The BESSPIN Scale

1 Introduction

The BESSPIN score is the security figure of merit that is used to evaluate the SSITH hardware. As any security metric, it is more subjective than empirical, but more structured than qualitative. Since the SSITH program fundamentally relies on MITRE CWEs to determine the various weaknesses targeted by SSITH protections, the BESSPIN scale is inspired by the structure of CWSS (MITRE's Common Weakness Scoring System). To make it more intuitive and more relevant to SSITH, we designed our own metrics and rules.

The scale is represented by a percentage, where 0% means there is no protection against any SSITH CWEs, and 100% means that the processor protects against all SSITH CWEs.

2 Formulation

To compute the BESSPIN scale, $\mathcal{B}$, for a given processor, we group the individual CWE test scores into weakness categories. We use a structurally similar method to CWSS in combining the category scores to obtain a separate score per vulnerability class, and the overall BESSPIN scale.

The SSITH CWE set, $\mathcal{S}$, includes all CWEs in the SSITH program picked from the MITRE CWEs. $\mathcal{S}$ is divided into a set of vulnerability classes (see Section 3). Each class, $\mathcal{V} \subset \mathcal{S}$, is also a set of CWEs. Each class is decomposed into some weakness categories or concepts. A category of weaknesses, $\mathcal{C} \subset \mathcal{V}$, is the set of CWEs whose descriptions intersect with the category definition; a CWE can belong to more than a single category. The class and SSITH CWEs can thus be written as:

$$\begin{aligned} \mathcal{V} &= \{cwe : cwe \in \mathcal{C} \ \wedge \ \mathcal{C} \in \mathcal{V}\} \\ \mathcal{S} &= \{cwe : cwe \in \mathcal{V} \ \wedge \ \mathcal{V} \in \mathcal{S}\} \end{aligned} \tag{1}$$

The BESSPIN security evaluation tool assigns each CWE a score value, $score_{cwe}$, from the SCORES enum object in scoreTests.py. Any value conveying a failure of any kind would disqualify the computation of the scale. A failure means that the test did run as expected, or the scorer was unable to score it properly. This leaves us with the following acceptable scores: NONE, DETECTED, LOW, MED, and HIGH. As explained in the security evaluation document, HIGH means the weakness is critical and leads to the lowest score on the BESSPIN scale, and either NONE or DETECTED denotes the best score. For the sake of the scale, the CWE score can thus be normalized as follows:

$$\begin{aligned} \mathbf{S}(cwe) = & \\ 0, \quad & score_{cwe} = \text{HIGH} \\ 1/3, \quad & score_{cwe} = \text{MED} \\ 2/3, \quad & score_{cwe} = \text{LOW} \\ 1, \quad & score_{cwe} = \text{NONE or DETECTED} \end{aligned} \tag{2}$$

The score of a category of weaknesses is the arithmetic mean of the scores of its CWEs:

$$\mathbf{S}(\mathcal{C}) = \overline{\mathbf{S}(cwe)}, \quad cwe \in \mathcal{C} \tag{3}$$

The BESSPIN coefficient, $\beta(\mathcal{C})$, is a measure that reflects the importance of a category of CWEs, $\mathcal{C}$, and its relevance to the SSITH program. It is formalized as:

$$\beta(\mathcal{C}) = \mathbf{TI}(\mathcal{C}) \times \mathbf{AV}(\mathcal{C}) \times \mathbf{ENV}(\mathcal{C}) \times \mathbf{SSITH}(\mathcal{C}) \tag{4}$$

where each of the factors can take the values 1/3, 2/3, and 1:

- **TI**($\mathcal{C}$): As defined by CWSS, the technical impact (TI) is the potential result that can be produced by the weakness, assuming that the weakness can be successfully reached and exploited. This is also related to the possible acquired privilege (AP). It can take the values: 1 for critical, 2/3 for moderate, and 1/3 for limited.
- **AV**($\mathcal{C}$): The access vector (AV) identifies the channel through which an attacker must communicate to reach the functionality that contains the weakness. It can take the values: 1 for user mode, network, or application, 2/3 for supervisor mode, i.e. operating system, and 1/3 for machine mode, i.e. hardware access.
- **ENV**($\mathcal{C}$): The environmental metric group for the BESSPIN scale constitutes of two concepts:

$$\mathbf{ENV}(\mathcal{C}) = \mathbf{BI}(\mathcal{C}) + \mathbf{LDX}(\mathcal{C}) \tag{5}$$

  The Business Impact (BI) is the potential impact to the business or mission if the weakness can be successfully exploited, and LDX is the Likelihood of the weakness to be Discovered and eXploited.

  To reduce the subjectivity of the evaluation, we limit each of these concepts to two scores; 1/2 for high and 1/6 for low. The summation of the two scores will thus be limited to the same three values as all other factors.
- **SSITH**($\mathcal{C}$): The SSITH factor is a measure of how relevant the weakness category $\mathcal{C}$ is to the SSITH program. It can take the values: 1 for strongly relevant, 2/3 for relevant, and 1/3 for somewhat relevant.

We can thus compute the BESSPIN scale for a particular vulnerability class, $\mathbf{S}(\mathcal{V})$:

$$\mathbf{S}(\mathcal{V}) = 100\% \times \frac{\sum_{\mathcal{C}_j \in \mathcal{V}} \Big(\beta(\mathcal{C}_j) \cdot \mathbf{S}(\mathcal{C}_j) \Big)}{\sum_{\mathcal{C}_j \in \mathcal{V}} \beta(\mathcal{C}_j)} \tag{6}$$

Similarly, we can compute the BESSPIN scale for a given processor:

$$\mathcal{B} = 100\% \times \frac{\sum_{\mathcal{V}_i \in \mathcal{S}} \sum_{\mathcal{C}_j \in \mathcal{V}_i} \Big(\beta(\mathcal{C}_j) \cdot \mathbf{S}(\mathcal{C}_j) \Big)}{\sum_{\mathcal{V}_i \in \mathcal{S}} \sum_{\mathcal{C}_j \in \mathcal{V}_i} \beta(\mathcal{C}_j)} \tag{7}$$

3 Vulnerability Classes and Categories

3.1 Buffer Errors

Buffer errors allow inappropriate read and/or write access to locations within memory associated with variables, data structures, or internal program data. Inappropriate access to memory is exploited to subvert the normal hardware operations creating security vulnerability in the hardware.

1. **Boundary Error:** The software accesses through an array a memory location that is outside the boundaries of that array.
   - Covered by CWEs: 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 786, 787, 788, 805, 806, and 823.
2. **Improper Size Handling:** The software either miscalculates the size of a memory buffer or offset, or blindly trusts a provided size or offset. This might lead to a buffer overflow.
   - Covered by CWEs: 120, 123, 124, 125, 126, 127, 129, 130, 131, 680, 785, 786, 787, 788, 805, 806, and 823.

3.2 Permission, Privileges, and Access Control (PPAC)

PPAC weaknesses allow the execution of unauthorized operations in a system. A privilege vulnerability can allow inappropriate access to system privileges. A permission vulnerability can allow inappropriate permission to perform functions. An access vulnerability can allow inappropriate control of the authorizing policies for the hardware.

1. **Authorization:** Access authorization to privileged resource access and authentication to grant such authorization.
    - Covered by CWEs: PPAC-1, PPAC-2, 284, 285, 287, 288, 668, 669, 862, and 863.
2. **Accountability:** Tracking of activities that were performed.
    - Covered by CWEs: PPAC-3, and 284.

3.3 Resource Management

Resource management weaknesses allow improper use of the hardware resources that, in turn, allow external takeover of hardware resources. This includes improper access to hardware resources such as memory, CPU, and communication and/or preventing valid users from gaining access to these resources.

1. **Memory/Data Layout:** The software makes invalid assumptions about how protocol data or memory is organized at a lower level, resulting in unintended program behavior.
    - Covered by CWEs: 188, 463, 467, 562, 587, 588, 590, 762, and 763.
2. **Resource Control:** The software does not properly control the allocation, initialization, maintenance, sanitizing, and release of a resource.
    - Covered by CWEs: 400, 404, 415, 416, 672, 770, 771, 772, 789, 908, 909, and 911.
3. **Pointers Misuse:** Software produces a faulty value or behavior as a result of pointer misuse.
    - Covered by CWEs: 415, 416, 468, 476, 588, 690, 761, and 825.

3.4 Information Leakage

Information leakage is the exposure of information to parties that are not authorized or intended to access it. Beyond leaking sensitive data, this can include revealing information that enables subsequent attacks. Information can be leaked by directly sending data to unauthorized parties, or through side or covert channels that indirectly allow parties to learn something about otherwise secret data.

1. **Information Exposure:** Data was *intentionally* disclosed to actors, but the data contained information that should not have been made available to those actors.
    - Covered by CWEs: 200, 201, and 202.
2. **Observable Discrepancy:** The product behaves differently in a way that is observable to an unauthorized user.
    - Covered by CWEs: 203, 205, and 206.
3. **Improper Sanitization:** Resources were made accessible to unauthorized users, but those resources were previously containing sensitive information, which *unintentionally* discloses them to those users.
    - Covered by CWEs: 200, 212, 226, 244, and 524.

3.5 Numeric Errors

Numeric errors allow exploitation of improper calculation or conversion of numbers and numeric types. Improper/incorrect calculations can allow subversion of security critical operational decisions and/or resource management.

1. **Range/Domain Errors:** The software produces a faulty result due to range or domain violations.
   - Covered by CWEs: 128, 190, 191, and 369.
2. **Type Errors:** The software produces a faulty result due to conversions between data types.
   - Covered by CWEs: 192, 194, 195, 196, 197, and 681.
3. **Value Errors:** The software used an incorrect value because a resource was not properly initialized, or function was called with incorrect arguments.
   - Covered by CWEs: 234, 456, 457, 665, and 824.

3.6 Hardware/SoC

Hardware/SoC weaknesses encompass several potential issues in the design of a piece of hardware.

1. **Improper Hardware Design:** Mistakes in hardware design leading to faults, disclosure of sensitive information, or complete system breaches.
   - Covered by CWEs: 1037, 1209, 1221, 1222, 1224, 1231, 1233, 1234, 1239, 1242, 1243, 1245, 1246, 1251, 1253, 1256, 1257, 1259, 1260, 1261, 1264, 1270, 1271, 1274, 1276, 1277, 1280, 1281, 1282, and 1283.
2. **Mishandling Permissions and Boundaries:** The SoC does not handle permissions and boundaries in a proper way, which may lead to either disclosure of sensitive information or privilege escalation.
   - Covered by CWEs: 1189, 1193, 1220, 1222, 1223, 1234, 1251, 1252, 1257, 1259, 1260, 1262, 1268, 1273, 1274, 1276, 1280, 1282, and 1283.
3. **Vulnerable Transitions:** During power or state transition, some assumptions made by the SoC design might not hold, which could be exploited to access sensitive data or alter the SoC operation.
   - Covered by CWEs: 1193, 1221, 1223, 1232, 1239, 1243, 1264, 1271, 1272, and 1280.
4. **DoS and Side Channels:** Weaknesses that lead to resource exhaustion and resulting denial of service, or discrepancies that might be exploited by a side channel attack.
   - Covered by CWEs: 208, 385, 920, 1050, 1246, and 1254.
5. **Crypto and Security Elements:** Improper design of cryptographic systems, and incorrect security assumptions, that are detrimental to security. This category is also concerned with the design and operation of any security elements in the SoC, such as secure boot.
   - Covered by CWEs: 1037, 1231, 1233, 1240, 1241, 1243, 1259, 1262, 1270, 1273, 1277, 1279, 1282, and 1283.

3.7 Injection

Injection weaknesses are architectural weaknesses that allow information from a less trusted domain to replace information in a more trusted domain in violation of an explicit or implied policy. Use of the injected information in the context of the more trusted domain could be detrimental to system operation or security.

1. **Trust Untrusted Data:** Untrusted data is accessed as trusted data.
   - Covered by CWEs: INJ-1, INJ-2, and INJ-3.

4 BESSPIN Coefficient Values

The BESSPIN coefficients document contains all values used to compute the BESSPIN coefficients for each weakness category. They were evaluated by the BESSPIN team and DARPA.