Empirical Study on BMC Firmware Vulnerabilities: Root Causes and Architectural Insights

Jihye Lee
Korea University
Seoul, Korea
jh_kor@korea.ac.kr

Chanhee Park
Korea University
Seoul, Korea
pch2180@korea.ac.kr

Youngjoo Shin
Korea University
Seoul, Korea
syoungjoo@korea.ac.kr

***Abstract*—Baseboard management controller (BMC) provides out-of-band management to datacenter infrastructures for IoT and mobile services. Because the BMC allows privileged access to hardware resources, it has become an attractive attack target. Despite its importance, little research has been conducted on the security of BMC firmware. In this paper, we examine the security vulnerabilities of BMC firmware by analyzing recent BMC-related CVEs from 2020 to 2025. Specifically, we analyze statistics classified by CWEs, severity, attack vectors, and vendor-specific factors. Furthermore, we examine the root causes of BMC firmware vulnerabilities and categorize them using our novel three-category taxonomy: architectural, implementation, and hardware issues. Our analysis shows that most vulnerabilities originate from implementation problems, while architectural issues such as weak privilege models and insecure update mechanisms in the BMC firmware were also identified, revealing BMC-specific security issues. We also discover different vulnerability patterns among vendors, which suggests that vendor-specific implementations and architectures substantially affect the security posture of the BMC. Our study identifies several future research directions, including the elimination of fundamental root causes and the implementation of BMC-specific static and dynamic vulnerability analysis techniques.**

***Index Terms*—BMC firmware, CVE measurements, Root cause taxonomy, Embedded systems security.**

I. Introduction

Baseboard management controllers (BMCs) are critical embedded components within modern data center infrastructures, offering out-of-band management capabilities essential for IoT and mobile services. Operating independently from the host OS, BMCs retain privileged access to hardware resources, enabling remote monitoring, control, and system maintenance even when the primary system is powered off. Given this high privilege and persistent availability, BMC firmware represents a significant and appealing target for attackers [1].

Despite their critical role, security considerations specific to BMC firmware have been largely overlooked. Previous research [2], [3] predominantly concentrated on identifying and analyzing observable vulnerabilities, often without delving deeply into their underlying causes. Thus, a comprehensive understanding of the root causes behind BMC firmware vulnerabilities remains limited.

The extended version of this paper is available at https://github.com/koreacsl/BMC_dataset/blob/main/icoin26-full.pdf.

To bridge this gap, this study presents a systematic and large-scale empirical analysis of BMC firmware vulnerabilities by reviewing 358 Common Vulnerabilities and Exposures (CVEs) [4] disclosed from 2020 to 2025. Each CVE is analyzed according to its Common Weakness Enumeration (CWE) type, severity, attack vector, and, most crucially, its root cause. We introduce a novel taxonomy specifically designed for BMC firmware vulnerabilities, categorizing them into architectural, implementation, and hardware-related issues. Our findings reveal that implementation flaws overwhelmingly dominate, particularly due to inadequate input validation and unsafe memory handling. However, architectural issues, such as insufficient privilege separation and insecure firmware update mechanisms, are also significant and highlight systemic weaknesses unique to BMC systems. Vendor-specific analyses indicate substantial variations in security postures, suggesting that individual design and implementation decisions significantly influence vulnerability profiles.

This research highlights the urgent need for targeted, BMC-specific security methodologies. We propose future research directions aimed at addressing fundamental vulnerabilities through specialized static and dynamic analysis frameworks, source-to-sink models, and robust fuzzing environments tailored to the complex, multi-binary nature of BMC firmware. Our work thus provides foundational insights and concrete pathways for enhancing the resilience of BMC systems, contributing significantly to the security of modern data center infrastructures. The main contributions of this work are threefold: (1) a large-scale empirical analysis of 358 BMC firmware vulnerabilities reported from 2020 to 2025, (2) a root-cause taxonomy covering architectural, implementation, and hardware-level issues, and (3) a vendor-specific comparison revealing recurring vulnerability patterns and systematic design flaws.

II. Background

A. Baseboard management controller

A baseboard management controller (BMC) is an independent hardware controller embedded in server motherboards, designed to provide out-of-band system management functionalities regardless of the host system's operational state. Internally, BMCs typically run a lightweight embedded Linux operating system. BMC firmware includes several software

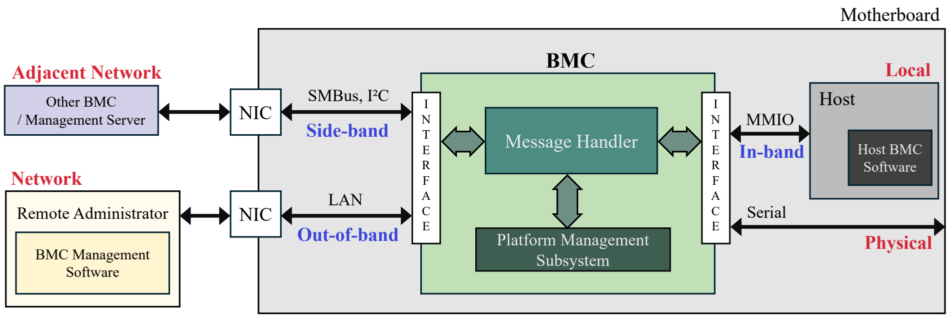


Fig. 1: Overview of BMC and BMC's attack surface

components such as a minimal kernel, basic system utilities, and various user-space services. Figure 1 shows an overview of its communication architecture. The BMC firmware consists of two key modules: the Platform Management Subsystem, which is responsible for core functionalities such as sensor data collection, power control, event logging, and firmware updates; and the Message Handler, which interprets incoming message (e.g., IPMI, Redfish) and routes them to the appropriate subsystems or generates responses. BMCs mainly use protocols such as IPMI and Redfish, and the latter is standardized by DMTF to offer a more secure, RESTful API with role-based access control [2], [5]–[7].

BMC communicate with internal and external components through three channels: (1) in-band links connecting to the host via memory-mapped I/O (MMIO), (2) side-band hardware paths such as SMBus or I²C connecting to system sensors and controllers, and (3) out-of-band management ports accessed over IPMI, Redfish, or HTTPS. These channels enable remote administration but also expand the system's attack surface.

Attack surface. BMCs are subject to multiple attack surfaces due to their exposure to both internal and external interfaces. Based on the attacker's proximity to the system and their required level of access to BMC interfaces, the attack surfaces can be categorized into network, adjacent network, local, and physical vectors (See Fig. 1).

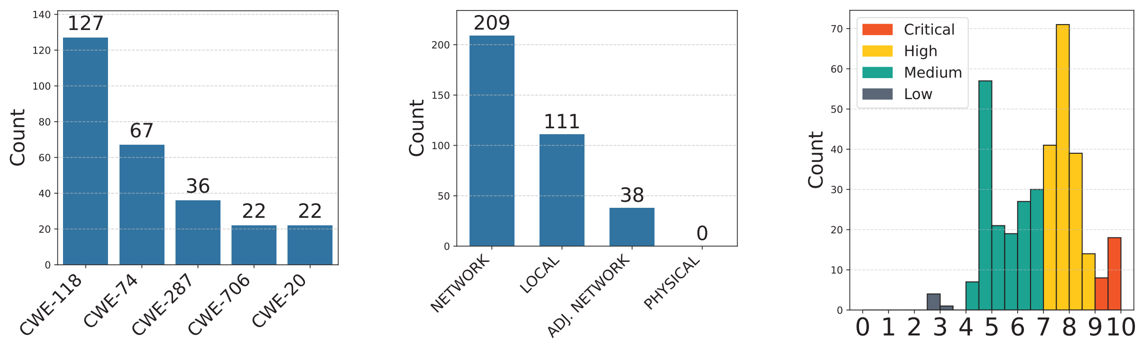
Network vector includes remote interactions over management protocols such as IPMI over LAN, Redfish, and web interfaces. These services often remain active by default, exposing attack points for input handling flaws or weak authentication that can lead to unauthorized access or remote code execution. Local vector refers to threats originating from the host or privileged software communicating with the BMC through in-band channels such as PCIe or MMIO. Excessive host access or misconfigurations can blur privilege boundaries and expose sensitive BMC state information. Finally, physical vector involves direct hardware access via debug ports (UART, JTAG) or tampering with SPI flash storage. Such attacks are rarer but serious, since physical access allows firmware modification and bypass of software-level defenses.

B. *CWE and CVSS*

The Common Weakness Enumeration (CWE) provides a taxonomy of software weaknesses using the CWE-1000 view [8], which groups related issues such as improper input validation or authentication. The Common Vulnerability Scoring System (CVSS) rates vulnerability severity from 0.0 to 10.0 based on impact and exploitability. This study focuses on two CVSS elements—attack vector and severity level—classified as network, adjacent network, local, and physical vectors, and as Critical, High, Medium, or Low severity categories.

TABLE I: Taxonomy of root causes in BMC vulnerabilities

Issues	Subclass (Code)	Description
Architectural	Excessive Attack Surface (A-E)	Multiple remote management interfaces (e.g., IPMI, Redfish, SSH) are exposed without proper isolation or access control
	Host–BMC Boundary Breakdown (A-H)	Shared access to memory or control channels (e.g., PCIe) undermines isolation and enables lateral privilege escalation
	Lack of Secure Mechanism, Update and Boot chain (A-L)	Failures in firmware validation and rollback protection allow persistent compromise via malicious updates
	Privilege Model Misdesign (A-P)	Incomplete privilege separation or flawed authorization logic enables unauthorized access
Implementation	Validation Bugs (I-V)	Inadequate input sanitization or boundary checking leads to injection or memory corruption
	Functional Bugs (I-F)	Broken authentication or misconfigured session handling
	Resource Management Failures (I-R)	Mishandled memory or file descriptors cause denial-of-service
	Extrinsic Bugs (I-E)	Race conditions or unstable runtime state trigger unexpected behavior
Hardware	Debug Port Exposure (H-D)	Accessible UART or JTAG interfaces provide root access in production systems
	Unprotected Flash or Memory Regions (H-U)	SPI flash or DRAM can be tampered with due to lack of protection
	Side-channel and Fault Injection (H-S)	Power analysis or glitching bypasses secure boot or leaks sensitive data



(a) CWE type (b) Attack vector (c) Severity score

Fig. 2: Statistics of BMC-related CVEs

III. Root Cause Taxonomy

This section introduces the taxonomy of root causes that make BMC firmware vulnerable to attack. We classify vulnerabilities into three main categories—architectural, implementation, and hardware issues—each comprising several subclasses as summarized in Table I. Architectural issues stem from flawed design and weak trust boundaries, such as excessive attack surfaces or insecure update mechanisms. Implementation issues arise from improper code execution, including missing validation, faulty authentication, and poor resource handling. Hardware issues involve physical or board-level weaknesses, such as open debug ports and unprotected memory regions. This taxonomy serves as the analytical basis for our CVE-based study and distinguishes vulnerabilities that can be addressed through software-level mitigation from those that require architectural or hardware-level solutions.

IV. Analysis on BMC-related CVEs

A. Dataset construction

Preprocessing. Our analysis used the raw CVE data from the MITRE [4], covering the years 2020 to 2025. Since the MITRE does not have a separate category specifically for BMC vulnerabilities, we filtered the dataset using relevant keywords to isolate BMC-related entries. The keywords includes vendor names, product identifiers (e.g., iDRAC, MegaRAC), protocols (e.g., IPMI, Redfish), platforms (e.g., Yocto), and threat-related terms (e.g., UART, Serial, Firmware). Subsequently, we performed vulnerability type classification and severity scoring according to the CWE and CVSS, respectively. When both CWE and CVSS information were missing, we manually supplemented the data using vendor advisories and third-party security feeds such as Red Hat and Tenable.

Vulnerability classification. We then classified the preprocessed CVEs according to its vulnerability type. In our study, we used the class level of CWE-1000 to organize vulnerability types, as CWE-1000 structures weakness types by abstract behavior and conceptual relationships. This allowed us to group diverse CWE entries into meaningful categories based on shared operational traits. The mapping of individual CWE types to the CWE-1000 view is documented by MITRE and can be accessed at [8].

Root cause classification. Finally, we conducted root cause classification based on the taxonomy presented in Section III. In particular, we reviewed manually each CVE and assigned to one of these categories based on its technical description. This classification forms the basis for the statistical analysis and visualization presented in the next section.

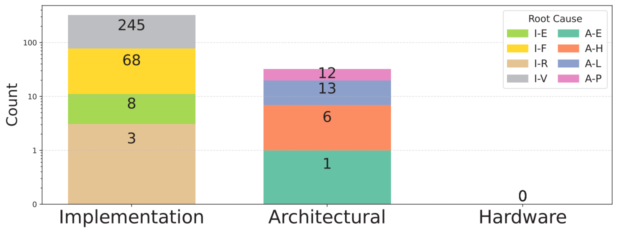
B. Insights into BMC vulnerabilities

Based on the structured dataset constructed in Section IV-A, we present an in-depth analysis of 358 BMC-related CVEs, revealing critical insights into prevalent vulnerability types, attack vectors, and associated severity levels.

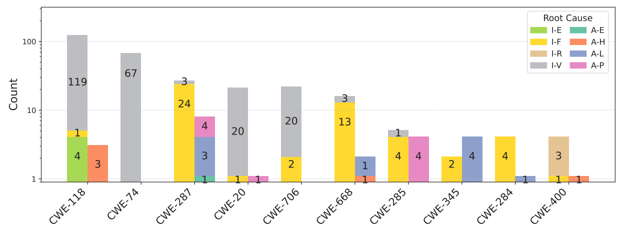
1) Statistical overview: Figure 2 provides a statistical overview of the 358 BMC-related CVEs from three key perspectives: CWE type distribution, attack vector distribution, and severity distribution. First, Figure 2a shows the distribution of the five most frequently occurring CWE types among the 33 observed types. CWE-118 (127 cases) is the most common, followed by CWE-74 (67 cases), CWE-287 (36 cases), and both CWE-706 and CWE-20 (22 cases each). These reflect widespread issues in boundary checks, input validation, and authentication logic.

BMC firmware is typically implemented in type-unsafe languages such as C/C++ and operates within embedded environments with limited memory and strict real-time constraints. As a result, boundary checks are often omitted or minimized. In addition, management protocols such as IPMI and Redfish rely on custom text-based parsers that are frequently extended by vendors, increasing the likelihood of missing input filtering. Furthermore, because BMC integrates all functionality—from initial boot to hardware control—into a single firmware image, authentication and privilege logic is often entangled, raising the risk of default credentials and hardcoded tokens being

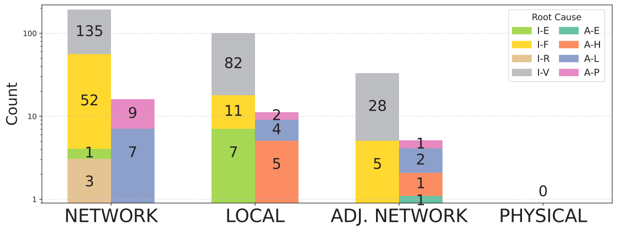
CWE-118: Incorrect Access of Indexable Resource ('Range Error')
CWE-74: Improper Neutralization of Special Elements in Output Used by a Downstream Component ('Injection')
CWE-287: Improper Authentication
CWE-706: Use of Incorrectly-Resolved Name or Reference
CWE-20: Improper Input Validation



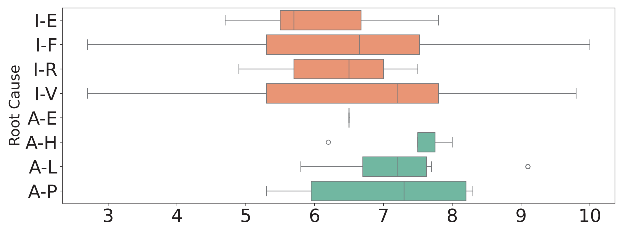
(a) Root Cause Distribution



(b) Root Cause - CWE



(c) Root Cause - Attack Vector



(d) Root Cause - Severity

Fig. 3: Root cause analysis

left behind. These design limitations and code reuse patterns contribute to name or path resolution issues (CWE-706) and improper input validation (CWE-20), where normalization and validation logic tend to be weak or entirely absent.

Figure 2b shows that the NETWORK vector dominates (209 cases), followed by LOCAL (111) and ADJACENT_NETWORK (38) vectors. Network-based vulnerabilities are especially critical because exposed management interfaces such as IPMI and web UIs enable remote code execution or malicious firmware uploads. The LOCAL vectors highlight risks from insiders or compromised hosts, while the ADJACENT_NETWORK attacks indicate lateral movement within internal management domains. The prevalence of remotely exploitable vulnerabilities underscores the importance of isolating BMC interfaces from production networks and enforcing strict access controls [9].

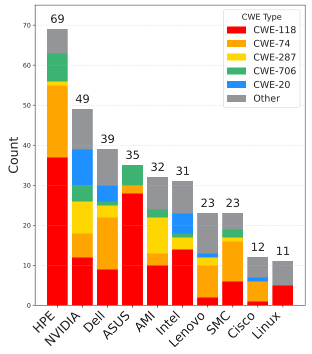
Over half (53.5%) of the analyzed CVEs are rated High or Critical, demonstrating the severe impact of these weaknesses. Because many BMCs remain accessible over management networks, timely patching and layered security measures, including isolation, access control, and vulnerability management, are essential to reduce risk.

2) Root cause analysis: Figure 3 presents the root cause analysis based on our proposed taxonomy. Approximately 90% of the analyzed CVEs originate from implementation flaws, mainly validation bugs (245 bugs), while about 9% are architectural, including privilege model misdesign and insecure update or boot mechanisms. Although hardware-related CVEs were not found, past cases such as CVE-2018-15776 confirm their relevance. This distribution reflects the nature of BMC development. In practice, most firmware is written in C/C++ and assembled from reused modules and vendor extensions, where strict input validation, memory safety, and privilege separation are rarely enforced. Limited runtime resources and tight integration with hardware further discourage robust boundary checking, making implementation defects both common and systemic.

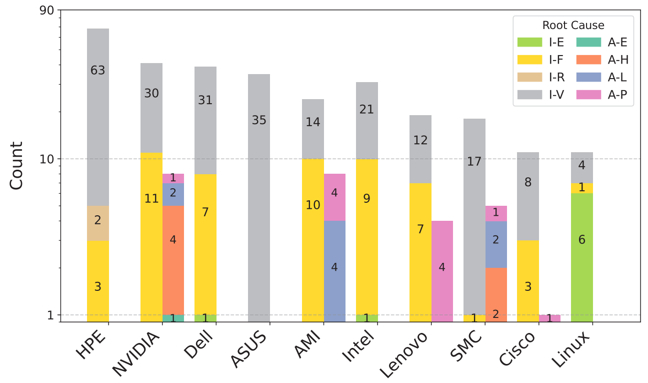
Figure 3b maps the most frequent CWE types to root causes. CWE-118 and CWE-74 mainly involve boundary and input validation errors, while CWE-287 and CWE-20 span both functional and architectural flaws related to privilege design. Representative examples include buffer overflows in OpenBMC's `slpd-lite` daemon (CVE-2024-41660) and command injection in Lenovo's XCC firmware (CVE-2024-8278). These recurring problems arise because BMC firmware often combines kernel-level code and user-space services within a single image, blurring privilege boundaries and allowing unsafe data handling to propagate across components.

As shown in Figure 3c, validation bugs dominate across all attack vectors. Network-based vulnerabilities are most prevalent, exploiting weak input handling in interfaces such as IPMI and Redfish; LOCAL vectors typically involve host-BMC boundary flaws that enable privilege escalation, while ADJACENT_NETWORK attacks expose architectural weaknesses like hardcoded credentials or certificates. These patterns underscore how BMC's permanent exposure and multi-channel connectivity amplify the security impact of implementation bugs, turning simple validation errors into remotely exploitable entry points.

Finally, Figure 3d shows that architectural issues, though fewer, tend to have higher severity scores. For example, CVE-2022-22374 in IBM Power9 systems (CVSS 9.1) reflects insecure firmware downgrade protections. Overall, implementation-level flaws dominate both in count and in impact, whereas architectural weaknesses that stem from fundamental design and update processes tend to cause broader system compromise when exploited. These findings highlight that BMC firmware's monolithic structure, long vendor reuse cycles, and limited verification pipelines collectively foster persistent, high-severity vulnerabilities.



(a) Vendor - CWE



(b) Vendor - Root Cause

Fig. 4: Top 10 Vendors CWE and Root Cause Distribution

3) Vendor-specific analysis: To examine vendor-specific vulnerability patterns, we categorized major BMC vendors by their target markets and product focuses, as summarized in Table II. This categorization contextualizes the differences in vulnerability types and root causes shown in Figure 4, focusing on the ten vendors with the highest number of reported CVEs.

Figure 4a compares CWE distribution among vendors. HPE and ASUS show frequent CWE-118 and CWE-74 cases, indicating persistent boundary-check and input filtering weaknesses. ASUS also exhibits many CWE-706 cases related to improper input validation and limited use of static analysis. In contrast, NVIDIA shows a more balanced distribution across CWE types, reflecting complex command parsing in high-performance BMCs.

Figure 4b shows that implementation flaws, mainly validation bugs, dominate most vendors; over 80% of vulnerabilities in HPE, Dell, and ASUS fall into this category. NVIDIA and AMI, however, exhibit more architectural issues, including weak host-BMC isolation, insecure update mechanisms, and misconfigured privilege models. Lenovo presents a mix of privilege model and flaws with inconsistent API-UI validation. These variations stem from each vendor's development priorities and market focus. Performance-oriented vendors like NVIDIA tend to suffer complex architectural issues due to feature-rich and high-throughput firmware design. OEM vendors such as AMI, prioritizing compatibility and flexibility, frequently face authentication and privilege management vulnerabilities. General-purpose server vendors, including HPE and ASUS, show recurring implementation errors like input validation and memory safety issues, often linked to cost-driven development and limited security testing.

Overall, vendor-specific design choices and market strategies substantially shape BMC security. Addressing these recurring flaws requires tailored detection and prevention strategies that reflect each vendor's design philosophy, alongside standardized security guidelines applied at the development stage to improve the collective security posture of BMC firmware.

TABLE II: BMC vendors

Type	Vendor
General-purpose / Enterprise	HPE, Dell, ASUS, Lenovo, Cisco
OEM	AMI, Supermicro, Intel
High-performance	NVIDIA
Open-source	Linux

V. Future Research Directions

Based on our analysis in Section IV, BMC firmware security challenges primarily stem from implementation flaws, architectural weaknesses, and potential hardware risks. Addressing these problems requires targeted research into detection methodologies, secure design principles, and hardware-level safeguards tailored to the unique characteristics of BMC systems.

Implementation-level research. Since nearly 90% of vulnerabilities originate from implementation errors such as boundary checking, injection, and input validation flaws, future work should focus on specialized static and dynamic analysis frameworks for BMC firmware. Static methods must improve loop and bounds verification, while dynamic approaches should analyze input handling and API behaviors through targeted payload testing and runtime monitoring. Given BMC firmware's multi-binary composition and vendor-specific APIs, research into BMC-aware multi-binary analysis, source-to-sink modeling, and hardware-integrated fuzzing frameworks is especially critical to uncover hidden vulnerabilities.

Architectural research. Although architectural flaws represent only about 9% of cases, their severity is high. Future studies should develop secure architectural guidelines encompassing privilege model design, role-based access control, and verified update and boot chains. Achieving this requires clear trust-boundary definitions and formal verification of privilege enforcement and firmware integrity. Such methods would prevent design flaws that enable privilege escalation or insecure firmware rollback.

Hardware-level research. Even though hardware-related CVEs were not observed in our dataset, past cases reveal critical risks at the physical layer. Research should focus on securing or disabling debug interfaces (UART, JTAG), enforcing firmware encryption and signature verification, and implement strict board-level access control. In addition, supplychain assurance and pre-deployment penetration testing should be standardized to detect hardware tampering and firmware injection during manufacturing and delivery.

VI. Related Work

A. Empirical studies on software vulnerability

Prior research has examined software vulnerability trends using CVE datasets, mostly focusing on IoT and firmware-related systems [10]–[14]. These studies analyzed severity distributions, access vectors, or exploitability metrics but did not investigate architectural or root causes. In contrast, our study introduces a taxonomy-driven approach that analyzes BMC firmware vulnerabilities from a root-cause perspective, offering a deeper understanding of why such flaws persist.

B. Vulnerability analysis on BMC firmware

Existing research on BMC security has mainly explored protocol and management interface vulnerabilities. Large-scale scans revealed widespread IPMI misconfigurations and default credentials [2], [15], while subsequent works demonstrated remote exploitation and privilege escalation in OEM implementations such as HPE iLO and Dell iDRAC [16], [17]. Network-level studies [18], [19] uncovered exposed or reachable BMCs, and more recent tools like BMCDea [3] automated firmware decryption and secret extraction. Unlike these case-specific efforts, our work performs the first large-scale CVE-based statistical analysis across multiple vendors, correlating vulnerability types with architectural and implementation-level causes.

VII. Conclusion

Strengthening BMC security is essential as these components increasingly underpin large-scale computing infrastructures for IoT and mobile services. This study analyzed 358 BMC firmware vulnerabilities reported between 2020 and 2025, classifying them by root cause into architectural, implementation, and hardware categories. Most vulnerabilities arose from implementation issues, notably input validation and memory errors, though some critical vulnerabilities reflected deeper architectural flaws like inadequate privilege separation and insecure update mechanisms. Our structured root-cause taxonomy highlights systemic patterns, guiding future development of targeted security measures such as source-to-sink modeling, scalable multi-binary analysis, and hardware-aware fuzzing.

Acknowledgment

This research was supported by a National Research Foundation of Korea (NRF) grant, funded by the Korean government (MSIT) (RS-2023-NR077166, RS-2023-00227165)

References

[1] Y. Zhai, F. Cao, P. Zhang, G. Zhang, W. Yao, and Y. Hao, "A survey of out-of-band management vulnerability analysis based on bmc," in *Proceedings of the International Conference on Electronics Technology*, 2024, pp. 653–658.

[2] A. Bonkoski, R. Bielawski, and J. A. Halderman, "Illuminating the security issues surrounding Lights-Out server management," in *USENIX Workshop on Offensive Technologies*, 2013.

[3] Y. Zhai, F. Cao, P. Zhang, and S. Yue, "Bmcdea: Decryption and extraction analysis of bmc firmware," in *Proceedings of the International Workshop on Computer Science and Engineering*, 2024, pp. 356–364.

[4] CVE, "Common vulnerabilities and exposures," https://www.cve.org/.

[5] NVIDIA, "Dgx h100/h200 user guide: Redfish apis support," https://docs.nvidia.com/dgx/dgxh100-user-guide/redfish-api-supp.html.

[6] HPE, "Redfish: a more secure alternative to ipmi," https://community.hpe.com/t5/servers-systems-the-right/redfish-a-more-secure-alternative-to-ipmi/ba-p/7044568.

[7] Supermicro, "Supermicro server management (redfish® api)," https://www.supermicro.com/en/solutions/management-software/redfish.

[8] MITRE Corporation, "Cwe view: Research concepts," https://cwe.mitre.org/data/definitions/1000.html.

[9] National Security Agency and Cybersecurity and Infrastructure Security Agency, "Harden baseboard management controllers," https://media.defense.gov/2023/Jun/14/2003241405/-1/-1/0/CSI_HARDEN_BMCS.PDF, 2023.

[10] A. Gueye and P. Mell, "A historical and statistical study of the software vulnerability landscape," arXiv:2102.01722, 2021.

[11] B. Dodiya, U. K. Singh, and V. Gupta, "Trend analysis of the cve classes across cvss metrics," *International Journal of Computer Applications*, vol. 183, no. 33, pp. 23–30, 2021.

[12] J. Glyder and Others, "Some analysis of common vulnerabilities and exposures (cve) data from the national vulnerability database (nvd)," in *Proceedings of the Conference on Information Systems Applied Research*, 2021, p. 1508.

[13] R. Khoury, B. Vignau, S. Hallé, A. Hamou-Lhadj, and A. Razgallah, "An analysis of the use of cves by iot malware," in *Foundations and Practice of Security*, ser. Lecture Notes in Computer Science, vol. 12819, 2021, pp. 47–62.

[14] A. T. Sheik, M. Hooper, C. Maple, and J. Crowcroft, "Cyber threat observatory for national identity systems quarterly report: Drawing insights from vulnerability patterns in digital identity, government, healthcare, and finance sectors," https://www.turing.ac.uk/sites/default/files/2025-05/threat_report_cve_april-4.pdf, 2025.

[15] D. Farmer, "Ipmi: Freight train to hell or linda wu & the night of the leeches," http://fish2.com/ipmi/itrain.pdf.

[16] A. Gazet, J. Czarny, and F. Perigaud, "Subvert server bmc: How to replace your server's brain," https://recon.cx/2018/brussels/talks/subvert_server_bmc.html.

[17] N. Iooss, "idrackar, integrated dell remote access controller's kind approach to the ram," https://github.com/nico/idrackar.

[18] J. Zhang, Y. Wang, J. Liu, and K. Xu, "On the mismanagement and maliciousness of networks," in *Network and Distributed System Security Symposium*, 2014, pp. 23–26.

[19] O. Gasser, F. Emmert, and G. Carle, "Digging for dark ipmi devices: Advancing bmc detection and evaluating operational security," in *Traffic Measurement and Analysis Conference*, 2016.