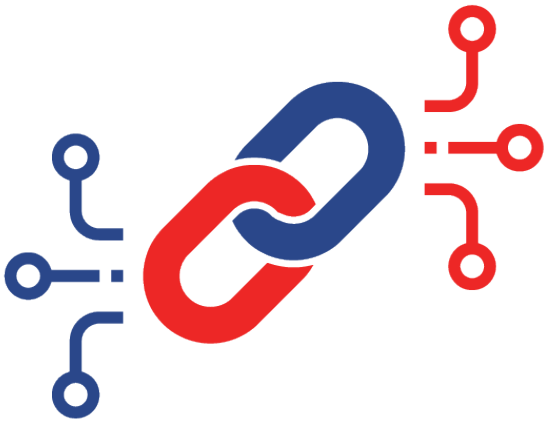


CYBER SCRM

FOR CRITICAL SYSTEMS

Proceedings of the Cyber Supply Chain Risk Management for Critical Systems (CySCRM '24)

Presented October 29 & 30, 2024,
at Pacific Northwest National Laboratory
Richland, Washington

Organizing Committee

CySCRM '24 was jointly led by Pacific Northwest National Laboratory (PNNL), Lawrence Livermore National Laboratory (LLNL), and the University of Texas El Paso (UTEP). The support of these institutions, in funding, promotion, and resources, was critical to the success of the event.

PNNL
Jess Smith, PhD (Chairperson)
Rebecca Redeker
Lisa Campbell
Animesh Pattanayak

LLNL
Bob Hanson
Judd Morse
Ashley Zepeda
Mike Nygaard

UTEP
Deepak Tosh, PhD

Invited Speakers

Dr. Adam Kimura, Research Leader at Battelle Memorial Institute

Dr. Adam Kimura is a research leader at Battelle Memorial Institute and has been working in the field of trusted and assured microelectronics since 2013. He currently serves as the technical director for Battelle's cyber trust and analytics research and development portfolio and is the principal investigator for Battelle's post-silicon verification and validation programs. Dr. Kimura is an inventor on fifteen pending or issued patents and currently holds his BS, MS, and PhD in electrical and computer engineering from The Ohio State University.

Justin Pascale, Principal Industrial Consultant at Dragos

Justin Pascale is a Principal Industrial Consultant at Dragos, Inc. and serves as the primary subject matter expert with customers to perform architecture assessments, network vulnerability assessments, consequence driven modeling, etc. of their industrial environment.

Prior to joining Dragos, Justin worked as a cyber security consultant for General Electric. In this role he was responsible for leading cybersecurity risk assessments and advising ICS/OT customers on practical solutions to their security challenges. Justin began his career in the United States Army as a Military Intelligence Officer, where he conducted cyber threat analysis in support of intelligence deliverables.

Bob Kolasky, Senior Vice President of Critical Infrastructure at Exiger

Bob Kolasky is Senior Vice President for Critical Infrastructure at Exiger where he focuses on developing cutting-edge risk management solutions for critical infrastructure companies and supporting government agencies. In this role, Mr. Kolasky leads market strategy for addressing third party and supply chain risk in critical infrastructure and delivering analysis.

Mr. Kolasky also serves as a Nonresident Scholar in Technology and International Affairs Program at the Carnegie Endowment for International Peace, as a Senior Associate for the Center on Strategic and International Studies (CSIS), and a Senior Fellow at Auburn University's McCrary Institute. He is the former Chair of the High-Level Risk Forum for the Organization of Economic Cooperation and Development (OECD).

Mr. Kolasky joined Exiger after 15 years as a senior leader in the Federal government, where he was responsible for foundational work in national security risk management and election security. He was the founding Director for the Cybersecurity and Infrastructure Security Agency's (CISA) National Risk Management Center at the Department of Homeland Security. As one of CISA's Assistant Directors, he oversaw efforts to facilitate a strategic, cross-sector government and industry risk management approach to cyber and supply chain threats to critical infrastructure.

Mr. Kolasky has served in a number of other senior leadership roles for DHS, including Acting Assistant Secretary and Principal Deputy Assistant Secretary for Infrastructure Protection.
Earlier in his career, Bob was a management consultant, a journalist and an entrepreneur. He graduated from Dartmouth College and from the Harvard Kennedy School of Government.

Table of Contents

Papers

Abstracts

Abstracts, Cont.

DOD Product Assurance Playbook Process for Commercial-Off-The-Shelf Products
C. Crossley

Leveraging SBOMS for Vulnerability Management
C. Crossley

PHICS: Programmable Hardware Image Collection System
C. Weitz, G. Gloria, M. Kirkland

Software Composition Analysis Tools: SCRM Value Add or Lossy Noise Machines
R. Erbes, M. Gallegos

The Current State of C-SCRM
N. Henderson

The Software Supply Chain Business Case
D. Sparrell

PAPERS

A Framework for Post-silicon Analog Design Verification and Validation

Vince McKinsey[1], Andrew Elliott[1], Benjamin Hayden[1], Yaser Helal[1],
Adam R. Waite[1], Jon Scholl[1], Jamin McCue[2], Shane Smith[3], and Adam Kimura[1]
[1]Battelle Memorial Institute, Columbus, OH, USA
[2]Air Force Research Lab, Dayton, OH, USA
[3]SenseICs, Columbus, OH, USA

***Abstract*—A post-silicon Analog, Mixed-Signal, and RF (AMS/RF) design verification and validation method was developed called CHARGE (Circuit Hierarchy Analysis, Review, and Graph Evaluation). AMS/RF design verification and validation is important due to the critical role of AMS/RF designs in DoD systems. The CHARGE framework utilizes a Parametric Graph Isomorphism (PGI) algorithm that enables detection and identification of design deviations in AMS/RF circuits. An experiment was devised that tested the CHARGE framework against two AMS/RF designs with deviations in the recovered design and contrasted against golden versions of the same. Experiments in this paper have demonstrated that the CHARGE framework successfully identifies and spatially highlights both structural and parametric deviations in two AMS/RF test articles.**

***Keywords*—verification, validation, hardware assurance, trust, microelectronics, integrated circuits, RF, mixed-signal, analog, GDSII, layout, untrusted foundry**

I. Introduction

Over the last few decades, global economics and market trends within the semiconductor industry have driven modern microelectronics to offshore and untrusted locations for fabrication [1]. With virtually no visibility into the manufacturing supply chain, it is nearly impossible for designers or program offices to know, with any level of confidence, if the integrated circuit (IC) chip has been compromised at a point in the manufacturing process. To address this challenge, post-silicon verification and validation (V&V) techniques have been developed for assuring the manufactured design's equivalence to the trusted golden design [2], [3]. Significant progress has been made over the years with digital design V&V, developing tools that scale to perform equivalence checks between the recovered and golden design across physical layout, function, logic, graph, and timing modalities [4], [5]. These techniques, however, do not map well into the analog domain due to fundamental differences between digital and analog designs. One example of a fundamental difference is the discrete nature of digital versus the continuous nature of analog, which confounds logical and functional checking algorithms. Thus, this limits the number of tools, techniques, and approaches for assuring analog, mixed-signal, and radio frequency (AMS/RF) designs. Microelectronics in modern DoD systems are often complex Systems on a Chip (SoC) that contain sizable percentages of AMS/RF components. Assuring these design modules are equivalent to the golden is critical for a program office to have confidence in the chip's assurance prior to its deployment.

In this paper, we introduce the Circuit Hierarchical Analysis, Review, and Graph Evaluation (CHARGE) framework, a first of-its-kind V&V framework that performs post-silicon V&V on AMS/RF designs. We provide an in-depth review of the CHARGE framework, outline the methodology, and discuss how it aligns and integrates into the larger portfolio of post-silicon V&V techniques. We demonstrate the capability of the CHARGE framework with an experiment that ingests two different AMS/RF designs. The first is a fabricated 14 nm FinFET Configurable Ring Oscillator (CRO) containing design deviations that were fabricated at the foundry. These deviations, however, are not present in the golden GDSII layout. The second is a 14 nm FinFET 4-bit flash Analog-to-Digital Converter (ADC) intended as the mixed-signal component to an RF receiver chain. The deviations elude the traditional functional and logical equivalency checks used in digital V&V, however, the CHARGE framework for AMS/RF V&V is able to identify and flag them for deeper analysis.

II. Limitations of Existing Verification and Validation Approaches

The central issue that differentiates AMS/RF design from digital design is the way analog relates to different mathematical and physical phenomena. This impacts how AMS/RF V&V must be performed, compared to Digital V&V. Digital circuits are restricted to handling boolean algebra through discrete logic. AMS/RF cannot be simplified down to boolean algebra. This can be verified if we consider the Effective Number Of Bits (ENOB) that analog signals can contain due to their continuous nature, which is usually more than one ENOB. Put another way, analog signals can contain more than one bit of information at any given moment, digital signals only ever contain one bit of information. Traditional V&V approaches, such as logical equivalence, solely consider the case when ENOB equals one, which is Boolean Logic. Thus, AMS/RF's continuous nature makes traditional V&V approaches not applicable. AMS/RF circuits can accomplish

certain tasks in far less Power, Performance, and Area (PPA) compared to a digital circuit (e.g., Analog Adder vs Digital Adder or Analog Pulse Width Modulator (PWM) vs Digital PWM). AMS/RF circuits can also perform tasks that digital circuits are incapable of achieving (e.g., Radiating power into space). Consequently, sources of deviation in a design can expand beyond logical failures (e.g., changes in binary logic) to include performance changes (e.g., bandwidth in a filter, gain in amplifiers, slew rate) and physics related changes (e.g., the introduction of negative capacitance, mobility change, or doping change). As such, traditional design verification tools such as Layout vs Schematic (LVS) and Logical Equivalence Check (LEC) do not apply directly to AMS/RF V&V in a post-silicon fabrication context. This limitation of digital design verification tools is illustrated in Figure 1, which also illustrates how the CHARGE framework's new paradigms expand on the existing Digital V&V techniques to include new equivalence check methods that provide coverage over AMS/RF designs. To fully cover the spectrum of functions enabled by AMS/RF systems, a comprehensive solution would compare many electrical effects of the golden design and the recovered design, within a specified tolerance range. For example, solving electromagnetics within materials and surfaces on a large scale with modern computation is currently an active research topic [6]. To that end, we seek computationally feasible methods or techniques that can cover some part of the space covered by AMS/RF circuits.

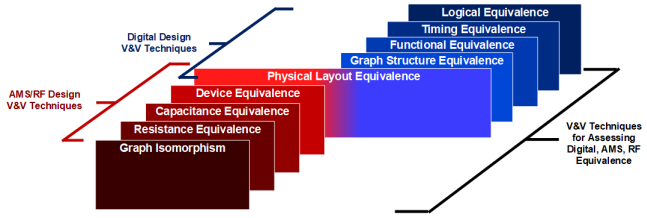


Fig. 1. The CHARGE framework integrates new AMS/RF V&V Techniques into the existing Digital Design V&V Equivalency Checking Techniques from [3], [4] providing a comprehensive post-silicon V&V tool suite for performing assurance on any type of design.

III. AMS/RF V&V Framework

The CHARGE framework introduces a novel approach to post-silicon AMS/RF V&V incorporating concepts across the AMS/RF disciplines. These disciplines usually start at the system specifications, then build up schematics with simulations, and then transition to a parasitic representation that captures first order resistance, capacitance, and inductive effects for greater precision of circuit behavior. Accordingly, the CHARGE framework first analyzes the recovered design schematically for topology and component parameter verification, then validates it at the parasitic level. Schematic level V&V quickly captures deviations in the electrical domain, whereas parasitic level V&V captures deviations in both the physical and electrical domains at the expense of computational complexity. In both cases, the CHARGE framework leverages practices from the existing field of graph theory, by representing extracted circuits as graphs to perform V&V.

Before getting to a graph representation, AMS/RF circuit graph analysis requires Simulation Program with Integrated Circuit Emphasis (SPICE) netlists from both the schematic level and parasitic level. A SPICE netlist is a textual representation of a circuit and includes all necessary components with parameters, component models, and connections. These SPICE netlists are extracted from both the recovered and golden layouts using Cadence Pegasus (or other schematic extraction Electronic Design Automation (EDA) tools) to create SPICE netlists for the schematic. Cadence Quantus (or other parasitic extraction EDA tools) is used to create SPICE netlists that have parasitics included for the layout. Once the proper SPICE netlist is generated, the SPICE netlist is converted into a graph. An example of converting from SPICE to a graph in GraphML format is shown by Figure 2. The conversion is done by taking each net, or electrical device, and creating a node in the graph. Then, edges are connected between graph nodes that represent a connection between a net and a device pin that existed in the SPICE netlist. Additional metadata about position and device parameters are then embedded in the nodes. Once the nodes, edges, and metadata are made, the graph is formed. This graph now enables performing parametric isomorphism checks. Through parametric isomorphism checks, detailed in Section IV, deviations in circuit topology and component parameters can be quickly identified.

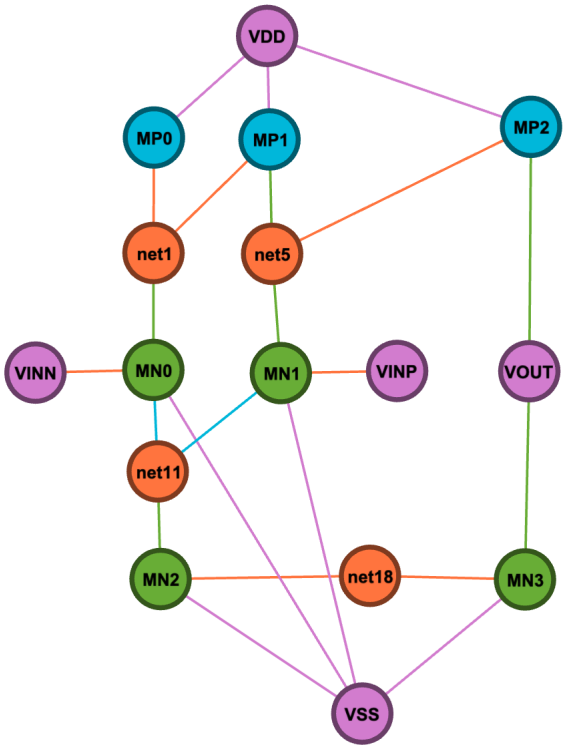
IV. Analysis Methods

The CHARGE framework expands into AMS/RF V&V by introducing new paradigms into the V&V space that traditional V&V approaches can't handle when dealing with AMS/RF designs. This new approach, called Parametric Graph Isomorphism (PGI), is applied on circuit graphs that are generated from SPICE netlist files. This graph then undergoes partitioning to reduce the complexity of the search space. The resulting graphs are then digested by PGI to perform the comparison and equivalence checks to find deviations in the recovered design. The comparison and equivalence checks expand beyond pure graph isomorphism by additionally being able to compare the metadata of the nodes. This means the tools can be used to compare component values (such as resistance or capacitance) or any other data that is included in the metadata (such as location of the devices). The process for this analysis is shown in Figure 3, wherein two layouts, one for the golden design and one for the recovered design, are ingested into a parasitic extraction (PEX) tool to produce a SPICE netlist.

The concept of PGI is a derivative of Graph Theory's isomorphism [7], wherein we compare the graph structure between the golden graph and the recovered graph. In PGI, two nodes, one in the golden graph and one in the recovered graph, are considered "matches" if the nodes have the same connectivity and metadata within a specified tolerance. The specified tolerance is to account for process variations and imaging errors, which allows PGI to show gradients of change

```
.SUBCKT opamp VDD VINN VINP VOUT VSS
MP2 VOUT net5 VDD VDD pfet m=1 l=14n nf=1 nfin=2 fpitch=48n cpp=78n ngcon=1
+ p_la=0 plorient=0 analog=-1.0
MP1 net5 net1 VDD VDD pfet m=1 l=14n nf=1 nfin=2 fpitch=48n cpp=78n ngcon=1
+ p_la=0 plorient=0 analog=-1.0
MP0 net1 net1 VDD VDD pfet m=1 l=14n nf=1 nfin=2 fpitch=48n cpp=78n ngcon=1
+ p_la=0 plorient=0 analog=-1.0
MN3 VOUT net18 VSS VSS nfet m=1 l=14n nf=1 nfin=2 fpitch=48n cpp=78n ngcon=1
+ p_la=0 plorient=0 analog=-1.0
MN2 net11 net18 VSS VSS nfet m=1 l=14n nf=1 nfin=2 fpitch=48n cpp=78n ngcon=1
+ p_la=0 plorient=0 analog=-1.0
MN1 net5 VINP net11 VSS nfet m=1 l=14n nf=1 nfin=2 fpitch=48n cpp=78n ngcon=1
+ p_la=0 plorient=0 analog=-1.0
MN0 net1 VINN net11 VSS nfet m=1 l=14n nf=1 nfin=2 fpitch=48n cpp=78n ngcon=1
+ p_la=0 plorient=0 analog=-1.0
.ENDS
```

(a) The SPICE netlist extracted from a generic amplifier made in 14 nm FinFet technology.



(b) The equivalent graph of the previous SPICE netlist.

Fig. 2. An example of an amplifier SPICE netlist (a) being converted to a graph (b). The edges of the node lead to the voltage nodes (orange) of the device in SPICE. Data about the device given on the SPICE line (such as number of fingers, Length, or Width) is stored in the metadata of the node. Note that the voltage nodes themselves are also nodes in the graph. Those voltage nodes then lead to other nodes via edges.

on component and parasitic values. This enables highlighting regions of interest for highly deviated components. However, these tolerance values must be chosen carefully. Too large of a tolerance range could result in failure to identify a deviation (Type II), but setting the threshold too tightly may result in large amounts of false positives (Type I) due to small process variations that would not be expected to cause deviated behavior. An example of the various deviations that can be captured by this tool is illustrated in Figure 4, wherein we see not just missing nodes, but nodes that have various design parameters altered. PGI effectively enables equivalence checking of electrical, physical, and other relevant parameters in not only designed devices of post-silicon layouts but also parasitic elements.

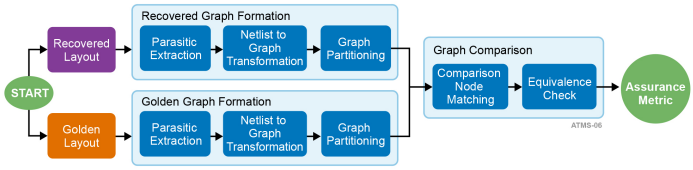


Fig. 3. Overview flow diagram showing the general flow of the analysis. Two layouts, one for the golden design and one for the recovered design, are ingested into a PEX tool to produce a SPICE netlist. That SPICE netlist is transformed into GraphML and undergoes graph partitioning to reduce the complexity of the search space. The resulting graphs are then digested by PGI to perform the final comparison and equivalence checks to find deviations in the recovered.

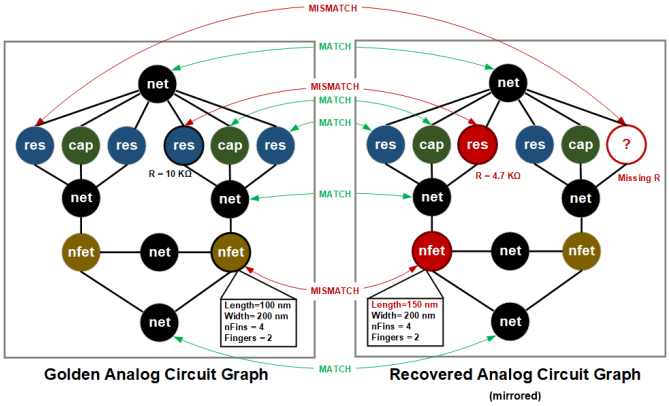


Fig. 4. Example golden vs. recovered parametric graph isomorphism, showing a mismatch on one of the resistors, a match on two of the nets, and a mismatch on a parameter in one of the NFET devices. Mismatches can also be found in capacitance values, voltage nodes, etc.

V. Experimental Setup

In order to demonstrate the CHARGE framework's ability to ingest and parse an AMS/RF design, compare it to a trusted golden reference version, and identify deviations, an experiment was performed on two AMS/RF designs as test cases for validating our post-silicon AMS/RF assurance approach. These chosen designs were a CRO from the Headache chip and a 4-bit flash ADC from the Insomnia chip.

A. 14 nm FinFET Headache Configurable Ring Oscillator

The first design, dubbed the Headache chip, was designed and fabricated in a 14 nm FinFET technology and contained a variety of AMS/RF circuits. Our work focused on a ring oscillator design, which serves as an entropy source for a True Random Number Generator circuit that generates cryptographic keys. Headache version A represents the golden reference layout sent to the foundry. Headache B is a cloned version of Headache A that contains small, stealthy variations in the AMS/RF circuitry that impact the performance of the ring oscillator circuit. The Headache B chip is representative

of the threat case where a modification was made at some point in the manufacturing supply chain. These changes elude the traditional digital post-silicon V&V techniques.

B. 14 nm FinFET Insomnia Analog-to-Digital Converter

The second design considered, called the Insomnia chip, implements a 14 nm RF receiver chain, consisting of a Low Noise Amplifier (LNA), Mixer, Lowpass Filter (LPF), and a 4-bit flash ADC. The Insomnia chip has two versions of this RF chain implemented on-chip. The first version is a "golden" reference chain, which serves as the "ground truth" for the entire chain. The other version is a "deviated" chain, which represents a scenario where a modification was made at some point in the manufacturing supply chain. Each block of the chain (the LNA, the mixer, the LPF, and the ADC) had deviations introduced that were analog in nature. This work focuses on analyzing the variant ADC block of the Insomnia chip.

C. Setup Summary

The Headache chip's CRO deviation replaces several fill cells, which is a change that is detectable using existing analysis techniques [3], [4], allowing us to validate our methods against a known result. Insomnia's flash ADC deviation is purely resistive in nature, made by increasing the length of metal lines to increase resistance. This deviation would elude traditional logic and function V&V approaches used for digital circuits.

Each GDSII file was loaded into Cadence Quantus in order to perform a PEX. In all cases, capacitive and resistive extraction was performed using identical settings. These extracted SPICE netlists were then converted into graphs and the graphs were processed using PGI to detect and identify anomalies in the deviated layouts. The question we ask is: *can the CHARGE framework capture those modifications that elude other established V&V techniques, enabling AMS/RF V&V?*

VI. Experimental Results and Analysis

A. 14 nm FinFET Headache Configurable Ring Oscillator

We first recovered the full circuit design files by applying sample delayering, imaging, and feature extraction techniques across the entire 14 nm FinFET design stack-up of Headache B [8], [9]. We then extracted the as-fabricated layout from Headache B and ingested it and the golden Headache A layout into the CHARGE framework for comparison, according to the flow diagram shown in Figure 3. Once analyzed, the framework identified the deviations within the AMS/RF circuit graphs and highlighted the areas requiring deeper inspection. Using our PGI approach, we detected and determined that the changes made to Headache B were decoupling capacitor fill cells being switched out for normal fill cells with no capacitors. We found 492 deviated devices due to the missing via connections in the ring oscillator. Figure 5 shows the region of interest and the extracted cells of the ring oscillator as well as highlighted transistors impacted by the modifications.

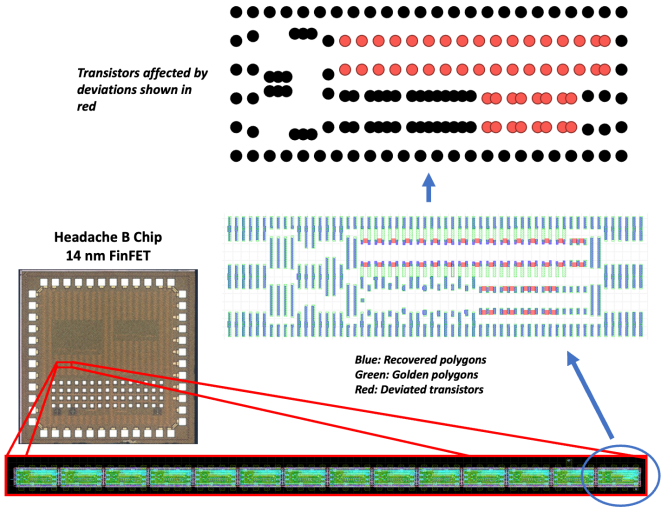


Fig. 5. Headache B was ingested into the CHARGE framework and analyzed against Headache A for deviations in the AMS/RF circuitry. One stage of the ring oscillator is shown with the affected transistors in red and non-affected transistors in black due to the introduced deviations.

B. 14 nm FinFET Insomnia Analog-to-Digital Converter

Next, we took a single variant of the 4-bit ADC from the Insomnia chip, seen in Figure 6, and applied PGI against the golden 4-bit ADC from the same chip. The PGI algorithm then produced a graph with marked nodes which represent devices that had parameters out of tolerance as seen in Figure 7. Using the positions of the devices, the out-of-tolerance devices can be located in the layout for further inspection.

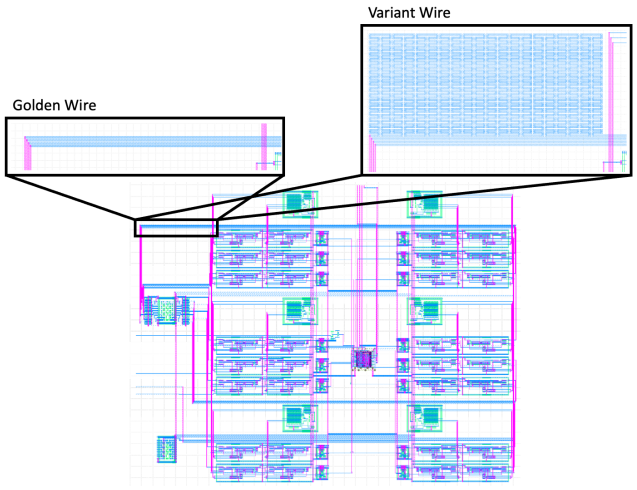


Fig. 6. ADC Variant layout showing both the golden and variant versions of the affected wire.

C. Experimental Summary

These two experiments show that PGI is an effective method for determining AMS/RF deviations in designs. Using PGI,

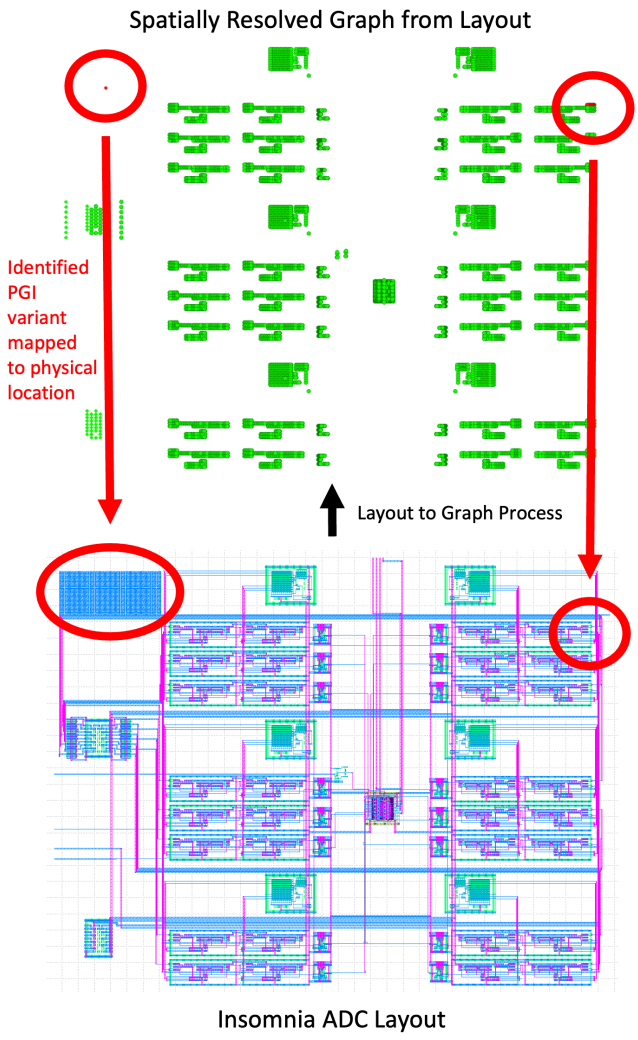


Fig. 7. ADC Variant layout (shown at the bottom) contrasted with the PGI generated graph (shown at the top) showed the approximate location of the variant that have been marked due to parameters being out of tolerance. The left most circle is the actual wire that the variant changed. The right most circle is the effect the variant had on the connectivity of the layout.

the CRO in Headache completed in 1.47 minutes, which had a layout area of 560 μm^2. The ADC in Insomnia completed in 45.31 minutes, which had a layout area of 21,970 μm^2. The PGI tool runtime and design size table is shown below in Table I along with additional information about graph sizes and layout area.

VII. Discussion

Experiments have demonstrated that the CHARGE framework successfully identifies and spatially highlights both structural and parametric deviations in two AMS/RF test articles. The introduction of PGI into the problem of post-silicon AMS/RF V&V represents a powerful capability by being able to detect and identify deviations that are present due to topological and parasitic effects in a design, in addition to connectivity and component-value checks present in a standard LVS comparison. Besides the example of resistance shown in this work, PGI can detect and identify deviations in other electrical properties such as capacitance, or any other numerical property, in the same way. By visually highlighting the deviated area for the user and reporting the type of deviation encountered, the framework can be used by a V&V team to pinpoint the deviation for further analysis.

After PGI approximately located the deviations, a circuit designer with AMS/RF experience analyzed the function of the deviations that were found. For the CRO, it was concluded that certain FILL cells were replaced with changed FILL cells that presented different capacitance. The changed capacitance would change the CRO's typical oscillation frequency, which was subsequently validated using Cadence Spectre. For the ADC, it was determined was that additional resistance was being added into the feed wire from the reference circuit to the ADC's comparators. The additional resistance would impact the kickback from the ADC's comparators in the comparator ladder, which was subsequently validated using Cadence Spectre.

A factor limiting the effectiveness of the CHARGE framework is the scaling of the approach. While the graph isomorphism problem is of NP-intermediate complexity [10], PGI reduces this complexity by further constraining node matches based on parametric values. Runtime can be reduced by increasing the minimum parasitic thresholds during PEX, at the cost of decreased resolution of the parasitic graph. In practice, default PEX settings for the PDK used to design the test articles resulted in graphs of a size that were not tractable for our current implementation. By reducing minimum PEX values, we extracted graphs that PGI checks were able to process in under an hour on our test articles running on a standard workstation.

While the CHARGE framework was developed with AMS/RF V&V in mind, the methodology described here can be applied to digital circuits as well. Deviations in a digital circuit that arise due to complex analog behaviors are not detectable when modeling the circuit logically. By considering a digital design as an analog system, the CHARGE framework can be applied to digital circuits to identify deviations that exist due to these analog effects, thereby providing additional modalities of deviation detection and identification. Once deviations are detected in a digital design, information provided by the CHARGE framework about the location and nature of the deviations can be utilized to further investigate the logical effect that the deviations may cause. Due to the complexity and density of modern digital circuits, further optimizations may be necessary to ensure analysis can be completed in a timely fashion.

VIII. Conclusions

This work extends existing post-silicon V&V science beyond the digital domain and into the AMS/RF domain. Here, we provided a newly developed proof-of-concept framework designed to detect and identify deviations in AMS/RF circuits after fabrication. For this demonstration, a real-world

TABLE I
PGI TOOL RUNTIME AND DESIGN SIZE

Device	Area (μm^2)	# of Devices	# of Nodes	# of Edges	Time (mins)
CRO Variant	∼560	2160	4744	8640	1.47
CRO Golden		2160	3883	8640	
ADC Variant	∼21970	9440	12022	36920	45.31
ADC Golden		9439	12020	36918	

Note: "# of Devices" is the number of resistors + number of capacitors + number of transistors

fabricated chip that contained modifications in the AMS/RF circuitry, in addition to a newly developed AMS/RF chip currently being fabricated, were selected for V&V. We applied our physical design decomposition and design file extraction approach to recover the as-fabricated layout and ingested it with the golden layout into the CHARGE framework for detection and identification of unknown design deviations. Finally, we successfully identified and located all the deviations, thus demonstrating its potential to address the challenge of ensuring the integrity of AMS/RF components on manufactured chips. There are several avenues that could be explored for future work. Research for techniques and methods to capture radiative effects, such as those experienced by antenna and RF circuits in general, is ongoing. Future research should also seek to cover material effects.

ACKNOWLEDGEMENT

Funding Acknowledgement: This research was sponsored by the Air Force Research Lab as part of the OUSD Trusted and Assured Microelectronics Program.

REFERENCES

[1] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.

[2] A. R. Waite, J. H. Scholl, J. Baur, A. Kimura, M. Strizich, and G. D. Via, "IC decomposition and imaging metrics to optimize design file recovery for verification and validation," in *ISTFA 2020: Papers Accepted for the Planned 46th International Symposium for Testing and Failure Analysis*. ASM International, Dec. 2020.

[3] A. Kimura, J. Scholl, J. Schaffranek, M. Sutter, A. Elliott, M. Strizich, and G. D. Via, "A decomposition workflow for integrated circuit verification and validation," *J. Hardw. Syst. Secur.*, vol. 4, no. 1, pp. 34–43, Mar. 2020.

[4] K. Liszewski, T. Mcdonley, J. Delozier, A. Elliott, D. Jones, M. Sutter, and A. Kimura, "Netlist decompilation workflow for recovered design verification, validation, and assurance," *Cryptology ePrint Archive*, pp. 1–8, 2021.

[5] A. Waite, J. Kelley, J. Scholl, Y. Patel, J. Baur, J. Schaffranek, V. Mckinsey, G. D. Via, and A. Kimura, "Verification and validation of a 45 nm node device utilizing a novel homeomorphic error checking tool for intelligent feature extraction," in *Government Microcircuit Applications & Critical Technology (GOMAC Tech) Conference*, 2021.

[6] D. P. Zoric, D. I. Olcan, and B. M. Kolundzija, "Solving electrically large em problems by using out-of-core solver accelerated with multiple graphical processing units," in *2011 IEEE International Symposium on Antennas and Propagation (APSURSI)*, 2011, pp. 1–4.

[7] A. Bondy and U. Murty, *Graph Theory*, 1st ed. Springer London, 2008.

[8] A. R. Waite, Y. Patel, J. J. Kelley, J. H. Scholl, J. Baur, A. Kimura, E. D. Udelhoven, G. D. Via, R. Ott, and D. L. Brooks, "Preparation, imaging, and design extraction of the front-end-of-line and middle-of-line in a 14 nm node finfet device," in *2021 IEEE Physical Assurance and Inspection of Electronics (PAINE)*, 2021, pp. 1–6.

[9] A. Kimura, A. Waite, J. Scholl, J. Kelley, J. Schaffranek, E. Haines, T. McDonley, K. Liszewski, Y. Patel, J. Baur, and et al., "Applied microelectronics assurance verification and validation for unknown variant identification and analysis in a fabricated 14 nm finfet design," *Government Microcircuit Applications & Critical Technology Conference (GOMAC Tech)*, Mar 2022.

[10] U. Schöning, "Graph isomorphism is in the low hierarchy," *Journal of Computer and System Sciences*, vol. 37, no. 3, pp. 312–323, 1988. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0022000088900104

Comparing Bills of Materials

Lucas Tate
Pacific Northwest National Laboratory
lucas.tate@pnnl.gov

Rebecca Jones
Pacific Northwest National Laboratory

Doug Dennis
Pacific Northwest National Laboratory

Tatyana Benko
Pacific Northwest National Laboratory

Jody Askren
Pacific Northwest National Laboratory

***Abstract*—Bills of materials (BOMs) are quickly becoming an effective tool for managing supply chain risk. As more BOMs enter circulation, the ability to compare them will be crucial to understanding how products differ and in managing BOMs from different tools or sources. This paper will describe some of the challenges of comparing BOMs followed by a discussion of several comparison methods.**

***Index Terms*—bill of materials, BOM, HBOM, SBOM, comparison, graph comparison**

I. Introduction

Modern supply chains are increasingly complex. A supply chain for a single product can include multitudes of suppliers, manufacturers, distributors, and more. Common components that drive efficiency and reduce costs also serve to increase the potential damage of any one compromised component. This complexity poses significant challenges for managing risk because a vulnerability in any one component may have outsized consequences and the knowledge of what's inside any given product may be distributed across different companies or across the globe.

Understanding supply chains risks is more important today than it has ever been. Significant vulnerabilities and breaches continue to highlight the growing risks that supply chains face whether they were introduced maliciously or unintentionally. Well known events such as Log4 Shell [18] or Spectre/Meltdown [9] demonstrated how weaknesses in components could leave millions of products susceptible to attack. Solar winds [29] and the recent XZ Utils backdoor [10] demonstrated how malicious actors could subvert elements of a supply chain in an attempt to exponentially increase their reach. The increasing persistence and sophistication of supply chain threats requires new tools to combat them.

Recently, BOMs have been gaining traction as a tool to increase our supply chain understanding and help respond to this threat. A BOM contains details of the components that are used in building a product. While certainly not a silver bullet, understanding what is inside systems is a first step toward protecting them and responding once protections have failed. Work around software BOMs (SBOMs) far outpaces other proposed BOMs such as hardware (HBOMs) or artificial intelligence (AIBOMs) with regulations such as Executive Order 14028 [12] and the European Union (EU) Cyber Resilience Act (CRA) [6] helping to further global adoption. Encouragingly, interest in BOMs has spawned an abundance of new research dedicated to better understanding how to generate, exchange, store, and operationalize BOMs to improve risk management. As BOMs become ubiquitous, we anticipate a growing need for the ability to compare them which will be the focus of this paper.

Due to the nature of BOMs, comparing them is effectively looking at how the composition of two products differs. There are a variety of cases where that might be useful. It may be important to understand how the composition of a product changed with a version update or patch. Another use case might be evaluating a received BOM against an authoritative reference BOM. Comparisons can also be utilized to understand temporal changes that arise in dynamic systems or variation across a family of products. Beyond these use cases, we also find comparison methods useful for identifying inconsistencies in the creation of BOMs themselves. These inconsistencies are discussed later, with **the irony being that many of these inconsistencies that make comparison difficult are most easily discovered via comparison**.

In this work, we'll start by discussing previous work on comparing BOMs in Section II, followed by some of the barriers to comparing BOMs in Section III. Section IV discusses select methods for comparison followed by several examples that illustrate the comparison of two SBOMs in Section V and two HBOMs in Section VI. Lastly, Section VII will summarize our conclusions.

II. Previous Work

As the application of BOMs continues to evolve, particularly in the context of cybersecurity and supply chain management, significant research has been dedicated to understanding and improving how BOMs are generated, compared, and utilized.

Early research on BOMs predominantly focused on enhancing data management techniques to cope with the complexity of large-scale manufacturing environments, such as implementing a control system to manage BOMs throughout a company [25] and automating the creation of BOMs using an object-oriented model [2]. The object-oriented programming model, similar to the graph method used today, allows for semantic relationships that can create multi-level BOMs with sub-components of components [3]. However, since relational databases like SQL are ubiquitous, using them to create

BOMs became more common [20]. Algorithms were invented to automate the creation of BOMs by determining which products were in a product order and then pulling the required component information from the database of parts [1]. While this method is efficient, flexible and simple, it does not allow for parts explosion or complex computations, especially when analyzing or comparing BOMs [21].

To tackle the growing complexity of intricate products, advancements in BOM structures have been proposed. The multi-level BOM model is particularly effective in managing software, hardware, or system variations, supporting efficient design and planning [31]. Its strengths lies in the ability to handle complex, hierarchical data that can be represented as a graph.

Further development in this area focuses on using graph databases, which store node and relationships, instead of relational databases to integrate product development with production planning [13]. BOM management, found at companies such as Neo4j and OpenBOM take this approach of using a graph database.

The main advantage of turning BOMs into graphs is pairing the knowledge of what's inside something with information about how those components relate to each other. Using graphs, multiple BOMs can easily be combined to gain new information on a larger system, especially when combined with graph visualization tools. Even when used on a single BOM, graph analysis techniques can provide new insights into a system [4]. Another contribution to the graph-theoretic approach is where BOMs are converted into generic BOM graphs using data mining techniques [24]. This method leverages graph theory to identify common substructures within BOMs, facilitating the detection of component reuse across different products. While the ability to uncover hidden patterns and relationships within BOMs is useful, its application could be hindered by computational complexity and graph scalability.

There have been a few attempts to leverage graph theory for BOM comparison.

Graph based similarity analysis has been used to highlight the importance of reducing unnecessary production variations [26] and derived graph similarity metrics have been used to describe the similarity of two BOMs to place them into their product families [23, 17, 27]. Tree reconciliation, matching components in one graph to the components of another, has been used in biology to compare phylogenetic trees and extended to BOMs [16]. Similar methods can be applied to BOMs to create new products quickly [15]. This is all of the literature we could find on applying graph comparisons to BOMs.

However, graph comparisons have a rich history that could be explored for comparison of BOMs [7]. Work has been done on comparing and visualizing trees, a specific type of graph with no cycles [11]. This is might be especially applicable to HBOMs since they tend to be more hierarchical while SBOMs have a tendency to create loops making it less suitable. Comparison methods range from similarity metrics, like the ones referenced for BOM graph comparison to unknown node comparisons, which attempt to create a mapping between the nodes of the two graphs [19]. In the latter, some algorithms use attributes while others focus solely on the graph structure. Taking advantage of the attributes is more computationally complex, but is important when comparing BOMs due to the metadata often captured in BOM components. Since there is not a lot of research on efficient and effective ways of comparing BOMs, graph comparison literature may offer promising methods for future application research.

III. Barriers to Comparing BOMs

BOMs today are extremely heterogeneous which makes subsequent comparison very difficult. Before undertaking comparison, it's important to understand some of the sources of variability [28, 32]. While addressing these differences will be outside of the scope of this paper, considering them will likely be a prerequisite to meaningful comparison. This list is not exhaustive but captures some of the variability inherent in HBOMs and SBOMs.

A. BOM Standards and Versions

Currently in the field there are not single authoritative standards describing the structure or contents of an HBOM or an SBOM. For software, the NTIA Minimum elements [5] has been an influential guidance document outlining a set of generally accepted minimum elements. The two leading standards, SPDX [30] and CycloneDX [22], provide detailed schemas that describe a data structure for the capture of SBOM information but the mapping between them can be lossy. Despite the fact that BOMs have existed in manufacturing for decades, development of HBOM conventions has not reached full maturity. In addition to the CycloneDX and SPDX standards, the Information and Communications Technology (ICT) Supply Chain Risk Management (SCRM) Task Force and Department of Homeland Security Cybersecurity & Infrastructure Security Agency (DHS CISA) released a comprehensive HBOM framework that differs from those standards, although it attempts to provide mappings to them as applicable [8]. Even within the same format, major and minor versions describe BOM changes that can impede direct comparison.

B. SBOM Types

Despite some foundational work defining SBOM types, little has been done to formally differentiate them within real world SBOMs. The result is that two SBOMs for the same software can be markedly different. As an example, a *source* SBOM created directly from the source code will include named dependencies that are imported or loaded. This will look very different from a *build* SBOM which will describe a specific release and may include information on the build process and produced files.

C. Naming Conventions

Naming challenges permeate every aspect of BOM generation and despite being a known problem, it is extremely difficult to solve. Software names remain an open challenge.

Efforts such as the common platform enumeration (CPE) and package uniform resource locator (PURL) have helped machine-to-machine readability, but they deviate from how people would colloquially refer to software. Other information such as a vendor is complicated by lack of authoritative conventions. As an example 'MSFT', 'Microsoft Corporation', and 'Microsoft' are all defensible values but the inconsistent recording makes systematically disambiguating them difficult. As a last example, hardware component identifiers have a tendency to describe a family of components. This means that sub-strings of the name can still accurately describe components, but additional characters identify it with increasing specificity. The 'AD7579' from Analog Devices describes a LC^2MOS 10-Bit Sampling A/D Converter, but 'AD7579JN' distinguishes it as having a specific temperature range, integral nonlinearity, and package. Neither name is incorrect, but they utilize different levels of specificity which makes comparison more challenging.

D. Hashing Approaches

Well known hashing approaches such as MD5, SHA1, SHA256, SHA512 are extremely useful for providing easily matchable fingerprints of files. Their reproducible and static nature make them much more attractive in certain cases than names. One problem is that they are susceptible to dynamic information such as timestamps that often appear in files. Since hashes don't convey why the files are different, it won't be obvious whether the difference is meaningful in a specific comparison. Furthermore, existing standards offer a lot of flexibility in choosing hashing methods which means a different method might have been used from one SBOM to the next reducing comparability.

E. Structure

Structure in this context describes the relationships between components within a BOM. This structure gives us additional information such as where a dependency is introduced into our software or which board a specific component is mounted on. The problem is that methods for describing these structures are not rigid leading to expected variability in how they are described from one BOM to the next.

F. Scoping

In this context, scoping describes the boundaries of a BOM; what goes inside a particular BOM and what falls outside. This is a surprisingly hard problem. As an illustrative hardware example, we could describe a Raspberry Pi with an HBOM. If that Raspberry Pi is mounted inside a consumer product, should the HBOM for the consumer product include an external reference to the Raspberry Pi HBOM? Should it duplicate the information from the Raspberry Pi? From a software perspective if a software application requires the use of a shared library in the operating system, should that be included in the SBOM? The lack of a clear and accepted answer to these scoping questions result in variability that needs to be considered.

G. Quantities

Quantities are an interesting property that appear in HBOMs. Rather than listing a component *n* times, we can indicate how many of them are present with an integer value. However, if one HBOM opts to list the components individually and another HBOM leverages the quantity field, then you have to rectify these different representations when conducting a comparison.

H. Order

BOMs are unordered. This makes sense because there isn't a correct order to describe components. This property however immediately adds a lot of variability to the files which poses some challenges for simple comparative methods such as tabular comparison especially in conjunction with name variation that will stymie attempts to sort the data.

IV. Comparing BOMs

Unfortunately there is no single method that can be used to compare BOMs. Instead, strategies need to be specifically chosen based on the data available in the BOMs in conjunction with consideration of the questions that need to be addressed. In a simple example, if we want to understand the difference in *licenses* between two BOMs that utilize a well-formed ontology such as the SPDX License List [30], a set comparison using exact match of the values can be employed successfully.

In other cases, understanding quantities can be important. If we consider two HBOMs and want to understand how the components differ, we may opt for a list comparison of *component names* which will tell us if there are different components, as well as whether there were a different number of them used. We know from earlier discussion that *component names* can have a lot of inherent variability, so depending on the consistency of the data, a fuzzy matching technique may be needed. Fuzzy matching allows for some threshold of leniency in matching values that are 'close enough' at the expense of possibly making errant matches.

Direct element comparisons are not the only useful comparisons to be made. Creative use of redundant or complimentary information can be exploited to great effect. This is especially useful when comparing SBOMs where *component names*, *hashes*, *cpes*, or *purls* can be used together to gain additional insights. As an example, matching *hashes* provide some level of guarantee that the contents of a software component match. When compared to *component names* this can identify interesting situations where 1) *component names* are the same, but the contents differ, 2) *component names* are different, but the contents are the same, or 3) offer consensus between *component names* and *hashes*. These comparisons can often uncover unexpected results that are invaluable for assessing quality and consistency of BOMs.

So far, the comparisons that have been discussed implicitly assume a comparison of two similar BOMs, but other comparisons can be useful as well. SBOM practitioners will likely be intimately familiar with the variability of generation tools. Despite the monumental efforts around standardization,

SBOMs tend to vary greatly from one tool to the next. This can happen for many reasons, but some examples include inconsistent assumptions, different methods or levels of technical ability, different opinions on the boundaries of an SBOM, and opinions on whether transitive dependencies should be included. Further exacerbating the issue is the fact that the details driving the variability are often proprietary or black-box. By comparing the lists and sets of elements within the BOMs it is possible to gain insights into design choices and accuracy of various tools.

BOMs of different size can also be compared. This can suggest that the components of one BOM are a subset or contained within another BOM. It is also possible to explore subsets of a BOM; in a system with built-in redundancy it may be useful to look at how duplicated modules or sub-assemblies compare. Much of the previous work done on comparing BOMs has been used to cluster or identify product families that contain similar components in a similar structure. Comparing products within a family can lead to quicker generation of new products, as well as streamlining supply chain processes.

A. List and Set Comparisons

A straightforward approach to comparing two BOMs is by simply comparing lists. Due to the popularity of JSON and XML file types for use in BOMs, this will often require parsing and/or flattening of the data to obtain the unordered lists. An example this could be comparing all the *component names* in one BOM to the *component names* in another BOM. List comparisons aid in understanding differences and quantities of components which can be particularly useful if multiples of a single component are present. It should be noted that there are cases where comparing multiple elements simultaneously is necessary. For example, two manufacturers could use the same name for a particular component in which case it may be more useful to compare the manufacturer and name at the same time so as to differentiate one component from the other.

Beyond simple lists it can also be useful to only consider the unique values or sets. This representation sacrifices information about the frequency of values, but can greatly reduce the burden of comparison. Set comparison might be useful when looking at something like *licenses* where knowing that a license appears in *n* dependencies is probably less important than just having the list of unique licenses.

B. Graph Comparisons

Graph matching techniques can provide useful insight into the comparison of two BOMs. The list comparison approach does not take advantage of the relationships which are present between elements in a BOM. For example, if there are duplicates of a chip on a piece of hardware, set comparisons will not capture that information, while a graph comparison will. Importantly, it will also show where the chips are physically in the hardware. This can be done through text like the list comparison or crucially, visualization techniques that are intuitively easy to understand.

In order to take advantage of this visualization ability, the BOMs are converted to a graph by making the components into nodes and the relationships between two elements as edges. Information about each element can be recorded in the graph by associating node attributes, and relationship types can be given by edge attributes. Then a node mapping where one set of nodes is mapped to the other is created by using the edges and the node attributes.

V. SBOM Example

To illustrate the SBOM comparison methods, we used Trivy to generate SBOMs from two versions (3.6.4 and 3.7.0) of Thingsboard, an open-source IoT platform for data collection, processing, visualization, and device management. Thingsboard is largely written in Java and uses Maven to manage the project. Several modifications were made to the SBOMs. First, duplicate software components that shared the same purl and metadata were collapsed to a single component. If relationships existed to the duplicates that were removed, they were tied to the remaining copy. While the exact nature of the duplication in these SBOMs wasn't clear, it should be noted that removing them could hinder certain analyses such as finding multiple copies of a dependency. Next we opted to remove all the npm front-end dependencies. This was only done to make the SBOMs a little smaller for illustration.

The comparison of the resulting SBOMs (results in Table I) showed that out of the 230 components, there were 214 unique names detected in version 3.6.4 and 218 unique names from the 234 components in version 3.7.0. Interestingly, the number of unique purls was also four apart: 170 to 174. We note that the difference between the number of components and purls was due to 60 components that did not contain purls in the SBOM. The number of unique purls is identical to the total purls which is expected after the deduplication, meaning that each purl appears only once. There were 10 component names that were duplicated; each one had a different unique purl, with a total of 16 duplicates. Looking at the comparisons of the unique names, there were 201 names that appeared in both SBOMs. Finally, doing a Jaro-Winkler string comparison with a threshold of greater than 0.85 on the node names between the two sets resulted in a total of 887 matches.

TABLE I
COMPARISON RESULTS FOR SBOM WHERE 3.6.4 (ONLY) INDICATES THE DIFFERENCE BETWEEN 3.6.4 AND 3.7.0

	3.6.4	3.7.0	3.6.4 (Only)	3.7.0 (Only)
Name	230	234	13	17
Unique Names	214	218	13	17
Purls	170	174	158	162
Unique Purls	170	174	158	162

Comparing two SBOMs does not have to be constrained to differences in components. We also considered whether the licenses reported were different between the generated SBOMs. Because the primary interest is whether there are any different licenses to consider, using set comparison (comparing the unique values) of recorded license would seem the obvious

choice. Version 3.7.0 contains only a single unique license: Apache-2.0. However, version 3.6.4 contains two unique licenses: Apache-2.0 and MIT. While this may look like version 3.6.4 is more complete, it is also important to know that only six dependencies in the version 3.6.4 SBOM had a license recovered by the tool. Version 3.7.0 had four dependencies with recovered licenses. This largely indicates that neither SBOM has a complete picture of the licensing exposure in Thingsboard.

The potential lack of information prompted us to manually review the licenses for the listed dependencies. We discovered that both versions of the software contained several other licenses such as the Eclipse Public License (EPL) and the Lesser GNU Public License (LGPL). These licenses have additional disclosure and representation requirements that may not be satisfied with the same rules as Apache-2.0 or MIT. Additionally, it is not inconceivable that an organization may apply additional scrutiny to licenses from the GNU Public License (GPL) family and would want to know that LGPL code is being used. While set comparison identified a notable difference in recorded licenses, a list comparison would have highlighted how few of the components captured license information.

We also examined the organizations of the dependencies. They were detected by using the first two segments of the package name in the purls. For example, given the purl "pkg:maven/com.example.foo@1.2.3", the organization is "com.example". There were fifty unique organizations in version 3.6.4 and 52 in version 3.7.0. Excluding Java standard library packages, we found that when moving to version 3.7.0 Thingsboard gained four additional external organizations and lost one. This information could be useful for situational awareness or subsequent corporate analysis where some producers may imply an increased/decreased level of assurance.

For the graph comparison, we converted the SBOMs into graphs using components as nodes and dependencies as edges. We then merged them on the name field using a depth-first search matching algorithm [14] with exact matching. A quick calculation shows us that 217 nodes were matched, 13 appeared only in version 3.6.4 and 17 appeared in version 3.7.0, matching the comparison in Table I. The visualization of the compared graphs did not add to the analysis and was therefore not included. To understand if the identified differences between the two SBOMs was due to small variance in the names, we employed fuzzy matching. It should be noted that with a priori knowledge of the name structure, some fuzzy matching approaches may be more successful, but here we naively employed Jaro-Winkler on the node names with an arbitrary threshold of .85 and found that there are 7 similar packages (see Table II). While we note that the graph wasn't particularly useful for visualization, the added constraint of the structure reduced the number of possible matches by 880 because rather than just finding similar names, the structure requires the names to also be in the same place as defined by the relationships in the SBOM.

In comparing these SBOMs, a combination of methods proved useful. List and set comparisons provided useful characterization of the SBOMs and identified some license irregularities. The graph method allowed us to ignore dependencies with similar names and focus on the differences we are more interested in, which were updated dependencies.

TABLE II
SIMILAR NAMES OF PACKAGES IN EACH THINGSBOARD VERSION WITH THE DIFFERENCES HIGHLIGHTED.

3.6.4	3.7.0
bcpkix-jdk15on	bcpkix-jdk18on
bcprov-jdk15on	bcprov-jdk18on
commons-collections	commons-collections4
hypersistence-utils-hibernate-55	hypersistence-utils-hibernate-63
javax.annotation-api	jakarta.annotation-api
swagger-annotations	swagger-annotations-jakarta
springfox-boot-starter	spring-boot-starter-webflux

VI. HBOM EXAMPLE

In this example, two HBOMs were created from two distinct instances of the same hardware product. The identities of the devices and components have been obfuscated, but the real characteristics of the comparison were preserved. Visualizations of the two graphs are shown in Figures 1 and 2.

A list comparison of the component names for the HBOMs immediately conveys differences shown in Table III. Despite being the same product, we see 35 components that only appear in HBOM 1 and 46 components that only appear in HBOM 2. Because the comparison of unique names reflects different counts, we can infer that some of the differences include components that appeared multiple times. In reviewing the differences, the most noteworthy finding was that one of the components was a circuit board which was especially surprising. Fuzzy matching in this case was not particularly useful because it flagged 743 possible matches which is difficult to sift through.

A comparison of the vendor revealed that there were 17 unique vendors in HBOM 1 and 14 unique vendors in HBOM 2. with only 13 vendors shared between the two. We offer no explanation as to why the vendors differ, but interestingly despite the BOMs representing the same product, the component supply chain looks different and may result in varying levels of risk exposure.

TABLE III
COMPARISON RESULTS FOR HBOM EXAMPLE WHERE HBOM 1 (ONLY) INDICATES THE DIFFERENCE BETWEEN HBOM 1 AND HBOM 2

	HBOM 1	HBOM 2	HBOM 1 (Only)	HBOM 2 (Only)
Name	156	169	35	46
Unique Names	99	108	17	26

Repeating the same graph comparison approach as in Section V, we generated the merged graph shown in Figure 3. Blue nodes indicate that the node names matched exactly, while the thick yellow edges indicate the node names matched approximately (Jaro-Winkler with a threshold of .85).

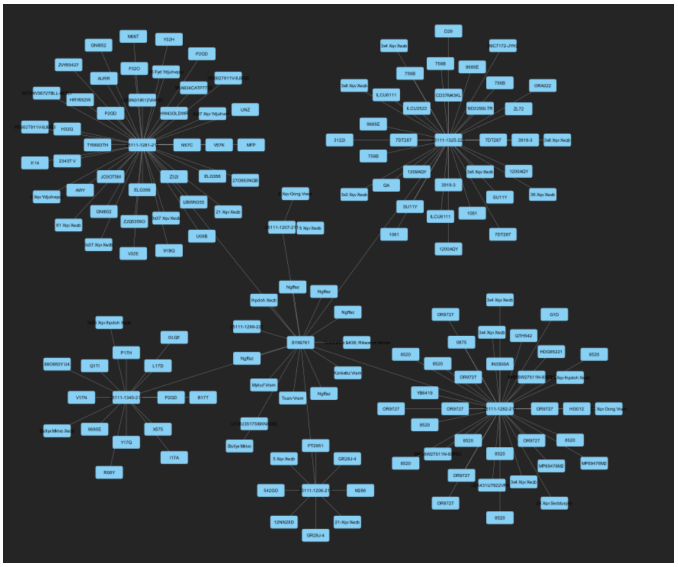


Fig. 1. Visualization of HBOM 1.

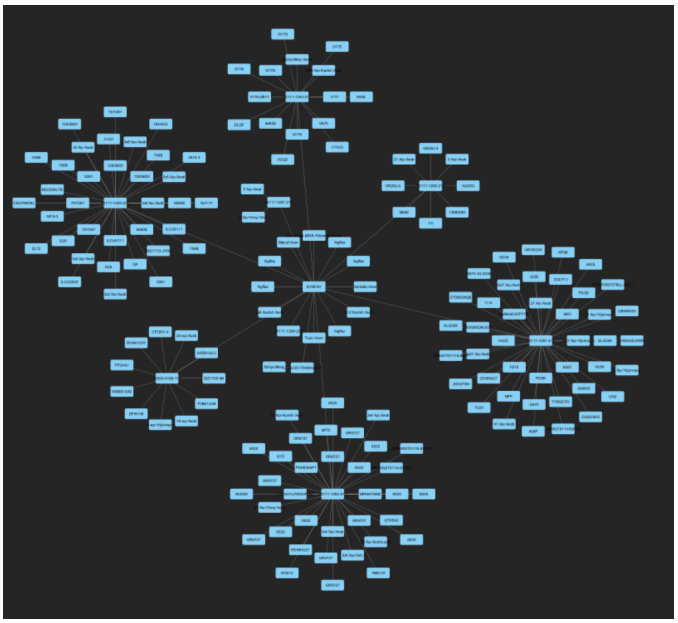


Fig. 2. Visualization of HBOM 2

This visualization is immediately useful. The most notable difference is the presence of a yellow circle in the top of Figure 3. This turned out to be the additional circuit board that was unexpectedly present in one of the devices, and we can quickly understand which of the different components correlate to the addition of that board. The remaining differences are drastically reduced by enforcing the graph structure (e.g. the component must be on the same board). The differences identified fell into one of three categories:

1) The component name was transcribed incorrectly. For example, the names were recorded as $IN3S00A$ and $IN3500$, where a 5 is switched for an S.
2) The difference was real and describes a component that had been switched out in production with a different but equivalent component.
3) The name in one HBOM was recorded with more specificity than the name of the equivalent component in the other HBOM. One component was named $V17N$ and one was recorded as $V17N - ZB11$.

Fuzzy match on the component names using the graph was particularly useful for this comparison. Instead of 743 possible matches, there are only 21 matches, identifying transcription errors and incremental component variations with high precision. As with the SBOM comparison, leveraging multiple comparison methods proved to be useful, but notably the graph provided significantly increased utility in the hardware example.

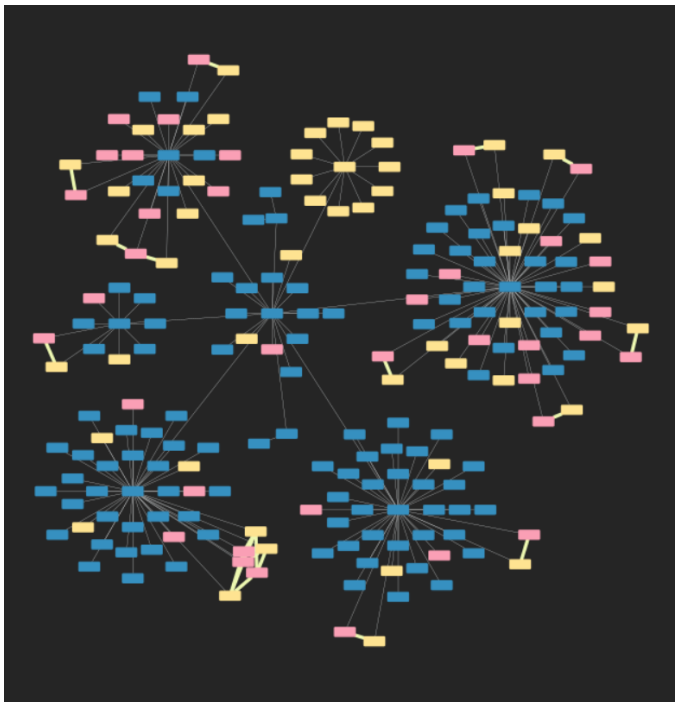


Fig. 3. Visualization of Merged Graph. Blue represents nodes found in both HBOM 1 and HBOM 2. Pink nodes are only found in HBOM 1 and yellow nodes are only in HBOM 2. Yellow edges indicate nodes where the names fuzzy matched.

VII. Conclusion

This paper provides an introductory discussion of the methods and challenges associated with comparing BOMs. Despite the recent abundance of energy and research in BOMs by government, industry, and academia, tools and methods to effectively compare BOMs lag behind. There is no single method of comparison that can effectively compare BOMs today. List, set, and graphical comparisons are complimentary and contribute to a foundational capability. As reference BOMs become more readily available, comparison methods will be vital to leveraging them. Ultimately BOM adoption, spurred by policy and regulation will continue to grow. The

ability to compare BOMs will be essential for understanding and reasoning about BOMs for supply chain risk management.

Acknowledgment

The authors wish to thank the Department of Energy (DOE) Cybersecurity, Energy Security, and Emergency Response (CESER) and the Cyber Testing and Resilience of Industrial Control Systems (CyTRICS) Program including Idaho National Laboratory (INL), Lawrence Livermore National Laboratory (LLNL), National Renewable Energy Laboratory (NREL), Oakridge National Laboratory (ORNL), and Sandia National Laboratory (SNL). Information Release: PNNL-SA-202952.

References

[1] A. O. Aydin and A. Güngör *. "Effective relational database approach to represent bills-of-materials". In: *International Journal of Production Research* 43.6 (2005), pp. 1143–1170. DOI: 10.1080/00207540512331336528.

[2] Sheng-Hung Chang, Wen-Liang Lee, and Rong-Kwei Li. "Manufacturing bill-of-material planning". In: *Production Planning & Control* 8.5 (Jan. 1997), pp. 437–450. ISSN: 0953-7287, 1366-5871. DOI: 10.1080/095372897235019.

[3] Yunkung Chung and Gary W. Fischer. "A conceptual structure and issues for an object-oriented bill of materials (BOM) data model". In: *Computers Industrial Engineering* 26.2 (1994), pp. 321–339. ISSN: 0360-8352. DOI: https://doi.org/10.1016/0360-8352(94)90065-5.

[4] Matteo Cinelli et al. "A network perspective for the analysis of bill of material". In: *Procedia CIRP* 88 (2020), pp. 19–24. ISSN: 22128271. DOI: 10.1016/j.procir.2020.05.004.

[5] The United State Departement of Commerce. *The Minimum Elements For a Software Bill of Materials (SBOM)*. July 12, 2021. URL: https://www.ntia.doc.gov/files/ntia/publications/sbom_minimum_elements_report.pdf.

[6] European Commission. *Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on horizontal cybersecurity requirements for products with digital elements and amending Regulation (EU) 2019/1020*. Sept. 15, 2022. URL: https://eur-lex.europa.eu/resource.html?uri=cellar:864f472b-34e9-11ed-9c68-01aa75ed71a1.0001.02/DOC_1&format=PDF.

[7] D. CONTE et al. "THIRTY YEARS OF GRAPH MATCHING IN PATTERN RECOGNITION". In: *International Journal of Pattern Recognition and Artificial Intelligence* 18.03 (2004), pp. 265–298. DOI: 10.1142/S0218001404003228.

[8] Cybersecurity and Infrastructure Security Agency. *A Hardware Bill of Materials (HBOM) Framework for Supply Chain Risk Management*. Sept. 2023. URL: https://www.cisa.gov/sites/default/files/2023-09/A%20Hardware%20Bill%20of%20Materials%20Framework%20for%20Supply%20Chain%20Risk%20Management%20%28508%29.pdf.

[9] Cybersecurity and Infrastructure Security Agency. *Meltdown and Spectre Side-Channel Vulnerability Guidance*. May 1, 2028. URL: https://www.cisa.gov/news-events/alerts/2018/01/04/meltdown-and-spectre-side-channel-vulnerability-guidance.

[10] Akamai Security Intelligence Group. *XZ Utils Backdoor — Everything You Need to Know, and What You Can Do*. Apr. 1, 2024. URL: https://www.akamai.com/

blog/security-research/critical-linux-backdoor-xz-utils-discovered-what-to-know.

[11] John Alexis Guerra-Gómez et al. "TreeVersity: Interactive Visualizations for Comparing Hierarchical Data Sets". In: *Transportation Research Record: Journal of the Transportation Research Board* 2392.1 (Jan. 2013), pp. 48–58. ISSN: 0361-1981, 2169-4052. DOI: 10.3141/2392-06. URL: http://journals.sagepub.com/doi/10.3141/2392-06.

[12] The White House. *Executive Order on Improving the Nation's Cybersecurity*. The White House. May 12, 2021. URL: https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/.

[13] Xiaodu Hu et al. "Graph Model Based Bill of Material Structure for Coupling Product Development and Production Planning". In: *Intelligent and Transformative Production in Pandemic Times*. Cham: Springer International Publishing, 2023, pp. 593–605. DOI: 10.1007/978-3-031-18641-7_55.

[14] Rebecca Jones and Lucas Tate. "Visualizing Comparisons of Bill of Materials". In: *2023 IEEE Symposium on Visualization for Cyber Security (VizSec)*. 2023, pp. 12–16. DOI: 10.1109/VizSec60606.2023.00008.

[15] M. Kashkoush and H. ElMaraghy. "Product Design Retrieval by Matching Bills of Materials". In: *Journal of Mechanical Design* 136.1 (Jan. 1, 2014), p. 011002. ISSN: 1050-0472, 1528-9001. DOI: 10.1115/1.4025489.

[16] Mohamed Kashkoush and Hoda ElMaraghy. "Matching Bills of Materials Using Tree Reconciliation". In: *Procedia CIRP* 7 (2013), pp. 169–174. ISSN: 22128271. DOI: 10.1016/j.procir.2013.05.029.

[17] Mohamed Kashkoush and Hoda ElMaraghy. "Product family formation by matching Bill-of-Materials trees". In: *CIRP Journal of Manufacturing Science and Technology* 12 (Jan. 2016), pp. 1–13. ISSN: 17555817. DOI: 10.1016/j.cirpj.2015.09.004.

[18] Edward Kost. *Log4Shell: The Log4j Vulnerability Emergency Clearly Explained*. June 20, 2023. URL: https://www.upguard.com/blog/apache-log4j-vulnerability.

[19] S. Melnik, H. Garcia-Molina, and E. Rahm. "Similarity flooding: a versatile graph matching algorithm and its application to schema matching". In: *Proceedings 18th International Conference on Data Engineering*. 2002, pp. 117–128. DOI: 10.1109/ICDE.2002.994702.

[20] G. Nandakumar. "The design of a Bills of Material Processor using a relational data base". In: *Computers in Industry* 6.1 (1985), pp. 15–21. ISSN: 0166-3615. DOI: https://doi.org/10.1016/0166-3615(85)90066-1.

[21] Ganesan Nandakumar. "Bills of material processing with a SQL database". In: *Computers Industrial Engineering* 18.4 (1990), pp. 471–483. ISSN: 0360-8352. DOI: https://doi.org/10.1016/0360-8352(90)90005-7.

[22] OWASP Foundation. *OWASP CycloneDX Software Bill of Materials (SBOM) Standard*. URL: https://cyclonedx.org/.

[23] C.J. Romanowski and R. Nagi. "On Comparing Bills of Materials: A Similarity/ Distance Measure for Unordered Trees". In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 35.2 (Mar. 2005), pp. 249–260. ISSN: 1083-4427. DOI: 10.1109/TSMCA.2005.843395.

[24] Carol J. Romanowski and Rakesh Nagi. "A data mining and graph theoretic approach to building generic bills of materials". In: 2002. URL: https://api.semanticscholar.org/CorpusID:17069084.

[25] P.S. Rusk. "The role of the bill of material in manufacturing systems". In: *Engineering Costs and Production Economics* 19.1 (May 1990), pp. 205–211. ISSN: 0167188X. DOI: 10.1016/0167-188X(90)90044-I.

[26] Michael Schmidt et al. "Graph-based similarity analysis of BOM data to identify unnecessary inner product variance." In: 21st International Conference on Engineering Design (ICED17). Vol. 1. Vancouver, Canada, Aug. 2017.

[27] Han M. Shih. "Product structure (BOM)-based product similarity measures using orthogonal procrustes approach". In: *Computers & Industrial Engineering* 61.3 (Oct. 2011), pp. 608–628. ISSN: 03608352. DOI: 10.1016/j.cie.2011.04.016.

[28] Trevor Stalnaker et al. "BOMs Away! Inside the Minds of Stakeholders: A Comprehensive Study of Bills of Materials for Software Systems". In: ICSE '24: IEEE/ACM 46th International Conference on Software Engineering. Feb. 6, 2024, pp. 1–13. DOI: 10.1145/3597503.3623347.

[29] Dina Temple-Raston. *A 'Worst Nightmare' Cyberattack: The Untold Story Of The SolarWinds Hack*. https://www.npr.org/2021/04/16/985439655/a-worst-nightmare-cyberattack-the-untold-story-of-the-solarwinds-hack. Last accessed 2023-06-30. Apr. 16, 2021. URL: https://www.npr.org/2021/04/16/985439655/a-worst-nightmare-cyberattack-the-untold-story-of-the-solarwinds-hack.

[30] The Linux Foundation Projects. *International Open Standard (ISO/IEC 5962:2021) - Software Package Data Exchange (SPDX)*. URL: https://spdx.dev/.

[31] Yao Wang, Wei Guo Wang, and Song Mao. "A New Type of BOM Model and its Application". In: *Applied Mechanics and Materials* 347-350 (Aug. 2013), pp. 1234–1238. ISSN: 1662-7482. DOI: 10.4028/www.scientific.net/AMM.347-350.1234.

[32] Boming Xia et al. "An Empirical Study on Software Bill of Materials: Where We Stand and the Road Ahead". In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). IEEE, May 2023, pp. 2630–2642. DOI: 10.1109/ICSE48619.2023.00219.

Potential and Problems in Detecting Privacy Risks in Source Code Using Large Language Models

Ambarish Gurjar
Department of Intelligent Systems Engineering
Indiana University, Bloomington, USA
agurjar@iu.edu

Xinyao Ma
Department of Informatics
Indiana University, Bloomington, USA
maxiny@iu.edu

Anesu Chaora
Department of Informatics
Indiana University, Bloomington, USA
achaora@iu.edu

Vinayak Kumar
Department of Computer Science
Indiana University, Bloomington, USA
kvinayak@iu.edu

Abhijeet Muralidharan
Department of Computer Science
Indiana University, Bloomington, USA
abhmura@iu.edu

L. Jean Camp
Department of Informatics
Indiana University, Bloomington, USA
ljcamp@indiana.edu

***Abstract*—Developers regularly rely on third-party code obtained from collaborative software platforms and code repositories. Despite the benefits, this introduces potential security and privacy risks. The Software Bill of Materials requires a manifest of all code components; however, understanding the possible risks from the inclusion of the components is a deeper challenge. Both privacy and security risks may be contextual, depending on operational environments. In order to evaluate the potential for large language models to identify potential risk, we focus on privacy risks as these are contextual and nuanced. We focus on risks from the inclusion of data types that have been adjudicated as personally identifiable information (PII). We explore the efficacy of Large Language Models (LLMs) in meeting this challenge by testing their performance in the classification of code snippets as processing sensitive data types, or not. We report on the feasibility of using a range of zero-shot and few-shot approaches to automate the detection of source code that includes processing of sensitive data. For this purpose, we developed a labeled corpus of code snippets from GitHub. We report on the accuracy of fine-tuned LLMs (GraphCodeBERT, LongCoder, Mistral, LLama2, and CodeLLama) and commercial prompt-based LLMs (OpenAI's ChatGPT and Google's Gemini). We conclude that language models have the potential to identify privacy risks; however, such models must be trained to meet specific regulatory requirements. Moreover, we find that commercial LLMs may need further development before they are suitable for general use in identifying privacy risks in code.**

***Index Terms*—secure by design, data minimization, privacy, risk management, AI, SBOM, compliance, code quality, LLM**

I. Introduction

Developers frequently rely on code reuse, leveraging collaborative code management systems such as GitHub, SourceForge, or Stack Overflow. However, the advantages of code reuse which add speed and ease to the task of coding also create the disadvantage of incurred security and privacy risks [1]. The inclusion of code that references sensitive data can expose organizations to compliance risk or legal liability. This underscores the pressing need to ensure that organizations have a clear grasp of what data they're compiling, not only for purposes of respecting user's rights but also for avoiding operational risks.

The definition of what constitutes privacy risk varies depending on the context, making it challenging for developers to reach consensus on which data compilations and uses to consider privacy sensitive. Moreover, different jurisdictions and cultures may have distinct guidelines for quantifying the sensitivity of information. Due to the contested and complex nature of privacy, here we focus on potentially sensitive data types from explicit legal categorizations and enumerations. The legal definition of sensitivity of data types also varies across contexts of use; for example, healthcare provision and healthcare research have different standards. Consequently, organizations face difficulties in safeguarding against compliance risks. Even large organizations with extensive compliance processes, including Google, Amazon, H&M, British Airways, and Marriott, have - in the past few years - been fined substantial amounts in excess of 20 million pounds for violating the General Data Privacy Regulation [2] . Identifying indicators of sensitive data, such as variable names or the code in which a variable is embedded, could enable developers to avoid incidental inclusion of code with potential privacy and compliance risks. It could also inform operator compliance decisions when code goes between jurisdictions (e.g., US/EU or across state borders), or across business functions (e.g., patient care to public health dataset). The potential for one-shot and few-shot learning offers the possibility that organizations can train for their own unique combination of code and context.

The overall goal of this work is to explore the possibility of automating identification of sensitive data in code. Traditional methods, like string matching or regular expressions search, have been routinely employed to identify the presence of named variables in code [3]. This study aims to document the efficacy of more sophisticated approaches in capturing potentially sensitive data types. The insights provided by the comparisons of the accuracy of the models and failure modes can inform software development choices and risk management.

Through our study, we seek to answer the following research questions:

RQ1: Is it possible to identify potential privacy-related data risks, arising from third-party code, by applying a Natural Language Processing (NLP) based approach to search for for indicators of sensitive data in code?

RQ2: How well do different artificial language models compare in terms of their ability to identify variables that correspond with the handling of sensitive data?

RQ3: Are there identifiable patterns, or types of failures characteristic of different models?

II. Related Work and Motivation

In this section, we begin with a discussion of how we define sensitive data types. Subsection II-A describes the basis for the labelling of different data types as sensitive. The identification and labelling of data as sensitive is grounded in GDPR and CCPA. Our goal is to both provide a motivate for our selections of data types and to describe our method in enough detail, to enable the reproduction of this research with any enumerated list of sensitive data types.

A. Defining Sensitive Data

In this Subsection II-A, we define what constitutes sensitive data in this experiment. We begin with an overview of two privacy regulations that have well-defined categories for personally identifiable information, i.e., data that is considered personal or protected. Given the variation in definitions across different jurisdictions, we refer to these data types as *sensitive* or *privacy sensitive*.

The General Data Protection Regulation (GDPR) of the European Union and the California Consumer Privacy Act (CCPA) of the State of California are used to define what data are sensitive in our analysis. These two privacy regulations have significant requirements constraining the compilation and processing of Personal Identifiable Information (PII). These regulations are significant factor in software production as California produces roughly 5% of the total global economy and the EU accounting for between 15% and 20%. [4]. Both emphasize foundational principles such as data minimization, which calls for the least amount of data collection necessary, and purpose limitation, to ensure data is used solely for the reasons it was collected (GDPR Article 5 [2]; CCPA Section 1798.100(d)(1) [5]). More importantly, for our purposes, both also clearly enumerate what data is considered sensitive. This method can be reproduced by smaller jurisdictions or industry segments, as long as there are clearly enumerated data categories. Concurrently, the Software Bill of Materials (SBOM) initiative has emerged as a complementary effort aimed at transparency in the software supply chain. As SBOM matures, it has expanded beyond licensing and vulnerability tracking to considerations of development practices. compliance. and privacy [6].

The GDPR was grounded in the 1995 Data Protection Directive, which was an early attempt to identify privacy-sensitive data types and data processing. The GDPR was adopted in 2016 to define individiual data rights. It was intended to provide a unified, consistent, comprehensive framework to safeguard the rights and freedoms of individuals within the European Union. The GDPR defines the responsibilities of organizations compiling, processing, storing, and managing personal data. CCPA and GDPR compliance require that organizations confirm that personal data processed through code dependencies aligns with the regulations and the organizations' stated privacy policies. As a result, organizations are obligated to determine the potential impact of code dependencies on user data privacy and to institute measures that protect personal data throughout software development and supply chain processes. Recognizing that a particular data type covered by the GDPR or CCPA is in a library, utility, or other dependency remains an open challenge.

The CCPA defined statutory principles of data minimization and purpose limitation (Section 1798.100(d)(1) [5]). As with GDPR, it identifies a range of data types. The statute endows consumers with multiple rights, including the right to be informed, to access, and to delete their personal information. In terms of code analysis, aligning with the CCPA requires organizations to evaluate data compilation and processing in code, including dependencies, where the code is used by Californians. The Act has an expansive definition of sensitive data types and defines consumer rights over those types.

There are additional regulations based on specific contexts, notably for health, biometrics, education, employment, children's data, and regulations in other legal contexts [7], [8]. This work contributes to understanding the potential, and the risks, of using few-shot and zero-shot techniques described in this work to identify sensitive data types in code. Few-shot training in the context of this paper refers to training samples on a very small dataset. Zero-shot learning refers to prompting LLMs like ChatGPT to leverage their pre-existing knowledge without providing them with any labeled training samples.

III. Methodology

Our research methodology was structured into five stages. First, we conducted a literature survey, as summarized in related work (Section II). We then compiled a data corpus using the GitHub search API for code. We validated all samples through human review of both the code and its implicit or documented use context. With the data validated, we then proceeded to the third stage of fine-tuning large language models. Initially, our evaluation process focused on CodeBERT and GraphCodeBERT [9], [10]. However, the results were not impressive. With the open-sourcing of LLaMA2 by Facebook and the release of Mistral 7B we were able to expand our work [11]–[13]. Our final results include evaluations of GraphCodeBERT, LongCoder, Mistral, LLaMA2, and CodeLLaMA. LongCoder performed particularly well, validating its potential across diverse coding tasks. Results from salesforces T5,T5+ and CodeBERT have been reported [9], [14], [15].

Due to their robust performance and widespread adoption, we included ChatGPT and Gemini Advanced as the fourth stage of our method. This enabled us to evaluate their advanced zero-shot capabilities and benchmark their effectiveness in our specific use case. Lastly in the final stage, we further analyze the responses and the data that we compiled during these

stages to answer the research questions that we outline in the Introduction/

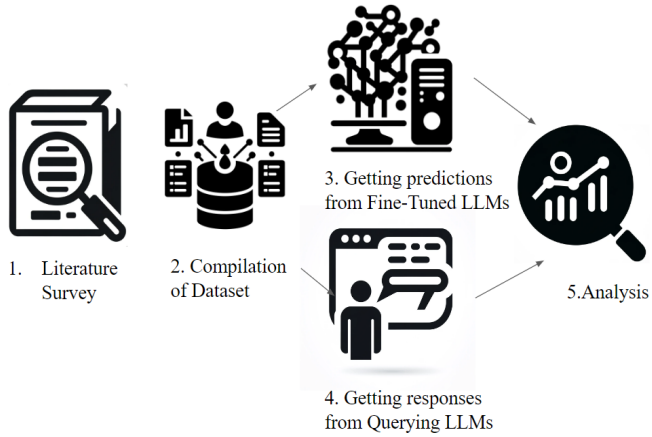


Fig. 1. Overview of methodology

A. Data Corpus

To create a corpus of code containing sensitive information, we began by selecting a subset of categories that are identified in both the California Consumer Protection Act (CCPA) and the European Union's General Data Protection Regulation (GDPR) as sensitive. The ten data information categories we chose were as follows: These were 1. Unique Device ID, 2. Account/Individual Identifier, 3. Demographics, 4. Internet traffic, 5. Commercial/Financial information, 6. Biometrics, 7. Multimedia data, 8. Employment information, 9. Educational information, and 10. Location information.

We leveraged common expressions (e.g., DoB) and code functionality (e.g., payment) to search for code candidates in repositories, using GitHub's code search API. We list the search terms in the appendix. We selected code from commercial public repositories and other popular open-source code. While no small sample can be representative of every production environment, our goal is to explore the potential of zero-shot and few-shot learning for different contexts so we sought widely used code.

To create a control group, we included an equal number of including a large number of code snippets that were not considered sensitive. These code snippets were selected from popular code, and include diverse functionality. The inclusion of non-sensitive code snippets enabled us to assess the specificity and accuracy of our models in distinguishing between sensitive and non-sensitive code.

To ensure the contextual sensitivity of the identified variables these were manually reviewed. The research team (including a professor, a post-doctoral scholar, three interdisciplinary doctoral candidates, one computer science graduate researcher, and one undergraduate computer science researcher). Each individually reviewed the code snippets and determined whether the identified variables were indeed sensitive within the given code contexts. We began with only those samples where the Fliess Kappa score of agreement amongst researchers was equal to 1. Any differences in opinion were identified and discussed in real-time meetings until there was consensus. This collaborative knowledge construction method [16], [17] was a verification stage to add a layer of accuracy to the dataset. Further validation serendipitously occurred when

Category	Repos	Lines	Variable	Code Snippets
Unique Device ID	6	958	27	25
Individual Identifier	9	1643	29	28
Demographics	6	179	20	21
Internet Traffic	7	2549	46	57
Financial Information	8	1068	44	50
Biometrics	5	490	32	38
Multimedia Data	4	1388	20	31
Employment	7	943	29	31
Location	8	1472	50	58
Education	2	102	7	6

TABLE I
SUMMARY OF SEARCHING CODE CORPUS

For the scope of this study, we collected Python code samples to create our dataset as it is widely used and easy to understand for most of the human participants. We searched for identifiable variables in the selected categories, based on CCPA, GDPR, Privado rules, and previous literature. GitHub, as the largest open-source code-sharing platform, offers an extensive repository of publicly accessible code contributed by a diverse community of developers. Its vast collection of code makes it an ideal resource for acquiring a large-scale public dataset suitable for research and analysis in various domains. As we focused on code with potential privacy implications, we searched all Python GitHub repositories for a small set of data privacy-related keywords and data types. Searching for these keywords resulted in a corpus of 53 repositories, 30 keywords, and 86 variables with their related variables and code expressions in total, see Table I for details. We used different keywords for each category domain to search the repositories on the GitHub platform. We also extract the nearby variables and code segments that have the value transmission with the picked variable. The final dataset consisted of 57 code fragments containing sensitive data and 57 fragments without, each roughly 100 lines. Preprocessing was applied to the collected code fragments, involving the removal of comments and non-essential characters.

B. Fine-Tuning Large Language Models for Identification of Privacy risks

Fine-tuned models have exhibited a strong proficiency in few-shot learning, where they can quickly generalize to new tasks by learning from a limited number of examples like our dataset. This contrasts with zero-shot learning scenarios where models apply learned patterns without additional training data. Few-shot learning exploits the models' pre-trained knowledge base, enabling them to identify underlying patterns in sparse data and effectively adapt to novel programming tasks or languages with remarkable efficiency using relatively smaller training samples [18]–[20].

We began our evaluation process with CodeBERT and GraphCodeBERT (described above) [9], [10]. To further re-

fine these models, fine-tuning with additional neural network layers, often termed "task-specific heads" or "classification layers," as employed by [21]. These layers are crucial for adapting the model's expansive knowledge to specific tasks, allowing it to detect subtle, task-dependent patterns. With just a handful of examples, models like GraphCodeBERT, LLAMA, LongCoder, CodeLlama, and Mistral can be fine-tuned to significantly enhance their proficiency in making precise code-related predictions or classifications. Through few-shot learning and strategic fine-tuning, these models can become exceptionally adept at interpreting and analyzing code, even in areas with little to no prior exposure.

Preprocessed code fragments underwent tokenization, resulting in sequences of tokens compatible with the respective models. We then processed each tokenized code snippet through all the models. These models transformed the code tokens into high-dimensional vectors, known as embeddings, which effectively encapsulate the syntactic and semantic nuances of the code. To streamline the computational process and manage the high dimensionality of the data, we computed the mean of the tokens for each snippet. This averaging technique yielded a single vector of fixed dimensions per code fragment, which succinctly represented the original richly exhibited data [22]. These condensed vector representations were then used as input in the subsequent stages of sensitivity classification.

In our research, we initially implemented a fully connected neural network (FCN) model. Our preliminary architecture was composed of three layers, containing 64 and 32 hidden units, respectively, with the Rectified Linear Unit (ReLU) serving as the activation function [23]. The neural network predicts the probabilities that the input sample is sensitive or not sensitive. The model was designed and executed using the PyTorch deep learning framework and was optimized using the Adam optimizer [24]. In our model, we used binary cross-entropy as the loss function, which is a standard approach for binary classification tasks [25], [26]. However, we transitioned to a more robust FCN model, consisting of two layers each with 512 nodes. This enhanced network was capable of learning and predicting more intricate patterns in our data, thereby improving the accuracy of sensitivity discernment for variables.

The hyperparameter choices for the models were determined by iterative experimentation and tuning to achieve satisfactory performance. Our initial methodology utilized an 80:20 ratio for training and testing to evaluate the model's performance. To enhance this evaluation, we adopted a k-fold cross-validation method, with k designated as 5 [27]. This method divides the dataset into k equally sized portions, known as 'folds.' In a series of iterations, each fold is used once as the test set while the remaining folds (k-1 in number) serve as the training set. This approach does not just yield an averaged value for metrics such as accuracy, precision, recall, and F1-score, but also captures the variability within the data. It ensures that our performance metrics are not skewed by any specific partition of the dataset. The average values reported from the iterations of the 5-fold cross-validation present a more thorough and reliable measure of the model's performance, having been stringently evaluated across various data subsets.

C. Prompting Large Language Models

In this phase of our methodology, we harnessed the prompt-based classification capabilities of advanced language models, namely ChatGPT 3.5, ChatGPT 4, and Google's Gemini Advanced to assess the privacy sensitivity of the collected code fragments. Zero-shot learning allows these models to perform downstream classifications they were not specifically trained to recognize, by drawing on their extensive pre-training on diverse data. This capability is critical for identifying sensitive information in code (as per GDPR and CCPA guidelines) without the models having previous exposure to such classification tasks [28].

Prior to classification, we assessed the models' knowledge of GDPR and CCPA regulations to ensure their capability to make informed judgments about data sensitivity by asking questions about these regulations. Then, to facilitate classification, we crafted and refined the prompts through an iterative query engineering process. The final prompt directed the models to examine four code fragments, each separated by a specified delimiter symbol in the input, and to determine the presence of sensitive data as per the mentioned guidelines without providing explanations or summaries of the code. We structured the prompt to maximize generalizability and accommodate the language models' interface limitations, asking for a straightforward response indicating which fragments contained sensitive information. Previous work has illustrated that LLMs may validate questions, so our queries were strictly neutral.

The initial query we presented to the models was, "Given below are 4 fragments of code. Each of them is separated by a delimiter. Considering the first code as '1' and the fourth as '4,' identify if the fragments contain sensitive data according to GDPR or CCPA guidelines, indicating your response as 'code fragment number: Sensitive or Not Sensitive'." This query-based approach ensured consistency and uniformity in the sensitivity classification process across the entire dataset. Each code fragment was presented to ChatGPT 3.5, ChatGPT 4, and Gemini models using the same prompt, enabling standardized classification based on their understanding of GDPR and CCPA guidelines. While GPT 3.5 initially showed limitations in performance, its subsequent paid version demonstrated significantly enhanced higher-order reasoning abilities. This iterative refinement of our query approach was informed by comparative studies of large language models in tasks such as sentiment analysis and prompt engineering surveys [29], [30].

In this study, the specific query was carefully formulated to address the challenges posed by the generative nature of language models. Language models have a tendency to provide detailed explanations or summaries of code, which may introduce bias or compromise the objectivity of the analysis. This query maintains a clear restraint over the query parameters by explicitly instructing the language model to refrain from providing explanations or summaries. This ensures that the

language model's responses focus solely on assessing the presence of sensitive data in the code fragments. Furthermore, by referencing the GDPR and CCPA guidelines, the query aligns with the legal frameworks for data privacy, enabling a targeted evaluation of code sensitivity. Overall, this query was iteratively optimized to mitigate the risks of information bias, maintain objectivity, and ensure that the analysis is conducted within the scope of the research objectives.

We analyzed the responses generated by ChatGPT 3.5, ChatGPT 4, and Gemini to determine the sensitivity classification of the code fragments. Accuracy, consistency and any challenges associated with the models' inclination to provide detailed explanations rather than straightforward sensitivity labels were considered.

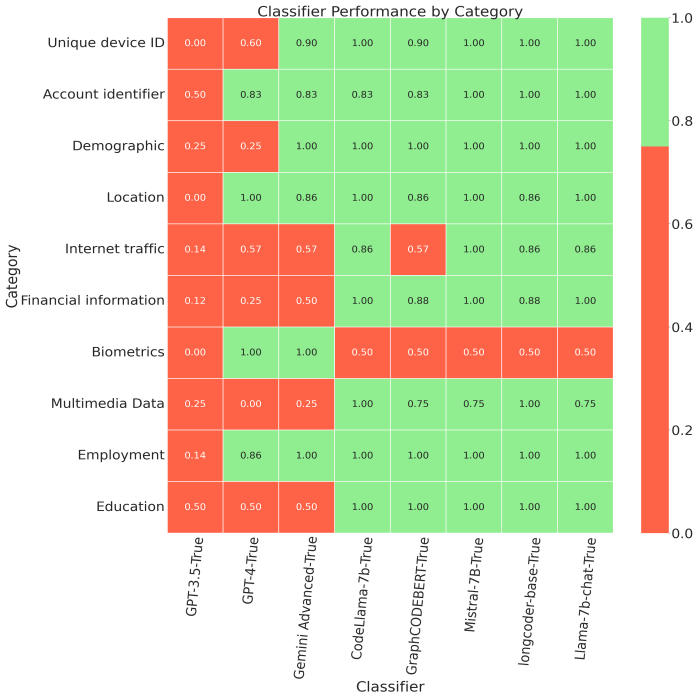


Fig. 2. Heatmap showing how various categories of privacy risks were misclassified. Red squares show poor performance of the language model in the category as the threshold for a good score is 75% accuracy and above

IV. Results

The performance of various models was evaluated using metrics such as Accuracy, F1 score, Precision, and Recall. The results are summarized in Table II. Overall, the results indicate that Mistral 7B and CodeLlama 7B are highly effective at classification of code as containing potentially sensitive data processing. The overall accuracy, precision, and recall of these two LLMs were high, with other models offering varying trade-offs in their levels of performance. Given tolerances for different types of errors, other models may be suitable for different applications.

Mistral 7B outperformed all other models, achieving the highest scores across all metrics with an accuracy of 92.09%. CodeLlama 7B followed closely with an accuracy of 91.26%,

TABLE II
Comparison of Model Performance Metrics of finetuned

Model	Accuracy	F1 score	Precision	Recall
Mistral 7B	**92.09%**	**92.04%**	**92.57%**	**92.04%**
CodeLlama 7B	91.26%	91.12%	91.97%	91.12%
Llama 2	90.39%	90.29%	91.62%	90.30%
Long Coderbase	88.61%	88.06%	91.01%	88.48%
GraphCodeBERT	80.71%	80.60%	81.31%	80.51%
CodeT5-220m	78.93%	78.48%	80.47%	78.86%
CodeT5plus-770m	81.54%	81.43%	82.39%	81.66%
CodeBERT	72.80%	72.26%	74.61%	72.87%

and Llama 2 also performed well with an accuracy of 90.39%. Long Coderbase and GraphCodeBERT displayed moderate performance, with accuracies of 88.61% and 80.71%, respectively. The Salesforce models, CodeT5-220m and CodeT5p-770m, showed competitive results, particularly the latter with an accuracy of 81.54%. CodeBERT, while achieving the lowest performance among the evaluated models, still provided useful insights with an accuracy of 72.80%. LLM modes, particularly general chat models, did not perform as well with ChatGPT 3.5, Chat GPT 4.0, and Gemini achieving accuracy of 50%, 73%, and 58% respectively.

V. Discussion

Our results in Table II report a high level of accuracy for fine-tuned models trained on a small corpus. The accuracy of popular chat-based models remained low, regardless of the query. One of our initial limitations was a concern that the corpus of data samples was too small. In contrast, we hypothesized that NLP models that have already been pretrained on extensive datasets could transfer learned knowledge effectively, even when applied to smaller, domain-specific datasets. The accuracy of our results aligns with recent findings in the field, where models trained on less in-distribution data have shown better out-of-distribution performance. These results mitigate concerns about the capabilities of pretrained models [31]. However, the analysis of the results as reflected in the confusion matrices, revealed a concerning trend of high false negatives. Inaccurate classification of sensitive information of Language Models (LLMs) spanned across multiple categories: Unique Device ID, Account/Individual Identifier, Demographics, Internet Traffic, Financial Information, Biometrics, Multimedia Data, Employment Information, Location Information, and Educational Information.

In the case of UniqueID, all the LLMs classified three of sixteen as not sensitive regardless of the fact that such unique identifiers are central to differentiating identifiable from nonidentifiable (and thus less sensitive) data.

We identify the limitations of prominent large language models including ChatGPT and Gemini. These models displayed systematic patterns of failure in classifying code snippets related to both demographic data and geofencing operations. In particular, none of the code snippets which indicated geofencing processes were not categorized as privacy-sensitive by these LLMs. This is not only notable because of extensive research that emphasizes the high privacy risks associated

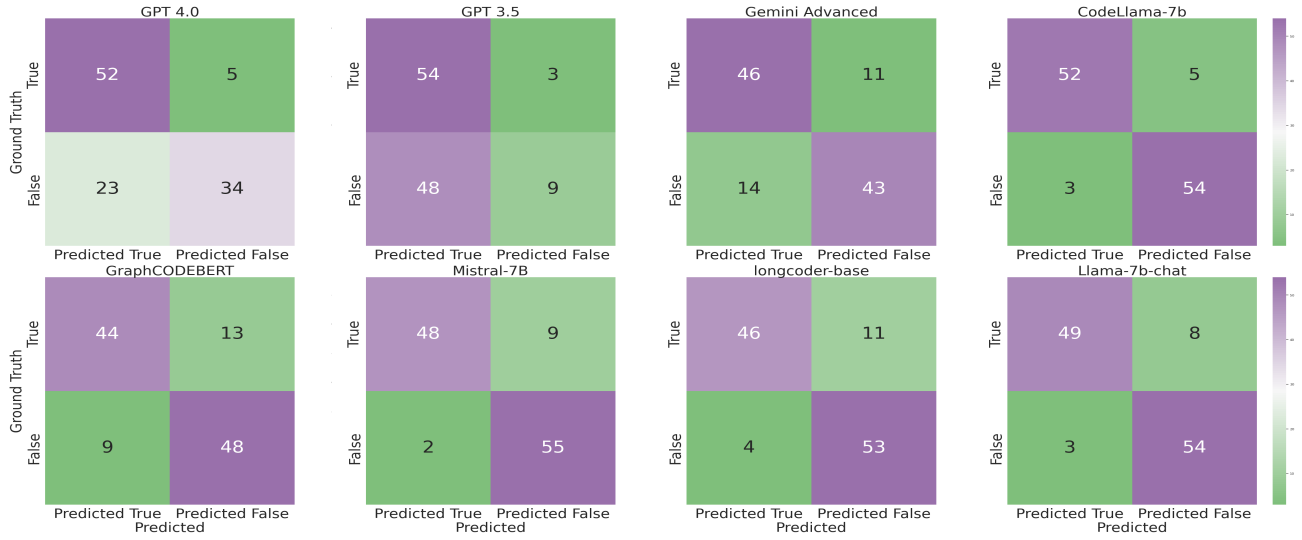


Fig. 3. Confusion Matrices showing the false positive, false negative, true positive, and true negatives of each model. These confusion matrices provide the raw percentages used to calculate precision and accuracy, illustrating the need to explore beyond the report of summary statistics.

with location data (e.g., [32]) with even four spatiotemporal points adequate to uniquely identify an individual within a large dataset, thereby escalating privacy risks [33]. Recall that location data is defined as sensitive.

The processing of demographic data was also frequently misclassified. Three of four instances were subject to incorrect classification. This high rate of failure may be attributed to the small sample size, as discussed in the limitations section of this paper. Our analysis revealed that certain code snippets containing this data type were erroneously classified as non-sensitive by both ChatGPT and Bard. This trend is more notable in light of extensive research documenting the risks of mishandling such sensitive demographic information. Previous studies have indicated that even simple demographic characteristics can uniquely identify individuals within large populations, thus exacerbating privacy concerns [34], [35]. Moreover, careless handling of demographic data not only jeopardizes individual privacy but also has broader societal implications, such as exposing communities to discrimination, stereotyping, and targeted surveillance [36], [37]. Eckersley emphasized that even seemingly innocuous information could uniquely identify users, illustrating the covert ways in which privacy can be compromised [38]. Further adding complexity to this issue are the risks associated with microtargeted ads on platforms such as Facebook, which have been shown to exploit demographic data [39]. Korolova specifically addressed the potential for privacy violations through targeted advertising, shedding light on how such platforms could inadvertently facilitate unauthorized data sharing and re-identification [40]. Given these considerations, our findings underscore the necessity for improved measures to ensure that demographic data are managed as sensitive data, not systematically identified as non-sensitive.

Other categories where all the LLMs misclassified included financial and biometric data. These are contextual in the sense that financial data that was used for payment (other than fraud) are used with consent. Similarly, biometrics may be recognized by LLMs as being components of authentication. Regardless of the presumption of use, financial and biometric data can be sensitive. The systematic misclassification of these categories in comparison with other approaches illustrates the applicability of criticisms of unexamined adoption of LLMs.

These reflect the classic critiques of LLMs listed in [41]. An essential argument of that paper is that the models can be too large to be accurate. Our results can be read as reifying this. The second argument is that social change cannot easily be embedded in a model; consequently, LLMs may be inherently unresponsive to social change. This may apply to the case of privacy, for example, consider network information such as IP addresses. There was a long-standing argument in the first two decades of the century if IP addresses were sensitive, and the differences between jurisdictions remain [42].

The third critique is that LLMs are majoritarian and often inadequately respond to less frequent events (or populations). There is a general problem in privacy (and security) that most code does not have privacy concerns, resulting in biased datasets from the real world. The confusion matrices illustrate this, in the prevalence of false negatives.

Conversely, there were instances where these models displayed an overly cautious approach, erring on the side of over-classification of non-sensitive content as sensitive. Gemini, in particular, demonstrated this flaw when it incorrectly classified an ML classification and cropping algorithm as identifying sensitive financial data in receipts. Upon closer inspection, it was evident that the algorithm in question was solely focused on generic image classification and was not associated with any receipt dataset. Such false positives underline the importance of refining model criteria for sensitivity assessments and ensuring that their determinations are grounded in accurate contextual understanding.

An obvious solution to this would be to simply ask developers to hand label code as sensitive or not sensitive. Yet, previous research on that question illustrated that individual developers disagree about sensitive information, and often do not know that entire categories of data are sensitive. A 2018 survey of 36 developers found that many had never heard of a privacy impact assessment (47.2%), fair information practices (38.9%), nor privacy by design (36.1%) [43]. A later evaluation of 99 developers showed that developers with more experience were not better at identifying code as processing sensitive data [44]. Agreement comparisons between the more expert human coders (0.2033) was slightly higher than the agreement between the models evaluated here (0.2028). (Scores greater than 0.2 indicate consensus.) Human experts categorized sensitive data as not sensitive more often than LLMs and more often than non-experts. Non-expert consensus was measured as 0.2171. The results from LLMs are closer to those of non-experts, perhaps as a result of their training corpus.

VI. Limitations and Future Work

We report that LLMs exhibit systematic failures in identifying sensitive information and thus privacy risks in code. We identified patterns in these errors. However, our data set was too small to determine if these were systematic or random. Some of the errors may indicate an embedded assumption of legitimate use, such as payment information. In other cases, there is no clear reason. Consider that, aside from targeted advertising, the business case for demographic data is limited while the privacy implications are substantial. Some data types, like the date of birth, are widely classified as sensitive personal information and have limited commercial use.

The failures by the models to correctly categorize some code snippets as sensitive underscore a pivotal challenge: despite their ability to process vast quantities of information and generate coherent responses, LLMs are limited in their ability to identify nuanced concerns surrounding data privacy. This gap highlights the urgent need to avoid the unexamined uniform adoption of LLM for the identification of nuanced categorizations. A focus on the recognition and classification of potential privacy threats is needed, especially in light of the substantial risks associated with the mishandling of demographic and geolocation data. Our results support integrating robust location and demographic privacy measures into the ethical guidelines and coding practices that govern LLMs [45].

In future work, we will expand our corpus. We seek a partner to explore how that organizations might employ a Grammarly-like framework to signals parts of code that might violate privacy compliance requirements. We are particularly interesting in working with organizations that share data across operations and research functions.

VII. CONCLUSIONS

There is a need to detect the inclusion of Personally Identifiable Information (PII) in source code. We have reviewed past research focused on identifying privacy risks in code and highlighted the existing gaps and challenges. We report on the accuracy of emerging technologies, such as Generative AI and Large Language Models (LLMs), in offering solutions for automated privacy risk detection in source code. Specifically, we explored the potential for one-shot and few-shot learning for the identification of code that processes sensitive data using a custom-labeled corpus. LLMs varied in their accuracy in the correct classification of sensitive code but were higher than previous research using human evaluators, including previous research on software developers.

Our results underscore the challenges in automated code sensitivity detection and emphasize the need for continuous model refinement. Our results also illustrate that while advancements in language models have provided us with powerful tools for code analysis, understanding their limitations and biases is crucial in deploying them in real-world applications This is arguably especially true for tasks with significant legal and ethical implications such as compliance with privacy laws.

The larger question is if it would be effective for organizations to train LLMs for their own unique combination of code and context. While this justifies the use of a small, carefully labeled dataset, we also acknowledge the size of the dataset is a limitation. Our sample dataset can be augmented with more detailed analysis and is available on the author's Github.[1]

Part of the significance of our work lies in the use of one-shot and few-shot training methodologies to identify privacy sensitivity. A single organization may have different privacy constraints for different units or processes so a single model might not be able to capture the required nuances for all business purposes. Similarly, individuals have differing rights based on their jurisdictions, as well as differing individual personal preferences. Our contributions are to the challenges of selecting products and managing privacy risks based on these constraints. The comparison of the categories and patterns of failures can contribute to the identification of sensitive information in code, and provide insights into the effectiveness of different methods of analysis.

Acknowledgment

This research was supported in part by CTIA and the Comcast Innovation Fund. We would like to acknowledge the contributions of Prof. Tatiana Ringenberg. We acknowledge support from the US Department of Defense [Contract No. W52P1J2093009]. This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Number 17STQAC00001-07-00." The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of all of these organizations or Indiana University.

References

[1] F. Fischer, K. Böttinger, H. Xiao, C. Stransky, Y. Acar, M. Backes, and S. Fahl, "Stack overflow considered harmful? the impact of copypaste on android application security," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 121–136.

[1] https://github.com/ambarishgurjar/PrivacySensitivityDataset

[2] European Parliament and Council of the European Union, "Regulation (EU) 2016/679 of the European Parliament and of the Council (General Data Protection Regulation (GDPR))." [Online]. Available: https://data.europa.eu/eli/reg/2016/679/oj

[3] Privado. [Online]. Available: https://github.com/Privado-Inc/privado

[4] J. Desjardins, "The $86 trillion world economy in one chart," 2019.

[5] California Civil Code, "California Consumer Protection Act, (CCPA)," July 2018.

[6] Cybersecurity and Infrastructure Security Agency $SBOM$ Tools Working Group, "Types of Software Bill of Materials (SBOM)," Apr. [Online]. Available: https://www.cisa.gov/resources-tools/resources/types-software-bill-materials-sbom

[7] 93rd United States Congress, "Family Educational Rights and Privacy Act (FERPA) of 1974," 1974, available from: https://www2.ed.gov/policy/gen/guid/fpco/ferpa/index.html.

[8] 104th United States Congress, "Health Insurance Portability and Accountability Act (HIPAA), 1996," accessed on 10-31-2023 Available from: https://www.hhs.gov/hipaa/index.html.

[9] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, and M. Zhou, "CodeBERT: A Pre-Trained Model for Programming and Natural Languages," *Findings of EMNLP 2020*, p. 12, Sep. 2020. [Online]. Available: https://github.com/microsoft/CodeBERT

[10] D. Guo, S. Ren, S. Lu, Z. Feng, D. Tang, S. Liu, L. Zhou, N. Duan, A. Svyatkovskiy, S. Fu *et al.*, "GraphCodeBERT: Pre-training code representations with data flow," *International Conference on Learning Representations*, 2021.

[11] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier *et al.*, "Mistral 7b," *arXiv preprint arXiv:2310.06825*, 2023.

[12] e. a. p. Hugo Touvron et al, year=2023, "Llama 2: Open foundation and fine-tuned chat models."

[13] B. Roziere, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, T. Remez, J. Rapin *et al.*, "Code llama: Open foundation models for code," *arXiv preprint arXiv:2308.12950*, 2023.

[14] Y. Wang, W. Wang, S. Joty, and S. C. Hoi, "Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation," *arXiv preprint arXiv:2109.00859*, 2021.

[15] Y. Wang, H. Le, A. D. Gotmare, N. D. Bui, J. Li, and S. C. H. Hoi, "Codet5+: Open code large language models for code understanding and generation," *arXiv preprint*, 2023.

[16] M. J. Belotto, "Data analysis methods for qualitative research: Managing the challenges of coding, interrater reliability, and thematic analysis," *The Qualitative Report*, vol. 23, no. 11, pp. 2622–2633, 2018.

[17] C. E. Hmelo-Silver, "Analyzing collaborative knowledge construction: Multiple methods for integrated understanding," *Computers & Education*, vol. 41, no. 4, pp. 397–420, 2003.

[18] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[19] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–34, 2020.

[20] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:52967399

[22] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu, "Towards universal paraphrastic sentence embeddings," in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[23] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, 2010.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[27] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. New York, NY, USA: Springer, 2013.

[28] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 9, pp. 2251–2265, 2019.

[29] J. Lossio-Ventura, R. Weger, A. Lee, E. Guinee, J. Chung, L. Atlas, E. Linos, and F. Pereira, "Sentiment analysis of COVID-19 survey data: A comparison of ChatGPT and fine-tuned OPT against widely used sentiment analysis tools," 2023.

[30] B. Liu, Y. Wang, H. Jin, and P. He, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *arXiv preprint arXiv:2107.13586*, 2021.

[31] N. F. Liu, A. Kumar, P. Liang, and R. Jia, "Are sample-efficient nlp models more robust?" 2023.

[32] R. Shokri, G. Theodorakopoulos, J.-Y. L. Boudec, and J.-P. Hubaux, "Quantifying location privacy," in *Security and Privacy (SP), 2011 IEEE Symposium on*. IEEE, 2011, pp. 247–262.

[33] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the crowd: The privacy bounds of human mobility," *Scientific Reports*, vol. 3, p. 1376, 2013.

[34] L. Sweeney, "Simple demographics often identify people uniquely," *Health (San Francisco)*, vol. 671, no. 2000, pp. 1–34, 2000.

[35] P. Golle, "Revisiting the uniqueness of simple demographics in the us population," ser. WPES '06. New York, NY, USA: ACM, 2006, p. 77–80. [Online]. Available: https://doi.org/10.1145/1179601.1179615

[36] S. Barocas and A. D. Selbst, "Big data's disparate impact," *California law review*, pp. 671–732, 2016.

[37] D. Lyon, *Surveillance, Snowden, and Big Data: Capacities, consequences, critique*, 2014, vol. 1, no. 2.

[38] P. Eckersley, "How unique is your web browser?" in *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, ser. PETS'10. Berlin, Heidelberg: Springer-Verlag, 2010, p. 1–18.

[39] S. C. Matz, M. Kosinski, G. Nave, and D. J. Stillwell, "Psychological targeting as an effective approach to digital mass persuasion," *Proceedings of the National Academy of Sciences*, vol. 114, no. 48, pp. 12 714–12 719, 2017.

[40] A. Korolova, "Privacy violations using microtargeted ads: A case study," in *2010 IEEE International Conference on Data Mining Workshops*. IEEE, 2010, pp. 474–482.

[41] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?" in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '21. New York, NY, USA: ACM, 2021, p. 610–623. [Online]. Available: https://doi.org/10.1145/3442188.3445922

[42] Court of Justice of the European Union, "Judgment of the Court (second chamber) on case c-582/14: Patrick Breyer v. Bundesrepublik Deutschland," European Court Reports, 2016.

[43] A. Senarath and N. A. Arachchilage, "Why developers cannot embed privacy into software systems? an empirical investigation," in *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*, 2018, pp. 211–216.

[44] X. Ma, A. Gurjar, A. Chaora, and L. J. Camp, "Programmer's perception of sensitive information in code," in *Symposium on Usable Security and Privacy (USEC) 2024*. Network and Distributed System Security, 2024.

[45] Shokri, Reza and Theodorakopoulos, George and Boudec, Jean-Yves Le and Hubaux, Jean-Pierre, "Privacy games: Optimal user-centric data obfuscation," *Proceedings of Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 299–315, 2015.

Supply Chain Risk Management: Enumeration

Animesh Pattanayak
Pacific Northwest National Laboratory
Richland, WA, USA
Animesh.pattanayak@pnnl.gov

Nina Lopez
Pacific Northwest National Laboratory
Richland, WA, USA
Nina.lopez@pnnl.gov

Lucas Tate
Pacific Northwest National Laboratory
Richland, WA, USA
Lucas.tate@pnnl.gov

Travis Anderson
Lawrence Livermore National Laboratory
Livermore, CA, USA
Anderson300@llnl.gov

***Abstract* —** *The systems we use on a day-to-day basis consist of hundreds, if not thousands, of individual components. When considering system security, the inherent risk of each of these components must be considered. A single vulnerability within a component can have a permeating impact throughout the system. Supply chain risks to modern devices can lead to attacks that affect millions of people, as seen in recent years through the Colonial Pipeline attack. To consider the risk posed by individual components, these components must first be identified. Identifying and understanding these risks can minimize damages by providing organizations with methods to reduce risk. One of the first steps toward identifying risks is performing enumeration of the components in a system. While several internet forums mention performing enumeration of hardware systems, no formal or scientific presentation of hardware enumeration processes were found in academic and industry publications. Furthermore, tools for software enumeration do not provide comprehensive insight of the contents of software packages. This paper describes the processes, procedures, and outputs of hardware and software enumeration and proposes a standardized enumeration process that ensures consistency and reproducibility for use in supply chain risk management. The enumeration process proposed creates a more thorough process of data collection and analysis for hardware and software related products. As supply chain security matures, additions to tools and available information could further improve the ability to craft more complete enumerations and analytic conclusions.*

Keywords—component, supply chain risk management, system, critical infrastructure, publicly available information, enumeration, hardware enumeration, software enumeration, bill of materials, software bill of materials, hardware bill of materials

I. INTRODUCTION

Traditional supply chain management (SCM) focuses on logistics for maximized efficiency, reliability, and revenue; the newer research area, supply chain risk management (SCRM), additionally aims to identify the risks within a supply chain, such as vulnerabilities, weak points, and break points. In an effort to improve the risk assessment, SCRM aims to look at individual nodes within the supply chain to identify the risk introduced. This in-depth risk assessment may enable an organization to better understand how a product influences its reputation, bottom line goals, and trust in products.

Events in recent years, such as the Colonial Pipeline attack [1], involving supply chain failures have raised awareness of the need for security within supply chains. This priority shift has been recognized by government and industry alike, as single faults within supply chains can have significant negative impact. Efforts to minimize the damages to security, finances, and reputations require identifying and understanding the risks within a supply chain. Because SCRM is relatively new, it is important to understand the taxonomy surrounding the topic and attempt to maintain some standardization. Terms such as elements, processes, and network are used in this context [2]. Though these terms may not always be consistent, the concept of an element, process, and network are maintained. Element refers to the individual components, process refers to the way two or more components may relate to each other, and network defines the grouping of multiple elements and their respective processes [2].

Traditional SCM may indicate where single points of failure exist within a supply chain; consider a critical component with only a single vendor. After identification of these single points of failure, corrective actions can be taken, such as selecting multiple vendors for the component. This is where SCRM builds on SCM.

A single fault in a supply chain can have lasting and noticeable impacts on a business and its product lines. Identifying and understanding various risks can minimize reputation, security, and financial damages by providing organizations within a supply chain with methods to reduce its susceptibilities.

Cyber Testing for Resilient Industrial Control Systems (CyTRICS) [3] is a Department of Energy multi-laboratory effort to increase energy sector cybersecurity and reliability. CyTRICS has strategic partnerships with key stakeholders, including technology developers, manufacturers, asset owners and operators, and interagency partners. CyTRICS has taken initiative on driving the key methodology for enumerating systems. Enumeration is the dissection and documentation of a given device or software program. Enumeration is necessary in SCRM as it enhances accountability, awareness, and anomaly detection by requiring a thorough understanding and analysis of the part being enumerated. Additional research into the product during or after the enumeration process also increases the insight of the product's capabilities and vulnerabilities.

While enumeration of a single component can be useful, developing a consistent and reproducible approach to enumeration enables comparability across enumerations, aggregation across multiple collection teams, and the exchange of enumeration data. By aggregating enumeration

data, we can begin to investigate broader questions about supply chains and provide insight into combined attack surfaces. In this paper, efforts to standardize enumerations are presented by detailing the processes, procedures, and outputs to ensure consistency and reproducibility.

When performing an enumeration, it is important to have clear understanding of the scope of enumeration. There may be cases in which most components should be enumerated such as processors, memory chips, and ports. In other cases, chips may be necessary to record but ports are not. One particular delineation with respect to scope is destructive vs. nondestructive enumeration. This scoping indicates whether enumeration should continue when further disassembly of components would potentially cause irreversible damage to the system. One action that may be considered destructive is removing a chip from the circuit board. If an enumeration is requested with the stipulation that the system still be functional upon completion, it is indicative of a nondestructive enumeration.

While the act of performing an enumeration is not inherently illegal, it is strongly suggested that enumeration teams take appropriate measures to understand the licensing information and rights of the hardware or software owner. In short, the enumeration team must ensure that due diligence is practiced before, during, and after the enumeration to ensure appropriate collection methods and data storage.

II. Literature Survey

In preparation of both developing the CyTRICS program and writing this paper, a survey was completed to determine existing practices, theories, and opinions regarding SCRM. As both industry and government continue to place a larger emphasis on securing critical supply chains, there will continue to be a rise in available literature. At this time, the majority of documentation regarding enumeration is specific to software. Less information is available regarding hardware and standardized practices for securing hardware components. Neither software nor hardware have a standardized and documented process that develops a comprehensive view of a given system.

Information regarding supply chain security is also starting to see a larger subset of academic papers. “What do you mean, Supply Chain Security” [2] focuses on defining the framework and taxonomy of supply chain security to better facilitate a common language and set the foundation for standardized practices. This paper defines security in terms of the confidentiality, integrity, and availability triad. Supply chain is divided into elements, processes, and networks. This framework and taxonomy translate to the enumeration process as each step needs to be defined in a reproducible and standardized way.

The literature surrounding enumeration efforts is constantly evolving as enumeration is a somewhat novel process. The enumeration process exists to produce bills of materials (BOMs) for the hardware or software under inspection. Recent articles from OpenBOM, CycloneDX, Fortress, and CISA outline efforts to develop hardware bills of materials (HBOMs) and software bills of materials (SBOMs).

As technology has evolved, the distinction between hardware and software has blurred. Much of modern hardware includes software that runs on it. For example, modern cars may include upward of 100 million lines of code [4]. The OpenBOM article describes the need for a multi-disciplinary BOM to gather and join data together. OpenBOM is a software tool described as a “flexible data model” [4] used to create BOMs.

Another resource for composing BOMs is the CycloneDX project by the Open Web Application Security Project (OWASP). CycloneDX is a model that is capable of bringing together HBOMs, SBOMs, and vulnerability exploitability exchange (VEX) documents, among other data. The CycloneDX BOMs are based on a high-level object model containing data including metadata, components, services, dependencies, compositions, vulnerabilities, and extensions [5].

Once of the primary outcomes of enumerations is the generation of a BOM that provides a detailed view of the exact components that make up a software package or hardware system. The current efforts to generate SBOMs require some form of enumeration, albeit less structured and well defined. By referencing existing SBOMs and efforts to produce SBOMs, the enumeration process can be adjusted to better collect meaningful data.

Part of generating effective SBOMs is to have consistent and standardized terminology and data collected. A well-defined schema can be effective at providing this consistency by specifying the fields to be collected. In addition to stating the terminology, the schema can also define what attributes are to be collected for each of the fields. Efforts to perform enumeration have existed in some capacity prior to being titled enumeration in the form of logical and physical inspection [6] [7], which describe the process for software and hardware enumeration, respectively. Some portions of software enumeration can be automated; however, hardware enumeration can be a more daunting task, particularly for larger systems. For this reason, it may be useful to provide guidance on how to down select the components to be enumerated if resources are not available to perform a full enumeration. Because of their increased potential to be used in an attack, inclusion of all logic-bearing components should be considered a baseline for hardware enumerations [8]. Logic-bearing components include components that possess the capability to perform computation, store data, or communicate with other components, in other words, networking capability.

The National Telecommunications and Information Administration (NTIA) has spearheaded efforts to explore the effectiveness of SBOM within a medical environment [9]. The primary intent of the SBOM is to provide an inventory of the software components and dependencies that make up software systems and define the relationships between the components. Having an inventory provided by the SBOM enables an organization to determine whether certain supply

chain concerns will affect their products. Among other benefits, NTIA identifies reduction of cost and reduction of risk as primary benefits. The NTIA SBOM lists the following as baseline information that can be collected for a component:

- Author name
- Supplier name
- Component hash
- Component name
- Version string
- Unique identifier
- Relationship

NTIA performed a proof-of-concept SBOM for medical devices in a healthcare setting to test the efficacy of its SBOM [10]. Through the process, the generation of the SBOM used both manual and semi-automated processes, including scripting languages and software composition analysis tools. The fields identified as important to collect during the proof of concept were as follows:

- Author
- SBOM document name
- List of SBOM components

After completion of the proof of concept, it was determined that the SBOM had cybersecurity benefits across the procurement process, asset management, and enterprise risk management activities. NTIA identified the following strengths, weaknesses, opportunities, and threats associated with the proof of concept.

Strengths included successful access, ingest, parsing, and querying of the SBOM data. The lack of standard format for data, the lack of consistency in naming devices across organizations, and a lack of authoritative sources for certain fields were identified as weaknesses. The proof of concept identified opportunities to create a standard format, use globally unique component identifiers, and begin discussion of including an HBOM. The lack of a defined auditing and validating process presents a threat associated with the data collected as it may not be accurate or complete.

The NTIA, after analyzing use cases, identified the following benefits for three categories of software users: producers, choosers, and operators [11].

- Software producers
- Software choosers
- Software operators

Similar to NTIA, this white paper emphasizes the importance of creating an SBOM as developers often create products that incorporate open-source and commercial software with proprietary code. Identifying and understanding what is in the SBOM is the first step in addressing challenges associated with hardening systems. Using SBOMs allows components to be compared against known databases so vulnerabilities and counterfeit components can be identified more easily.

III. Foundational Data Structures

At the highest level, the object received for enumeration is known as the system, as shown in figure 1. There is only one system per enumeration, which consists of one or more devices. In this framework, a system is an abstract grouping intended to encompass the breadth of the enumeration activity. A device is then comprised of one or more components identified through the enumeration process. For each component, metadata is gathered and stored in isolation. To the extent possible, further decomposition of the component produces more components for which metadata is subsequently captured. This process continues until all components within the system under test have been enumerated. The hierarchical structure of the components as they relate to one another is then described by a set of pairwise relationships where each relationship indicates membership, direction, and nature of the relationship.

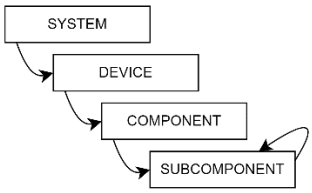


Figure 1: Component Relationship Hierarchy

At the conclusion of an enumeration, what is left is a system containing a list of components that are tied together via relationships. In this way, the nested components of the system and the underlying structure that ties them together are preserved.

IV. Purpose of Enumeration

The purpose of enumeration is to identify all the components within a system. This data then enables us to have a better understanding of how the system works and allows for further research into the system. The components that make up a given product can heavily influence the product's overall exposure to risk based on the component's origin. Understanding how a system works cannot always be assumed by reading the owner's manual, as unintended uses need to be known in critical infrastructure. Enumeration can support security testing by identifying unintended functionality and other security concerns. These unintended uses could have a lasting negative impact on a system, so identifying them is the first step in mitigating any gaps in security. Further research on a system is also needed to best understand all the parts that contribute to the final product. Each component added to a system plays a specific role and can influence the overall security of the system. As a system is typically made up of hardware and software, we have developed enumeration processes specific to both.

A. Awareness and Accountability

Enumeration plays a large role in bringing awareness to both the vendor or manufacturer and the consumer in what actually makes a system and how it can positively or negatively affect their overall network. For hardware this can include memory, processors, ports, and other forms of communication. Software can include types and sizes of files, ports in use, and how the device executes functions, classes, and imports. Each item should be considered a potential entry point for risk and its ability to secure itself and the system

from external threats should be scrutinized. Still, inherent risks can occur with a component. Take for example a field programmable grid array, which is intended to be changed post-production. While this allows for flexibility in functionality, it also introduces new risk as it can be manipulated at any time. These types of inherent risks may not be readily known unless all components within a system are cataloged and researched through enumeration. By having awareness of the components that should exist within a system based on manufacturer claims, an organization can hold itself and the manufacturer accountable for verifying the validity of the claims.

B. Documentation

A core outcome of enumerations is to produce an easily readable output of the data collected. The output of the enumerations is a nested directory structure with JavaScript Object Notation (JSON) files that contain the details of the enumerated components as well as any images and documentation associated with the component. In order for the data in the JSON files to be easily searchable and meaningful, it must be standardized and consistent. To ensure this, the data must pass through a validator that compares it to a predefined JSON schema. This is a data structure that describes what is considered acceptable input data. Among other capabilities, the schema defines what field names are valid, the types of those fields, whether the field is required, and the valid values for a given field. After the data in the JSON files have passed the validation step, they are ready to be stored in a data repository. The inclusion of data in a repository enables centralized access to the data in a query-friendly medium.

C. Anomaly Detection

After data collection, documentation, and research have been completed, anomaly detection can more easily be performed. Anomaly detection is identifying what the system is expected to have or do versus what it actually has or does. In some cases, this could mean discovering the system has missing functionality in the software or that counterfeit components have been used in the hardware. It could also mean discovering the system is capable of more than what was intended or that unexpected ports can add further access via communication.

A more serious example of anomalous behavior would be behavior injected from a malicious actor. In December of 2020, FireEye published research of a software supply chain vulnerability being used to insert malicious behavior into the SolarWinds Orion product [12]. Many government agencies were affected by this incident.

A similar technique implemented on systems within critical infrastructure could have much more serious and visible consequences, such as the disruption of critical services as seen in the Colonial Pipeline ransomware attack. Anomaly detection is key to securing critical infrastructure supply chains, so using enumeration to collate data in a standard form is the logical first step in strengthening a system.

V. Process of Enumerations

The enumeration processes for both hardware and software follow an iterative process of unpackaging, research, and documentation that is repeated until the entire product has been enumerated. The processes differ in how the product is unpackaged and documented because of the variance in what data is collected.

A. Hardware Enumeration

1) Unpackaging

When a system is received for enumeration, the first step is to unpackage the system from the shipping container it is received in. The system may be received in a package with supplementary material such as additional cords, power cables, disassembly tools, and documentation. It is important to make note of these supplementary materials as they can be instrumental in proper disassembly or operation of the system. The documentation may provide insight for how certain components need to be removed from the system and in some cases, there are special tools for removing pieces from the unit. After determining which parts of the packaging received are actually part of the system to be enumerated and which parts are supplementary materials, the enumeration process can begin. During the unpackaging process, an inventory should be kept of the parts that arrived in the package and which parts are considered part of the system versus supplementary.

2) Imaging

Imaging includes taking photographs or other visual captures of the system and its components. Imaging should be completed every time a previously undocumented subcomponent is identified during the disassembly phase of enumeration. Images taken should include all critical or relevant information about the component; for example, an integrated circuit where writing on the component is readable should be photographed in a manner that allows the writing to be referenced for information in the future. The time and tools needed for imaging varies based on what is being photographed. For example, capturing an overview image of a system is often as easy as point and shoot, while taking a close-up image of an integrated circuit on a circuit board takes additional stabilization and polarization to optimize the image. Regardless of what is being photographed, the image should be clear and easy to view. An image naming convention is also invaluable when enumerating as recalling specific images needs to be efficient so additional processes can run in a timely fashion.

3) Research

Additional research is almost always required to find key identifying information about a component or system. Identifying information can include part names, part numbers, and the part manufacturer. Research is conducted on the internet using publicly available information (PAI) and results usually include datasheets, manufacturer information

releases, and third-party resellers. Discrepancies found in datasheets as compared to the physical component can be critical in anomaly detection.

4) Documentation

Data collected through hardware enumeration needs to be documented thoroughly and in a manner that allows for it to be easily ingested and analyzed. A schema, as described in the purpose of enumeration section, gives a way to standardize data so it can be inserted into a repository and ensure all data points are collected. Further information about data structuring related to enumeration is detailed in *Supply Chain Risk Management: Data Structuring* [13].

5) Disassembly

While the primary goal is data collection, a good enumeration has a secondary goal of ensuring the system is functional post enumeration. Therefore, disassembly of a device is done in a nondestructive manner that ensures the device can be reassembled and function the same as before disassembly. Each device is disassembled into individual components until doing so would result in the destruction or damage of the device or component. Before beginning disassembly and after each step of the process, documentation and imaging of every new component identified is needed.

6) Repeat as Necessary

The previous steps should be repeated recursively until the device has been completely disassembled and all components have been documented. Imaging should also be completed after the system has been reassembled to ensure the system is back in its original condition.

B. Software Enumeration

Software enumeration is the process of identifying the individual software files, collecting metadata about the files, and documenting the relationships between them. Because software is not a tangible object like hardware, the enumeration process is inherently different.

1) Unpackaging

Generally, software will be provided in a zipped folder. The first step in software enumeration is to handle unzipping the software package as received. These zipped packages may be in a variety of file formats including .zip, .tar, and .7z. There are standard tools that can be used to unzip these packages. After unzipping the package, the tester should have some directory structure with software files. The directory structure will vary with each software package, but generally there will be several levels of nested folders with software files, documentation, and licensing information.

2) Basic Detail Enumeration

Software enumeration initially collects details about each file: the file name, hash values, file path, and file type. Collection of these details can be automated using a Python script.

Additional information related to the specific contents of each file can be extracted using knowledge of the specific file format and headers. This can be used to get file type specific metadata such as the vendor, version number of the software, and imported shared libraries. Going a step further with binary files to disassemble/decompile the file can be used to collect information on the functions, classes, strings, and vulnerabilities associated with each file. After running the enumeration script and performing the additional steps listed, a tester should ensure that the fields recorded meet the minimum set of requirements for a complete SBOM as defined by NTIA, described in the Literature Survey.

a) File Name, File Path, and File Type

The file name attribute of a file is self-explanatory. The file path attribute is the relative path from the root folder of the software package to the location where the file is located. For basic details enumeration, the file type can be determined based on the extension on the file when an extension is present. For example, a .py file is a Python file and a .docx file is a Microsoft Word file. However, file types can be spoofed by changing the extension to make a file appear as a different type than it actually is.

b) File Hashes

The last of the basic details to collect are the hash values associated with a file. Hash values are intended to produce a unique string value based on the contents of the file. A good hashing function will produce a hash such that even a small change in the contents of the file will produce a different hash value. The hash values can be used to verify the identity or ensure the integrity of a file. If the manufacturer claims that the file hash will be *a1b2c3d4* and the computed hash value differs, the tester has reason to believe that some contents of the file have been modified from what the manufacturer released and requires further analysis. There are numerous hashing functions commonly used, but for our enumerations we use SHA1, SHA256, and MD5 because of their current and historic use.

3) Documentation

All the details captured must be documented, preferably in a specific format that conforms with the schema mentioned in the earlier section of this paper. By conforming to the schema, data collected can easily be entered into the data repository with limited preprocessing. The last step in software enumeration is to perform any research necessary to ascertain further details about the software package and individual files. The primary goal of this research is to discover known vulnerabilities, datasheets, web pages, and open-source repositories, but any additional information found is useful. The process described should be recursively repeated for each individual file in the directory structure.

VI. Tools and Techniques

A. Physical Tools for Enumeration Assistance

The hardware enumeration process requires multiple physical tools to better deconstruct and document a given device. These tools include the workstation setup, tool kits, cameras, and microscopes. Other physical tools may need to be considered depending on an organization's goals for device enumeration.

1) Workstation Setup

The workstation setup will vary depending on the device to be enumerated. Additionally, the number of workers who may be at one table contributes to the size and shape of the desk needed to accommodate people, tools, and the device. After the desk itself has been chosen, all workstations should include antistatic mats to reduce device damages and bodily injuries. Antistatic sheets, often made of foam, should also be used when devices are being deconstructed in order to reduce potential static interactions.

2) Tool Kit

Designated tool kits per station are needed for teams planning to work at the same time. The kit should be designed for electronic disassembly and reassembly to best prepare for varying screw heads and sizes, as well as niche tools such as spudgers and angled tweezers. An all-in-one kit can be extremely useful when it comes to organization and storage, but often additional tools such as wrenches may need to be purchased outside of a kit designed specifically for electronics.

3) Camera

A camera with the capability to change lenses to take better images is a must. A mirrorless camera is more expensive than a digital single-lens reflex (DSLR) camera and does not have an optical viewfinder. A DSLR camera with a cross-type autofocusing system, tripod mount capabilities, self-cleaning sensor unit, and a minimum of 12 megapixels is recommended for enumerations. The cross-type autofocusing system will ensure that the camera will be in focus, especially when the subject of an image contains vertical lines. This is especially useful when photographing printed circuit boards and other small electronic components. A self-cleaning sensor unit will help ensure that there are minimal dust particles present, which allows for clearer images. Any dust particles on the lens can cause interference and hinder the process of capturing an image of the device. A tripod can be used to help stabilize the camera over the device to allow for a more focused image. By selecting a camera with a minimum of 12 megapixels, the enumeration team can be confident there are enough details captured in the image. If scope and budget allow, a camera with a higher megapixel count should be chosen to enable the enumeration team members to capture additional detail in their images. An 18–55 mm f/3.5–5.6 lens was used for taking images of the component under test. This lens is versatile, allowing for images of the overall device and adequate images of the printed circuit board (PCB). Conformal coating, a protective film placed over integrated circuits and PCBs, can present an issue when taking photographs. Namely, the conformal coating can produce a glare when attempting to capture images of components, causing the text on the component to be unreadable in the image.

4) Magnification

Some form of magnification will be required to read identifying text on components, identify anomalies, and count pins, in some cases. Various tools such as microscopes, magnifying glasses, or jeweler's loupes can be used for this purpose. The power of magnification needed depends heavily on the types of components and subcomponents expected to be enumerated. Additionally, lighting and stage options need to be considered as tints of lighting can affect conformal coating differently and stage sizes can impede microscope use. Versatility is also key in identifying key features accurately and efficiently. The team currently uses a jeweler's loupe to move the viewer easily and uses a binocular compound microscope for areas that require a higher magnification. The team is considering more advanced options that can be found in the Conclusions and Future of Work section.

B. Electronic Tools for Enumeration Assistance

Enumeration can be a manually intensive task. The hardware systems and software packages received can range from small with fewer than ten components up to very large with hundreds of components. Documenting all of these components, handling the images and datasheets associated, and ensuring the data conforms to the schema and JSON format can be a challenging task. For this reason, eliciting the use of electronic tools can ease the burden on the tester to ensure each of these challenges are met appropriately.

1) Output Generation

One electronic tool that can be helpful for ensuring information is formatted correctly is an output generation tool. In this case, the tool may present the user with a frontend web page in which the tester inputs details into appropriately labeled text fields. These values can be used to generate the JSON output file in such a way that there are no concerns about correct formatting. The tool can contain more complexity to represent the relationships between two components but ensure appropriate and consistently formatted output is necessary.

2) Software Enumeration Scripts

As discussed in the software enumeration section, a significant portion of the basic software enumeration can be automated with a script to collect the file name, file path, file type, and hashes. This script takes advantage of several Python modules to collect information such as hashes and extension determination. The software enumeration script also outputs to the desired JSON format. The aforementioned script is shown in the Basic Detail Enumeration subsection.

VII. Results and Output

The standardized process of enumeration is still being developed and agreed upon by high levels of government and industry. Due to this, comprehensive examples of enumerations are not publicly available due to the sensitivity of the information.

1) JSON Structure

After an enumeration is performed, the data collected and generated needs to be stored in a consistent structure that will make entry into a database seamless. One way to structure data is to use a JSON file format. JSON stores information in key-value pairs in which the key is unique and the value can either be a static value or another JSON object.

For example, the following JSON is valid in that the value associated with the key ***name*** is a static value.

```
{
        “name”: “Alice”
}
```

The next example shows a more complex JSON structure in which the value associated with the key ***person*** is another JSON object.

```
{
        “person”: {
                “name”: “Alice”,
                “age”: 30,
                “profession”: “programmer”
        }
}
```

The data from an enumeration will be stored in a JSON structure, which contains all the pertinent details collected and generated throughout the enumeration process. Each component (hardware), file (software), and relationship will have a separate JSON object in the file. Relationship objects exist to allow a user to understand how two components, or two files are related to each other. For example, if a microprocessor is on a circuit board, the microprocessor is a child of the circuit board. In the case of software, if a file ***hello.c*** exists within the directory ***C_Files***, then the file hello.c is a child of the directory C_Files. By leveraging these relationships, a user is able to represent the structure of a software package or hardware system and understand where within the larger system/package an individual component/file exists.

2) Proposed Data Schema

JSON is a useful format for structuring the data, but to achieve consistency, we need to define what fields we expect and what the content of those fields should look like. For this we need to define a JSON schema. The schema allows us to define:

- Fields to be collected
- Human-readable descriptions of the fields
- Examples for data collectors to see
- Expectations for tracking changes
- Logical relationships/dependencies between fields.

Perhaps more importantly, the schema facilitates rapid quality control because it can be used to validate submitted JSON data and point to errors made in collection. This provides a level of assurance that the data submitted from enumeration matches expectations. Enforcing this consistency makes entry into a database automatable. If the JSON conforms to the schema, then each field in the JSON file can be mapped to a field in a database.

3) Repository Motivation and Objectives

Repositories can come in many forms and are generally shaped by the nature of the data that will be stored in them. Some examples of potential repository forms include databases (ex: MySQL, Postgres, NoSQL, MariaDB). While specifics of the storage are outside of the scope of this document, the primary motivations for the repository are important to understand as part of the complete process.

The primary objectives of the repository are to archive the data and perhaps more importantly to make it accessible for reuse. Using the well-structured data collection can facilitate direct ingestion into a database. From there it is important to consider how the data will be indexed for search and to ensure that the structure you choose aligns well with the anticipated research. Answering the following questions at the onset will help ensure that the resulting data is well aligned with the intended use.

- Is string searching sufficient?
- Do you need to traverse relationships in the query?

VIII. Conclusions and Future Work

Developing and integrating a standard enumeration process is essential to securing critical infrastructure and the associated supply chains. The proposed enumeration is well formatted to service hardware and software related products to create a more thorough method of data collection and analysis. Current methods are sufficient but additions to accessible tools and available information would better situate teams in crafting complete enumerations, and ultimately, analytic conclusions.

While enumeration is an integral step in understanding the risk associated with a system, it is only one of the first steps in completing a full risk analysis. Enumeration informs the tester of the system components but does not provide any risk level indicator. Some current challenges and limitations being experienced by those developing and implementing enumeration procedures include differences in terminology and catering to varying levels of expertise and tools available. In order to have a longstanding and successful methodology, terminology must be standardized across industry so cross pollination of information and ideas can occur. Additionally, this standardized process needs to be scalable for complexity of the device as well as the existing capabilities an enumeration team may have. After enumeration, performing PAI research on each component will inform the tester of any known and documented susceptibilities associated with the component that influences the overall risk. In addition to what is publicly available, performing hands on analysis can assist in discovering new risk indicators. When enumeration is performed in conjunction with PAI research and analysis, a more complete risk profile can be generated for the system.

Future work should include the funding for an enumeration on a piece of obsolete hardware and software to thoroughly demonstrate and document the process.

A. Tools to Consider

1) Handheld Microscope

Adding a handheld microscope to physical tools could prove to be beneficial in not only imaging but also documentation. There are a wide variety of handheld microscopes on the market with differing specifications based on intended use. A handheld microscope with dynamic range, polarization, large working distance, and a wide-angle lens will allow closer and clearer imaging than a camera with an 18–55 mm lens. Additionally, the versatility of a handheld device allows the user to angle the microscope in a manner that a traditional microscope would be unable to reach.

2) Digital Microscope

Still, there are more advanced digital microscopes that can be used to construct virtual three-dimensional representations of the product as well as up to 6,000 times magnification. Determining which microscope, if any, will enhance current enumeration procedures will depend on cost, the ability of team members to learn how to use, how the added features influence data collection, and any added security risks.

3) Additional Lens and Lighting Options

A 100 mm f/2.8 macro lens would be a great additional lens. A macro lens would enable capturing high-quality images of the text on small components, which are traditionally hard to read by the human eye. Additionally, having multiple LED lamps or overhead lights would allow for better images by providing proper lighting from several angles without emitting much heat. An LED light and the macro lens would allow taking quality images of the PCB with minimal problems from conformal coating.

4) Vendor/Manufacturer Supplied BOMs

Detailed BOMs from vendors would help confirm the contents of a device and make finding discrepancies easier. Additionally, having access to such information could allow enumeration teams to have varying levels of disassembly performed depending on the client's needs. For example, confirming parts are present by comparing to the BOM from the vendor or manufacturer would be a faster process than discovering, identifying, and confirming a component is supposed to be in the system. The time spent on enumeration would depend heavily on the type of information already provided and also would influence what types of tools would be necessary for the type of enumeration.

ACKNOWLEDGMENT

The authors would like to thank their partners throughout the U.S. Department of Energy, Office of Cybersecurity, Energy Security, and Emergency Response (DOE CSESER)'s CyTRICS program, including Idaho National Laboratory, Lawrence Livermore National Laboratory, Oak Ridge National Laboratory, Pacific Northwest National Laboratory, and Sandia National Laboratories.

REFERENCES

[1] G. M. Makrakis, C. Kolias, G. Kambourakis, C. Rieger and J. Benjamin, "Industrial and Critical Infrastructure Security: Technical Analysis of Real-Life Security Incidents," *IEEE Access,* vol. 9, 2021.

[2] J. Smith and J. Teuton, "What do you mean, Supply Chain Security? A Taxonomy and Framework for Knowledge Sharing," in *Hawaii International Conference on System Sciences*, Waikoloa Village, 2017.

[3] Idaho National Laboratory, "CyTRICS: Cyber Testing for Resilient Industrial Control Systems," U.S. Department of Energy: Office of Cybersecurity, Energy Security, and Emergency Response, [Online]. Available: https://cytrics.inl.gov/. [Accessed 05 01 2023].

[4] O. Shilovitsky, "OpenBOM," 27 05 2021. [Online]. Available: https://www.openbom.com/blog/bill-of-materials-lifecycle-software-vs-hardware-and-multi-disciplinary-bom. [Accessed 10 01 2023].

[5] CycloneDX, "Hardware Bill of Materials," OWASP, [Online]. Available: https://cyclonedx.org/capabilities/hbom/. [Accessed 10 01 2023].

[6] N. Lopez, E. Pollans, M. Kirkland, J. Seaman, A. Pattanayak and L. Tate, "Landscape Analysis: Supply Chain Risk Management Established Methodologies," Unpublished., 2022.

[7] S. Pal, "Technology for Reducing Risks of Equipment Failures and Cyber Incident," New Orleans, 2022.

[8] N. Lopez and E. Pollans, "Site Best Practices," Unpublished, 2023.

[9] National Telecommunications and Information Administration, "SBOM at a Glance," 27 April 2021. [Online]. Available: https://www.ntia.gov/files/ntia/publications/sbom_at_a_glance_apr2021.pdf. [Accessed 9 January 2023].

[10] National Telecommunications and Information Administration, "Software Component Transparency: Healthcare Proof of Concept Report," 1 October 2019. [Online]. Available: https://www.ntia.gov/files/ntia/publications/ntia_sbom_healthcare_poc_report_2019_1001.pdf. [Accessed 9 January 2023].

[11] National Telecommunications and Information Administration, "Roles and Benefits for SBOM Across the Supply Chain," 8 November 2019. [Online]. Available: https://www.ntia.gov/files/ntia/publications/ntia_sbom_use_cases_roles_benefits-nov2019.pdf. [Accessed 9 January 2023].

[12] C. Jaikaran, "SolarWinds Attack—No Easy Fix," Library of Congress, Washington, D.C, 2021.

[13] N. Lopez, A. Pattanayak and J. Smith, "Supply Chain Risk Management: Data Structuring," in *2022 Resilience Week* , National Harbor, 2022.

Using an SBOM to Mitigate a Lemons Market

Peter J. Caven
Luddy School of Informatics, Computing, and Engineering
Indiana University
Bloomington, Indiana, United States
pcaven@iu.edu

Xinyao Ma
Luddy School of Informatics, Computing, and Engineering
Indiana University
Bloomington, Indiana, United States
maxiny@iu.edu

Vafa Andalibi
Luddy School of Informatics, Computing, and Engineering
Indiana University
Bloomington, Indiana, United States
vafandal@iu.edu

L. Jean Camp
Luddy School of Informatics, Computing, and Engineering
Indiana University
Bloomington, Indiana, United States
ljcamp@iu.edu

***Abstract*—The Software Bill of Materials (SBOM) has emerged as a possible tool to mitigate information asymmetry within the security market. By promoting transparency throughout the supply chain, stakeholders now have crucial information that can support decisions throughout a product's lifecycle. This pre and post-procurement decision support aligns with the evolving cybersecurity paradigm and supports well-established economic models. Our research identifies the need for more effective communication within the current security market. While SBOMs may present an effective and viable option, their current instantiation is not suitable for all consumers. We explore how SBOMs can be made more usable. This paper seeks to draw from our research to discuss the economic benefits of communicating security. Particularly, we focus on how both visualizing SBOMs and integrating SBOM information into labels can increase transparency, which increases consumers' willingness to pay.**[1]

***Index Terms*—Security, Labels, SBOM, Permissions, Secure Supply Chain**

I. Introduction

The concept of a *lemon's market* was introduced in 1970 to describe the U.S. second-hand car market [2]. The term *lemon* refers to a used vehicle with hidden issues or defects that may not be readily apparent to a buyer. Unscrupulous sellers may attempt to mask underlying issues by investing in cheaper superficial improvements (e.g., new exterior paint or interior detailing) [16].

A lemon market can be characterized by three key factors. First, information asymmetry exists, wherein the seller holds more information about the quality of the goods than the buyer. Second, the buyer rationally assumes that the goods offered for sale are of inferior quality since the seller has not sufficiently proved the quality. Finally, the development and sale of high-quality goods become financially impractical, as there are no reliable means for a buyer to assure quality [2], [16], [28], [31]. Essentially low and high-quality goods become indistinguishable. This results in a situation where buyers are unwilling to pay a premium for a potentially higher-quality good, fearing it may end up being low quality. This threat of inconsistent quality leads to a lack of consumer confidence and decreased demand across the entire market.

A lemons market is normally a two-sided market, where one person is selling a good and one person is buying a good. However, technology creates more complicated markets [23]. For example, the mobile app market is more intricate due to the existence of three major stakeholders: the developer, the buyer, and the marketplace competition (often in the form of a duopoly) [49]. Mobile app marketplaces need to simultaneously cater to the needs of developers, while at the same time instilling trust in buyers. More explicitly, a marketplace must appeal to developers, since app availability increases a smartphone's usability and functionality, which drives hardware purchases; simultaneously, the marketplace must maintain buyers' trust in the hosted applications to ensure continued marketplace utilization. This leads to the need for clear benefit communication [49]. Without it, consumers may lack confidence, especially when there are disparate ratings across multiple marketplaces (i.e., one has only positive reviews and the other has negative reviews).

Information asymmetry is the result of a communications problem. In this paper, we will use experiments designed to peel back the layers of the lemon market to foster better communication. We focus on how visualizing SBOMs and integrating SBOMs into labels can both be used to increase transparency and mitigate the lemons market in terms of security and privacy. In doing so, we demonstrate that while SBOMs may be complex, they hold the key to providing a more safe and secure technology market.

[1]This research was supported in part by CTIA and the Comcast Innovation Fund. We acknowledge support from the US Department of Defense [Contract No. W52P1J2093009]. This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Number 17STQAC00001-07-00. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security, Comcast, CTIA, Indiana University, nor the U.S. Department of Defense.

II. Background & Related Works

A challenge arises in how privacy and security risks are communicated to users. Various measures, such as runtime permissions, manifest presentations, Apple tracking transparency, and Apple privacy labels have been implemented to address this issue [6], [29], [34], [45]. The core argument is that providing more information to users enables them to make better decisions. When users have access to better risk communication, they can demand higher security and privacy, which increases their willingness to pay for secure products [12], [15], [22]. This creates an incentive for developers to develop better products. Without this signal, developers with superior products may never enter the marketplace, leading to a decrease in overall quality and the persistence of a security lemons market [2].

A. Risk Communication

To improve risk communication and change market behavior, we can start with some assumptions from the early days of risk communication – show users they are choosing a risky option and that it is better not to take risks. But if users have no choice (e.g., they have to share phone contacts in order for the application to work), they will acquiesce. Therefore, we need to create partnerships to encourage developers to request fewer permissions, thus building long-term trust in marketplaces. Additionally, it empowers users to select the most secure and privacy-preserving apps, in turn decreasing information exfiltration. This is, of course, the goal of risk communication.

Risk communication also tells us that security should be a gain. We can leverage prospect theory, which suggests that people prefer gains over the probability of loss [27]. Currently, all users see are gains. They are making decisions on marginal gains and losses, not the final outcome. No one is choosing to be vulnerable allowing hackers access to their data. However, they are making incremental gain and benefit choices that lead to this outcome. Therefore, we should present security and privacy as gains in our risk communication strategy. We need to communicate more effectively using models that are shown to work in benefit communication and apply them to risk communication.

Studies have shown that offering phone owners better choice points for permissions, and ways to prioritize higher-quality permissions, can lead to more informed decisions [5], [32], [34], [45]. This technical problem can be addressed through an economics-based solution. Currently, decisions are solely based on benefits, and developers have no incentive to have correct privileges or protect data. There is no cost for being risk-maximizing. By using simple indicators to communicate risks and benefits at the moment of decision, customers are more informed [5], [20]. Our underlying assumption is that app purchases drive developers' engineering choices. To achieve better risk communication, we must provide clear information about risks and benefits, empowering users to make secure decisions and prioritize their privacy.

B. Security Labels

To address these challenges, the National Institute of Standards and Technology (NIST) has proposed the use of security labels [37], [38]. In 2023, the Federal Communications Commission (FCC) revealed the U.S. Cyber Trust Mark, stemming from NIST's security labeling recommendations [18], [52]. On March 14, 2024, the FCC voted to use this label for wireless consumer Internet of Things (IoT) products [19].

Again, the same multiple decision-makers are playing a role in this process (i.e., developers, buyers, and marketplaces). And again, developers will not invest in creating secure, less privileged products if buyers do not prioritize security. If we want buyers to care more about security, labels and risk communication must be attention-grabbing, easy to comprehend, and aligned with users' mental models [6], [9]. Presenting this information at the time of decision-making is crucial.

A good example is the United States' *Smoking Kill* label, which is clear and straightforward, effectively conveying the risk without requiring extensive understanding of medical harms [21]. There are similar labels in Australia, which are even more effective by using graphic imagery [24], [50]. All these labels serve as a warning, conveying the message that smoking leads to death without the need for detailed or convoluted medical jargon. However, the field of computer science still struggles to achieve such clear and urgent risk communication; it expects users to understand technical terms or implicit impacts associated with risk.

C. Software Bill of Materials

Complementary to NIST's labeling effort, the National Telecommunications and Information Administration (NTIA) is working on a software listing called a Software Bill of Materials (SBOM). This has been described as an ingredient list for systems. NTIA defines an SBOM as a nested inventory of all the components, information, and supply chain relationships that contribute to a piece of software [41].

Bills of materials have long been standard practice in manufacturing environments to identify all materials used in the manufacture of a product. Similarly, an SBOM enables secure use by identifying all software components; thus, it can be used to trace vulnerabilities embedded in complex code packages [51]. The minimum level of information creates interoperability and allows for the traceability of components through a product's supply chain. While an SBOM is important, by itself it is not sufficient to create a market for safe, secure software and operations.

D. The SBOM Lifecycle

The SBOM lifecycle occurs over four cyclic phases: software evaluation, generation, operations, and verification. In the generation of an SBOM, we want to identify the components, represent them correctly, and prune them so that components that are of no concern are removed. For verification of the SBOM, we might do static analysis, configuration file analysis, real-time runtime analysis; we will want some cryptographic attestations of these. For using SBOMs in operation, we

integrate it with our threat modeling, map vulnerabilities to service, and identify mitigations. Then we update the code and do software evaluation, where we map against current SBOMs, identify differentiations, look at how the dependencies occur for different services, and evaluate customer-specific use-cases.

Fig. 1: The four phases of the SBOM lifecycle.

In theory, this idea of a well-defined software lifecycle, consisting of requirements, planning, software design, testing, and release, is sound. In practice, software development and deployment can be more chaotic and iterative than this simplified model suggests. It requires constant vigilance to identify vulnerabilities and maintain a secure codebase throughout the software's lifecycle. Especially when we discover something wrong started in the requirements phase and we have to redevelop it, iteratively updating the code as we learn more. How then do we support users and developers in this constant feedback, nonlinear lifecycle? In this paper, we conduct experiments to identify ways of communicating these vulnerabilities. Applying the lessons learned can make unfriendly SBOMs more friendly to a variety of stakeholders.

However, we need to maintain some level of complexity to have value for technical stakeholders. For example, security labels only serve the least expert and do not address operational concerns. They are inadequate for experts needing technical information. SBOMs address this requirement gap, though their current instantiation is not designed to be integrated into labeling efforts. Another consideration is SBOM's consistent update and validation, providing a more accurate representation of a product's current security posture. A static label, on the other hand, represents the validation within a moment in time, which may not accurately reflect the reality of a dynamically evolving security ecosystem.

E. Vulnerability Disclosures

The significance of patching can be traced back to events like the Therac-25, a medical linear accelerator used for cancer radiation therapy. Eleven machines were installed, and six people were seriously injured, with numerous other complaints about the device. Still, Therac did not fix it until the U.S. Food and Drug Administration (FDA) forced them to do so [30]. The lack of timely resolution resulted in severe injuries to patients, highlighting the critical need for efficient vulnerability disclosure and response processes.

While often a vendor will acknowledge and quickly patch a vulnerability, other times they may take months to respond, and sometimes outright deny the existence of said vulnerabilities. Still, vulnerability reporting and 3rd party analysis create tremendous value in the ecosystem [26]. However, the scale of vulnerabilities has surpassed human evaluation capacity. In 2020, the National Vulnerability Database identified 18,349 vulnerabilities; in 2023, it had grown to 28,819 a year [39]. It is no longer possible for humans to adequately evaluate all these issues, and in 2024, NIST paused its enrichment efforts of Common Vulnerabilities Exposures (CVEs) [40]. This highlights the importance of automation in assessing, framing, and prioritizing vulnerabilities.

The discovery of a vulnerability is often challenging, but once it is identified, applying that vulnerability and determining the scope and impact on systems is a bigger challenge. And when we introduce bugs into this ecosystem, the complexity increases again. The SBOM is critical in managing increasingly complex software ecosystems [51]. And it is a critical tool for future automation efforts. Currently, SBOMs are represented by a machine-readable file structure such as JavaScript Object Notation (JSON). This allows machines to quickly parse and build nested inventories so that when a vulnerability is identified it can be properly managed. SBOMs can be used to determine when to invest based on the status, technical impact, level of access, and likelihood of an attack.


```
{
     "SPDXID": "SPDXRef-Package-14-numpy",
     "name": "numpy",
     "versionInfo": "1.26.4",
     "primaryPackagePurpose": "LIBRARY",
     "supplier": "Organization: Travis E. Oliphant et al.",
     "downloadLocation": "https://pypi.org/project/numpy/1.26.4",
     "filesAnalyzed": false,
     "checksums": [
       {
         "algorithm": "SHA1",
         "checkumValue": "9815c16f449e12915ef35a8255329ba26dacd5c0"
       }
     ],
     "licenseConcluded": "NOASSERTION",
     "licenseDeclared": "NOASSERTION",
     "licenseComments": "numpy declares Copyright (c) 2005-2023, NumPy Developers.
which is not currently a valid SPDX License identifier or expression.",
     "copyrightText": "NOASSERTION",
     "summary": "Fundamental package for array computing in Python",
     "externalRefs": [
       {
         "referenceCategory": "PACKAGE_MANAGER",
         "referenceType": "purl",
         "referenceLocator": "pkg:pypi/numpy@1.26.4"
       },
       {
         "referenceCategory": "SECURITY",
         "referenceType": "cpe23Type",
         "referenceLocator": "cpe:2.3:a:travis_e._oliphant_et_al.:numpy:1.26.4:*:*:
*:*:*:*:*"
       }
     ]
   },
   {
     "SPDXID": "SPDXRef-Package-15-psutil",
     "name": "psutil",
     "versionInfo": "5.9.0",
     "primaryPackagePurpose": "LIBRARY",
     "supplier": "Person: Giampaolo Rodola (g.rodola@gmail.com)",
     "downloadLocation": "https://pypi.org/project/psutil/5.9.0",
     "filesAnalyzed": false,
     "checksums": [
```


Fig. 2: An example of what a few packages in an SBOM look like in JSON format.

III. Method & Experiments

In this section, we discuss how SBOMs can be used to mitigate a *lemons market*. First, we draw a similarity to a visualizer we developed for Manufacturer Usage Description and how SBOM data can be represented to provide a more

complete understanding of its interaction. Next, we will determine which security factors users may find the most salient within the presented SBOM. Finally, we seek to determine if consumers would be willing to pay for more secure products. These experiments demonstrate that while SBOMs in their current form may not be usable, they have significant value if they can be communicated in the right way to the right users.

A. Visualizing SBOMs

We can gain valuable insights from app permissions, software development practices, and access control through the Manufacturer Usage Description (MUD). MUD is a machine-readable access control list designed for IoT devices, ensuring secure device-specific access control without the need for customizing hardware controls. The idea is to simplify onboarding and facilitate device integration by sharing MUD files. The goal is to enable easy plug-and-play usage for all users [3].

1) Method: To assess the usability of the MUD-Visualizer, we conducted a comparison study where 52 participants were divided into two groups: a control group using standard textual MUD files and an intervention group using the MUD-Visualizer, shown in Figure 3. We use the control to evaluate the effectiveness of visualization. Both groups were tasked with answering 23 questions regarding the information presented in the MUD. For example, participants were tasked with identifying which remote servers or local devices were allowed to interact with other devices on the network based on their MUD file. Additionally, we asked basic questions related to the protocols permitted by the devices, such as IP version, port numbers, and whether TCP or UDP was used. The MUD-Visualizer has been developed to simplify information presentation, allowing increased usability and understanding of interconnected information.

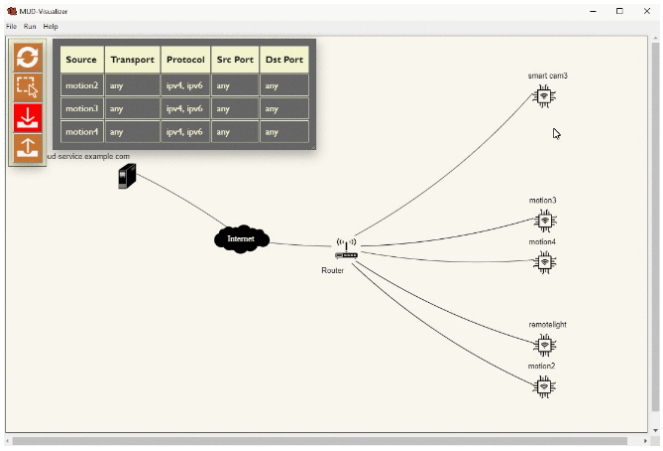


Fig. 3: The MUD-Visualizer can help visualize the connected nodes as well as display the traffic data.

2) Results: Using the Software Usability Scale (SUS) for both the control and intervention, we can determine the groups' usability; an aggregate score of 68 is considered to have average usability [4]. Participants in the control group scored 55.19, while those in the intervention scored 77.02. This higher score is indicative of increased usability of complex information. We used the Mann-Whitney rank sum test to determine that this difference between the groups was statically significant, $p < .001$. When analyzing the time to selection, as well as the accuracy of selection, we again see the intervention outperforming the control. The MUD-Visualizer took significantly less time to use, almost a third less time than the control. Additionally, the accuracy difference was statistically significant, where participants using the MUD-Visualizer had a median accuracy of 100.00%, while the control only had a median accuracy of 78.26%.

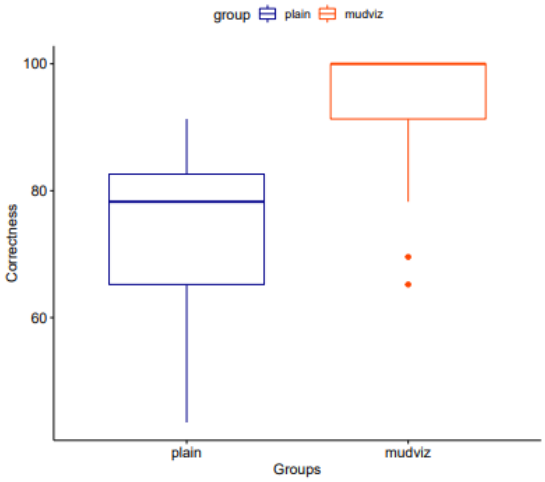


Fig. 4: Median accuracy between the control (i.e., plain) and the intervention (i.e., mudviz).

3) Implications: This visualization approach has the potential to expand to be used for SBOMs. When users receive the manufacturer's usage device access control list, they gain insight into the components of their device. Similar to an SBOM, the MUD is machine-readable but not easily accessible to humans. To support developers in generating and visualizing MUDs, they require proper support. As demonstrated in our study, an effective method to do this was the MUD-Visualizer. Simply allowing visual manipulation instead of relying solely on written access control rules increased accuracy. This visualizer concept aligns with SBOM generation; creating a map of dependencies, where all other dependencies naturally flow as part of the overall SBOM's upstream connections. Adding or removing components becomes intuitive, as users do not have to review the entire SBOM for each change. This not only makes it computationally efficient but also useful to those interacting with the code.

B. SBOM Visualization Tools

While automation will drive the initial vulnerability identification, human interaction is important to determine the criticality of the vulnerability, scope, and impact. Organizations have finite resources and cannot remediate every risk. Visualization tools help map complex interdependencies, allowing analysts

to assess the broader implications beyond automated outputs. Combining automation for detection and human expertise for contextual analysis creates a more comprehensive approach to managing security risks. However, for human expertise to be valuable, decisions must be accurate. Recall, that the MUD-Visualizer demonstrated a high correlation between accuracy and accessibility, and we can apply this finding to SBOMs.

1) Method: In this study, we compared two open-source SBOM visualization tools (i.e., It-Depends and DeepBits) against a machine-readable JSON file generated with the Software Package Data Exchange (SPDX) standard. It-Depends focuses on identifying dependencies and flags packages with vulnerabilities, as shown in Figure 5. This tool lacks detailed information, such as the source of the vulnerability [53]. In Figure 6, we show the use of DeepBits, a commercial AI-based SBOM generation suite [13]. This tool provides source information on package vulnerabilities, including CVE identifiers. For our study's control, we use JSON files using the SPDX standard, an open standard for representing SBOMs [48]. We use the JSON file format to provide a more *traditional*, machine-readable SBOM to compare against the other two visualization tools. An excerpt of the JSON file format is shown in Figure 2.

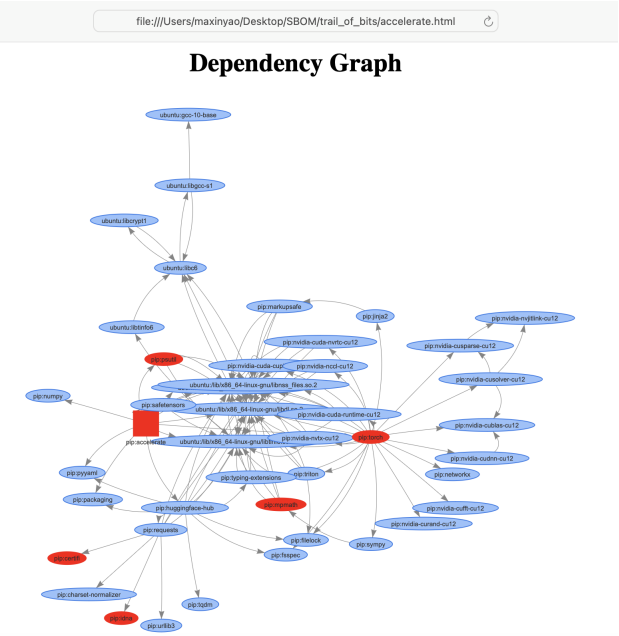


Fig. 5: It-Depends shows the interconnected dependencies between vulnerable packages.

This study was based on vulnerability identification and mitigation tasks. Specifically, we evaluated SBOM's acceptability and accuracy by randomly distributing participants into one of the three conditions (i.e., It-Depends, DeepBits, JSON). Within each condition, participants were presented with a series of code components and asked to determine the existence of a vulnerability, any dependencies, and any mitigation steps.

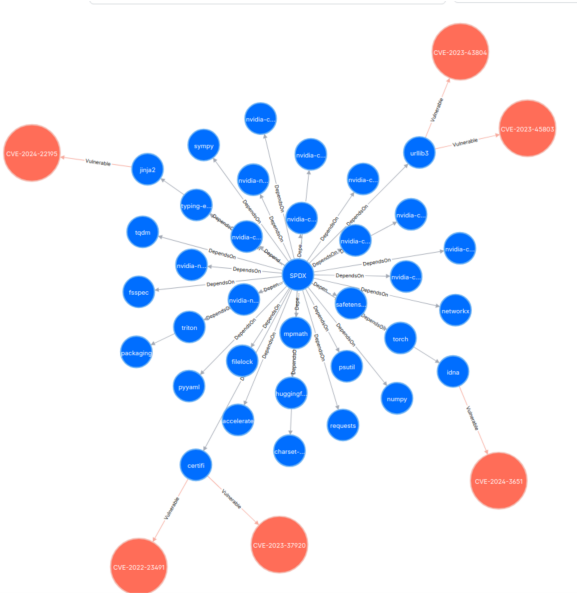


Fig. 6: A DeepBits generated graphic can show CVEs in code packages

2) Preliminary Results: This project is currently ongoing; however, a primary objective of this study is to determine whether visualizations enable SBOM usability. We have recruited 70 participants, who were randomly distributed to one of the three groups. All participants assigned to It-Depends (19) and DeepBits (22) completed the task. However, of the 29 participants assigned to the JSON file, only 17 completed the tasks. Both visualizations demonstrate how users can more efficiently engage with vulnerability information. Further analysis is needed to assess how these relate to performance and user experience. Additionally, the current results are based on 70 participants, where the majority of participants lack experience with SBOMs. As SBOMs become more common in software development, we anticipate more familiarity with SPDX's JSON file format.

3) Implications: The lower completion rate from the JSON condition indicates that users experience more cognitive load and frustration when working with less visual tools. In practice, this may delay the assessment of a vulnerability's scope and impact, prolonging exposure to an organization. However, while visualizations are a solution, this integration is still a challenge. With the complexity and necessity of tools to make SBOMs *usable*, is it realistic that SBOMs will ever be used? Only if they can be readily identified and integrated. When users are burdened with a vast number of fragmented SBOMs, regardless of format, they will never use them. Instead, having a comprehensive view of all the components and their dependencies creates usability.

C. Integrating SBOMs into Labels

Labels are designed to be used at the point of purchase, as it synthesizes technical information into a simple, graphical design. On the other hand, SBOMs, in their current form,

are not user-friendly, as they are designed for longer-term use to track vulnerabilities throughout the supply chain. Ideally, combining SBOMs with labels will strengthen the security of products, encouraging consumers to make more security-conscious choices over a product's lifecycle. The efficacy of both labels and SBOMs will be a function of their reliability and relevance. In this experiment, we explore which security features are most important to consumers and how they can be used to convey implicit aspects of an SBOM [8], [10].

1) Method: This experiment began with a study of security guidelines to gauge their efficacy in practice [14], [33], based on sources from the Federal Trade Commission (FTC) [11], National Highway Traffic Safety Administration (NHTSA) [35], Federal Bureau of Investigation (FBI) [17], Online Trust Alliance (OTA) [42], NIST [47], and Open Web Application Security Project (OWASP) [43]. The resulting union of the 131 best practices was 56 unique recommendations. As federal labeling efforts would primarily impact users within the federal acquisition system (i.e., suppliers, vendors, and buyers), we include additional federal guidelines [36]–[38], [46] and AI & ML factors [48].

We used these 73 security factors in a virtual card sorting exercise to identify insights into participant decision-making. The 66 participants were recruited and asked to identify the relative importance of a given factor for a specific purpose (e.g., supports transport encryption as a component of secure operations). Specifically, participants were presented with a security feature that was associated with one of five security categories (e.g., authentication, secure onboarding). Each feature was placed into one of four categories (i.e., very important, important, less important, or not important) according to the degree to which they would influence a participant's purchasing decision.

2) Results: As we assume consumers of SBOMs would have some technical literacy, we selected participants with some expertise in coding. In addition, previous work had found that only the more technically literate users would engage with the technical information presented on labels. We identify the attitudes towards a specific category using measures of central tendency and dispersion to rate the distribution of responses. Results from the study are aggregated into five separate security categories and represented in Figure 7. The initial identification of *top ten* items for label design are:

1) Sensitive personal information
2) Two-factor authentication
3) Brute force protection
4) Transport encryption
5) Standards compliance
6) Vulnerability process
7) Specialized hardware requirements
8) Encryption at rest
9) Required consent of data sharing
10) System backups

The study also aimed to identify the relationship between security factors and different groups. The results show only education and technical acumen had any significant correlation with any security factors or categories. And even then, it was only marginally influential between variables.

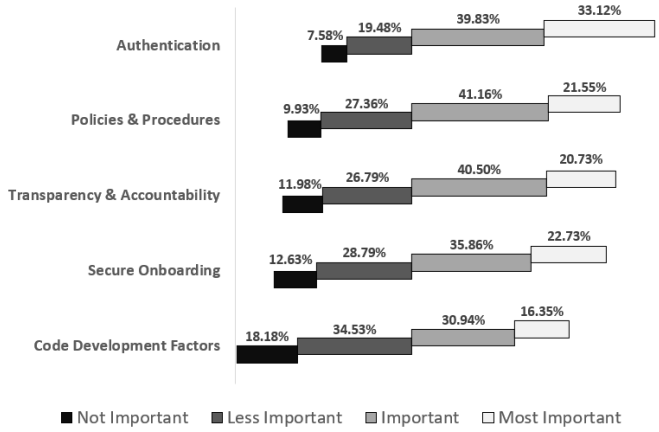


Fig. 7: Aggregate responses are categorized into security categories.

3) Implications: From this study, we can see that security labels, and to a higher degree, SBOMs, are a significant research challenge. However, SBOMs have significant upside if leveraged properly, extending beyond purchasing decisions to secure operations post-purchase. Ensuring secure operations involves integrating with threat modeling, mapping vulnerabilities to services, linking with the attack chain, and identifying mitigation options. A critical question then emerges: will companies invest in security? Knowledge of vulnerabilities is essential to evaluate appropriate return on investment, and sharing this information across organizations could increase investment within the ecosystem. The existence of SBOMs, as a mechanism of information sharing, increases investment simply by decreasing uncertainty. SBOMs should ideally enhance the situation by improving information flow. Next, we will determine if consumers would be willing to pay for more transparent security.

D. Willingness to Pay for Security

Willingness-to-pay is the maximum amount of money a consumer is willing to spend to acquire a good or service, denoting the value they place on a particular item [25]. Empirical results of multiple laboratory investigations illustrate that consumers will pay for security and demonstrate the importance of quality and brand to consumers [1], [7], [8], [12], [15], [22]. Additional research in willingness to pay for security and privacy indicates a greater willingness to pay for privacy [1], [22], [44]; however, security and privacy are complex, ever-changing, and often intertwined. Modern privacy includes dimensions that are clearly aligned with security: risk, integrity, and trust. Moreover, if privacy is the goal, security is the enabler. In this experiment, we seek to understand if consumers will pay more for increased security.

1) Method: To determine consumers' willingness to pay, we conducted a simulated purchasing experiment. We surveyed 599 participants and gave each $15, informing them

they would receive their product and any remaining funds at the end of the study. Participants were then divided into six experimental groups: five interventions based on different security indicators and one control with no indicators. Products were labeled with these security indicators to determine if they would influence choice. We utilized Amazon marketplace listings to provide external validity since these listings already have built-in economic and product trade-offs. Additionally, participants are more likely to authentically engage with these listings to process information across product descriptions, pricing structure, customer reviews, product ratings, and product features/designs. Since these listings mirror actual economic influences consumers face, representing more multi-dimensional decision dynamics, it allows us to better determine the impact of security in the marketplace.

2) Results: A key research question centers around whether security labels can increase a consumer's willingness to pay for a product. To assess this we evaluated the pricing for each label compared to the control and employed a Mann-Whitney U Test, which allowed us to compare the distributions of our samples to determine whether they were statistically different from the control. Given the design of our experiment, any deviation in spending between the labeled groups and the control group could reasonably be theorized to reflect more security-conscious decision-making, as security labeling was the only variable introduced. Our results indicate that in terms of general consumers, there are no statistically significant differences when compared to the control. This suggests that, overall, security labels do not drive a collective increase in participants' willingness to pay for more secure products.

However, when focusing on consumers who are already predisposed to caring about security, we saw an increase of 16.5% above non-security-aware consumers and 11.3% above the control. These results were statistically significant, indicating a willingness to pay among Security-Aware participants. The increase was less pronounced within the Privacy-Aware participants, and not statistically significant. This suggests that privacy concerns alone do not substantially elevate willingness to pay for security features.

Willingness to Pay

	SA	S&PA	PA
Non-Security-Aware	+16.5%	+22.0%	+4.4%
Control	+11.3%	+20.7%	+4.9%
Intervention	+10.4%	+19.3%	+3.4%

TABLE I: Percent increase for Security-Aware (SA), Privacy-Aware (PA), and Security&Privacy-Aware (S&PA) participants vs. Non-Security-Aware, Control, and Intervention participants. For example, SA participants were willing to pay +16.5% more for security compared to non-security-aware consumers.

3) Implications: We can apply three lessons learned from this experiment – the importance of simplicity, the need to communicate economic value to the consumer, and the role of familiarity. An effective awareness campaign could not only emphasize the significance of security but also provide clear and accessible information about the benefits and implications of cost savings. Over time, increased awareness and trust in security could significantly enhance the overall security market, as individuals would come to expect it. Communicating security's value proposition is critical as general consumers often do not understand underlying benefits and the economic trade-offs (i.e., higher upfront investment in security may save more money in the long term).

IV. Conclusion

In this paper, we reviewed how information asymmetry can lead to market failure. We explored the historical underpinnings of a lemons market, used to describe a used car market, and we applied a similar framework for new technologies. Studying this phenomenon highlighted that even in the digital age, where a vast amount of information is easily accessible, information asymmetry can persist. This persistence of a lemons market results in adverse consumer interaction, due to hidden or obscured information, ultimately deteriorating the overall quality of goods in the market. We reviewed two methods that could mitigate this: visualizing SBOMs and using SBOM information in security labels. In doing so, it should drive an increase in consumers' willingness to pay.

Ultimately an SBOM, supported by user-friendly interactions and the right tools, has the potential to foster a market for safe and secure software across its lifecycle, from development and verification to operations and purchase support. By providing transparent and easily consumable information about complex software, the SBOM becomes a catalyst for informed decision-making. This peels back the layers of the lemons market for any device that is reliant on software code as a foundation. Overall, the SBOM can correct how the marketplace interacts with stakeholders, aligning itself with established mental models.

Acknowledgment

We would like to thank Atmikha Jeeju for her assistance in reviewing this paper.

References

[1] Alessandro Acquisti, Leslie K. John, and George Loewenstein. What Is Privacy Worth? *The Journal of Legal Studies*, 42(2):249–274, 2013.

[2] George A Akerlof. The Market for "Lemons": Quality Uncertainty and the Market Mechanism. In *Uncertainty in Economics*, pages 235–251. Elsevier, 1978.

[3] Vafa Andalibi, Jayati Dev, DongInn Kim, Eliot Lear, and L Jean Camp. Is Visualization Enough? Evaluating the Efficacy of MUD-Visualizer in Enabling Ease of Deployment for Manufacturer Usage Description (MUD). In *Annual Computer Security Applications Conference*, pages 337–348, 2021.

[4] Aaron Bangor, Philip T Kortum, and James T Miller. An Empirical Evaluation of the System Usability Scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[5] Kevin Benton, L. Jean Camp, and Vaibhav Garg. Studying the Effectiveness of Android Application Permissions Requests. In *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 291–296, March 2013.

[6] L. Jean Camp, Shakthidhar Gopavaram, Jayati Dev, and Ece Gumusel. Lessons for Labeling from Risk Communication. In *Workshop and Call for Papers on Cybersecurity Labeling Programs for Consumers: Internet of Things (IoT) Devices and Software*, pages 1–3, Washington D.C., September 2021. NIST.

[7] Peter Caven, Zitao Zhang, Jacob Abbott, Xinyao Ma, and L. Jean Camp. Comparing the Use and Usefulness of Four IoT Security Labels. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery.

[8] Peter J. Caven, Jacob Abbott, and L. Jean Camp. Towards a More Secure Ecosystem: Implications for Cybersecurity Labels and SBOMs. In *TPRC 51*, pages 1–15, Rochester, NY, 2023. SSRN.

[9] Peter J Caven, Shakthidhar Gopavaram, Jayati Dev, and L Jean Camp. *SoK: Anatomy of Effective Cybersecurity Label Development*. SSRN, Rochester, NY, 2023.

[10] Peter J. Caven, Shakthidhar Reddy Gopavaram, and L. Jean Camp. Integrating Human Intelligence to Bypass Information Asymmetry in Procurement Decision-Making. In *MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM)*, pages 687–692, Rockville, MD, 2022. IEEE.

[11] Federal Trade Commission. FTC Report on Internet of Things Urges Companies to Adopt Best Practices to Address Consumer Privacy and Security Risks. January 2015.

[12] Lorrie Faith Cranor, Yuvraj Agarwal, and Pardis Emami-Naeini. Internet of Things Security and Privacy Labels Should Empower Consumers. *Communications of the ACM*, 67(3):29–31, 2024.

[13] Deepbits Technology. Generate, Distribute and Monitor SBOMs in One Platform. https://www.deepbits.com/sbom.

[14] Andrew Dingman, Gianpaolo Russo, George Osterholt, Tyler Uffelman, and L. Jean Camp. Good Advice That Just Doesn't Help. In *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 289–291, Orlando, FL, 2018. IEEE.

[15] Pardis Emami-Naeini, Janarth Dheenadhayalan, Yuvraj Agarwal, and Lorrie Faith Cranor. Are Consumers Willing to Pay for Security and Privacy of IoT Devices? In *In Proceedings of the 32nd USENIX Security Symposium*, pages 1–18, Anaheim, CA, 2023. USENIX Association.

[16] Winand Emons and George Sheldon. The Market for Used Cars: New Evidence of the Lemons Phenomenon. *Applied Economics*, 41(22):2867–2885, 2009.

[17] Federal Bureau of Investigation. IoT Poses Opportunities for Cyber Crime, September 2015. https://www.ic3.gov/media/2015/150910.aspx.

[18] Federal Communications Commission. Rosenworcel Announces Cybersecurity Labeling Program for Smart Devices, 2023.

[19] Federal Communications Commission. FCC Creates Voluntary Cybersecurity Labeling Program for Smart Products, 2024.

[20] Baruch Fischhoff. *Communicating Risks and Benefits: An Evidence Based User's Guide*. Government Printing Office, 2012.

[21] Sabine Glock, Simone Maria Ritter, RCME Engels, AJ Dijksterhuis, Rick Bart van Baaren, and Barbara Caterina Nadine Müller. 'Smoking kills' vs.'Smoking Makes Restless': Effectiveness of Different Warning Labels on Smoking Behavior. 2013.

[22] Shakthidhar Gopavaram, Jayati Dev, Sanchari Das, and L. Jean Camp. IoT Marketplace: Willingness-To-Pay vs. Willingness-To-Accept. In *20th Annual Workshop on the Economics of Information Security*, 2021.

[23] Andrei Hagiu and Julian Wright. Multi-Sided Platforms. *International journal of industrial organization*, 43:162–174, 2015.

[24] David Hammond, Geoffrey T Fong, Ann McNeill, Ron Borland, and K Michael Cummings. Effectiveness of Cigarette Warning Labels in Informing Smokers About the Risks of Smoking: Findings from the International Tobacco Control (ITC) Four Country Survey. *Tobacco control*, 15(suppl 3):iii19–iii25, 2006.

[25] W Michael Hanemann. Willingness to Pay and Willingness to Accept: How Much Can They Differ? *The American Economic Review*, 81(3):635–647, 1991.

[26] Allen D Householder, Garret Wassermann, Art Manion, and Chris King. The CERT Guide to Coordinated Vulnerability Disclosure. *Software Engineering Institute (Carnegie Mellon University). Disponible en https://bit. ly/3CSCaz5*, 2017.

[27] Daniel Kahneman and Amos Tversky. Prospect Theory: An Analysis of Decision Under Risk. In *Handbook of the fundamentals of financial decision making: Part I*, pages 99–127. World Scientific, 2013.

[28] Jae-Cheol Kim. The Market for "Lemons" Reconsidered: A Model of the Used Car Market with Asymmetric Information. *The American Economic Review*, 75(4):836–843, 1985.

[29] Konrad Kollnig, Anastasia Shuba, Max Van Kleek, Reuben Binns, and Nigel Shadbolt. Goodbye Tracking? Impact of iOS App Tracking Transparency and Privacy Labels. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 508–520, New York, NY, USA, 2022. Association for Computing Machinery.

[30] Nancy Leveson et al. Medical Devices: The Therac-25. *Appendix of: Safeware: System Safety and Computers*, 1995.

[31] Jonathan Levin. Information and the Market for Lemons. *RAND Journal of Economics*, pages 657–666, 2001.

[32] Behnood Momenzadeh and Jean Camp. Peeling the Lemons Problem with Risk Communication for Mobile Apps.

[33] Behnood Momenzadeh, Helen Dougherty, Matthew Remmel, Steven Myers, and L. Camp. Best Practices Would Make Things Better in the IoT. *IEEE Security & Privacy*, PP, May 2020.

[34] Behnood Momenzadeh, Shakthidhar Gopavaram, Sanchari Das, and L Jean Camp. Bayesian Evaluation of User App Choices in the Presence of Risk Communication on Android Devices. In *International Symposium on Human Aspects of Information Security and Assurance*, pages 211–223. Springer, 2020.

[35] National Highway Traffic Safety Administration. Cybersecurity Best Practices for Modern Vehicles. *Report No. DOT HS*, 812:333, 2016.

[36] National Information Assurance Partnership. Protection Profile for Application Software. Technical report, NIAP, October 2021.

[37] National Institute of Standards and Technology. Recommended Criteria for Cybersecurity Labeling for Consumer Internet of Things (IoT) Products. Technical report, U.S. Department of Commerce, February 2022.

[38] National Institute of Standards and Technology. Recommended Criteria for Cybersecurity Labeling of Consumer Software. Technical report, U.S. Department of Commerce, February 2022.

[39] National Institute of Standards and Technology. National Vulnerability Database (NVD) Statistical Summary. Technical report, 2024.

[40] National Institute of Standards and Technology. NVD News. Technical report, 2024.

[41] National Telecommunications and Information Administrator. Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM). 2020.

[42] Online Trust Alliance. OTA Internet of Things, accessed April 2019. https://www.internetsociety.org/ota/.

[43] Open Web Application Security Project. OWASP IoT Project, accessed April 2019. https://www.owasp.org/index.php/IoT_Security_Guidance.

[44] Prashanth Rajivan and Jean Camp. Influence of Privacy Attitude and Privacy Cue Framing on Android App Choices. In *Authentication Workshop of the 12th Symposium on Usable Privacy and Security*, Denver, CO, 2016. USENIX Association, USENIX Association.

[45] Prashanth Rajivan and L. Jean Camp. Influence of Privacy Attitude and Privacy Cue Framing on Android App Choices. In *Twelfth Symposium on Usable Privacy and Security*. USENIX Association, 2016.

[46] Secure Communications Alliance and IoT PP Working Group. IoT Secure Element Protection Profile (IoT-SE-PP). Technical report, Smart Communication Alliance, December 2019.

[47] Adam Sedgewick. Framework for Improving Critical Infrastructure Cybersecurity, Version 1.0. 2014.

[48] SPDX AI ML Working Group. SPDXv3 AI SBOM, 2022. https://lists.spdx.org/g/spdx-ai.

[49] Lara Stocchi, Naser Pourazad, Nina Michaelidou, Arry Tanusondjaja, and Paul Harrigan. Marketing Research on Mobile Apps: Past, Present and Future. *Journal of the Academy of Marketing Science*, pages 1–31, 2022.

[50] Kamala Swayampakala, James F Thrasher, David Hammond, Hua-Hie Yong, Maansi Bansal-Travers, Dean Krugman, Abraham Brown, Ron Borland, and James Hardin. Pictorial Health Warning Label Content and Smokers' Understanding of Smoking-Related Risks—a Cross-Country Comparison. *Health education research*, 30(1):35–45, 2015.

[51] National Telecommunications and Information Administration. Software Suppliers Playbook: SBOM Production and Provision. 2021. https://www.ntia.doc.gov/report/2021/minimum-elements-software-bill-materials-sbom.

[52] The White House. Biden-Harris Administration Announces Cybersecurity Labeling Program for Smart Devices to Protect American Consumers.

[53] Trail of Bits. It-Depends. https://github.com/trailofbits/it-depends.

ABSTRACTS

A Systems Engineering Approach to Policy Analysis: Addressing the Need for a Cyber Backstop

Jenna McGrath, Michael Grappone
LLNL

This report presents the process of designing a cyber backstop policy tool system. A cyber backstop provides guidance and requirements for the federal government and private reinsurance and insurance industry stakeholders to share the financial burden related to a catastrophic cyber attack. The proposed backstop framework is modeled after existing successful terrorism and natural hazard backstop programs. The backstop considers the unique requirements that arise with cyber threats; for example, the difficulty of attribution, the cascading consequences, and the quickly evolving nature of sophisticated cyber attacks.

Despite a growing cyber insurance market, a catastrophic cyber event targeting critical infrastructure would impact more cyber risk policy holders than can feasibly be covered by the insurance and reinsurance markets. Additionally, many (re)insurers do not cover acts of war, with the reasoning being that attacks of this nature are too costly to insure and the potential losses from such an event would be unsustainable for the (re)insurance industry. To provide adequate coverage for cyber catastrophes, a cyber backstop framework is required to determine how to share the burden of an extremely costly catastrophic cyber event. This cyber backstop will provide guidance into what portion of the recovery from the event is covered by private insurance and what portion will require government-backed financial intervention.

Our report analyzes a series of cyber backstop system designs. We use existing policy tools and insurance backstop programs as examples and models in designing our backstop system. These programs include: the Terrorism Risk Insurance Program (TRIP), implemented after the September 11th attacks under the Terrorism Risk Insurance Act (TRIA); the National Flood Insurance Program (NFIP) which aims to provide federal flood insurance to regions where private insurance is unavailable; and the United Kingdom-based PoolRe program, where private insurers are billed a premium in order to provide a buffer threshold of funds to cover insured losses from a terrorist attack.

We consider various pathways in designing our policy tool, including (1) an entirely new insurance concept, (2) adapting an existing framework, and (3) creating a new program based on an existing framework.

We use these acceptance criteria: (1) our system shall functionally handle the needs of all stakeholders, (2) cover every type of relevant cyber incident, and (3) maintain records of all interactions with the system. After multiple iterations to define and select our concepts and system context, we conclude that the best path forward is to design a new policy tool using an existing program's framework (specifically, TRIA/TRIP).

We demonstrate how our system could process various examples of operational scenarios and use cases. We define the system's requirements and provide a system architecture to demonstrate the tool's implementation. We address various risks to the system through mitigation efforts.

We conclude that the cyber security backstop we have described can provide a policy tool to successfully address the growing risks of cyber attacks and the financial burden, uncertainty, and risk they place on governments and private insurance.

Acceptability and Accuracy with SBOM Data and Visualizations

Xinyao Ma, Peter Caven, Zitao Zhang, Ambarish Gurjar, L. Jean Camp
Indiana University Bloomington

The Software Bill of Materials (SBOM) enables transparency throughout the supply chain. SBOMs have the potential to support actionable, timely, usable risk communication based on dependency information and data in the larger vulnerability ecosystem. With effective risk communication SBOMs can support risk-aware decision-making as code is authored and altered. Toward this goal, the Linux Foundation developed training materials for the software package data exchange (SPDX) SBOM generator. Similarly, the CycloneDX working group has training resources for its own generators. Beyond training, multiple visualizations have been developed. Our research compares the straightforward provision of SBOM data with two popular open-source visualization tools: ItDepends and DeepBits.

We implemented a human subjects experiment to determine the degree to which visualizations enabled our participants to accurately and efficiently identify the existence of vulnerabilities and their mitigation. (All human subjects research is approved by the IRB.) We asked participants to answer questions about specific real-world code using either one of the two visualizations and the raw SBOM provided in a JSON format. The JSON script was generated with SPDX. The purpose of this research is to answer the following research questions:
Q1: Do participants improve their efficacy when using visualizations?
Q2: How does any change in efficacy vary between different visualizations?
Q3: How usable are the visualizations?
Q4: Are participants' perceptions of their accuracy aligned with their accuracy? How does this vary between visualization and raw JSON?

\We have recruited 69 participants who either have a computer science background or coding experience. Of these, 49 stated that they were familiar with or had knowledge of SBOMs. All the participants were randomly distributed to one of the three conditions by the Qualtrics survey platform. In each condition (ItDepends, DeepBits, JSON), the participants were presented with a series of code components, one at a time. For each component, they were asked to identify a vulnerability and the dependencies that contained it. They were asked to determine if a different component contained any vulnerabilities, and if so, what these vulnerabilities were. Once this was completed, the participant was asked how to mitigate the vulnerability. In order to make the task feasible and ensure all participants were working from the same information base, every participant was provided a one-click link to the description and the mitigation of every vulnerability in the experiment. We then inquired about their experience using the standard NSA task load index.

We propose to present on the accuracy of participants, their perception of accuracy, as well as the reported usability. We propose to present a detailed description of the experiment design and a completed analysis of the results. Here we can only provide preliminary observations from current data. Our current results are that participants interacting with the ItDepends visualization are the most accurate. The accuracy across the three tasks was not uniform. The JSON format provided the least accuracy, unsurprisingly as it was the least human readable. In fact, 12 out of 29 participants did not complete the tasks after being randomly allocated to the JSON

experimental group. Conversely, those completing the tasks using JSON ranked its usability as higher than those using and ranking the visualizers.

The experiment is on-going. We are recruiting additional participants. It will be closed before the event. Based on the responses of the reviewers and organizers, we could demonstrate or invite participation. We are open to conversations about engaging with interested participants in think-aloud inperson walkthroughs of the tasks or collaborating to customize tasks for their organizations.

Future work includes design of effective visualization, with the goal of integrating more inclusive information and targeting it based on developer expertise. We will also evaluate the accuracy and consistency of the tools. We designed our research to focus on usability and acceptability. However, there was significant variance in the displays of what should ideally be the same data across the three presentations. Different visualizations identified different vulnerabilities. Neither the documentation of the generator nor the visualization tool provided clear explanations for the difference.

An Introduction to the Federal Acquisition Security Council (FASC)

Sangeetha Ranadeeve
DHS CISA

The Federal Acquisition Security Council (FASC) was established by the Federal Acquisition Supply Chain Security Act of 2018 in response to the need to ensure a government-wide approach to evaluate threats and vulnerabilities in government Information and Communications Technology and Services (ICTS) supply chains and address supply chain security risks. Chaired by OMB, the FASC is an interagency council with representatives from General Services Administration; Department of Homeland Security (DHS); Office of the Director of National Intelligence (ODNI); Department of Justice; Department of Defense (DOD); and Department of Commerce. This presentation provides an overview of the FASC, its authorities and functions, and the responsibilities of the Information Sharing Agency (ISA) – CISA on behalf of DHS.

Breaking Down the IIoT Cyber Labeling Effort for US Cyber Trust Mark Applied to Smart Meters and Solar Inverters

Ian Johnson, Animesh Pattanayak
PNNL

IoT devices were once an edge of technology, found only in the homes of the techy savvy individuals. Today, IoT devices have permeated into electronics, appliances, and consumer products. Most homes have an IoT device whether a smart TV or a voice assistant device like Amazon Alexa or Google Home. These devices, while convenient and entertaining, bring security concerns associated with the increased number of devices connected to one's home network.

A July 2023 press release from the White House announced a new Cybersecurity Labeling Program for Smart Devices. In this release, the Biden-Harris administration outlines the plan for the FCC's proposed U.S. Cyber Trust Mark to be incorporated into this plan to strengthen cybersecurity and improve privacy for individuals utilizing IoT devices in their home.

In support of this effort, over the course of the last year, a team of researchers, engineers, and analysts from six DOE National Laboratories have been investigating the applicability of a labeling program applied to IIoT. This research will be particularly pertinent to energy sector OEMs who manufacture IIoT products and end-users of these products, including both energy sector asset owners and home enthusiasts.

The intent of a cybersecurity label applied to IIoT is to provide information that could inform consumer choice and awareness when purchasing, installing, and operating internet connected energy devices. In this presentation, we aim to break down the following components of our research process:

- Research performed to understand existing standards applicable to a label for consumer-facing energy infrastructure along with voluntary input from vendor partners
- Development of initial list of proposed label elements
- Receipt of feedback through stakeholder workshops and
- Government coordination with key stakeholder offices including NIST, FCC, and SETO
- Receipt of public feedback and incorporation into our recommendations
- Ongoing process of revising the proposed label elements to better accommodate public feedback and vendor feedback – close alignment of initial label elements with NIST IR 8259 series
- Considerations taken during design and development of our recommendations
- Deep dive into the proposed recommendations to be delivered (the final report is still in progress but will be complete by end of FY24).

Closing the Visibility Gap in Critical Systems Software Supply Chains

Derek McCarthy
NetRise

This presentation will explore the critical software visibility gap in today's supply chains, especially within critical systems. We'll discuss how a lack of transparency in software components can expose organizations to significant vulnerabilities and risks. This presentation will highlight issues such as the age of software components, presence of thousands of known vulnerabilities in brand new firmware/software and identification and analysis of other supply chain artifacts such as credentials, misconfigurations and cryptographic material. Using real-world data, we will illustrate how addressing this visibility gap can bolster security measures and reduce the risk of supply chain attacks. Artifacts covered in the presentation will include, but not be limited to: industrial control systems, telecommunications equipment, server firmware, critical windows applications, containers and virtual machines.

DOD Product Assurance Playbook Process for Commercial-Off-The-Shelf Products

Cassie Crossley
Schneider Electric

The Product Assurance Playbook process, which is in design with the DOD CIO office, Exiger, Schneider Electric, and The Chertoff Group, evaluates the supply chain for commercial-off-the-shelf (COTS) products. This presentation describes the phases of the playbook, as well as details the method for examining the software bill of materials (SBOMs), hardware bill of materials (HBOMs), vendor development, and vendor manufacturing. Risks associated with the product such as foreign-owned controlling interest (FOCI) and tier X (4th+ party) suppliers are identified so the vendor can document mitigations and/or plans of action. To highlight the differences between the playbook and the CyTRICS process, an example of a COTS smart power meter that was piloted through this process will be shown.

Leveraging SBOMS for Vulnerability Management

Cassie Crossley
Schneider Electric

There is some debate as to how SBOMs can enhance vulnerability management practices, and some believe that collecting SBOMs from internal teams or suppliers is too difficult and time-consuming. Learn how Schneider Electric has collected thousands of our product SBOMs and how we are leveraging the SBOMs as part of our corporate product CERT to quickly analyze and focus our attention when time is of importance. This presentation describes how we modified our policies and processes to collect, generate, and store thousands of SBOMs. You will hear how we have leveraged SBOMs during the Log4j and OpenSSL vulnerability events. Then we will conclude with key learnings, suggestions, and opportunities for improvement.

PHICS: Programmable Hardware Image Collection System

C. Weitz, G. Gloria, M. Kirkland
PNNL

The increasing scrutiny of supply chain of critical infrastructure has made the development of Hardware Bill of Materials (HBOMs) an appealing solution for identification of risk. However, HBOMs are not currently provided by vendors and manufacturers of control system equipment at large. The lack of first-party HBOMs has created a vacuum of quality supply chain data in the hardware that comprises key energy-critical infrastructure. Filling this gap requires third-party HBOMs to be generated until vendor-provided HBOMs are available.

Third-party enumerations require identifying the electronic components that comprise the critical infrastructure technology in question. This usually requires physical disassembly and photographing of the device, a heavily manual process that does not scale easily. Other solutions avoid physical disassembly by utilizing X-ray, CT, and Ultrasound imaging technology to identify components. These solutions require expensive equipment, trained staff, and may have other technical limiting factors (i.e. limited sample size, struggle with shielded integrated circuits, etc.).

PHICS offers to solve the burden of manual, time-consuming enumeration by providing open-source, user-friendly, inexpensive hardware that can reduce the burden of manual enumeration photography and identification. This is just a prototype version designed by a Computer Numerical Control (CNC) - controlled routing table modified to utilize a digital microscope.

PHICS utilizes a combination of CNC via the Mostly Printed CNC project with a commercial Dino-Lite digital microscope. The software is created using Python to interface the hardware and perform semi-automated enumeration.

The digital microscope has been made fully functional and software controllable. As a result of finishing the functionalities in the microscope, PHICS has produced detailed imagery from commonly difficult component enumerations and has significantly decreased the amount of time necessary for recording traditionally lengthy enumerations.

Software Composition Analysis Tools: SCRM Value Add or Lossy Noise Machines

Robert Erbes, Micaela Gallegos
INL, LLNL

Software supply chain risk management (SCRM) depends upon accurate information regarding the software components that comprise any given software system. The collection of components included in a software package can be organized within a software bill of materials, or SBOM. SBOMs are ideally generated when the software components are put together, such as at compile time, but for many reasons that has not and is not always possible. For example, legacy or proprietary software packages often do not have SBOMs available to downstream consumers of that software. It's not just end users that are affected, manufacturers themselves also must deal with this problem.

To answer these questions, the market has seen the rise of several commercial software composition analysis (SCA) tools. These tools aim to peer into completed software systems, automatically identifying hidden software dependencies and looking up known vulnerabilities associated with those dependencies to enable end-users to enhance their cyber supply chain risk management processes. These tools are potentially a huge boon to end users of legacy and proprietary software – and a potential bane, depending on how accurate they are.

This research asks that question – how accurate are currently available binary SCA tools – and provides answers to several other questions: What does it mean to be "accurate"? What limitations do the tools have in identifying common edge cases that take place in modern software development? Can they help you avoid a devastating supply chain attack, or is it all just noise?

After researching SCA tools on the market, we identified three vendors that fit our use case and would provide analysis on compiled binaries. Using these tools, we submitted firmware for critical infrastructure devices for analysis and SBOM generation. The SBOM outputs were then cross referenced with SBOMs generated through manual analysis for comparison. In addition to the firmware samples, we also submitted edge case samples based off a popular open-source library that were specifically crafted to evaluate each tools' ability to accurately identify components. These samples were customized to be consistent with modifications we have seen in modern software development as well as a couple that are representative of supply chain attacks.

In the end, we found that accuracy in component identification varied across tools and edge case difficulty. To learn more, be sure to attend our talk!

The Current State of C-SCRM

Nikkia S. Henderson
DHS CISA

Nikkia Henderson will be providing an overview of C-SCRM. This presentation will serve as an opportunity to continue conversations around C-SCRM challenges and topics and explore ways to address these. It will explore the life cycle of previous incidents and uncover future considerations that could be taken to increase resiliency in our supply chains. The audience will be a deepened understanding of C-SCRM, current vulnerabilities and challenges, and left with a call to action to how to address these. Additionally best practices such as increased collaboration, holistic adoption, and increased collaboration will be strongly encouraged post conference attendance. Participants will be left with thought provoking ideas to operationalize C-SCRM and methods to implement and mature internal C-SCRM programs and initiatives.

The Software Supply Chain Business Case

Duncan K. Sparrell
sFractal Consulting

Building trust in critical digital systems costs money. Or does it save money? The definition of a business case is “a justification for a proposed project or undertaking on the basis of its expected commercial benefit”. This presentation will help the audience understand, and quantify, the commercial benefits of understanding your software supply chain, as well as the work, the tools, and the costs necessary to do so. The author will begin by describing the threats to critical systems and the risks of not addressing them – including how to quantify those risks for your particular business. Use cases will be reviewed, building on the work in the National Telecommunications and Information Administration (NTIA) and Cybersecurity and Infrastructure Security Agency (CISA) working groups. Example use cases will be shown from vulnerability management, licensing, regulatory compliance, and end of life; including how to leverage across all these use cases to build your business case; and how to craft the business case in terms a Board would understand and relate with. Work in various standards development organizations (SDO’s) will be reviewed, including reviewing the alphabet soup of acronyms in this space as well as various recent events such as SBOMarama and the Cybersecurity Automation Village. But understanding our supply does take investment in both staff and money. Various costs will be reviewed that must be taken into account, and tradeoffs among alternatives. The presentation will conclude by summarizing the elements to your business case for establishing the right level of investment to establish the right level of trust for your particular business.

CySCRM '24