Threat Modeling for SoC Security Design: IEEE P3164 SA-EDI Standardization

Wenzhen Li, IEEE P3164 working group

***Abstract*-The importance of security design for a System-on-Chip (SoC) is well known. For SoC, security design and threat modeling are integrated, proactive processes for building resilient systems that anticipate and mitigate attacks before the product is deployed. Threat modeling systematically identifies vulnerabilities, while secure design principles and features are implemented to counter them throughout the chip's lifecycle. In this paper, we introduce the latest progress in IEEE P3164 standardization on Security Annotation for Electronic Design Integration (SA-EDI), which provides a series of specifications for the threat modeling to facilitate SoC security design and integration catering to the emerging trends in the landscape of security threats and SoC architectures.**

I. INTRODUCTION

System-on-Chip (SoC) architectures have become the foundation of modern computing platforms, whose deployment spans mobile devices, automotive systems, industrial automation, and edge/cloud computing. SoC enables high integration of heterogeneous components such as CPUs, GPUs, accelerators, and memory subsystems on a single chip. This integration improves performance and efficiency, it also expands the attack surface, exposing tightly coupled hardware and software subsystems to increasingly sophisticated threats. Moreover, the rise of pervasive IoT deployments, the transition to 5G-enabled networks, and the reliance on AI-driven workloads introduce unprecedented requirements for data confidentiality, integrity, and system resilience. At the same time the integration of artificial intelligence (AI) and the rise of quantum computing also introduces new risks that challenge traditional security models. Consequently, SoC security has shifted from a feature set to a primary design requirement, demanding systematic approaches that integrate threat modeling and secure SoC design.

Threat modeling for a SoC uses a structured process to analyze the system architecture and identify potential security flaws. This process typically involves the following steps:

1. *Identify critical assets and attack surface:* Determine what sensitive data or functions need protection and map all potential entry points, including hardware and software interfaces, communication buses, and debug ports.
2. *Define threat actors and scenarios:* Consider who might attack the system (e.g., competitors, organized criminals, hobbyists) and what their motivations are (e.g., IP theft, data exfiltration, device hijacking).
3. *Use threat modeling frameworks:* Apply established methodologies to analyze potential threats. Common frameworks include STRIDE [5], PASTA [6], Attack Trees [7], and MITRE [8] etc. Specifically, Intel and other SoC design companies practice their own threat modeling frameworks for SoC design [9] with a series of EDA tools.
4. *Risk analysis:* Based on their likelihood and potential impact, rank the identified threats to focus on the most critical risks.
5. *Develop countermeasures:* Define specific design and implementation strategies to mitigate the prioritized threats.

All these available threat modeling frameworks provide schemes to help reason and find threats to a system; or a specific multi-step methodology and process for aligning business objectives and technical requirements; or even provide conceptual diagrams showing how an asset might be attacked. They are generally applicable to computer and information systems. They can also be extended to SoC domain. More relevantly, e.g. Intel's SoC threat modeling is a specific in-house workflow and framework for SoC security design.

Nowadays the semiconductor industry relies heavily on the supply chain. Attackers are increasingly targeting the semiconductor supply chain, exploiting the growing reliance on third-party components and EDA toolchains. On this emerging SoC security landscape, IEEE P3164 SA-EDI standardization is working for a holistic inter-operable high-level framework and introducing a security-centric design flow along with relevant specifications for all parties involved in SoC design and manufacture. This framework facilitates the building of an inter-compatible SoC security design eco-system by standardizing a uniform security design workflow particularly for IP providers, IP integrators and EDA vendors. IEEE P3164 SA-EDI is an open standard. It clearly specifies the boundaries and interfaces of IP provider and IP integrators. Within the specific boundary, all open source or proprietary threat modeling and workflow can be reused and annotated. The standard also defines rich data objects of assets and threats, which can be used by EDA vendors to develop toolchains for SoC security design and verification automation. The standard can also be exploited by AI and ML tools for complicated circuit/system analysis to identify and exposure the potential threat that would evade traditional security measures.

In DVCON 2024 [2], Miltos provided a detailed introduction to IEEE P3164 and SA-EDI standard, which at that time was still in its infant stage and focused on Asset Identification [3] for the Security Assurance collaterals of an internal or third-party IP used in the SoC. Later IEEE P3164 working group further refines the standard and specifies the data objects of asset, threat and interfaces of IP provider and integrator. The standard also provides the uniform methodology, framework and workflow for the creation of a threat model at different integration levels of SoC security design. The specific threat model can be created by examining the Weakness and Threat data objects. The created model should be analyzed and validated by the integrator to enable a secure SoC design.

This paper is organized as follows. Section I is the introduction of threat modeling to secure a SoC Design, and the overview of the unique features of IEEE P3164 SA-EDI Standard catering to the new trends in the landscape of security threats and SoC architectures. Section II provides a full picture of emerging threats and vulnerabilities in SoC security design. Section III presents the framework and workflow of threat modeling particularly for SoC security design. The details of IEEE P3164 standard progress on threat modeling for SoC security are presented in Section IV. The final section summarizes the new trends in SoC security design and the future work of IEEE P3164 SA-EDI standard.

II. Emerging threats and vulnerabilities to SoC security

Emerging threats and vulnerabilities for System-on-Chip (SoC) security are driven by increased complexity, supply chain reliance, and advanced hacking techniques. The integration of artificial intelligence (AI) and the rise of quantum computing also introduce new risks that challenge traditional security models.

Supply chain compromises: Attackers are increasingly targeting the semiconductor supply chain, exploiting the growing reliance on third-party components and fabrication services.

- *Hardware Trojans:* Malicious modifications can be inserted at any stage of manufacturing. Sophisticated, dormant Trojans are extremely difficult to detect with traditional testing because they may only be triggered under specific, rare conditions.
- *Counterfeiting and IP theft:* Counterfeit or overproduced chips can be introduced into the market, and design intellectual property (IP) is a high-value target for theft. Counterfeit ICs may contain unknown vulnerabilities.
- *Design tool manipulation:* Attackers can compromise Electronic Design Automation (EDA) tools to insert malicious code or introduce vulnerabilities into the final chip design, a risk often compounded by the reuse of unverified IP from third-party libraries.

AI and machine learning (ML) threats: The integration of AI into SoCs for applications like IoT and edge computing introduces a new set of vulnerabilities.

- Adversarial attacks: Adversaries can intentionally manipulate input data to cause AI models to produce incorrect or malicious outputs. For example, a small modification to an image could be used to fool a computer vision system.
- Data poisoning: Attackers can inject malicious data into an AI model's training set, leading to biased, compromised, or exploitable behavior. For a SoC, this could mean compromising the security of a smart home or automotive system.
- Increased attack surface: The immense complexity of AI and ML code increases the potential surface area for attacks and expands the timeframe during which vulnerabilities can develop.

Post-Quantum Cryptography (PQC) challenges: The arrival of quantum computers poses a long-term threat to current cryptographic standards.

- "Harvest Now, Decrypt Later" attacks: Adversaries may steal and store encrypted data now, intending to use a future quantum computer to decrypt it. This is particularly dangerous for SoCs in long-life embedded systems like cars and infrastructure.
- Implementation difficulties: New PQC algorithms are more complex and computationally intensive, creating performance challenges for resource-constrained embedded systems. Developers must balance cryptographic robustness with efficiency.
- New vulnerabilities: PQC algorithms introduce new security challenges, including different side-channel and fault injection attack vectors, requiring specialized countermeasures during implementation.

Firmware and software exploits: While not new, firmware and software vulnerabilities continue to evolve, especially with the democratization of IoT for personal/family, communities and industry.

- Evolving malware and ransomware: Attackers are using increasingly sophisticated malware and ransomware tailored for embedded systems, often targeting vulnerable or end-of-life devices.

- Third-party and legacy component risk: Many SoCs incorporate IP and code from third parties or use legacy systems that may contain unpatched, known vulnerabilities. Misconfiguration of these components is a significant risk.
- AI-powered attacks: AI is increasingly used to automate and enhance attacks, such as creating new ways to bypass traditional security measures.

Architectural and physical attacks: Advanced attacks continue to find ways to bypass traditional protection mechanisms.

- Microarchitectural attacks: These attacks exploit vulnerabilities in CPU architecture. For example, speculative execution attacks which exploit processors performance optimization to leak sensitive data.
- Row-hammer attacks: Repeatedly accessing a memory row can cause bit flips in adjacent rows, potentially allowing attackers to change data in other processes or virtual machines.
- Side channel attack/Fault injection: Attackers can conduct cryptographic key extraction by monitoring the power/temperature patterns/maps of the SoC or use lasers or electromagnetic pulses to induce faults during chip operation to bypass security mechanisms.

III. THREAT MODELING FRAMEWORK FOR SOC SECURITY DESIGN

The three extensively deployed threat modeling frameworks in the computer system are STRIDE, PASTA, and Attack Trees. The entry to the details of all these frameworks can be found in [12]. **STRIDE** [5] is a model for identifying computer security threats developed by Microsoft. It classifies security threats in six categories. STRIDE is used to help reason and find threats in a system. It is used in conjunction with a model of the target system that can be constructed in parallel. This includes a full breakdown of processes, data stores, data flows, and trust boundaries. **PASTA** [6] is a risk-centric methodology. It provides a seven-step process for aligning business objectives and technical requirements, taking into account compliance issues and business analysis. The intent of the method is to provide a dynamic threat identification, enumeration, and scoring process. Once the threat model is completed, security subject matter experts develop a detailed analysis of the identified threats. Finally, appropriate security controls can be enumerated. This methodology is intended to provide an attacker-centric view of the application and infrastructure from which defenders can develop an asset-centric mitigation strategy. **Attack Trees** [7] are conceptual diagrams showing how an asset, or target, might be attacked. Attack trees have been used in a variety of applications.

All these models can be extended to the SoC security design considering the new trends in hardware security for SoC design, i.e., modularization of chip-let architectures, hardware-level protection of data in use, and AI/ML hardware acceleration etc. In Table 1, we specifically compare these frameworks under SoC security usage scenarios.

TABLE1. Comparison of threat modeling frameworks for SoC security

Framework	Core Focus	SoC-Specific Application	Best Used When
STRIDE	General application security: **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service, **E**levation of privilege.	Useful for modeling threats related to data flows and interfaces, such as firmware updates and communication protocols.	Starting the threat modeling process for a new feature or component.
PASTA	Risk-centric, attacker-focused methodology spanning seven stages. Integrates business objectives with technical analysis.	Provides a strategic, high-level view of how business and technical risks intersect in an SoC design, considering attacker motivation.	Integrating threat modeling into a broader risk management strategy.
MITRE ATT&CK	Real-world adversary tactics and techniques. Offers a detailed, behavior-focused view.	Mapped to the specific TTPs (Tactics, Techniques and Procedures) used to exploit firmware, devices, and systems controlled by a SoC.	Conducting a deep, intelligence-driven analysis of adversary behavior.
MITRE EMB3D	Critical infrastructure and embedded systems, building on ATT&CK.	Identifies threats specific to the unique hardware, software, and physical interfaces of embedded systems.	Designing an SoC for critical infrastructure, where the system itself is a high-value target.
Attack Trees	Visual, goal-oriented representation of attack paths.	Used to model multi-stage physical and logical attacks, such as combining a software vulnerability with a side-channel attack.	Analyzing specific, high-risk attack scenarios in detail.

IEEE P3164 working group closely works together with MITRE (Row 3 and Row 4 in Table 1), whose list of hardware CWEs (common weakness enumeration) [8] is being used as a case-study example for the SA-EDI standard development. While it should be noted that IEEE P3164 SA-EDI is an open standard, and all threat modeling frameworks in Table 1 or other frameworks can be easily adopted under its high-level, well-defined interface specification umbrella.

It's worth noting that Intel's SoC threat modeling is an in-house workflow and framework specifically to secure the SoC design. They tailored the framework to their specific needs for hardware, firmware, and platform development in addition to software development. Under the threat modeling framework, the adversary models (or threat models, exchangeable in this paper) are referred to the specific attack tactics/techniques and the specific behaviors employed by adversaries to achieve the adversary goals. Intel has developed 9 adversary models, which are shown in Table 2. It also has adopted a series of open-source and in-house security tools to identify vulnerabilities at scale and help ensure third-party/open-source software meets the security requirements [11].

TABLE 2. INTEL HAS DEVELOPED NINE ADVERSARY MODELS FOR SOC DESIGN [9]

Adversary	Description
Unprivileged Software Adversary	Typically known as a "user-space" adversary; capabilities are limited by the instruction set architecture (ISA), or the hardware platform (x86/x64 or IA-32/Intel 64), or the capabilities granted by the system software.
System Software Adversary	Full control over the operating system, or virtual machine monitor. This adversary can manipulate x86/x64 in any manner allowed by the instruction set architecture specification.
Startup code and SMM Adversary	All capabilities of the System Software Adversary, as well as control over initial boot code and system management mode. This adversary can manipulate x86/x64 in any manner allowed by the instruction set architecture specification. This adversary also has the ability to compromise system and platform firmware.
Network Adversary	Access to and may have control over various network fabrics that are used to connect the platform to other platforms, intranet, or extranet resources. This adversary can also interact with remote systems through predefined APIs.
Software Side Channel Adversary	Able to gather statistics from the CPU regarding execution and may be able to use them to extract secrets from software being executed. This adversary can also observe hardware resource usage to infer information and secrets from software being executed. This adversary can often directly influence resource usage (e.g., by causing contention) or by modulating an input to a victim program.
Simple Hardware Adversary	Physical access to the system, which typically doesn't require expensive equipment or extraordinary training/specialty.
Skilled Hardware Adversary	Physical access to the system and additional equipment and/or training that isn't accessible to the average individual consumer.
Hardware Reverse Engineer Adversary	Physical access to the system, specialized tooling (which can be rented), and highly specialized expertise.
Authorized Adversary	Intel or partner-granted authority that has capabilities not available to unauthorized entities. This may include access to manufacturing facilities and systems, access to design facilities and systems, or access to devices that haven't completed all manufacturing steps.

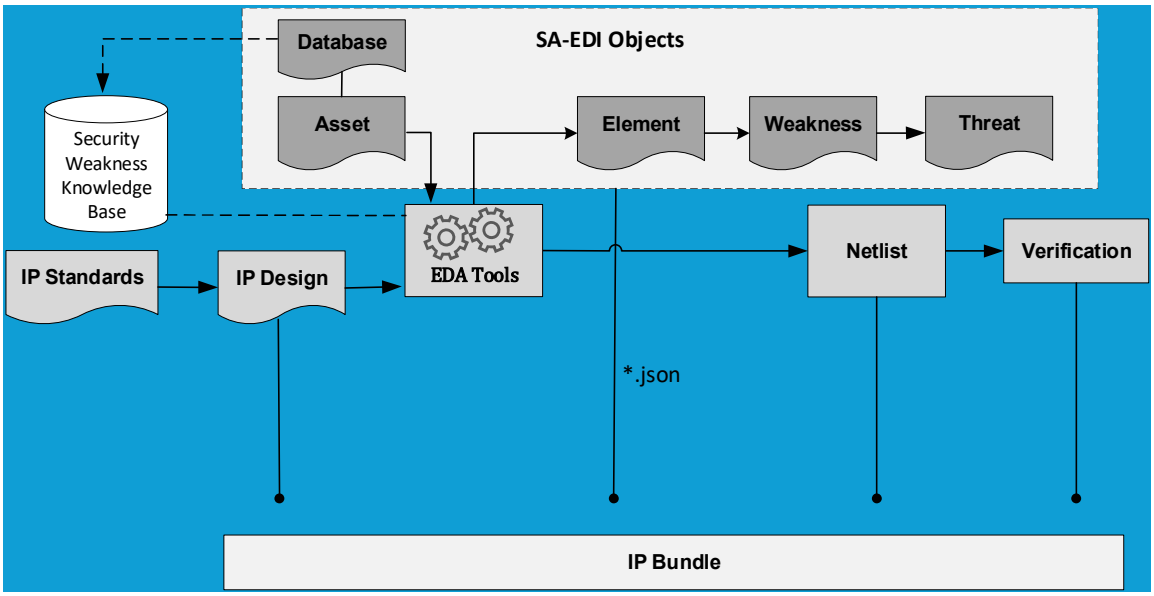

Figure 1. Security annotation design flow

IV. IEEE P3164 SA-EDI STANDARDIZATION ON THREAT MODELING FRAMEWORKS

IEEE P3164 SA-EDI standard is formed under the circumstances of,

- There are no standard and uniformly specified mechanisms of binding security assurance (SA) collateral to an IP, so SA verification is not thorough, and security related data mining is impossible at large scale.
- Different IP providers produce different collateral/formats for their security designs, which lacks consistent SA quality.

IEEE P3164 SA-EDI standard specifies a JSON-based data modeling framework to identify security assets in a module/ASIC IP. It collects all security-concern related data objectives via open-source or proprietary tools to accomplish a security goal. Moreover, this standard specifies how to bind these security data objects to the RTL and make it automated and verifiable. IEEE P3164 standard develops a security annotation methodology for IP providers, IP integrators, and EDA tool vendors in a generic SoC or subsystem integration flow, as shown in Fig.1.

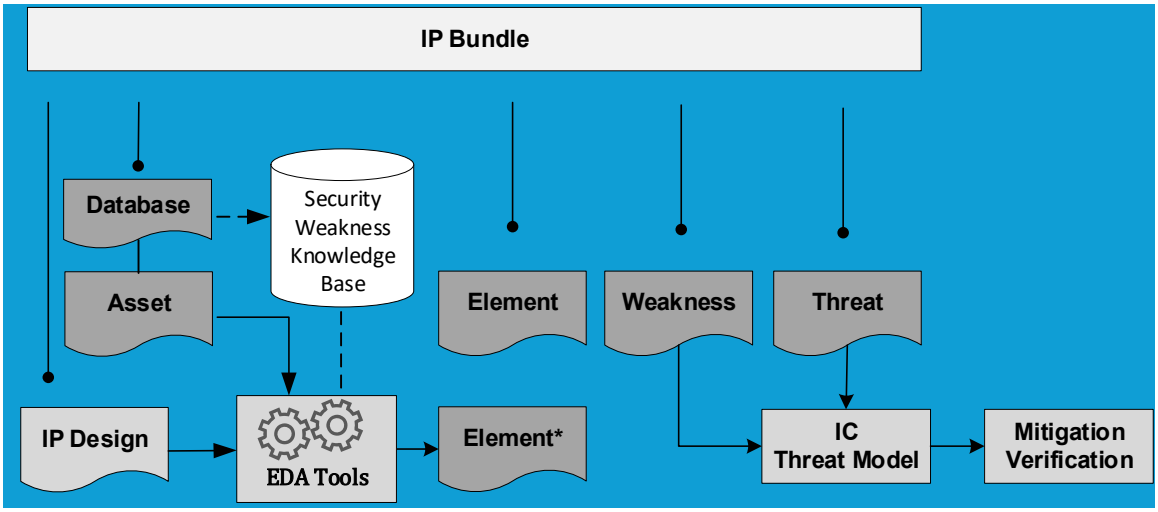

Figure 2. SA-EDI threat modeling framework

The SA-EDI is a lightweight standard, which defines only 5 data object types, Database Data Object; Asset Data Object; Element Data Object; Weakness Data Object; and Threat Data Object. The collected SA-EDI standardized data objects are packed into a single JSON file and are included in the released IP Bundle. This uniform format makes it highly portable for inter-operability and compatibility. It should also be noted that Element objects can be created manually, without an EDA tool.

TABLE 3. WEAKNESS DATA OBJECT

Attribute	Required	Type	Definition
Weakness Name	Yes	String	Unique identifier of the weakness for *Asset Name*. The *Weakness Name* need not be unique across multiple assets.
Description	Yes	String	Details about the weakness and how the security objective may be compromised
Asset Name	Yes	String	Reference to the attribute *Asset Name*
Security Objective	Yes	Array of Strings	Reference to attribute *Security Objective*.
Side-Channel	Yes	Enumeration	Indicates if this data object captures a side-channel weakness: 1. Yes 2. No
Attack Points	No	Array of Strings	Ports listed are associated with this security objective and the security weakness of this object
Condition	No	String	Reference to the attribute *Condition*.
Security Weakness Reference	No	Array of Strings	Reference to attribute *Security Weaknesses*.
Recommend ation	No	Array of Strings	Additional information that the integrator needs to be aware of, such as mitigations, configuration settings, etc.

The SA-EDI threat modeling framework is shown on Fig.2. Security annotation includes data objects that represent security assets, database entries, elements, weaknesses (or attack points), threats (adversaries), and security objectives providing an intent to mitigate the identified threats. Asset data objects identify sensitive (or critical) material within the specific subsystem. They are also related to data objects in a Security Weakness Knowledge Base, and they are inputs into EDA tools that create corresponding Element data objects. The database labeled "Security Weakness Knowledge Base" contains security weaknesses that are known from industry experience and/or security researchers as presented in Section III. The standard allows the use of multiple databases from multiple sources, e.g., MITRE [8]. The Element data objects contain ports and parameters that influence and/or observe the behavior of the asset and include potential security weaknesses that are associated with the asset. The Element data objects are used to create the Weakness data objects in Table 3. Weakness data objects associate a potential security concern to an asset by defining the ports and/or parameters from the Element objects. Weakness data objects are based on the architecture and design of IP (subsystem) as a standalone component. Additionally, the IP owner could create Weakness objects based on potential use cases of the module in an integrated SoC system.

The Threat data object is used to capture details about a specific exploitation of a weakness in a Weakness object. A Weakness object may be associated with several Threat objects because there can be multiple ways to exploit a weakness. The attributes of the Threat data object are listed in Table 4. The Weakness and Threat data objects contain the information required for adversary (threat) model. The integrator will use these objects to help create the IP/SoC's threat model. The integrator may also create additional Weakness and Threat objects that are IP specific, depending on which use cases are in scope.

Table 5 elaborates a basic SA-EDI threat modeling workflow step-by-step, which specifies the responsibilities and expectations of both the IP provider and integrator as it pertains to the SA-EDI threat modeling. The intent is to show step-by-step which party is responsible for creating the security assurance collateral and what the output should be. Here only the required steps are shown in Table 5. The optional steps are elaborated in the SA-EDI standard [4].

TABLE 4. THREAT DATA OBJECT [4]

Attribute	Required	Type	Definition
Threat Name	Yes	String	Unique identifier of the threat
Weakness Name	Yes	String	Reference to the attribute *Weakness Name*
Asset Name	Yes	String	Reference to the attribute *Asset Name*
Adversary	Yes	Array of Strings	The agent/attacker that can perform the technique detailed in the Technique attribute
Technique	Yes	String	Details about the attack itself such as actions required, equipment needed.
Mitigation	No	Array of Strings	Details the IC protection mechanism(s) that defends against the threat
Mitigation Verification	No	Array of Strings	Details how to verify the mitigation(s) described in the *Mitigation* attribute

V. KEY TRENDS IN HARDWARE SECURITY FOR SOC DESIGN AND FUTURE WORK

The major new trends in hardware security for SoC design include the modularization of chiplet architectures, the adoption of open-source RISC-V with enhanced security features, and the use of PUFs. Moreover, hardware-level protections are being strengthened with confidential computing for data in use, while AI and machine learning are being used to both defend against and enable new attacks.

On the one hand, open-source RISC-V and PUF help to improve the security of SoC design. For example, PUFs provide a device with a unique, unforgeable identity that is resistant to cloning and reverse engineering. The cryptographic keys derived from a PUF are never stored in a permanent memory and are destroyed when power is removed, making them "invisible" to attackers. Confidential computing focuses on protecting data while it is actively being used by an application. By isolating and encrypting sensitive data in a hardware-based Trusted Execution Environment (TEE), confidential computing ensures that data remains protected from unauthorized access, even from other software, the operating system, or privileged administrators.

On the other hand, some new trends also bring more challenges to SoC security design. For example, chiplet architecture requires a re-evaluation of the security model. Ensuring trusted execution and secure communication across multiple chiplets from different vendors is more complex than with traditional monolithic SoCs. Thus, security for chiplet-based architecture is now a fundamental design consideration, requiring robust measures to protect data as it moves between components.

AI and ML are double-edged swords for SoC security. They are used to enhance security features but also introduce new attack vectors. AI and ML can be used for real-time threat detection, identifying and mitigating complex attacks that would evade traditional security measures. New hardware-level adversarial attacks, such as side-channel attacks that exploit hardware vulnerabilities to compromise AI systems, are emerging. Adversaries may also use AI to accelerate attacks and evade detection.

The ongoing focus of the SA-EDI standard is catering for the new developments and requirements for secure AI hardware accelerators that resist new attacks and using AI to create more robust, self-defending security solutions, e.g., Advanced security features in Intel vPro hardware [10] is an excellent example, which can be leveraged by operating system (OS) and security software features across system attack surfaces to optimize mitigations against cyber threats. It demonstrated the practical application of hardware features by capabilities in Microsoft Windows 11 with Defender and CrowdStrike Falcon to assist defenders in understanding how these integrated capabilities can help mitigate real-world adversary behaviors as described in MITRE ATT&CK [13]. Surely SA-EDI standardization should accommodate all CPU architecture like ARM and RISC-V besides Intel's x86 CPU.

The future work of SA-EDI standardization also includes developing a series of specifications for security related tooling and automation and targeting a shifting-left security design and verification/validation. The standard will provide clear-cut and easily following interfaces for IP providers, IP integrators, and EDA tool vendors to practice a continuous process integrated early in the design phase to make sure advanced tools are used to model security test scenarios to ensure the integrity of the hardware. Therefore, IEEE P3164 standard workgroup is forming a taskforce to create supplemental material about the adversary models. The task may develop a case-study to demonstrate how EDA vendors concretize the SA-EDI standard by using security collaterals from a JSON file included in a specific released IP-bundle to accomplish a security verification automation via the related EDA toolchain.

TABLE 5. SA-EDI THREAT MODELING WORKFLOW

Step#	Owner	Required	Details	Output
1	IP provider	No	Identify a database of known weaknesses and create a Database object(s).	Database Data Object
2	IP provider	Yes	Identify asset(s) in the IP and create an Asset object(s) for each asset.	Asset Data Object
3	IP provider	Yes	Using the output of steps #1 and #2, generate the Element object(s) using an EDA tool. This step may also be done manually.	Element Data Object
4	IP provider	Yes	Using the output of step #3, create a Weakness object(s).	Table 3. Weakness Data Object
5	IP provider	No	Using the output of step #4, create a Threat object(s).	Table 4. Threat Data Object
6	IP provider	Yes	Bundle all the data objects created in steps #1-5 into the IP delivery package (i.e., IP bundle).	IP Bundle
7	Integrator	Yes	Using the output of step #6, determine which Weakness objects are in scope for the IC's threat model.	IC Threat Model
8	Integrator	Yes	Using the IC Threat Model, verify the security objectives are met by the mitigations in the design	Mitigation Verification

REFERENCES

[1] IEEE P3164 workgroup: https://sagroups.ieee.org/3164/
[2] Miltos Grammatikakis, Hellenic Mediterranean University "A Detailed Tour of IEEE standard P3164", DVCON, Munich, Germany, Oct 15-16, 2024, https://dvcon-proceedings.org/document/a-detailed-tour-of-ieee-standard-p3164/
[3] "Asset Identification for Electronic Design IP," in *Asset Identification for Electronic Design IP* , vol., no., pp.1-26, 5 April 2024.
[4] IEEE P3164 workgroup, Security Annotation for Electronic Design Integration (SA-EDI) Standard. [unpublished manuscript].
[5] Kohnfelder, Loren; Garg, Praerit (April 1, 1999). "The threats to our products". Microsoft Interface. Retrieved 13 April 2021.
[6] Ucedavélez, Tony and Marco M. Morana (2015). "Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis". John Wiley & Sons: Hobekin.
[7] Schneier, Bruce (December 1999). "Attack Trees". *Dr Dobb's Journal, v.24, n.12*. Archived from the original on 6 August 2007. Retrieved 2007-08-16.
[8] The MITRE Corporation (MITRE), "CWE Most Important Hardware Weaknesses," https://cwe.mitre.org/data/definitions/1194.html.
[9] https://www.intel.com/content/www/us/en/security/security-practices/secure-development-practices/threat-modeling.html.
[10] Intel-vpro-08.20.2024_attack-15.1-enterprise, Intel vPro Landing - Mappings Explorer
[11] https://www.intel.com/content/www/us/en/security/security-practices/secure-development-practices/security-development-tools.html.
[12] https://en.wikipedia.org/wiki/Threat_model
[13] MITRE ATT&CK® https://attack.mitre.org/