# Hardware Vulnerability Description, Sharing and Reporting: Challenges and Opportunities

Jeremy Bellay*, Domenic Forte†, Robert Martin‡ and Christopher Taylor*
*Battelle Memorial Institute, Columbus, OH
Email: {bellayj, taylorcp}@battelle.org
†University of Florida, Gainesville, FL
Email: dforte@ece.ufl.edu
‡MITRE Corporation, Bedford, Massachusetts
Email: ramartin@mitre.org

***Abstract*—Hardware, once deemed an immutable root-of-trust, has recently come under scrutiny as hardware vulnerability researchers have uncovered weaknesses, which can be exploited remotely, in the supply chain, or through through physical access. Given this revelation, interest has manifested in the concept of sharing hardware vulnerabilities and mitigations. Borrowing from the software domain, this work surveys the existing frameworks for public vulnerability and weakness sharing, examines their efficacy for hardware, and also identifies potential gaps. Additionally, due to the inherent apprehensiveness of sharing hardware vulnerabilities which is attributed to the fear of revealing hard to patch exploits, this work intends to address those potential risks and presents possibly benefits. Furthermore, the authors also analyze current hardware vulnerability reporting efforts and discuss how to quantify security for hardware.**

## I. Introduction

The field of computer and communications security has traditionally viewed hardware as an immutable root-of-trust. However, the last decade of hardware security research has led to the discovery of hardware-oriented weaknesses that can be exploited remotely, in the supply chain, through physical access, or through any combination of the three. In particular, the last few years have seen hardware vulnerabilities in several top processor vendors. By and large, these vulnerabilities were complicated to mitigate and incurred significant performance degradation as well as other unintended side effects [1].

In addition to simply offering an alternative attack surface beyond software, hardware has other attractions to attackers that will make it an increasingly common target. Since hardware is the foundation of computing and operates at the lowest abstraction layer, common software mitigations such as system updates and patches are inadequate. Moreover, while hardware attacks require physical access and have long been considered too expensive, the landscape is starting to change in the attacker's favor. As part of the emerging internet of things (IoT), the number of "smart things" is soon expected to dwarf mobile computing devices [2]. IoT products are often deployed in hostile environments where they can operate for years with minimal intervention. Furthermore, improvements in the capabilities of failure analysis tools, especially semi-invasive ones, coupled with lower barriers to their access make on-chip assets, secrets, and IP more vulnerable to fault injection, probing, and reverse engineering than ever before [3]. Finally, hardware itself is also evolving rapidly in new node sizes, neuromorphic computing, Carbon-Nanotube FET, and quantum computing. These advances create fundamentally new attack surfaces, many of which will probably never be entirely secured.

Left unaddressed, vulnerabilities expose critical systems, their information, and the people who rely on them. However, in order to be fixed, they must first be identified, and critical information must be shared among stakeholders. The rise of viruses, malware, and remote attacks in the 80's, 90's, and early 2000's led to efforts aimed at supporting and coordinating vulnerability education, discovery, advisories, and alleviation. These included a common language and taxonomies to describe software and software-related weaknesses [4], attack patterns [5] and vulnerabilities [6], [7], avenues for responsible disclosure of vulnerabilities and exposures [7], [8], entities to coordinate responses between researchers, vendors, and deployers, and approaches to prioritize mitigations [9]. MITRE has recently begun a major effort to extend CWE to hardware applications, but the particulars of hardware have made the wholesale transition of the software system to hardware difficult.

In addition to discussing current vulnerability databases, this work also analyses the current landscape of hardware vulnerability reporting, characterization, disclosing, as well as their risks and benefits to potential stakeholders. Moreover, this work also investigates the future of hardware vulnerabilities and weaknesses in relation to security estimations.

## II. Survey of Existing Vulnerability Reporting

The idea of a public national hardware vulnerability database is comparatively recent and comes two decades after the establishment of a vulnerability database focused on

This work was performed for AFRL, Contract No. FA8650-20-F-1894/Task Order 0003 under Nimbis IDIQ Contract No. FA8650-18-D-1604. 

software. For context, in the late 1990's there were several groups trying to discuss and share information surrounding software vulnerabilities, how they occurred, how they were attacked, how to recognize them, what to do about them, and how to prioritize them. Against this backdrop, the MITRE Corporation [10], a not-for-profit company that runs Federally Funded Research and Development Centers (FFRDCs) for multiple departments of the US Government, came up with the idea of a Common Enumeration of Vulnerabilities (CVE). The CVE was launched in September of 1999, however, years later the need to enumerate the weaknesses in software which created those vulnerabilities and how they could be attacked became apparent. As a result, MITRE launched Common Weakness Enumeration (CWE) and the Common Attack Pattern Enumeration and Classification (CAPEC) efforts while NIST established the National Vulnerability Database [11] to support a broader access to the CVE information and incorporate additional types of related information.

### *A. Common Vulnerabilities and Exposures (CVE®)*

The CVE website provides the widest used publicly available dictionary of known vulnerabilities in commercial and open source software, where a vulnerability is a specific type of weakness or weaknesses in a product that is exploitable by a threat. Furthermore, the CVE helps individuals and organizations correlate the numerous types of public and private information about public vulnerabilities in applications and other cyber-enabled capabilities. The dictionary is organized in an intuitive manner based on known vantage points and abides by a concise schema for describing related public information, while also offering a mechanism for correlating shared information. CVE's organization is based on the concept of a simple list of identifiers for publicly known vulnerabilities. which helps to correlate entries that are referring to or discussing the same issues [12]. Entries into the database, referred to as CVEs, are publicly known issues that need to be patched or mitigated to address an exploitable vulnerability. New CVEs are created through CVE Naming Authorities (CNAs), which is comprised of a web of organizations sharing the responsibilities of associating CVE identifiers with security issues. While MITRE is currently the root CNA, there are over 127 other organizations that have been identified as the preferred CNA which the community should work with to assign CVE identifiers. These organizations constitute security researchers and have specific products, product types, or other specified domains for which they address CVEs associated with their specific area.

Vulnerabilities characterized in the CVE are concrete examples of items described in the catalog of Common Weakness Enumeration (CWE) and exploitable by the attack patterns captured in the Common Attack Pattern Enumeration and Classification (CAPEC) collection. It is through the linking of public vulnerabilities in a specific product to the weakness in the CWE and attack patterns in CAPEC, organizations can leverage those collections and the information in them in their assessment and investigation into newly discovered examples of vulnerabilities. Additionally, the linked information also offers opportunities for organizations to examine their own code base for the same type of vulnerability.

### *B. Common Weakness Enumeration (CWE™)*

The CWE website provides a publicly available catalog of the weaknesses that occur in software architecture, design, code, or during deployment, where a weakness is a description of the common attributes and susceptibilities an adversary can capitalize on. These weaknesses can result in exploitable vulnerabilities and are meant to help individuals and organizations understand how weaknesses in applications and other cyber-enabled capabilities occur. The origin of CWE is derived from working with the real-world examples of the types of vulnerabilities that appear in software applications. These vulnerabilities are meant to generalize those real-world flaws into conceptual patterns of what makes software exploitable. Furthermore, the patterns allow people to learn and recognize them early in the lifecycle of software to either avoid introducing them or finding them quickly to address them before the software is put into operation. CWE entries, or CWEs, are a mixture vulnerability types found by exploiting them "in the wild" or through examination and testing of software by hackers, developers, and testers. New CWEs are created by generalizing a specific vulnerability in a particular product or through the examination of software architecture, design, code, or deployed applications and finding constructs that allow someone to do something that was not intended.

Weaknesses define the opportunities that an adversary may leverage and how the builder or defender can go about finding and removing them. In addition, if an organization is concerned with specific weaknesses because of the possible consequences from a successful attack on them, the relationships between CWEs and CAPECs can be used to identify likely attacks for a CWE which can inform the defender on options for defense. Moreover, CWE are organized by their properties where a list of possible properties is located in Table I, and definitions of the properties can be found on MITRE's website [13].

TABLE I
PROPERTIES OF WEAKNESSES IN CWE

| | | |
|---|---|---|
| Description | Common Consequences | Notes |
| Extended Description | Likelihood of Exploit | Alternate Terms |
| Taxonomy Mappings | Observed Examples | Memberships |
| Background Details | Potential Mitigations | References |
| Modes of Introduction | Related Attack Patterns | Detection Methods |
| Applicable Platforms | | |

### *C. Common Attack Pattern Enumeration and Classification (CAPEC™)*

The CAPEC website provides a publicly available catalog of common attack patterns that helps individuals and organizations understand how adversaries exploit weaknesses in applications and other cyber-enabled capabilities. CAPEC's origin is based on the concept of software design patterns

or templates for the design and implementation of commonly used software techniques. CAPEC entries or CAPECs are a mixture of attack patterns actively seen "in the wild", through proof-of-concept, or based on research surrounding what is theoretically possible to do. CAPECs are created by generalizing adversarial behaviors and can be thought of as an extrapolation of things that could happen or be observed. For instance, CAPEC aims to answer the following questions: how does an adversary take advantage of a software weakness, what are the steps taken, what is the knowledge needed to follow those steps, how difficult is it to achieve, and how likely is success.

Attack patterns define the challenges that an adversary may face and how they go about solving them. They derive from the concept of design patterns applied in a destructive rather than constructive context and are generated from in-depth analysis of specific real-world exploit examples, research, and technologies. If an organization is concerned with attacks on specific weaknesses because of the possible consequences, the relationships between CWEs and CAPECs can be used to identify the likely CAPECs for a CWE and that knowledge can inform the defender on options for mitigation. Table II shows a list of the properties of attack patterns in CAPEC[1].

TABLE II
ATTACK PATTERNS FOUND IN CAPEC DATABASE

| | | |
|---|---|---|
| Description | Prerequisites | Example Instances |
| Alternate Terms | Skills Required | Related Weaknesses |
| Likelihood of Attack | Resources Required | Taxonomy Mappings |
| Typical Severity | Indicators | References |
| Related Attack Patterns | Consequences | Execution Flow |
| Mitigations | | |

### D. CMU CERT/CC

The CERT Coordination Center (CERT/CC) is the coordination center of the computer emergency response team (CERT) for the Software Engineering Institute (SEI), a non-profit United States FFRDC. The CERT/CC researches software bugs that impact software and internet security, publishes research and information on its findings, and works with business and government to improve security of software and the internet as a whole. Furthermore, CERT/CC works directly with software vendors in the private sector as well as government agencies to address software vulnerabilities and provide fixes to the public. Through this process CERT/CC promotes a particular process of coordination known as Responsible Coordinated Disclosure. In this case, the CERT/CC works privately with the vendor to address the vulnerability before a public report is published, usually jointly with the vendor's own security advisory. However, in extreme cases when the vendor is unwilling to resolve the issue or cannot be contacted, the CERT/CC typically discloses information publicly after 45 days since first contact attempt [14]. Software vulnerabilities coordinated by the CERT/CC may come from internal research or from outside reporting. Moreover, depending on severity of the reported vulnerability, the CERT/CC may take further action to address the vulnerability and coordinate with the software vendor. Additionally, CERT/CC coordinates information with US-CERT in cases that concern US national security, whereas CERT/CC handles more general cases, often internationally.

[1]definitions of each of these properties see https://capec.mitre.org/documents/schema/index.html.

### E. National Vulnerability Database (NVD)

The NVD is a comprehensive cybersecurity vulnerability database that allows the tracking of vulnerability trends over time. This trending service allows users to assess changes in vulnerability discovery rates within specific products or within specific types of vulnerabilities. NVD data is represented using Security Content Automation Protocol (SCAP), which is a suite of specifications that standardize the format and nomenclature by which software flaw and security configuration information is communicated, both to machines and humans. Furthermore, the NVD includes databases of security configuration checklists for the National Checklist Program, listings of publicly known software flaws, product names, and impact metrics. Additionally, as of the end of July 2020, the NVD contained over 145,000 vulnerability advisories, over 150 SCAP-expressed checklists across 153 platforms and a product dictionary with over 500,000 operating system, application, and hardware name entries.

NVD is hosted and maintained by the National Institute of Standards and Technology (NIST) and is sponsored by the Department of Homeland Security's US-CERT. Despite being sponsored by a government entity, thousands of organizations worldwide use of SCAP data has effectively extended NVD's reach. Additionally, increasing demand for NVD XML data feeds (i.e., mechanisms that provide updated data from data sources) and SCAP-expressed content from the NVD website demonstrates an increased adoption of SCAP. Furthermore, in the past year, the NVD continued to see a significant upward trend in the number of vulnerabilities received. To support this increased workload, several strategies are in progress, including longer-term efforts to focus on vulnerability ontology and natural language processing to manage future growth. Overall, the NVD continues to experience an average download growth rate of over 10% per month.

A key component of the NVD is the Common Vulnerability Scoring System (CVSS), used for measuring the relative severity of software flaws. In 2017, the NVD began providing CVSS base scores following the CVSS v3 specification within the data feeds. Currently, NIST is working with the vulnerability community to enable the automated analysis of metrics, such as CVSS, establishing a baseline of the minimum information needed to properly inform the vulnerability management process, and facilitating the sharing of vulnerability information across language barriers. To assist in this work, a public draft of NISTIR 8138, Vulnerability Description Ontology (VDO): A Framework for Characterizing Vulnerabilities, was created to foster a conversation and collect feedback on the best mechanisms to improve the degree of automation within

vulnerability management processes. In FY 2019, NIST started to develop the VDO iteratively through collaboration with the security automation community on GitHub[2]. This approach will allow participation from as many stakeholders in the vulnerability community as possible.

## III. Hardware Particulars

### A. Trust-Hub

Trust-hub is a web portal[3] for the hardware security community to exchange of ideas, benchmarks, platforms, tools, and educational resources, which is led by the University of Florida and has been supported by funds from the National Science Foundation (NSF) for the last eight years. In 2018, Trust-hub added a Vulnerability Database sub-portal, which currently consists of a physical attack taxonomy that highlights the mechanisms used by attackers in IC and PCB reverse engineering, hardware Trojan insertion, fault injection, side channel analysis, etc. Additionally, Trust-hub contains a list of academic and commercial computer aided design (CAD) tools available for protecting hardware IP and assessing system-on-chip (SoC) susceptibility to information leakage, hardware Trojans, side channel attacks, fault injection, and probing.

While this sub-portal is a promising joint initiative from academia and industry that aligns well with the goals outlined in this paper, it remains a work-in-progress. For one, an additional SoC vulnerability database is mentioned as "coming soon", and physical attack taxonomy only includes attacks categorized in one manner. Furthermore, the sub-portal can be improved by adding short descriptions, execution steps, consequences, and details about the time, skills, and resources required for each attack. Moreover, the CAD tool list provides just the right amount of information about each tool, but it is not yet exhaustive. To expand its capabilities researchers and vendors are encouraged to send details to the Trust-hub Vulnerability Database organizers to gain exposure for their tools and so that community has a better idea of which tools still need to be developed. The University of Florida aims to upgrade the Trust-hub portal over the next several years in order to accept submissions, take requests, and facilitate more communication between users.

### B. Trusted Silicon Stratus

The Nimbis Services Trusted Silicon Stratus (TSS)[4] is a secure cloud services infrastructure intended to allow government agencies to design integrated circuits (ICs) in a private community cloud. The creation of the TSS had 4 main objectives. First is to enable the design and creation of Tech readiness level 8 ASIC and FPGA designs all the way through design, manufacture, and life cycle management through access to EDA tools and data repositories. Second, is to provide DoD Information Assurance Certification and Accreditation Process (DIACAP) cloud solutions for inter-agency collaboration for not only the Air Force but the rest of the DoD community. Third, the TSS is to function as a secure data repository which will enable the storing of data related to manufacturing, test, supply, field defects, fatigue analysis, inventory, deployment, re-fit and upgrade, and data provenance in the DoD supply chain and acquisition cycle. Finally, the TSS is an attempt to merge the disparate nature of the current DoD IC infrastructure to better enable leveraging economies of scale and help foster collaboration among the DoD community.

The TSS offers a secure user configurable Trusted Microelectronics chip design ecosystem hosted on Amazon's AWS GovCloud. This cloud based service enables the affordable multi-organizational chip design and verification platform required for today's complex microelectronics development cycle. Some of the features the TSS provides are as follows.

- EDA tools
- Scalable computing resources
- Life cycle management
- PDK and IP libraries
- Metered billing
- Tiered levels of security
- Archiving and version control

### C. Hardware Differentiation

*1) Overview of Hardware Security Taxonomies:* As described above, there are several description frameworks available for characterizing computer vulnerabilities in the form of the CVE ecosystem such as CVE, CVSS, CWE, CWSS, and CAPEC. CVE and CVSS have been used by the community to document well known vulnerabilities such as Spectre, Meltdown and Rowhammer on the NVD. While the CVE ecosystem is increasingly capable of usefully characterizing hardware vulnerabilities, there are many hardware specific attack and weakness concepts that have been already defined in the literature.

Taxonomies of hardware attacks and attackers have a long history [15], [16], and outside of the CVE ecosystem, there have been several efforts to create more detailed taxonomies for particular domains of hardware vulnerabilities. Currently the most well-known effort is Trust-Hub, (described in Section III-A) which contains taxonomies of physical vulnerabilities, hardware trojans, and hardware obfuscation. Additionally, in 2014 Guin, DiMase, and Tehranipoor [17], published a taxonomy of counterfeit ICs. Furthermore, there have been a variety of specialized taxonomies around specific weaknesses including a taxonomy of attacks based on speculative execution (e.g., Spectre/Meltdown) [18], general fault attacks [19], differential power analysis [20], and injection attacks [21] to name a few (as shown in Table III).

Altogether, these taxonomies give a detailed picture of the world of hardware security. However, with a few exceptions [22], the focus is on the definition of concepts for attacks, weaknesses, and attackers. Consequently, the attributes of these attacks are often missing that which would allow modeling and the development of best practices around each kind of attack. For example, while optical fault injection (OFI) attacks are well studied, no description framework, to the authors best

[2]https://github.com/usnistgov/vulntology

[3]available at www.trust-hub.org

[4]available at www.trustedstratus.com/

knowledge, has included the concepts necessary to describe when an OFI attack would be successful on what device.

TABLE III
EXAMPLES OF HARDWARE SECURITY TAXONOMIES

| Hardware Topic | Reference |
|---|---|
| Physical Vulnerabilities | Trust-Hub |
| Hardware Trojans | Trust-Hub |
| Obfuscation | Trust-Hub |
| Fault Attacks | Karaklajic et al. 2013 [19] |
| Speculative Execution | Canella et al. 2019 [18] |
| Counterfeits | Guin et al. 2013 [17] |
| Biased-Fault Attacks | Farhardy et al. 2015 [23] |
| DPA Analysis and Countermeasures | Marzouqi and Salah 2013 [20] |
| Fault Injection and Simulation | Piscitelli [24] |
| Signal Injection Attacks | Giechaskiel et al. 2019 [21] |
| Attackers | Anderson and Kuhn 1996 [15] |
| Hardware Attacks and Attackers | Rae and Wildman 2003 [25] |
| Hardware Attacks and Attackers | Keommerling et al. 1999 [16] |
| Smartcard | Yahay and Omar 2010 [22] |

*2) Hardware Weakness and Attack Pattern Descriptions:* The concepts necessary to describe software weaknesses are well documented in the CWE description framework described in Section II-B. In addition to software specific weaknesses, CWE includes many concepts required for computer security. Moreover, along with its taxonomies of concepts, it includes an ontology which allows for relations between those concepts and the concepts from CAPEC. This allows for the detailed description of hardware errors, even if hardware specific concepts are missing.

For example, the CWE concept "CWE-325: Missing Required Encryption Step" is described as "The software does not implement a required step in a cryptographic algorithm, resulting in weaker encryption than advertised by that algorithm." While this effect could be caused by an inherent software weakness it is often the end result of hardware glitching attacks. The CWE ontology contains the relation "resultant" and CWE-325 in its definition, can be described by that relation. Thus, CWE contains the concepts and relations to describe the action of a, for example, fault injection attack that causes an encryption algorithm to prematurely dump the plaintext to a recoverable media. However, CWE and CAPEC do not contain a concept for "Optical Fault Injection", though CAPEC does have a concept for the broader term "Fault Injection". From the hardware perspective, differentiating between types of fault injection is important, considering that timing faults and optical faults will have drastically different mitigation strategies.

A simple first step towards making the CVE ecosystem more relevant to hardware would be the incorporation of existing taxonomic concepts (e.g., those of Trust-Hub) into CWE and CAPEC. Additionally, once hardware vulnerabilities can be added, the process and procedures of how to properly disclosure new vulnerabilities into the CVE ecosystem, while taking into account the difficulties hardware vulnerabilities pose, must also be established. Notwithstanding, CWE released three updates in 2020 including many new hardware weaknesses. For example, version 4.0 added many hardware specific nodes such as "CWE-1247: Missing Protection Against Voltage and Clock Glitches" and "CWE-1239: Improper Zeroization of Hardware Register". Additionally, version 4.0 saw the introduction of a new "Hardware Design" view, which organizes weaknesses associated specifically with hardware. In version 4.3, a total of 95 additional hardware weaknesses were added to the CWE, which now are separated into a hardware section on the summary of changes page. However, these updates have not surfaced in CAPEC as of this writing.

## IV. VULNERABILITY DISCLOSURE

Once a vulnerability in a public system has been discovered, the question remains on how to best alert the relevant parties in order to produce a mitigation before the vulnerability can be exploited. In the event that a vulnerability is discovered internally, often a patch is developed and distributed without any public acknowledgement. However, even internally discovered vulnerabilities are often given CVEs and publicly acknowledged. Nonetheless, most cases of vulnerability disclosure arise when a 3rd party makes the discovery. In this situation, the reporting party must decide on how to proceed and whether to alert the product manufacturer or to disclose the vulnerability without input of the vendor (e.g., public disclosure). Historically, vulnerability disclosure was an extremely sensitive process with reporters being concerned about lawsuits brought on by vendors.

In the software domain, a formal vulnerability disclosure process has been largely recognized to be a benefit to both vendor and reporter [26]. However, this is not to say that disclosure is uncontroversial or without problems. Namely, the case has been made that disclosing vulnerabilities provides attackers with both a training ground and ammunition [27]. Moreover, due to the prestige associated with discovering major vulnerabilities, their exploitability can be exaggerated, and in some cases turn out not to be true [28]. However, without a disclosure process, users may be unaware of unpatched vulnerabilities within their system. This not only puts them at risk, but any network into which they have privilege as well, while also making them a target for botnet recruitment. Additionally, lack of disclosure also leaves system integrators at risk, as well as those who may use software without acknowledgement (often as a legally questionable practice). Several best practices have been published and collated in ISO/IEC report 29147 on Vulnerability Disclosure and the related report 30111 on Vulnerability handling.

Vulnerability disclosure can take one of several routes. The vender may discover the vulnerability and handle the disclosure and mitigation completely independently. A non-vender party may discover the vulnerability and report it to the vendor, who then handles the mitigation and disclosure independently. The discover may disclose the vulnerability without consulting the vender. Finally, the vender, discoverer and other stake holders may work together in mitigation and disclosure of the vulnerability in a process usually referred to as "Coordinated Disclosure".

Coordinated disclosure is of particular interest to hardware, where hardware vulnerability mitigation includes not only the already complicated ecosystem required for software vulnerability mitigation, but also the original equipment manufacturers (OEMs) and their network or suppliers. Additionally, researchers who report a hardware vulnerability often have already expended extensive effort to discover and characterize the vulnerability and would like to receive credit through publicity or other reward. Using a coordinated disclosure process allows the reporter to continue to participate in the process.

The mitigation of a hardware vulnerability often requires coordination between three types of technical parties: OEMs, which are the companies that brand and distribute the products in which a vulnerability may occur, but might not manufacture all (or even most) of the component parts; Hardware vendors, who manufacture the components but may have limited understanding of the larger functioning of the integrated product; and Software distributers, who produce the operating systems (and often firmware) that utilize the device, and as such, play a primary role in developing and testing software based mitigations.

## V. Hardware Security Estimation

Estimating the security and risk of electronic devices is key to addressing hardware security concerns. Notably, security estimates will allow for a rational distribution of resources towards high impact security problems. Additionally, such estimations are necessary to incentivize the adoption of secure designs by industry as a quantification of security can be used to justify the trade-offs. Ultimately security estimates must take several factors into account:

- Hardware security issues range from remotely accessible vulnerabilities such as Spectre/Meltdown, to counterfeit production, to subversion via fault injection. Therefore, generic security estimation is impossible, and a threat model must be specified.
- Security estimates must take preventative measures into account such as secure design practices, and testing/verification.
- Hardware has a distinct and complex lifecycle, which begins with multiple layers of design abstraction, and multiple phases of the design process leading to fabrication. Additionally, hardware can be used in multiple environments with various levels of physical security.
- Hardware usually consists of integrated components beginning with the inclusion of 3rd party IP at during IC design, to various components on printed circuit boards, and finally inclusion different hardware into large systems.

Integration poses an interesting problem for security estimation due to the possibility of combining relatively secure components with other (potentially much less secure) components. This situation presents both risk and opportunity, where restricting secure processes through trusted and secure components may allow for compensation for less secure components. However, an unsecure component may also serve as a "weakest link" for an otherwise secure system. Regardless, the ability to integrate security estimates across a system is essential to most applications.

## VI. Conclusions and Future Directions

Hardware vulnerability and weakness sharing has been a controversial topic due to their hard-to-patch nature. However, we identify several fruitful directions for research and collaboration that can improve the hardware security landscape. We have shown that the CVE ecosystem (CVE, CWE, CAPEC, etc.) has the underlying structure necessary for the reporting of hardware vulnerabilities, weakness, and attack patterns. However, the current concepts need to be expanded to better represent hardware specific weaknesses and vulnerabilities such as physical attacks. In fact, since the TAME forum and the publication of the accompanying report, MITRE has begun a vigorous effort to expand the CWE database to better cover hardware vulnerability concepts. Furthermore, another expansion is planned, and MITRE has volunteered to add the new hardware CWE terms to existing CVE instances.

Perhaps the common thread throughout all hardware differentiation is its immutability. While this raises real concerns around vulnerability sharing and disclosure, it also has a fundamental impact on the use cases around vulnerability and threat model usage. Because of the immutability of hardware, vulnerabilities almost always must be prevented and mitigated during design and manufacture. The current descriptive frameworks are primarily focused on reporting upon discovery, which is a fundamentally different application than prevention. Prevention requires detailed descriptions of the vulnerabilities/weakness, their parameterizations, and how they relate to verification and assurance tools. The Accellera Systems Initiative is working in this direction by developing a standard towards understand IP vulnerabilities between OEMs and their suppliers [30]. However, the space mapping hardware weaknesses to assurance tools and data is largely unmapped.

## References


[1] L. H. Newman, "The hidden toll of fixing meltdown and spectre," Jan 2018. [Online]. Available: https://www.wired.com/story/meltdown-and-spectre-patches-take-toll/

[2] L. Columbus. (2018, 12) 2018 roundup of internet of things forecasts and market estimates. [Online]. Available: https://www.forbes.com/sites/louiscolumbus/2018/12/13/2018-roundup-of-internet-of-things-forecasts-and-market-estimates/

[3] C. Boit, C. Helfmeier, and U. Kerst, "Security risks posed by modern ic debug and diagnosis tools," in *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*, 2013, pp. 3–11.

[4] CWE - common weakness enumeration. [Online]. Available: https://cwe.mitre.org/

[5] CAPEC - common attack pattern enumeration and classification (CAPEC). [Online]. Available: https://capec.mitre.org/

[6] J. D. Howard and T. A. Longstaff. (1998, 10) A common language for computer security incidents. [Online]. Available: https://digital.library.unt.edu/ark:/67531/metadc706351/

[7] CVE - common vulnerabilities and exposures (CVE). [Online]. Available: https://cve.mitre.org/

[8] R. Pethia, "Computer emergency responseâ€"an international problem," in *Proceedings of the 13th International Conference on Software Engineering*, ser. ICSE â€™91. Washington, DC, USA: IEEE Computer Society Press, 1991, p. 313.

[9] P. M. Mell, K. A. Scarfone, and S. Romanosky, "The common vulnerability scoring system (CVSS) and its applicability to federal agency systems," 08 2007. [Online]. Available: https://www.nist.gov/publications/common-vulnerability-scoring-system-cvss-and-its-applicability-federal-agency-systems

[10] "We operate ffrdcs." [Online]. Available: https://www.mitre.org/centers/we-operate-ffrdcs

[11] L. Brownsword, C. Woody, C. Alberts, and A. Moore, "A framework for modeling the software assurance ecosystem: Insights from the software assurance landscape project," 08 2010.

[12] T. L. S. D. Committee, "Test methods standard; general requirements, suspect/counterfeit, electrical, electronic, and electromechanical parts," 04 2018. [Online]. Available: https://www.sae.org/content/as6171

[13] "Common weakness enumeration." [Online]. Available: https://cwe.mitre.org/documents/glossary/index.html

[14] A. Manion, "Vulnerability disclosure policy." [Online]. Available: https://vuls.cert.org/confluence/display/Wiki/VulnerabilityDisclosurePolicy

[15] R. Anderson and M. Kuhn, *Tamper Resistance â€“ a Cautionary Note*, 1996.

[16] O. Kömmerling and M. G. Kuhn, "Design principles for tamper-resistant smartcard processors," in *Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology*, ser. WOSTâ€™99. USA: USENIX Association, 1999, p. 2.

[17] U. Guin, D. Dimase, and M. Tehranipoor, "A comprehensive framework for counterfeit defect coverage analysis and detection assessment," *J. Electron. Test.*, vol. 30, no. 1, p. 25â€“40, Feb. 2014. [Online]. Available: https://doi.org/10.1007/s10836-013-5428-2

[18] C. Canella, J. Van Bulck, M. Schwarz, M. Lipp, B. Von Berg, P. Ortner, F. Piessens, D. Evtyushkin, and D. Gruss, "A systematic evaluation of transient execution attacks and defenses," in *Proceedings of the 28th USENIX Conference on Security Symposium*, ser. SECâ€™19. USA: USENIX Association, 2019, p. 249â€“266.

[19] D. KaraklajiÄ‡, J. Schmidt, and I. Verbauwhede, "Hardware designer's guide to fault attacks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 12, pp. 2295–2306, 2013.

[20] H. Marzouqi, M. Al-Qutayri, and K. Salah, "Review of gate-level differential power analysis and fault analysis countermeasures," *IET Information Security*, vol. 8, no. 1, pp. 51–66, 2014.

[21] I. Giechaskiel and K. B. Rasmussen, "Sok: Taxonomy and challenges of out-of-band signal injection attacks and defenses," *CoRR*, vol. abs/1901.06935, 2019. [Online]. Available: http://arxiv.org/abs/1901.06935

[22] S. Yahya and N. A. N. Omar, "Security validation of smartcard: Mcos," 2010.

[23] P. S. Nahid Farhady Ghalaty, Bilgiday Yuce, "Analyzing the efficiency of biased-fault based attacks," Cryptology ePrint Archive, Report 2015/663, 2015, https://eprint.iacr.org/2015/663.

[24] R. Piscitelli, S. Bhasin, and F. Regazzoni, *Fault Attacks, Injection Techniques and Tools for Simulation*. Cham: Springer International Publishing, 2017, pp. 27–47. [Online]. Available: https://doi.org/10.1007/978-3-319-44318-8_2

[25] A. Rae and L. Wildman, "A taxonomy of attacks on secure devices," p. 14.

[26] A. D. Householder, G. Wassermann, A. Manion, and C. King, "The cert guide to coordinated vulnerability disclosure," Carnegie-Mellon Univ Pittsburgh Pa Pittsburgh United States, Tech. Rep., 2017.

[27] G. Uht, "Letâ€™s keep it to ourselves: Donâ€™t disclose vulnerabilities," Jan 2019. [Online]. Available: https://www.sigarch.org/lets-keep-it-to-ourselves-dont-disclose-vulnerabilities/

[28] B. Dickson, "Software vulnerability disclosure is a real mess," Aug 2019. [Online]. Available: https://www.pcmag.com/opinions/software-vulnerability-disclosure-is-a-real-mess#:~:text=Thistypeofvulnerabilityallows,maliciouspayloads,andstealinginformation.

[29] "Common weakness enumeration." [Online]. Available: https://cwe.mitre.org/data/definitions/1194.html

[30] B. Sherman, M. Borza, J. Pangburn, A. Sarkar, W. Chen, A. Nordstrom, K. H. Hayashi, M. Munsey, J. Hallman, A. Althoff, J. Valamehr, A. Sherer, I. Sobanski, S. Aftabjahani, and S. Nimmagadda, "Ip security assurance standard," Accellera, Tech. Rep., 09 2019.