Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 08, No. 01, March 2022.

AN IN-DEPTH VISION TO HARDWARE DESIGN SECURITY VULNERABILITIES

Zainab Younis and Basim Mahmood

(Received: 29-Oct.-2021, Revised: 20-Dec.-2021, Accepted: 11-Jan.-2022)

ABSTRACT

Hardware plays a major role in our everyday life. Despite the technological thrive, there remain various security issues regarding hardware weaknesses that needed to be addressed carefully. Hence, an in-depth vision of the vulnerabilities that may exist in hardware design is delivered in this study by generating a network model that contains the most common weaknesses reported in common weakness enumeration (CWE). The main goal of the generated network is to deeply analyze the relations between different hardware designs and security weaknesses. Based on the conducted analysis, recommendations and suggestions are given to benefit many parties including hardware security developers. Accordingly, the analysis approach depends on different concepts that are inspired by the field of network science. The generated model is illustrated in a graph, wherein the nodes are the weaknesses and the edges are created if two weaknesses have a relation to each other. Promising findings have been attained and can be observed in the given model. For instance, the weaknesses CWE-441, CWE-1189, CWE-276 and CWE-1304 have not been given enough attention by the CWE and should be highly considered by software developers. Moreover, a rank for the hardware vulnerabilities based on network metrics is provided and compared with the most recently announced list of top hardware weaknesses by CWE. It is found that only two weaknesses are in common between the two lists, which indicates that the CWE list does not highly consider the relations among the weaknesses.

KEYWORDS

Complex networks, CWE vulnerabilities, Data analysis, Hardware vulnerabilities.

1. INTRODUCTION

As is a well-known fact, computers consist of both software and hardware. The term hardware refers to the tangible components and devices a computer is made of. A computer is not a single device, but a system of amalgamated devices working together to achieve the desired tasks [1]. People use these devices in their day-to-day activities for a myriad of purposes, such as at work, for medicine or engineering tasks, for communication, for home activities and entertainment, or for implementing different types of software [2]. In a similar case to software, there are also security issues and risks when it comes to hardware, but the scope is much more loosely defined when discussing hardware security [3]. The physical chips and boards present in electronics, embedded/IoT devices, networks and even cyber-physical systems are also considered hardware components [4].

Security issues stem from vulnerabilities referring to the weaknesses found in the design process and implementation of hardware architecture, which can be exploited by a hacker or perpetrator to mount an attack. The common vulnerabilities enumeration (CVE) is the most accredited source of information about security vulnerabilities. Hardware is usually manufactured before or during software development, but cannot be easily updated like software, yet hardware executes the software that controls a cyber-physical system [5]. Hence, it is often the last line of defense against potential attacks; that is, if the attack reaches the hardware, the damage may be permanent or irreversible. Cyberattacks target software by targeting the flaws in hardware design since they are undetectable and do not leave a software trace in the log files of that system. Hence, attacks like this on hardware formulate dangerous risks on any system using flawed hardware designs. The MITRE corporation defines weakness as a weakness present inside a component of a computer that, "when exploited, results in a negative impact on confidentiality, integrity or availability" [6]. The common weakness enumeration (CWE) is operated by MITRE corporation and is supported by the department of homeland security in the U.S. [7].

CWE is a community-developed list that includes different types of ordinary software and hardware weakness that own various security implications. CWE was established as a support system for people

Z. Younis and B. Mahmood are with Department of Computer Science, College of Computer Science and Mathematics, University of Mosul, Mosul, Nineveh, Iraq. Emails: zainab.younis@uomosul.edu.iq and bmahmood@uomosul.edu.iq

with hardware, software systems, or networks that are vulnerable to attack. Potential attackers would target weaknesses in the form of bugs, faults, or other specific flaws in hardware or software architecture, design, code, or execution. There is what is called the CWE list along with an associated classification taxonomy, in which both act as terminology that can recognize and communicate such weaknesses in CWE terms [8]. The main aim of CWE is to support programmers, hardware architects, and designers on how to avoid, deal with and eliminate common mistakes before products are delivered. This way, vulnerabilities can be stopped at the source, which is the main target of CWE servicing developers and practitioners in security.

In the end, using CWE helps avoid and prevent various kinds of security weaknesses that have haunted the hardware and software industries putting enterprises at risk [9]. CWE has published a catalog of the hardware weaknesses and made it readily available for interested parties assisting individuals and organizations to fully understand the reasons for weaknesses occurrence in applications and other cyber-enabled capabilities. CWE takes pride in originating from the fact that their work is derived from actual real-world examples of various weaknesses that appear in software applications. Conceptual patterns that make software and hardware exploitable by potential attackers are then discovered and generalized out of these weaknesses to assist designers and developers in recognizing and learning them at an early stage in the development lifecycle of software or hardware for either their avoidance completely or their quick identification to address them before the software program is put into operation.

The entries in the CWE system, or what are simply referred to as CWEs, are an amalgamation of the types of weaknesses discovered by exploiting them “in the wild” or as they happen in different situations through investigative testing and examination of software by testers, developers and hackers. There are several specialists, including academics, representatives of government agencies and research institutions, information security tool vendors, and major operating system vendors that constitute the international CWE community. These representatives create the CWEs by discovering a particular weakness in a product and making it general information or through examining software architecture, code, design, or deployed applications and finding flaws that may permit a potential adversary to infiltrate the system and do undesirable things. Discovering these weaknesses early on presents an opportunity to identify how an attacker may leverage a weakness and how a CWE community member or defender can remove the weakness.

If a company or organization has experienced a previous attack on their software or hardware, consequentially making them interested in a particular type of weakness, then the CWE institution can exploit the relationships between CWEs and common attack patterns enumeration and classification (CAPEC) to predict or foresee future infiltrations and hence suggest defense methods. Moreover, CWE is organized by their properties where a list of possible properties and definitions of the properties can be found on MITRE’s website. CWE and SANS institute for security have established themselves as the most prominent organizations providing security and practical information on applications that are unbiased [4]. Open web application security projects (OWASP) and CWE/SANS are also mentioned by [5] as being the most popular entities in the field. There have been several different research works conducted in the past few years on CWE and some of such studies are reviewed in Section 2 of this study. When discussing hardware weaknesses, it is noteworthy to mention that they can be functional (e.g., the core function of the hardware) or nonfunctional (e.g., performance, availability, …etc.) depending on the nature of a system and its usage scenarios. Usually, an adversary identifies a weakness or sometimes even more than one weakness and then exploits that weakness and this is what constitutes a typical attack [5].

Occasionally, a piece of hardware -such as a computer’s memory- might experience unexpected behavior in certain situations, which in turn can be taken as an advantage by cyber attackers [10]. Previous research has revealed different kinds of attacks that have compromised various platforms, such as private computers [11], internet-related browsers [12]-[13], cloud-based virtual systems [14], and smartphones [15]. These attacks have been employed to intensify privileges [15], detect cryptographic keys [16], expose online connected systems [17], or lock down a processor [18]. Safety-critical hardware is also prone to attacks, as it can have weaknesses. An exploitable bug, for instance, was discovered in the Actel ProASIC3 and has been utilized by the military [19], automotive and medical applications [20], and the Boeing 787 aircraft [21].

Consequently, to avoid weaknesses, CWE and professional hardware designers provide best practices

and technical support for hardware weaknesses. Complex networks area is a field in computer science that models problems in the form of a graph with connected nodes and edges. The complex networks method is a multidisciplinary approach that can be used in addressing issues by investigating the relations among nodes and edges [22]. This approach is widely used in the computer security literature for investigating a variety of issues [19, 23]. Hence, this study utilizes the concepts of complex networks to investigate hardware vulnerabilities. This study is organized as follows: Section 2 contains some of the related works explained in a laconic way. Section 3 includes a description of the data collection process and the strategy followed in generating the network. Section 4 presents the visualizations and the obtained results. Finally, the paper's conclusion is given in Section 5.

2. RELATED WORK

Some of the most recent works on CWE weaknesses involved relating software weaknesses using complex networks [24] and making recommendations to software developers turning their attention to some of the neglected weaknesses when designing software. Later, the authors of [25] introduced a method to rank and prioritize weaknesses listed by CWE/SANS and OWASP. This method is usually used as a way for increasing CWSS accuracy. In other studies, such as [26], open-source tools are utilized for discovering, identifying, and classifying possible and emerging weaknesses, in addition to describing their advantages and disadvantages. The authors introduced a new weakness checking tool called *HardVul*, which is supposedly able to run on any kind of architecture, and then they reported what they found in their testbed. Similarly, the performance of different architectures was evaluated using the benchmark suite in [26], which was primarily produced to measure user-friendliness in the proposed operating system-friendly microprocessor architecture (OSFA). The authors ended up making it work on other architectures; therefore, studies that compare performance and reduce impact and overhead can be conducted. CWE has listed the findings in its memory corruption section.

Furthermore, research has also been conducted on both software and hardware security and their related weaknesses. For example, the authors of [27] proposed a contemporary and innovative methodology for an HLS-based security-aware system that creates architectures that are efficient (in terms of resources, energy, and performance) whilst also being secure. By doing so, the researchers highlighted then-emerging challenges that had to be faced and overcome by high-level synthesis (HLS) tools to have secure hardware accelerators. The discovery of these challenges led the authors to commence a discussion around hardware weaknesses mentioned in CWE list1 and they focused on how they can alter accelerator behavior through the exploitation of errors in hardware design. Other research articles have had a dissimilar approach, investigating the threats themselves and how to combat them or be proactive in their avoidance at the stage of software and hardware development. A general threat model is presented in [28] about visual sensor network (VSN) applications and their components exploiting their attack surfaces. The STRIDE taxonomy and CWE were used to classify the outlined threats and their weaknesses, respectively, both being considered as popular taxonomies for security weaknesses. After developing a threat model, which displays the possible methods an adversary can compromise the system, a tool for analyzing the threat is used to quantify the risks of a potential attack vector, after which priorities can be decided upon for the mitigation of the security issue. The authors of [28] bring forth a threat model that is relevant and complimentary for previous research in the fields of wireless sensor networks (WSNs) and the internet of things (IoT).

Related works of a slightly different purpose have investigated the efforts spent by academics and industry professionals in documenting and classifying the existing hardware security landscape. The contributions of [29] lie in the examination of joint efforts in a community in the categorization of hardware weaknesses paying particular attention to the CWE database. The authors discovered at the time that these efforts ranged from making classifications of known weaknesses and presenting them as a taxonomy to suggest best practices for the identification, mitigation, and prevention of these issues when designing a product. This is the aim that most research studies on software and hardware weaknesses have in common. The researchers of [30], for example, shed light on hardware weaknesses particular to the internet of things (IoT) and produced an ontology-driven storytelling framework (OSF). This OSF aims at identifying recurrent patterns revealing weaknesses over time, which in turn can be used to assist in mitigating the negative effects of weaknesses or at the least predict and prevent future weaknesses. To provide a profound analysis of the weaknesses and weaknesses found in IoT within CWE and CVE datasets and to be able to study how they are connected and related, in addition to

providing insight for the prevention of emerging weaknesses and their effects, the authors utilized contemporary natural language processing and machine-learning techniques.

Recent research such as [31] have tackled hardware security more generally and presented a basis or a so-called foundation that was established by industry researchers and academics that support the realization of an eco-system of CAD tools that are security-aware, especially covering hardware security and fault-injection assessment for SoC designs and security assurance standardization for electronic design integration. All this was to encourage the creation and development of what they named CAD tools for design for security and security validation and assurance. To polish off this review of related works, it is worth noting that [32] surveyed existing frameworks for public weakness and weakness sharing, examining their efficacy for hardware, and identifying potential gaps. The authors also intended to address potential risks and present potential benefits through analyzing hardware weakness reporting efforts and discussing how to quantify security for hardware. This work focuses on discovering relationships among hardware weaknesses and weaknesses listed by CWE using a complex network approach. From the reviewed studies, it is noticed that the research opportunities still stand for performing various research works in this field.

Accordingly, there is a severe lack in providing studies that consider the relations among hardware security weaknesses during the design phase. This is important, since a weakness may become a side effect of other weaknesses or cause others, which is not considered in the literature. Moreover, it is believed that the scoring system of CWE should focus more on the relations among weaknesses to have more dimensions about the severity of weaknesses. Hence, this work comes to deal with this issue and delivers an approach that models the hardware security weaknesses reported by CWE and reveals the most dangerous ones based on their relations to other weaknesses. The model depends on the notions of complex networks. Likewise, the proposed model can provide a deep view of the weaknesses and the relations among them. What makes this work unique is that the network science approach is utilized to investigate the relations among CWE hardware weaknesses profoundly and provide recommendations to software developers. Moreover, this procedure can be added to the scoring scheme of CWE aiming at having more dimensions about the severity level of weaknesses, which benefits software developers and hardware architects.

3. RESEARCH METHODOLOGY

3.1 Dataset Formation

In this study, the data was gathered from the most accredited source of security weaknesses, the CWE. The strategy followed in the data collection process was based on collecting all the security weaknesses, including the hardware and software ones. In the CWE, each weakness is classified and weaknesses are categorized under certain classes and categories based on the nature of weakness. According to CWE, the classes can be the following: (a) research concepts that deal with the theoretical aspect of the weaknesses; (b) software development that relates to weaknesses that are frequently faced during software design; (c) hardware design that deals with the weaknesses that are often faced through the design; (e) architectural concepts that relate to weaknesses of the software architectural design. It should be mentioned that many weaknesses in CWE are classified under more than one class (mixed).

This means that the weakness may belong to two or three classes at the same time. Besides, a category, in CWE, includes a group of weaknesses that are similar or highly related to each other. Moreover, the CWE provides detailed information about each weakness in terms of relation to other weaknesses. For instance, a weakness W_i from a particular class C_m and a particular category G_n may have relations to other weaknesses that belong to the same or different classes or categories. These relations can be one of the types of relations defined by CWE as follows: W_i is "*ParentOf*" W_j: W_i represents the parent of W_j; W_i is "*MemberOf*" W_j: W_i is in the same category as W_j; W_i is "*ChildOF*" W_j: W_i is a child of W_j; W_i is "*PeerOf*" W_j: W_i *is* like W_j.

Based on the above kinds of relations, the dataset was created accordingly. Hence, the dataset includes the following information for each weakness: *identifier (ID), CWE code, name, class, category, list of relations* to other weaknesses. The data collection process includes 1013 weaknesses from different classes and categories and 2913 relations connecting them. Furthermore, this work considers the hardware design weaknesses and their related attack patterns. However, the key aim is to investigate

hardware security weaknesses. For these weaknesses, the related attack patterns are given to provide the developers with more information about such weaknesses.

3.2 Network Generation and Metrics

As mentioned in Section 1, this study is inspired by complex networks, that is; in turn, based on graph theory. Using this theory, a given problem can be formalized as nodes connected by edges. Accordingly, each security weakness (hardware and software) is considered as a node. An undirected edge is created between two weaknesses *if and only if* one of the relation types in Section 2.1 is held (see Figure 1). This strategy was followed in [24] and [25] when generating a network of weaknesses. In contrast, if there is no relation between a given pair of weaknesses, then no edge is created for this pair. After considering this strategy, a complex network of weaknesses is generated. This network will be analyzed using network measurements at two levels of node and network. Figure 2 demonstrates the preliminary visualization of the network, in that, different colors reflect different classes considered in the created dataset, and nodes size reflects the frequency of connections with other weaknesses. In Figure 2, the colors of nodes refer to the class type of the weaknesses (Yellow = (Research Concepts-Software Development) class, Pink = Hardware Design class, Green = Software Development class, Blue = Attack Pattern class, Brown = Research Concepts class, Dark Green = Architectural Design class and Light Blue = (Research Concepts-Software Development-Architectural Design) class). Node's size reflects the frequency of relation of a weakness; larger sizes denote a high frequency of relations.

Figure 1. Edge creation between two weaknesses.

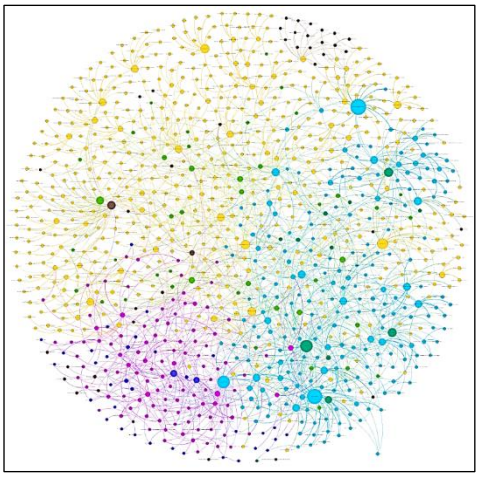

Figure 2. Basic visualization of the CWE network of weaknesses.

After generating the network of CWE weaknesses, the network metrics that are used in the evaluation can be summarized in the context of this work as follows [24]-[25]:

- *Average Degree*: the average number of connections of the weaknesses in the network.
- *Diameter*: the interval between the network's outmost weaknesses (longest connected weaknesses).
- *Density*: the actual number of connections among the weaknesses to the number of all possible connections when reaching a fully connected network.
- *Average Path Length*: the average shortest paths of any given pair of weaknesses in the network.
- *Clustering Coefficient*: the tendency of the weakness to cluster together with other weaknesses in the network and can be calculated as follows [22]:

$$C_i = \frac{2\left|\left\{w_{jk} : w_j, w_k \in N_i, w_{jk} \in R\right\}\right|}{k_i\left(k_i - 1\right)} \quad (1)$$

where R is network relations, w_{jk} is a weakness between the weaknesses w_j and w_k; N_i is the overall count of weaknesses and w_i is the neighbors' weakness. The mean of clustering coefficient (C) in the network is formulated as follows [22]:

$$C_G \frac{\sum_{i=1}^{n} C_i}{N} \quad (2)$$

where C_i is given in Eq. 1; N represents the weaknesses number.

- *Betweenness Centrality*: the number of times that the weakness occurs in the shortest paths of the other weaknesses in the network. In complex networks, this metric reflects the importance of a specific node within a network [22]. Accordingly, it shows how influential a weakness is to be a cause or a side effect of other weaknesses within the network and can be calculated using the following formula [22]:

$$B_c\left(W_j\right) = \sum_{i \neq j \neq k} \frac{\sigma_{i,k}\left(W_i\right)}{\sigma_{i,k}} \tag{3}$$

where $\sigma_{i,k}(W_i)$ is the number of shortest paths between weakness (W_i) and (W_k) passing through the weakness (W_j). The above equation is used for all the available pairs of weaknesses in the network.

- *Closeness Centrality*: reflects how close nodes are to each other [22]. It is an indicator of how close a weakness is to other weaknesses and can be calculated using the following equation [22]:

$$C_c\left(W_j\right) = \frac{N-1}{\sum_k \sigma_{j,k}} \tag{4}$$

where N is the number of weaknesses in the network, $\sigma_{i,k}$ is the shortest path between the weakness j and k.

- *Eigenvector Centrality*: reflects how well-connected a particular node is to the other nodes within a network. This metric shows the degree of connectivity a weakness has to the highly connected weaknesses in the network. If w and z are considered weaknesses, $a_{v,t}$ equals one if they are connected and zero otherwise. The u score for a weakness w is calculated as follows:

$$u(w) = \frac{1}{\lambda} \sum_{z \in M_{(w)}} u_z = \frac{1}{\lambda} \sum_{z \in G_{(w)}} a_{w,z} u_z \tag{5}$$

where G is the network of weaknesses, $M(w)$ is the adjacent of weakness w and λ is the eigenvalue.

4. RESULTS AND DISCUSSION

The first step in the analysis of this work is to visualize the network showing the hardware design weaknesses and attack patterns including the main characteristics of the generated network. Figure 3 shows the visualization of the CWE network of weaknesses with three main types of weaknesses: hardware design weaknesses (pink nodes), attack patterns (blue nodes), and the other weaknesses that belong to other classes (cyan nodes). Table 1 presents the main characteristics of the network.

Table 1. Characteristics of the CWE network of weaknesses.

# of Nodes	# of Edges	Average Degree	Average Clustering Coefficient	Diameter	Density	Average Path Length
1013	2913	5.571	0.155	10	0.006	5.064

Figure 3 reveals how the hardware design weaknesses and attack pattern are highly connected with the other classes of weaknesses (see also Figure 4). This can be considered as an indicator of the impact of non-hardware weaknesses on hardware security design, which is an interesting fact. Based on Table 1, the characteristics of the network also reflect some facts. The average degree of 5.571 shows the weak level of connections among the weaknesses, which is also confirmed when observing the average clustering coefficient that reflects a weak tendency of weaknesses to cluster together. The diameter reflects a long distance from the farthest weaknesses in the network. This is evident since the mean path size is 5.064 with a density level of 0.006. Moreover, the degree distribution of the nodes in the CWE network of weaknesses follows a power-law distribution as demonstrated in Figure 5, in that the x-axis represents the frequency of connections and the y-axis is the number of weaknesses. This means that few weaknesses appear with a high frequency of connections, while many of them have few connections. It can be inferred, according to the Pareto Rule [33], that 20% of the weaknesses dominate the

connections in the network. These metrics indicate that the relations among weaknesses are most likely restricted by the classes and the categories of the weaknesses, as shown in Figure 6. The relations are denser within the same class and less across classes. In Figure 6, the classes are encoded with different colors as follows: Yellow = (Research Concepts-Software Development) class, Pink = Hardware Design class, Green = Software Development class, Blue = Attack Pattern class, Brown = Research Concepts class, Dark Green = Architectural Design class and Light Blue = (Research Concepts-Software Development-Architectural Design) class). Node's size reflects the frequency of relation of a weakness; larger sizes denote a high frequency of relations.

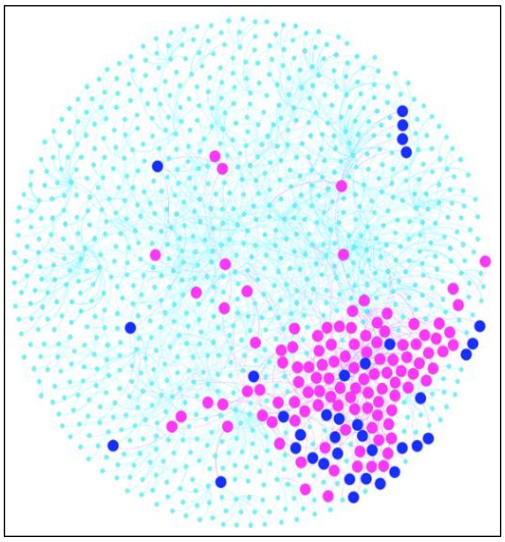

Figure 3. CWE network visualization showing the connections of the hardware design and attack patterns.

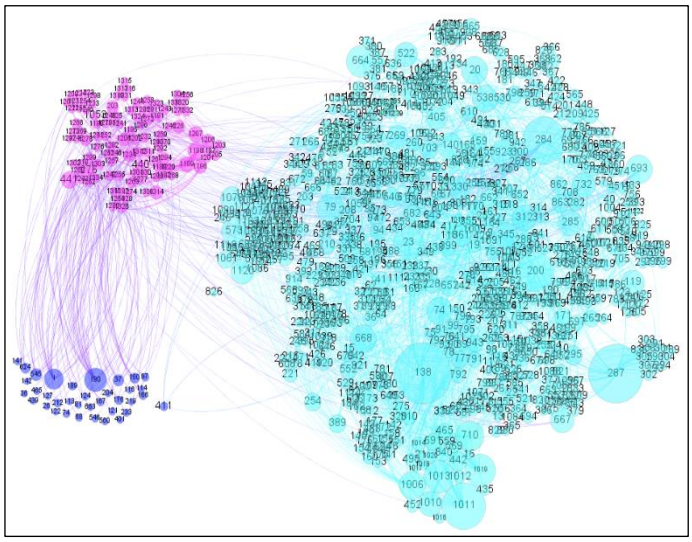

Figure 4. The relations between the hardware design weaknesses (pink color), attack patterns (blue), and other classes' weaknesses (cyan).

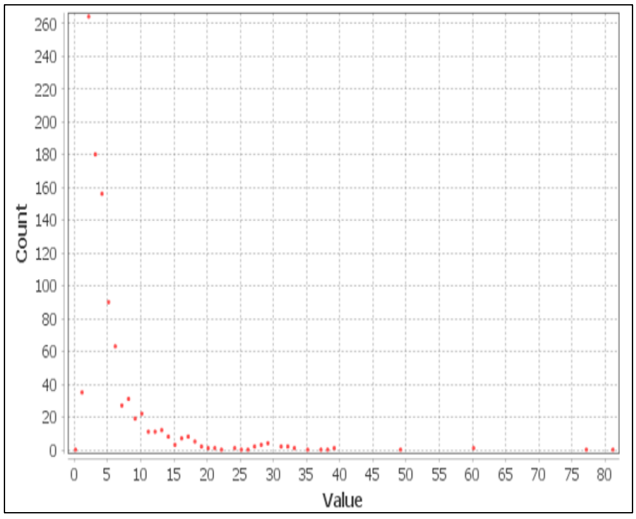

Figure 5. Degree distribution of the CWE network of weaknesses.

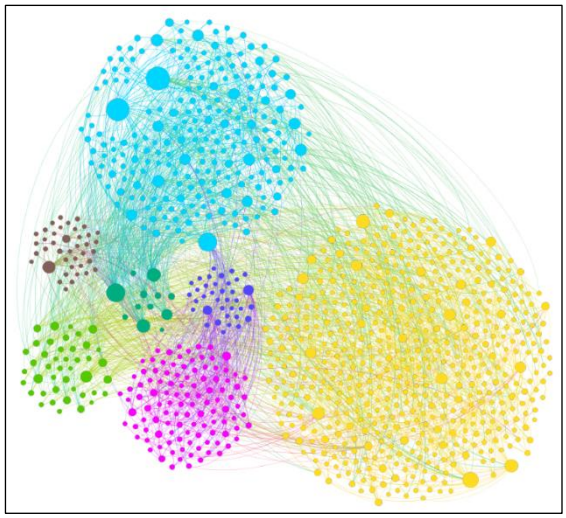

Figure 6. The density of the relations among different kinds of classes of weaknesses.

Now, the next step is to analyze the hardware weaknesses and attack patterns. Hardware weaknesses and attack pattern nodes are extracted from the main network, as shown in Figure 7. Surprisingly, the attack patterns do not have relations to each other, in that all their relationships are with weaknesses from other classes. The reason behind this case is that all the attack patterns are not original, and they are originated from other classes' weaknesses. For instance, the "Improper Resource Locking (CWE-413)" is a ChildOf the "Improper Locking (CWE-667)" weakness that belongs to the research concepts class.

Alternatively, the hardware design weaknesses are better connected compared to the attack patterns. Figure 8 depicts the hardware weaknesses and how they are connected. In this figure, nodes size reflects

the value of betweenness centrality (the larger the size, the higher the value of betweenness centrality). This means that the most influential weaknesses have larger sizes. Moreover, the number of weaknesses in the network is 107 connected by 140 relations (edges). As mentioned, the betweenness centrality measurement reflects how influential a weakness is in a community of weaknesses; that is, it represents the number of times a weakness is positioned in the shortest paths of the other pairs of weaknesses. Therefore, the hardware weaknesses are ranked using their betweenness centrality values as presented in Table 2. Besides, not all hardware weaknesses have appeared in the table, because values have dropped to zero. The table also presented the other metrics for each weakness (degree centrality, closeness centrality, eigencentrality, and clustering coefficient).

Based on Table 2, the CWE-441 (unintended proxy or intermediary ("confused deputy") obtained the highest betweenness value, which means that it is the most influential hardware design weakness. This weakness relates to the access control issues when an unintended proxy is performed. This matter should be given more attention by software developers since it appears more frequently in the shortest paths of the network pairs of weaknesses. In the second rank, the CWE-1208 appears, which is a category of weaknesses that relate to improper protection of hardware. As can be seen, many weaknesses have not been given enough focus in the literature, but they have a significant impact on the security of hardware design.

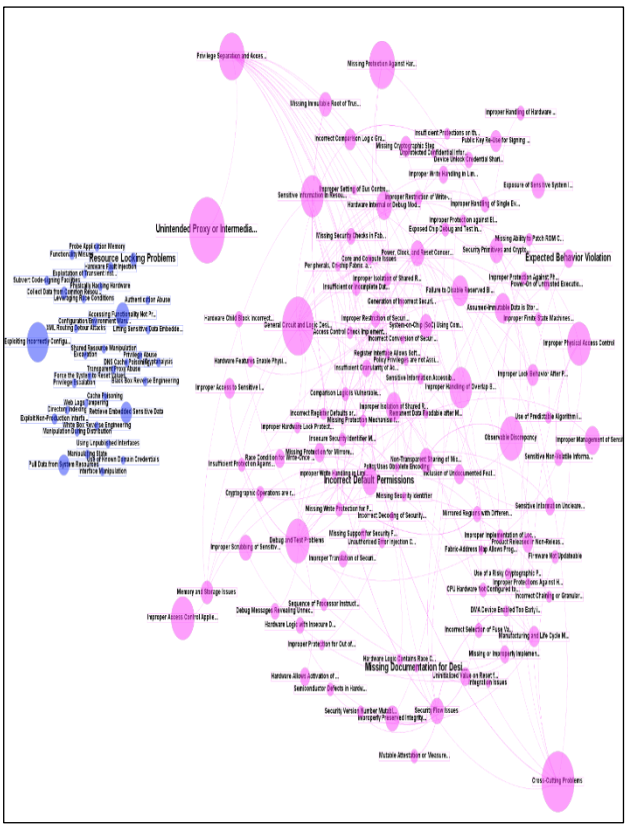

Figure 7. Visualization of the hardware design weaknesses (pink nodes) and attack patterns (blue nodes).

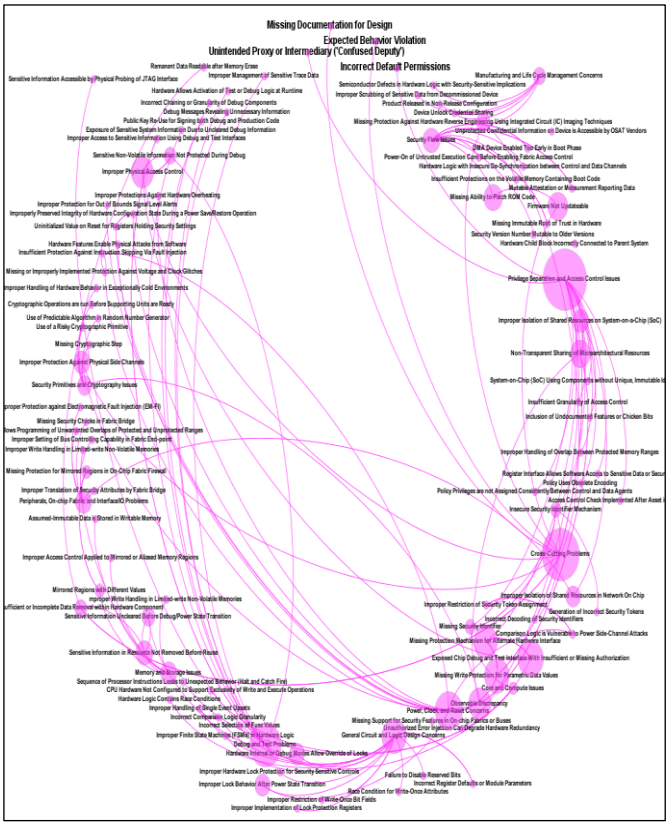

Figure 8. Visualization of the connections among hardware design weaknesses.

Furthermore, in terms of clustering coefficient, three weaknesses have strong tendencies to cluster with other weaknesses as a side effect of a cause. These weaknesses gained the highest levels of clustering coefficient; CWE-1189 (improper isolation of shared resources on system-on-a-chip (SoC)), CWE-276 (incorrect default permissions), and CWE-1304 (improperly preserved the integrity of hardware configuration state during the power save/restore operation). Software developers should be aware of the risk of these three weaknesses since they may impact the whole system in terms of security.

As can be observed in Table 2, the CWE-276 surprisingly has the highest value of eigencentrality. This means that it is connected to the highly connected weaknesses in the network, which makes it more dangerous when compared to the other network weaknesses. From the closeness centrality levels, it can

be noticed that most of the weaknesses in Table 2 have close levels, meaning that they are considered close to most of the weaknesses in the network. Many interesting facts can be extracted from Table 2 since it includes the best-connected CWE weaknesses. It should be mentioned that the values presented in Table2 were extracted from the whole network of weaknesses. In Table2, the CWE code of each weakness is hyperlinked to its web page CWE website. Table 3 presents the most recent list of the most important hardware weaknesses in CWE in 2021, in that only two weaknesses in Table 2 (bold and underlined) are shown in the 2021 most recent list of CWE (Table 3). This means that more attention should be given to the rank in Table 2 and the relations among weaknesses are crucial to be considered by software developers during the design phase.

By looking at the results, visualizations, and network metrics values, a better understanding can be obtained of the weaknesses and their relations to each other by hardware designers. Moreover, the outcomes provided in this work may consume time and effort during the design, testing, and maintenance phases. This is important since they reduce the total software cost. Finally, more analysis is needed of network edges that may hide some unseen patterns about the weaknesses. This can be performed by generating a network of all the weaknesses available in the CWE list.

Table 2. Prioritizing the hardware design weaknesses according to network centrality measurements and ranking them based on their betweenness.

RANK	CWE	Weaknesses	Category	Centrality Measurements				Clustering Coefficient
				Betweenness	Degree	Closeness	Eigen	
1st	441	Unintended Proxy or Intermediary ('Confused Deputy')	Privilege Separation and Access Control Issues	0.0154	16	0.2420	0.0559	0.0250
2nd	1208	Cross-Cutting Problems	Cross-cutting Problems	0.0140	10	0.2213	0.0218	0.0222
3rd	1199	General Circuit and Logic Design Concerns	General Circuit and Logic Design Concerns	0.0135	14	0.2295	0.0296	0.0110
4th	1278	Missing Protection against Hardware Reverse Engineering Using Integrated Circuit (IC) Imaging Techniques	Manufacturing and Life Cycle Management Concerns	0.0106	8	0.2291	0.0282	0.0000
5th	1198	Privilege Separation and Access Control Issues	Privilege Separation and Access Control Issues	0.0104	18	0.2287	0.0677	0.0261
6th	1207	Debug and Test Problems	Debug and Test Problems	0.0096	14	0.2114	0.0417	0.0000
7th	203	Observable Discrepancy	Security Primitives and Cryptography Issues	0.0096	10	0.2082	0.0160	0.0222
8th	1263	Improper Physical Access Control	Cross-cutting Problems	0.0094	7	0.2342	0.0480	0.0000
9th	1257	Improper Access Control Applied to Mirrored or Aliased Memory Regions	Memory and Storage Issues	0.0093	5	0.2350	0.0572	0.0000
10th	226	Sensitive Information in Resource Not Removed Before Reuse	Memory and Storage Issues	0.0090	10	0.2095	0.0232	0.0667
11th	**1260**	**Improper Handling of Overlap Between Protected Memory Ranges**	**Privilege Separation and Access Control Issues**	**0.0088**	**4**	**0.2360**	**0.0544**	**0.0000**
12th	1209	Failure to Disable Reserved Bits	General Circuit and Logic Design Concerns	0.0060	3	0.2145	0.0137	0.0000
13th	1282	Assumed-Immutable Data is Stored in Writable Memory	Memory and Storage Issues	0.0058	5	0.2417	0.0327	0.0000
14th	**1189**	**Improper Isolation of Shared Resources on System-on-a-Chip (SoC)**	**Privilege Separation and Access Control Issues**	**0.0055**	**6**	**0.2316**	**0.0259**	**0.2000**
15th	1234	Hardware Internal or Debug Modes Allow Override of Locks	Belonging to 2 Categories in Hardware Weaknesses.	0.0055	9	0.2113	0.0458	0.0000

17th	1323	Improper Management of Sensitive Trace Data	Debug and Test Problems	0.0053	5	0.2272	0.0441	0.0000
18th	276	Incorrect Default Permissions	Privilege Separation and Access Control Issues	0.0053	11	0.2316	0.1035	0.2182
19th	1196	Security Flow Issues	Security Flow Issues	0.0047	11	0.2041	0.0272	0.0000
20th	1304	Improperly Preserved Integrity of Hardware Configuration State during a Power Save/Restore Operation	Power, Clock and Reset Concerns	0.0047	6	0.2376	0.0517	0.1333
21st	1266	Improper Scrubbing of Sensitive Data from Decommissioned Device	Manufacturing and Life Cycle Management Concerns	0.0046	6	0.2036	0.0137	0.0000
22nd	1206	Power, Clock and Reset Concerns	Power, Clock and Reset Concerns	0.0046	11	0.2144	0.0333	0.0364

Table 3. The 2021 list of the CWE's most important hardware weaknesses.

#	CWE	Title
1	1189	Improper Isolation of Shared Resources on System-on-a-Chip (SoC)
2	1191	On-Chip Debug and Test Interface with Improper Access Control
3	1231	Improper Prevention of Lock Bit Modification
4	1233	Security-Sensitive Hardware Controls with Missing Lock Bit Protection
5	1240	Use of a Cryptographic Primitive with a Risky Implementation
6	1244	Internal Asset Exposed to Unsafe Debug Access Level or State
7	1256	Improper Restriction of Software Interfaces to Hardware Features
8	1260	Improper Handling of Overlap between Protected Memory Ranges
9	1272	Sensitive Information Uncleared before Debug/Power State Transition
10	1274	Improper Access Control for Volatile Memory Containing Boot Code
11	1277	Firmware Not Updateable
12	1300	Improper Protection of Physical Side Channels

5. CONCLUSIONS

In this study, a thorough sight of important hardware weaknesses is provided *via* the creation of a network model that contains the most common weaknesses reported in common weakness enumeration. In addition, the generation of the network considered three main types of weaknesses, which are hardware design, attack patterns, and other classes. The study showed how hardware design and attack patterns are highly connected. Moreover, it demonstrates how the hardware design weaknesses are better connected when compared to the attack patterns. The analysis revealed that the CWE-441is the most influential hardware design weakness. This weakness relates to the access control issues when the unintended proxy is performed. It also showed that the CWE-1189, CWE-276, and CWE-1304 gained the highest levels of clustering coefficient, which means that software developers should be aware of the risk of these three weaknesses, since they may impact the whole system security. As future work, we plan to combine all the weaknesses provided by CWE in one network model and deeply explore and reveal the unseen facts about the weaknesses.

ACKNOWLEDGMENTS

The authors are grateful to the staff and faculty members at the Department of Computer Science at the University of Mosul for supporting them constantly. The authors also would like to thank the contributors of the CWE for their efforts in providing worldwide researchers with data on security weaknesses.

REFERENCES

[1] T. Gaddis, Starting out with Python, ISBN-13: 978-0134444321, Harlow, UK: Pearson, 2018.

[2] A. Sengupta, "Hardware Vulnerabilities and Their Effects on CE Devices: Design for Security against Trojans [Hardware Matters]," IEEE Consumer Electronics Magazine, vol. 6, no. 3, pp. 126–133, 2017.

[3] M. Alenezi, M. Zagane and Y. Javed, "Efficient Deep Features Learning for Vulnerability Detection Using Character N-gram Embedding," Jordanian Journal of Computers and Information Technology, vol. 7, no. 1, pp. 25-38, 2021.

[4] P. A. Wortman, F. Tehranipoor and J. A. Chandy, "Exploring the Coverage of Existing Hardware Vulnerabilities in Community Standards," Proc. of the Silicon Valley Cybersecurity Conference (SVCC2020), pp. 87–97, DOI:10.1007/978-3-030-72725-3_6, 2021.

[5] G. Bloom, E. Leontie, B. Narahari and R. Simha, "Hardware and Security: Vulnerabilities and Solutions," Chapter 12, pp. 305-331, Handbook on Securing Cyber-Physical Critical Infrastructure, Morgan Kaufmann, 2012.

[6] CVE, "Terminology," [Online], Available: https://cve.mitre.org/about/terminology.html, [Accessed: 27-Oct-2021].

[7] B. Martin, "Common Vulnerabilities Enumeration (CVE), Common Weakness Enumeration (CWE) and Common Quality Enumeration (CQE)," ACM SIGAda Ada Letters, vol. 38, no. 2, pp. 9–42, 2019.

[8] CWE, "Common Weakness Enumeration," [Online], Available: https://cwe.mitre.org/index.html, [Accessed: 28-Oct-2021].

[9] S. Bhunia and M. H. Tehranipoor, Hardware Security: A Hands-on Learning Approach, ISBN-13: 978-0128124772, Cambridge, MA: Morgan Kaufmann Publishers, 2019.

[10] C. Li and J.-L. Gaudiot, "Detecting Malicious Attacks Exploiting Hardware Vulnerabilities Using Performance Counters," Proc. of the 43rd IEEE Annual Computer Software and Applications Conference (COMPSAC), pp. 588-597, DOI: 10.1109/COMPSAC.2019.00090, Milwaukee, WI, USA, 2019.

[11] M. Seaborn and T. Dullien, "Exploiting the DRAM Rowhammer Bug to Gain Kernel Privileges," Black Hat Briefings, pp. 1-71, [Online], Available: https://www.blackhat.com/docs/us-15/materials/us-15-Seaborn-Exploiting-The-DRAM-Rowhammer-Bug-To-Gain-Kernel-Privileges.pdf, 2015.

[12] E. Bosman, K. Razavi, H. Bos and C. Giuffrida, "Dedup Est Machina: Memory Deduplication As an Advanced Exploitation Vector," Proc. of the IEEE Symposium on Security and Privacy (SP), pp. 987-1004, DOI 10.1109/SP.2016.63, San Jose, CA, USA, 2016.

[13] D. Gruss, C. Maurice and S. Mangard, "Rowhammer.js: A Remote Software-induced Fault Attack in JavaScript," Proc. of the 13th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA), arXiv:1507.06955, pp. 300–321, 2016.

[14] Y. Xiao, X. Zhang, Y. Zhang and R. Teodorescu, "One Bit Flips, One Cloud Flops: Cross-VM Row Hammer Attacks and Privilege Escalation," Proc. of the 25th USENIX Security Symposium (USENIX Security 16), pp. 19-35, Austin, TX, USA, 2016.

[15] V. van der Veen, Y. Fratantonio, M. Lindorfer, D. Gruss, C. Maurice, G. Vigna, H. Bos, K. Razavi and C. Giuffrida, "Drammer: Deterministic Rowhammer Attacks on Mobile Platforms," Proc. of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 1675–1689, DOI: 10.1145/2976749.2978406, 2016.

[16] K. Razavi, B. Gras, E. Bosman, B. Preneel, C. Giuffrida and H. Bos, "Flip Feng Shui: Hammering a Needle in the Software Stack," Proc. of the 25th USENIX Security Symposium (USENIX Security 16), pp. 1-18, Austin, TX, USA, 2016.

[17] A. Tatar, R. K. Konoth, E. Athanasopoulos, C. Giuffrida, H. Bos and K. Razavi, "Throwhammer: Rowhammer Attacks over the Network and Defenses," Proc. of the USENIX Annual Technical Conference (USENIX ATC 18), pp. 213-226, Boston, MA, USA, 2018.

[18] Y. Jang, J. Lee, S. Lee and T. Kim, "SGX-Bomb: Locking Down the Processor *via* Rowhammer Attack," Proc. of the 2nd Workshop on System Software for Trusted Execution, pp. 1-6, DOI: 10.1145/3152701.3152709, 2017.

[19] A. Ferraiuolo, R. Xu, D. Zhang, A. C. Myers and G. E. Suh, "Verification of a Practical Hardware Security Architecture through Static Information Flow Analysis," ACM SIGARCH Computer Architecture News, vol. 45, no. 1, pp. 555–568, 2017.

[20] T. Yaqoob, H. Abbas and M. Atiquzzaman, "Security Vulnerabilities, Attacks, Countermeasures and Regulations of Networked Medical Devices—A Review," IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 3723–3768, 2019.

[21] A. Stander and J. Ophoff, "Cyber Security in Civil Aviation," Imam Journal of Applied Sciences, vol. 1, no. 1, pp. 23-26, 2016.

[22] R. Albert and A.-L. Barabási, "Statistical Mechanics of Complex Networks," Reviews of Modern Physics, vol. 74, no. 1, pp. 47–97, 2002.

[23] C. Li, Securing Computer Systems through Cyber Attack Detection at the Hardware Level, PhD Thesis, University of California, Irvine, 2020.

[24] Z. K. Younis and B. Mahmood, "Towards the Impact of Security Vulnerabilities in Software Design: A Complex Network-based Approach," Proc. of the 6th Int. Engineering Conf. "Sustainable Technology and Development" (IEC), pp. 157-162, DOI: 10.1109/IEC49899.2020.9122923, Erbil, Iraq, 2020.

[25] B. Mahmood, "Prioritizing CWE/SANS and OWASP Vulnerabilities: A Network-based Model," International Journal of Computing and Digital Systems, vol. 10, no. 1, pp. 361–372, 2021.

[26] S. Trecakov, C. Tran, H. Badawy, N. Siddique, J. Acosta and S. Misra, "Can Architecture Design Help Eliminate Some Common Vulnerabilities?" Proc. of the 14th IEEE Int. Conf. on Mobile *Ad Hoc* and Sensor Systems (MASS), pp. 590-593, DOI: 10.1109/MASS.2017.100, Orlando, FL, USA, 2017.

[27] C. Pilato, S. Garg, K. Wu, R. Karri and F. Regazzoni, "Securing Hardware Accelerators: A New Challenge for High-level Synthesis," IEEE Embedded Systems Letters, vol. 10, no. 3, pp. 77–80, 2018.

[28] J. Simonjan, S. Taurer and B. Dieber, "A Generalized Threat Model for Visual Sensor Networks," Sensors, vol. 20, no. 13, p. 3629, 2020.

[29] P. A. Wortman, F. Tehranipoor and J. A. Chandy, "Exploring the Coverage of Existing Hardware Vulnerabilities in Community Standards," Proc. of the Silicon Valley Cybersecurity Conference, Virtual, pp. 87–97, [Online], Available: https://svcc2020.svcsi.org/accepted-papers/Exploring-the-Coverage-of-Existing-Hardware-Vulnerabilities-in-Community-Standards, 2021.

[30] C. Bandi, S. Salehi, R. Hassan, S. M. P D, H. Homayoun and S. Rafatirad, "Ontology-driven Framework for Trend Analysis of Vulnerabilities and Impacts in IoT Hardware," Proc. of the 15th IEEE International Conference on Semantic Computing (ICSC), pp. 211-214, DOI: 10.1109/ICSC50631.2021.00045, Laguna Hills, CA, USA, 2021.

[31] S. Aftabjahani, R. Kastner, M. Tehranipoor, F. Farahmandi, J. Oberg, A. Nordstrom, N. Fern and A. Althoff, "Special Session: CAD for Hardware Security - Automation Is Key to Adoption of Solutions," Proc. of the 39th IEEE VLSI Test Symposium (VTS), pp. 1-10, DOI: 10.1109/VTS50974.2021.9441032, San Diego, CA, USA, 2021.

[32] J. Bellay, D. Forte, R. Martin and C. Taylor, "Hardware Vulnerability Description, Sharing and Reporting: Challenges and Opportunities," Proc. of Annual GOMACTech Conf., pp. 1-7, [Online], Available: http://dforte.ece.ufl.edu/wp-content/uploads/sites/65/2021/05/GOMACTech_conf.pdf, 2021.

[33] A. Clauset, C. R. Shalizi and M. E. J. Newman, "Power-law Distributions in Empirical Data," SIAM Review, vol. 51, no. 4, pp. 661–703, 2009.

ملخص البحث:

تلعب معدّات الحاسوب دوراً رئيسياً في حياتنا اليومية. وعلى الرّغم من النّهضة التكنولوجية، ثمّة مسائل تتعلّق بالأمان في معدّات الحاسوب ترجع الى نقاط ضعف في حاجةٍ الى معالجتها بعناية. لذا تقدم هذه الورقة رؤية متعمّقة لنقاط الضّعف في أمان تصميم معدّات الحاسوب، وذلك عبر تقديم نموذج لشبكة يحتوي على نقاط الضّعف الأكثر شيوعاً الواردة في سجلّ نقاط الضّعف الأكثر شيوعاً (CWE). ويتمثل الهدف الأساسي للشّبكة المقترحة في إجراء تحليل معمّق للعلاقات بين التّصاميم المختلفة لمعدّات الحاسوب ونقاط الضّعف المتعلّقة بالأمان في تلك التصاميم. وبناءً على نتائج التّحليل، تقدم هذه الورقة مجموعةً من التّوصيات التي من شأنها أن تفيد مختلف الأطراف، ومنهم مصمّمو الأمان في معدات الحاسوب. وبناءً على ذلك، يرتكز منهج التّحليل الى مجموعةٍ من المفاهيم المستوحاة من مجال علم الشّبكات. وقد تمّ إظهار النّموذج المقترح في شكل مخطط تكون فيه العُقَد هي نقاط الضّعف، في حين تتشكل الحواف إذا كانت نقاط الضّعف مرتبطة ببعضها البعض. وقد تمّ الحصول على نتائج واعدة من الممكن ملاحظتها في نموذج الدراسة. فمثلاً، نقاط الضّعف CWE-441 وCWE-1189 و CWE-276 و CWE-1304 لم تحْظ بالإهتمام الكافي، وعليه فإنّه يجب أخذها بعين الاعتبار من جانب مطوّري البرمجيات. كذلك تمّ تقديم لائحة تبين ترتيب نقاط الضّعف الخاصة بالمعدّات بناءً على قياسات الشّبكة، ومقارنتها بأحدث اللوائح الصادرة لترتيب نقاط ضعف المعدات عن سجلّ نقاط الضعف الأكثر شيوعاً (CWE). وقد وُجد أنّ هناك نقطتي ضعف مشتركتين فقط بين اللائحتين، مما يؤشر الى أنّ لائحة (CWE) لا تأخذ العلاقات بين نقاط الضّعف بعين الاعتبار.