Journal of Cybersecurity and Privacy

MDPI

*Article*

# An Evaluation of the Security of Bare Machine Computing (BMC) Systems against Cybersecurity Attacks

**Fahad Alotaibi [1,*], Ramesh K. Karne [1,*], Alexander L. Wijesinha [1], Nirmala Soundararajan [2] and Abhishek Rangi [1]**

[1] Department of Computer & Information Sciences, Towson University, Towson, MD 21204, USA; awijesinha@towson.edu (A.L.W.); arangi1@students.towson.edu (A.R.)
[2] Department of Computer & Physical Sciences, Southeastern Oklahoma State University, Durant, OK 74701, USA; nsoundararajan@se.edu
* Correspondence: falota3@students.towson.edu (F.A.); rkarne@towson.edu (R.K.K.)

**Abstract:** The Internet has become the primary vehicle for doing almost everything online, and smartphones are needed for almost everyone to live their daily lives. As a result, cybersecurity is a top priority in today's world. As Internet usage has grown exponentially with billions of users and the proliferation of Internet of Things (IoT) devices, cybersecurity has become a cat-and-mouse game between attackers and defenders. Cyberattacks on systems are commonplace, and defense mechanisms are continually updated to prevent them. Based on a literature review of cybersecurity vulnerabilities, attacks, and preventive measures, we find that cybersecurity problems are rooted in computer system architectures, operating systems, network protocols, design options, heterogeneity, complexity, evolution, open systems, open-source software vulnerabilities, user convenience, ease of Internet access, global users, advertisements, business needs, and the global market. We investigate common cybersecurity vulnerabilities and find that the bare machine computing (BMC) paradigm is a possible solution to address and eliminate their root causes at many levels. We study 22 common cyberattacks, identify their root causes, and investigate preventive mechanisms currently used to address them. We compare conventional and bare machine characteristics and evaluate the BMC paradigm and its applications with respect to these attacks. Our study finds that BMC applications are resilient to most cyberattacks, except for a few physical attacks. We also find that BMC applications have inherent security at all computer and information system levels. Further research is needed to validate the security strengths of BMC systems and applications.

**Keywords:** bare machine computing; operating system; computer architecture and design; security by design; Internet security; cybersecurity; bare Internet; cyberattacks



**Citation:** Alotaibi, F.; Karne, R.K.; Wijesinha, A.L.; Soundararajan, N.; Rangi, A. An Evaluation of the Security of Bare Machine Computing (BMC) Systems against Cybersecurity Attacks. *J. Cybersecur. Priv.* **2024**, *4*, 678–730. https://doi.org/10.3390/jcp4030033

Academic Editor: Steve Schneider

Received: 12 June 2024
Revised: 20 August 2024
Accepted: 10 September 2024
Published: 18 September 2024



## 1. Introduction

Cybersecurity has become a national and global problem due to globalization and billions of users (5.3 billion) on the Internet [1]. Also, the proliferation of IoT devices on the Internet has introduced new challenges for cybersecurity. There is a veritable ocean of information concerning cybersecurity spanning across many disciplines and systems. Cyberattacks impact all layers of computing and information, including hardware, software, firmware, applications, policies, and other system components. The three dimensions of cybersecurity consisting of information states (storage, processing, and transmission), security principles (confidentiality, integrity, and availability), and countermeasures (policies and practices, technologies, and users) [2] can be extended to ten dimensions, as shown in Figure 1.

There are many permutations and combinations of this ten-dimensional space. Many releases and versions {2} apply to all other nine dimensions during evolution. Many end-user applications {1} meet user requirements in many areas. Depending on their domains, these end-user applications may be implemented in various computer programming

languages {3}. A given end-user application may use a variety of networks and protocols {4} and user interfaces {5}. Most end-user applications run on top of some operating system (OS) {6}, and all computing systems run on some processor architecture such as Intel, AMD, and RISC {7}. Many processor chips exist {8} with multiple cores and other features. There are also many vendors {9} for software, hardware, and other components. Furthermore, there are a variety of cybersecurity methodologies, computer architectures, and implementations {10}. There is heterogeneity and redundancy in the above dimensions and rapid obsolescence. The ever-changing hardware, software, and related products contribute to increased complexity and related cybersecurity issues [3] in computer and information systems, resulting in a waste of resources and human skills, recurring costs, dumping of hardware, and reinventions in many dimensions. The attributes discussed here are mainly due to obsoleting products before their life span [4]. Surprisingly, many other aspects of obsolescence, including planned obsolescence [5–7], focus on sustained revenues and profits for the industry. Microprocessor and OS obsolescence are the main contributors to fast changes in computing and information systems.

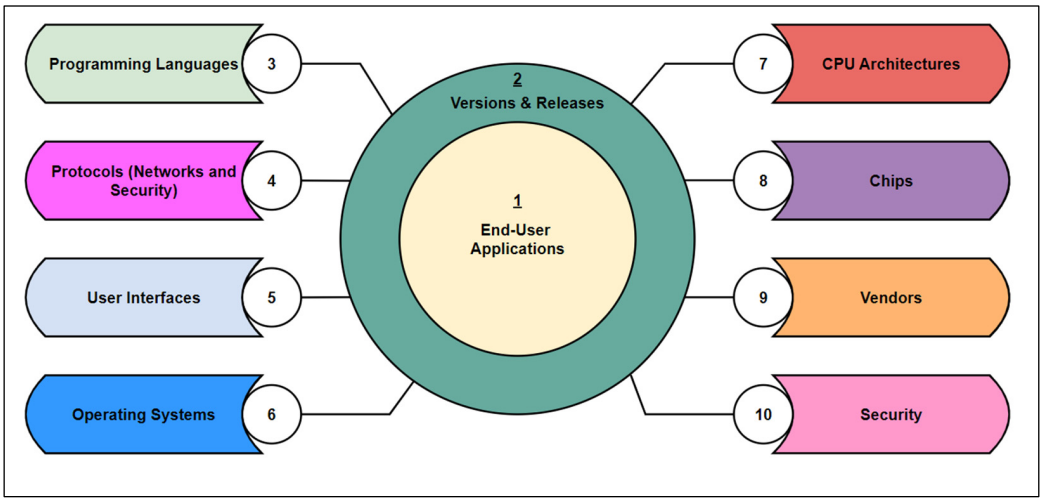


**Figure 1.** Ten dimensions of cybersecurity space.

Cybersecurity vulnerabilities are rooted in many dimensions and layers. Given the multidimensional nature of cybersecurity, it is possible to reduce this space by eliminating some of the dimensions. We present a radical solution that eliminates the OS. When the OS is eliminated, computing is based on the bare machine computing (BMC) paradigm, and applications become BMC applications, as discussed in Section 3 below. This paradigm makes computing devices bare and is inherently secure by design.

Some background on cyberattacks is necessary before discussing our methods to evaluate the BMC paradigm. We have chosen 22 cyberattacks to investigate in our study. In general, an attack always uses one or more vulnerabilities to create an attack. For example, a buffer overflow vulnerability can be used to craft many different types of attacks. Similarly, SQL injection may be used to attack a database system.

1. Buffer Overflow: A buffer overflow is a condition that exists when a program can put more data in the buffer than its allocated size. According to [8], "buffer overflow attacks played a major role in the propagation of malicious worms from machine to machine".
2. Phishing: This is one of the most used attacks in the cybersecurity world. Social engineering techniques are mostly used in phishing attacks [9]. Attackers persuade or lure users to share private information that can be used later to exploit their personal accounts, such as banking, credit cards, social security information, and personal assets. They can also use this information to damage their computer system.

3. Ransomware: Ransomware is malware that captures the victim's data or device and renders it unusable till some ransom amount is paid. There are two types of ransomwares: locker and cryptographic [10]. Locker ransomware is intended to lock computer access to users or delete files. Cryptographic ransomware encrypts files with hacker keys and demands a ransom to recover files. The main goal of ransomware is to extort money from victims.
4. Denial of Service (DoS): DoS [11] is an attack that can flood a victim's server with a variety of packets, such as UDP, SYN, HTTP, and ICMP. The goal is to slow down or crash the machine completely. The same attack can be carried out on a server using distributed systems known as distributed denial of service (DDoS) [11] to accomplish a larger and faster attack on a victim.
5. Man in the Middle (MitM): MitM [12] involves an attacker successfully inserting themselves between two communicating parties. The attack can be conducted within the same network or on the Internet. There are a variety of ways to conduct a MITM attack using protocols such as ARP, DNS, ICMP, DHCP, SSL/TLS, and BGP.
6. Password: This is the most common and obvious way to attack computer systems or smartphones. The attackers use a variety of techniques, including brute force, dictionary, phishing, shoulder surfing, key loggers, video recording, replay, credential surfing, and password spraying [13].
7. Trojan Horse: This is a malicious program in the guise of a standard harmless program. It can initiate actions without the user's approval once it is installed. Trojan horse attacks involve hiding a hacking program and dispatching it at the right time. There are many such Trojan horse techniques [14], including ArcBombs, Backdoors, Banking Trojans, Clickers, DDoS, Downloaders, Droppers, FakeAV, Game Thieves, Instant Messaging, Loaders, Mail Finders, Notifiers, Proxies, Password-stealing ware, and SMS. Almost any software is vulnerable to Trojan horse attacks.
8. Virus: A virus is malicious code that attaches to a host's executable file. The code executes whenever the file is opened. When the infected file is sent across computer systems, the virus spreads [15]. They are usually spread through emails or shared mass storage devices.
9. Worm: A worm is a type of malware that replicates itself and spreads across computers without any interaction from the user [16]. They consume memory and network resources, thus causing the system to hang. They can also allow access to attackers remotely.
10. Spyware: This is software secretly installed on the victim's computer or a computer belonging to an organization that monitors and gathers information on a user's online activity, websites visited, and other personal information. Spyware, also known as privacy-invasive software, is a prevalent issue in today's computing landscape, often installed without a user's full knowledge or consent [17].
11. Adware: This is a kind of spyware that runs unwanted advertisements and may contain malware that can get installed on the user's computer when the user clicks on the links [18]. It could also be installed due to unintentional drive-by downloads.
12. Rootkit: This is a software tool used to take over control of the intended computer and run commands on it with administrator privileges as it operates inside or near the kernel. It can hide keyloggers that capture the keystrokes and steal information such as passwords, credit card numbers, and online banking details [19].
13. Botnet: A botnet is a group of interconnected malicious computers that coordinate to attack other machines [20]. A single attacker (botmaster) can create an interconnected robot network (a botnet) with malicious code and control it for attacking. The attacker can initiate various types of cyberattacks, such as DDoS, phishing, click fraud, and spam.
14. Data Breach: A data breach is when unauthorized personnel gain access to sensitive data or confidential information. Some examples of sensitive data include social security information, bank accounts, healthcare data, and corporate information. Sensitive information can be stored in paper files, hard disks, thumb drives, and intellectual property. In these attacks, the confidentiality of breached data is lost [21].

15. Advanced Persistent Threat (APT): An APT remains undetected for an extended period after an attacker gains unauthorized access to a computer network [22]. This is a sequential and long-term attack that persists in the system. These attackers have a planned goal consisting of theft of data, gaining access to system resources, using social engineering techniques to lure users, staying in the system as long as possible, and moving around the system with the best attacking strategy without being noticed by any existing preventive tools.
16. SQL Injection: This attack consists of accepting SQL queries as inputs into a vulnerable database system that leads to exploitation of the database. The malicious SQL query allows the attacker to gain unauthorized access to the database. An SQL injection attack consists of the insertion or "injection" of a SQL query via the input data from the client to the application [23]. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administrative operations on the database (such as shutting down the DBMS), recover the content of a given file present on the DBMS file system, and in some cases issue commands to the OS.
17. Supply Chain: A supply chain attack [24] is one in which many vendors in the chain are rendered vulnerable when a single vendor is compromised. The attackers can obtain maximum benefit by concentrating their attack on a single vendor and then gain access to global data through the chain of connected vendors. A supply chain system consists of many vendors for a given customer. If one of the vendors is breached or attacked in the chain, it may influence other vendors. One of the vendors may have been breached due to security vulnerabilities in their site. When there are many suppliers, one or more of them may be prone to security vulnerabilities. Multiple targets can be compromised from a single vendor.
18. URL Interpretation: URL interpretation [25] exploits the vulnerability of URLs by changing and manipulating the URL meaning without changing or altering the syntax, which allows the attackers to access unauthorized data from the server associated with the URL. Attackers can alter and fabricate URL addresses to access victims' private data and information. It is also known as URL poisoning. An attacker may use this fake URL to gain administrative privileges and launch other attacks.
19. Insider Threat: Insider threats [26] happen when employees within an organization have privileged access to critical information and misuse their credentials for personal gain. This could render the organization system vulnerable to security attacks. These employees may have current credentials to access private data at an employer site. They may be seeking financial gains to extort the employer using the accessed private data.
20. Eavesdropping: Eavesdropping [27] is stealing transmitted information over an unsecured network. It consists of capturing network traffic. The attacks can be static or modification attacks. In static attacks, the data may be used later to capture valuable information. In modification attacks, the data are only useful if an attacker can modify it while the communication is in progress. A MitM attack can be used with the captured data.
21. Cookies: Cookies contain information about a machine, sites browsed, clicks made, location, and possibly login information [28]. Attackers can steal or hijack a cookie and capture this data. When a user visits a website, cookies may be enabled to play advertisements and capture the user's browser activity. Although cookies help speed up access to visited websites, they help attackers piggyback malicious programs along with legitimate ads.
22. Social Engineering: These attacks [29] are used to convince a victim to share private data or perform some actions that will enable an attacker to inject some illegitimate code into the system. This is one of the ways attackers can install malware on the victim's site.

There are many standards that classify security vulnerabilities and cyberattacks. With respect to these standards, such as CVE [30], CWE [31], and MITRE ATT & CK [32], their root causes and mapping to attacks are based on current systems, products, and software engineering methodologies. Bare machine computing is a different concept and approach. In this paper, we only consider generic attributes that can be compared with the

BMC paradigm and its applications. To provide a comprehensive evaluation of the BMC paradigm, we chose to compare conventional systems and BMC systems. First, we identify and discuss conventional system guidelines and conventional system characteristics. Next, we give an overview of the BMC paradigm, the compilation process in BMC development, and the integration of direct hardware interfaces in BMC code. We include code snippets that give insight into how BMC applications directly control the hardware. Then, we identify and discuss BMC system guidelines and characteristics and compare them with conventional system guidelines and characteristics to determine the resilience or exposure of each system to cybersecurity vulnerabilities. Using the above 22 common cyberattacks, we examine their primary root causes and preventive mechanisms given in the existing literature. We evaluate the BMC paradigm with respect to these cyberattacks by analyzing how the attacks are related to their root causes and preventive mechanisms. We then look at the applicability of root causes and preventive mechanisms to BMC systems. This enables us to evaluate the security strengths of the BMC paradigm and its ability to resist these cyberattacks. We also discuss the significant contributions of this research. Our evaluation using the relevant existing literature demonstrates that BMC systems are more resilient than conventional systems to cybersecurity vulnerabilities. Overall, this comparison highlights the advantages of using the BMC paradigm to build more secure computing systems.

The rest of this article is organized as follows: Section 2 provides an overview of conventional computing systems, including their applications, guidelines, and characteristics. Section 3 provides details on BMC systems, including their background, guidelines, and characteristics. It also establishes a knowledge base for evaluating the security potential of BMC systems. Section 4 compares BMC systems and conventional computing systems with respect to cybersecurity vulnerabilities using the respective guidelines and characteristics. Section 5 identifies cyberattacks' root causes and preventive mechanisms and evaluates the resilience of BMC systems to these cyberattacks. Section 6 discusses the significant contributions of this study. Section 7 concludes the article and suggests more research in secure computing systems.

## 2. Conventional Systems

Conventional systems evolved from mainframes, resulting in desktops, laptops, smartphones, and IoT devices. Most of these systems use some form of an OS or kernel. During the mainframe era, there was no Internet and no online users. The OS was used as a mediator for many users and application programs to provide their services. The OS protected itself from users and protected users from each other. As computers evolved over the years, OSs continued to play a significant role in building computer and information systems. Later, when the Internet became the primary vehicle for communication between users, communication merged with computing, thus extending the OS's role to provide communication. Now, most computing systems have computing and communication on a single device. For example, smartphones perform computing, communication, and control as part of the same system, and they are seamlessly homogenized to work on the Internet. Smartphones resulted in user-centric computing serving client applications.

Figure 2 illustrates how computer applications use system calls or APIs provided by the OS. Application programmers are isolated from hardware and focus on writing applications in each programming language. There is heterogeneity at every level, including communication protocols, system libraries, device drivers, third-party software, network interface cards, standards, OSs, and computer architectures. They follow an evolutionary path and continue changing and enhancing products rapidly.

During this evolution, open systems, standards, free software, free access, remote control of devices, and globalization proliferated worldwide, resulting in billions of users [1] and the IoT [33]. A given application's security depends on the security of the underlying OS and may be compromised due to OS vulnerabilities [2]. As OS versions and releases often change, its platform-related entities change accordingly. These fast changes in hardware and software may also introduce more security vulnerabilities.

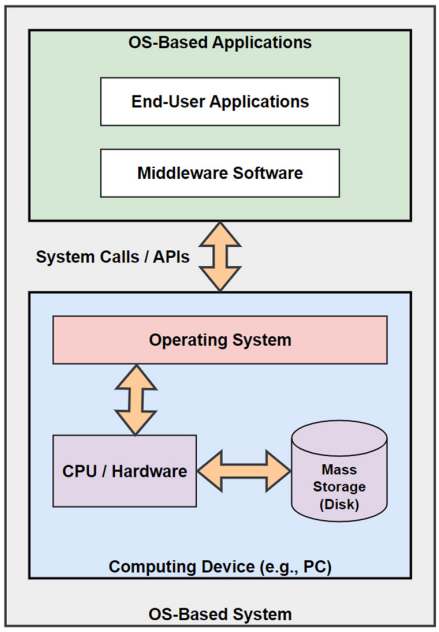


**Figure 2.** OS-based system.

Figure 3 illustrates a typical application development process and its development and execution environments. A given application only controls the functional flow and plays no role in maintaining an execution environment. When an application is created, it is linked with the system or external libraries. When an application is running, its execution is dependent on a variety of OS components. In addition, there may be other applications running in a multi-programming model. Today's Internet applications running on a device may be affected by cookies, network programs, sockets, shared memory, and background downloads. As the current OS provides computing, communication, and control in each box, a running application may be exposed to the side effects of the host OS and other applications running at the same time.

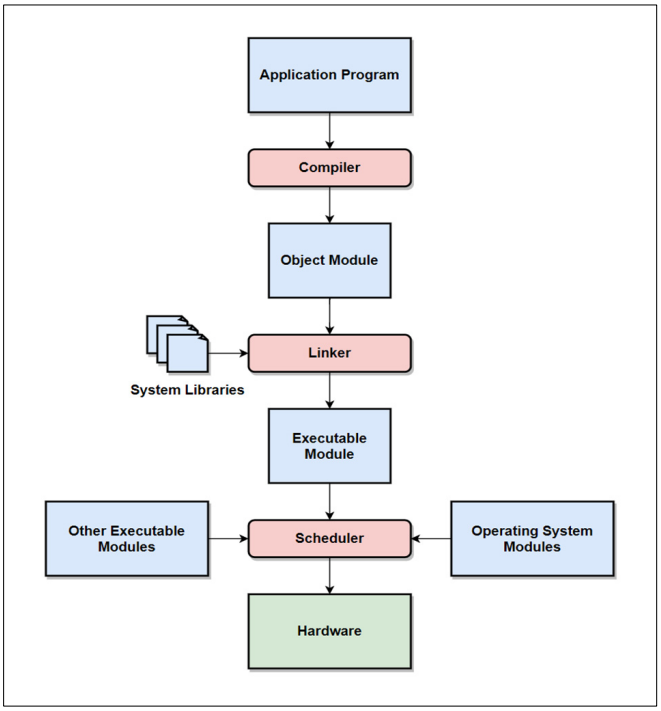


**Figure 3.** Conventional application development process.

Conventional system properties are typically classified into two categories: system guidelines and system characteristics.

### 2.1. Conventional System Guidelines

In our literature review spanning many references, some of which are included in this article, we have identified 20 conventional system guidelines as described below. This list is not exhaustive; however, it serves as a basis for comparison with BMC system guidelines.

1. System Approach: Over a 50-year period, computer architectures have been following an evolutionary path in processors and OSs, which are the backbone of all computing and information systems.
2. Open Systems: During the 1960–1990 period, most computer and information systems were closed systems and proprietary in nature, thus protecting a given industry's intellectual property. These systems and their know-how were limited to confidential and authorized users based on their need-to-know policy. Due to the emergence of the Internet and globalization, most of the systems and their knowledge became open to the world. While an open system allows for easier resolution of security vulnerabilities, it also means that attackers and developers have access to the same information, which implies that it is not more secure.
3. Global Focus: Many information systems are architected, designed, and implemented for a global market, as this market is much larger than a given country's local market. This resulted in large-scale commercialization on the Internet and more revenues and profits for related industries.
4. Global Users: Computer and information system users are all over the world, and they are not always easily identifiable and trackable.
5. Equal Access: With an Internet connection, any global user has access to most information on the Internet; this includes attackers as well.
6. Learning and Knowledge: Information about system internals and system security is often posted on the web. This helps the attackers to easily access the information they need at a faster pace. Defenders also put problems and fixes on the Internet as soon as they are discovered. This enables attackers to create new attacks with less effort. The following two quotations from [34] clearly illustrate the side effects of free learning and knowledge on the Internet. (a) "Open-source Intelligence (OSINT) tools could gather data from various publicly available platforms and thus help hackers identify vulnerabilities and develop malware and attack strategies against targeted CI sectors." (b) "OSINT tools: indirect reconnaissance data, proof-of-concept codes, and educational materials. The thematic results from this study reveal an increasing amount of open-source information useful for malicious attackers against industrial devices, as well as the need for programs, training, and policies required to protect and secure industrial systems and CI".
7. Unrestricted Internet access: Many Internet and web applications are available for all users worldwide with an Internet connection. Advertisements swamp systems almost instantly after access to the Internet. Searching online for a particular product or service immediately results in advertisements for related products or services. The advertisements may continue persistently for an extended period. This may be beneficial for some users to quickly learn about products and services. Still, it has many side effects, including a waste of time for users, distraction from a user's current tasks, and loss of privacy of email addresses, phone numbers, and other personal information. Personal information shared with online vendors to conduct any transaction on the Internet may be distributed to other vendors.
8. Layered Systems: All computer and information systems are layered with respect to architectures, protocols, interfaces, and security. Attackers may exploit these layers to suit their attack environment.
9. Heterogeneity: Current computer and information systems encourage heterogeneity at every stage of building computer and information systems, assuming that it will

promote innovation and productivity [35]. Instead, heterogeneity and incompatibility at every level interfere with the main goal of building end-user applications. With today's high-performance processors coupled with cheap hardware, due to heterogeneity, tweaking small improvements may not pay off as much as before. There is a need to reduce heterogeneity by developing basic building blocks for computing at every level that are long-lived. Heterogeneity exists in processors, OSs, protocols, user interfaces, programming languages, hardware, software, including third-party software, and more. This increases the complexity of computers and information systems [36] and provides more avenues for attackers to exploit cybersecurity vulnerabilities.

10. User/Developer Convenience: User convenience takes higher priority over other design issues in defining system guidelines. For example, it is convenient for users to choose a short-length PIN code for bank transactions, which may not be secure. Online teaching is convenient for faculty and students, but it may not be secure. As it is convenient for businesses to use the Internet for all transactions, physical security and privacy may be ignored. Many online transactions require personal information, causing a loss of privacy. In fact, without a smartphone, it is impossible to do many Internet transactions today. These actions may indirectly result in system vulnerabilities and cybersecurity weaknesses. In addition, developer conveniences, such as using a different programming language for a special purpose, using DLLs, downloading files and software, remote logins, remote administration, and third-party software, may also affect cybersecurity [37].
11. Training: All users and customers must obtain training as needed.
12. Software Installation: This is conducted online due to convenience and frequent changes.
13. Wi-Fi: Wireless connections are commonly used for computing devices such as a mouse, keyboard, headphones, and Internet access. Most Internet transactions are conducted using Wi-Fi and often with a smartphone.
14. Scripts and Batch Files: Scripts and batch files are used by administrators locally and remotely for convenience. This enables them to work remotely.
15. Attachments and Links: Email attachments are commonly used for convenience. Web links are also allowed in the emails for user convenience.
16. Social Media Platforms: They have become a part of everyday lives due to globalization, enabling speedy communication and sharing of information online.
17. Advertisements: There are advertisements appearing continually in social media, smartphones, TVs, computers, and websites.
18. Unsolicited Web Sites: Mostly for commercial and marketing reasons, unsolicited links to websites frequently appear in email and other Internet applications. It is not easy to distinguish between a link to a good site or a bad one.
19. Automated Tools: Automated tools such as Wireshark and reverse engineering tools such as Object-dump are available on the Internet. There are many such tools available for attackers to use.
20. Cookies: Cookies are used to track visits to websites for marketing purposes and user convenience, but they can also be used to launch malware.

### *2.2. Conventional System Characteristics*

The following characteristics were obtained from our literature review of the security papers listed in this article. We have identified 20 conventional system characteristics as described below. This list is by no means exhaustive; however, it serves as a basis for comparison with BMC system characteristics.

1. OS/Kernel/Embedded: Most systems have an OS or kernel. They can also be embedded systems. Due to the complexity and size of modern OSs, lean operating systems were designed to improve security and reduce complexity. Some middleware (e.g., the OS) makes software systems easy to build and isolates programmers from hardware intricacies.

2. Applications: Computer and information system applications are platform-centric and computing device-dependent. No proper abstractions have been used to build applications. Thus, applications are redundant on different platforms. A web browser on a smartphone is different from that on a PC. Email systems are also different on various platforms. Due to this redundancy and lack of proper abstractions, an end-user application may be duplicated many times, resulting in a large application space.
3. Programming Languages: Multiple programming languages are used in conventional systems. Different languages are often chosen based on the type of application, programmer's skills, convenience, and performance. Performance may not make much difference as current processor speeds are increasing in a short period.
4. Executables: Different operating systems support different executable formats. Some applications may have many executables, thus creating multiple address spaces. Inter-module communication may require message passing between modules.
5. Linking: Dynamic linking is used for many reasons. It reduces the executable sizes, as some modules can be linked at runtime. It is also convenient, as the compiler does not know where the system libraries and other modules are loaded in memory at compile-time. In addition, in multi-module design, the interfaces to other modules are not known at compile time.
6. Loading: Dynamic loading and linking are related. Dynamic loading at run time is also very convenient, as noted above for linking. In modern OSs, dynamic loading and linking (DLL) is commonly used to make the executables smaller and to enable use of external modules as needed at run time.
7. Multi-tasking: Multi-tasking or multi-programming is necessary for all computing systems, as the CPU cannot be kept busy all the time. Most programs need I/O; they cannot run when an I/O request is pending. In conventional systems, when one program is idle, other programs and the OS can run. A given program may not be running alone until its completion.
8. System Calls/API: OSs provide system calls to communicate with the hardware. OSs also provide APIs, enabling programmers to access hardware interfaces. System calls use interrupts to invoke OS services.
9. Sockets: Operating systems provide socket interfaces to communicate between processes within a node or with a remote node.
10. Open Ports: When a packet arrives at an OS, it uses the destination port number in the packet to deliver the packet to its application. A given application running in the machine uses a designated port number to communicate with other nodes. There may be some ports open in an OS to accommodate unexpected applications running in the machine.
11. During Execution: In an OS environment, a given application, other applications, and OS processes and threads run concurrently. Some of them may interact with others intentionally or unintentionally.
12. Event/Interrupt Driven: Most conventional systems use event-based and interrupt-based processing.
13. Shared Memory/Message Passing: Both are used for local communication, and message passing is used for remote communication.
14. Concurrency Control: Inter-process communication in an OS environment may require concurrency control mechanisms. Locking and semaphores are used to handle concurrency.
15. I/O: Most I/O is interrupt-driven in an OS.
16. Third-Party Software: Usually, there is software from many vendors in current systems.
17. Network Interfaces: Protocols such as TCP or QUIC have many interactions between a server and a client during connection establishment, data transfer, and connection termination. Security protocols such as TLS also require many interactions to ensure secure data transfer.

18. Internet Communication: In conventional systems, most communications are conducted through the global Internet. There are billions of users on the Internet while a given communication is in progress.
19. Internet Downloads: For convenience and data sharing, users download software, applications, files, pictures, private data, etc. on the Internet. During downloads, a user may have to enter personal data, resulting in a loss of privacy.
20. IoT: The IoT is growing at an exponential rate. Currently, there are 17 billion IoT devices [38].

## 3. Bare Machine Computing (BMC) Systems

This section describes the BMC paradigm [39] and gives some insight into BMC system development. The following subsections provide some details on building applications using this paradigm. A variety of real-world BMC applications have been built, demonstrating the feasibility of this paradigm. BMC is an alternative approach for building computer applications without running a centralized kernel or an OS. The BMC paradigm provides other benefits, including significantly reducing obsolescence and cybersecurity risks.

### *3.1. BMC Paradigm*

The BMC paradigm was originally motivated by the rapid obsolescence [4] and continuously emerging security vulnerabilities in computer and information systems. Obsolescence impacts computers and information systems, causing hardware and software to be replaced before their useful life span. The topic of obsolescence is not within the scope of this paper. The BMC approach resembles computing when it started in the 1960s using bare hardware operated with switches and buttons. In current technology, hardware is cheap, provides many functionalities, and allows more logic to be placed on a single chip. It also allows computing devices to be bare, as a processor can provide needed hardware interfaces to application programmers.

In a BMC system, as shown in Figure 4, applications directly communicate with the hardware without any middleware. For example, suppose we want to print the text "Hello" on the screen. Figure 5 illustrates how a programmer can directly control writing on the screen without using an OS. This simple example shows how a programmer can directly access the hardware through C++ -> C -> assembly as a single thread of execution without interruption. It also shows how a programmer views the underlying hardware in the BMC paradigm. It is possible to automate this process to make it user-friendly for developers. In this example, the assembly call writes string data to the video memory location at 0x000b8000. A BMC programmer controls all the code shown here. There are no other dependencies in the BMC code. This philosophy is carried throughout the development process of any BMC application. Using this approach, we can write all direct hardware interfaces needed for information systems applications and development. Eventually, these direct hardware interfaces can be built on the processor chip. If the output must go to graphics memory, the same logic is used with the help of a bare graphics driver. A BMC system is not the same as a bare metal system or embedded Linux [40], as BMC is a general-purpose computing paradigm. BMC servers are not the same as bare-metal servers or virtual servers in the cloud [41], which require some OS to provide services.

The BMC paradigm is based on an application-oriented approach. It divides all computer applications into domain-specific applications using object-oriented abstractions. Examples include banking, online transactions, manufacturing, payroll, desktop and office applications, accounting, email, chat, and browsers. A given domain-specific application may contain one or more applications bundled as a single domain application suite used at a given time. One such example is text processing, spreadsheets, and Internet search. Each domain-specific application suite contains all the necessary code to run on a given bare machine. Although BMC currently chooses Intel processors and their instruction set architecture, it can be adapted to work with any other computer architecture as needed. The BMC paradigm assumes hardware integrity. The paradigm is applicable to any computing device, including a smartphone.

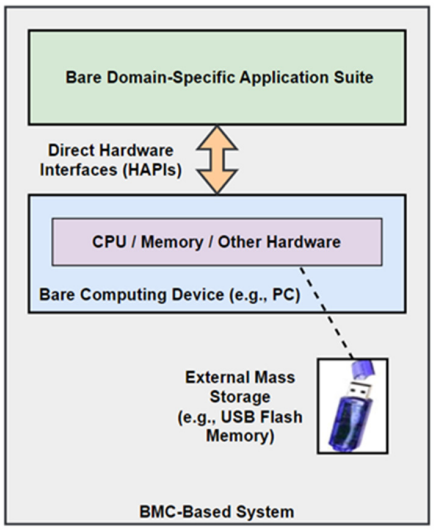


**Figure 4.** BMC-based system.

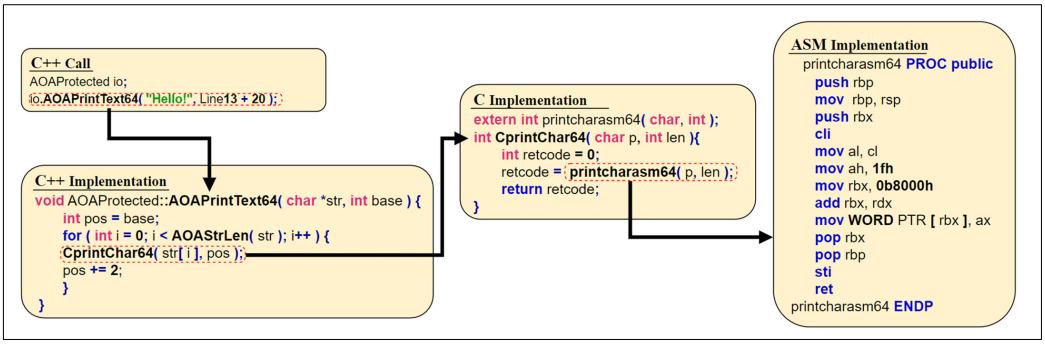


**Figure 5.** Hello program using the BMC paradigm. * Refers to a pointer declaration in C++.

In BMC, the first step is to make the computing device bare. This means it has no operating system, no hard disk (such as mass storage), and uses the BIOS for now. Eventually, the BIOS can be bundled with the associated application suite. The code can run on older or newer Intel processors, which are ×86 and ×64 compatible. The second step is to write the software in C/C++ and a small amount of assembly code where needed. The BMC paradigm is based on a single programming language environment. Boot, loader, and interrupts are written in assembly. One or more applications can be compiled as an application suite, generating a single monolithic executable. This is statically compiled and linked with no external software or libraries. The small executable contains all the required code to run its application suite. A given application suite executable is typically less than 2 MB. It carries the necessary bare drivers [42] and code to support the required functions. There are no kernel or system calls and no OS-related management functions.

The application programmer controls the hardware and required interfaces, and the application suite directly communicates with the hardware without using any middleware. The BMC paradigm also uses bare devices to communicate on the Internet, forming a bare Internet [43]. Only authorized bare users can communicate on bare Internet.

Each application suite can define its own memory map, as shown in Figure 6. This memory map shows how content in an application suite USB is copied into main memory. The boot code from sector 0 is copied into memory location address 0×7c00 (for IBM Compatible PCs). The start program from the USB is copied at memory location 0×600 and includes two header sectors. The start program is loaded in the real memory area. After the boot program runs, it jumps to 0×3900 in real memory to run the start program. When the start program runs, it copies the domain-specific application suite executable from the USB into the memory location starting at 0×00111000, which is above 1 MB and is in protected

memory. When the start program is completed, it starts executing at 0×00111000 (which is a main () entry point). All these numbers are extracted by using Microsoft tools to identify entry points and other components in the executable. Notice that there is no need for virtual memory in BMC systems, as there is no OS and only a few applications are bundled and run at a time. Each domain-specific application suite has its own characteristics and can be loaded in custom locations in memory accordingly. This process can also be automated in BMC systems. The above memory map can be customized for domain-specific application suites, which shows simplicity and flexibility in building BMC applications.

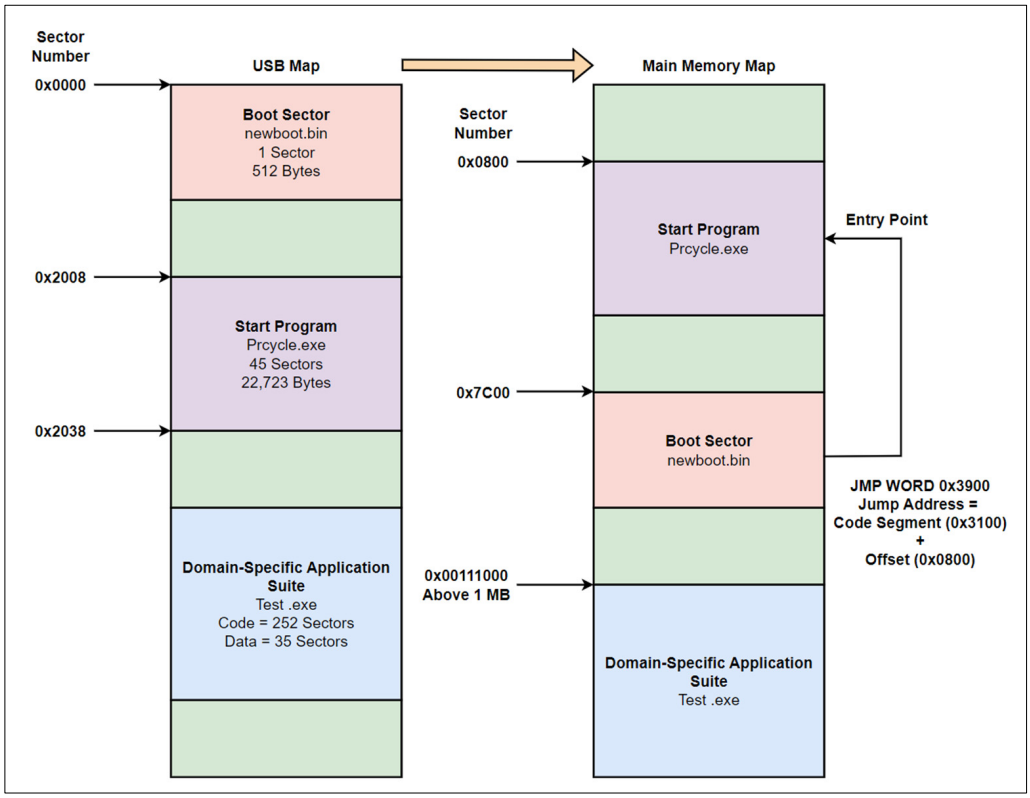


**Figure 6.** Memory map.

As per our knowledge, there is no true bare machine research/development/product like BMC. However, there is ample related work in making the OS lean, moving OS functions to application space, bypassing the OS to gain fast access to hardware resources, and designing an OS for IoT devices. Examples include the exokernel OS architecture enabling applications to manage the hardware and improve performance [44]; Tiny-OS for low-power wireless devices [45]; and Palacios and Kitten, a lightweight OS and VMM (virtual machine monitor) for high-performance computing [46]. Microkernel and related approaches [47,48] keep only essential OS services in the kernel to increase performance in applications such as cloud computing. IO-Lite and kernel bypass techniques [49] provide direct I/O access to applications to speed up I/O. For data center servers, a customized OS [50] can be used to bypass kernel services. RIOT is a small OS suited for IoT devices [51]. Tradeoffs in designing minimalist instead of functionality-rich platforms are analyzed in [52]. The BMC approach is a minimalist approach, where there is no OS or kernel.

### *3.2. Compilation Process*

This section illustrates compilation and the BMC development process. It shows that developing BMC applications is simple and independent of other development environments, which change often. It also shows that the BMC application suite is a single monolithic executable with one address space for a given set of applications. All the batch files used are shown in Figure 7.

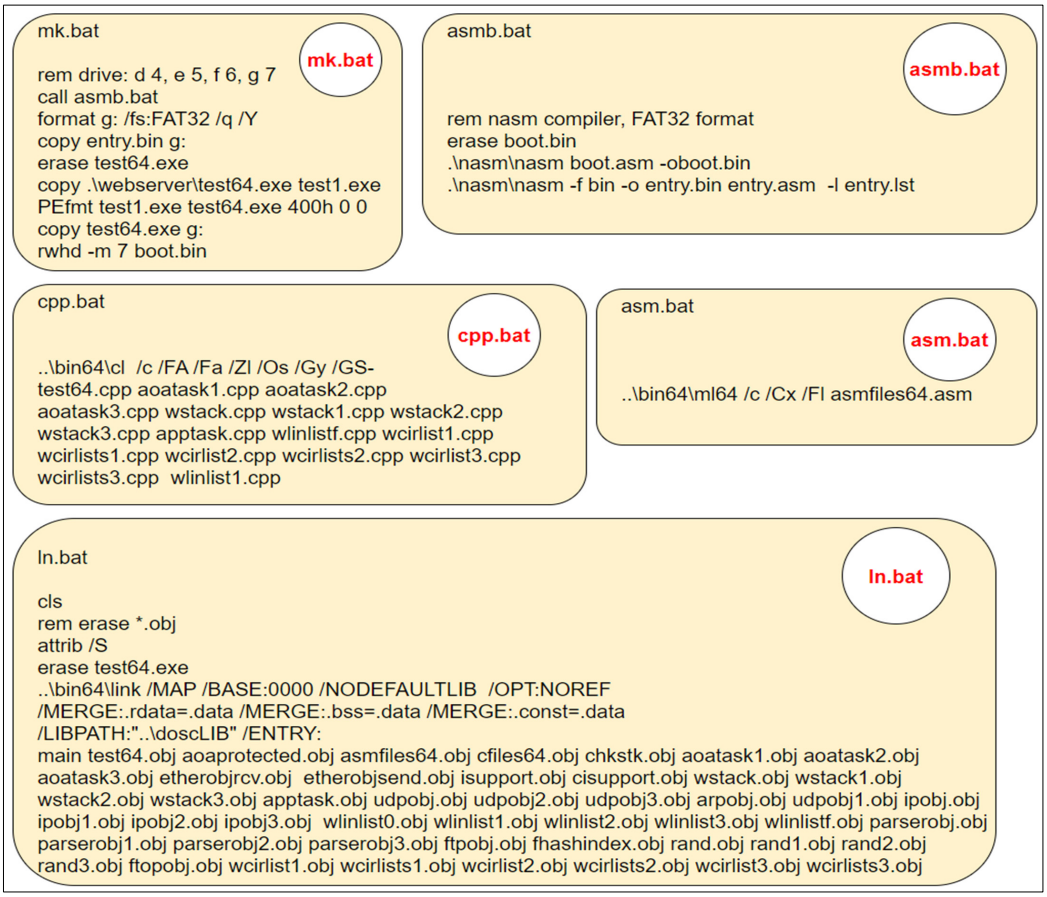


**Figure 7.** Compilation process. * Refers to all file names with the .obj extension in the command rem erase *.obj.

Currently, the development process in the BMC uses batch files and takes complete control by using a particular version of Visual Studio (VS 64-bit) as it changes often. These batch files allow us to conduct a compilation process independent of any version of the OS release. At present, we use a Visual Studio/bin directory to compile and link our bare code. The development environment does not use/lib and/include files. A boot.asm has a boot code, and entry.asm has an entry code for the 64-bit application suite. The asmb.bat file is used to generate these binary files, which are written in the NASM assembler. The mk.bat file is used to create a bootable USB, which has boot, entry, and test64.exe (application suite). The PEfmt tool is written for BMC to properly organize code and data segments as needed for the Microsoft PE format.

The asm.bat file is used to compile MASM assembly code that provides direct hardware interfaces. There is a cpp.bat file in every directory that has C/C++ code. The cpp.bat file shown below is for the main directory in the application suite. Similarly, there is a separate cpp.bat file in each directory, where there is C/C++ code. The object files from each directory are used together to run a link process. The ln.bat file creates an application suite executable named test64.exe. This description of batch files refers to creating a multicore web server using the BMC paradigm. This total control of compiling, linking, and creating a static executable prevents other malware from being linked with this application suite at run time.

### *3.3. Sample Direct Hardware Interfaces*

Some sample direct hardware interfaces, such as tasking, reading data from an Ethernet buffer, and processing a received packet, are discussed in the following sections. The code snippets demonstrate how the BMC code integrates all components seamlessly without any layers. It also shows that a programmer controls all hardware elements and software structures, such as Ethernet buffers.

### 3.3.1. Creating, Inserting, and Running a Task

The code snippets in Figure 8 from a 32-bit application suite for a web server show how a task is created in the BMC paradigm. Notice that task generation is bundled with the application suite and is not visible to outsiders, which prevents intrusion into the code. This code is generated, statically compiled, and linked. A given application suite decides the task creation and its control flow. The deleting task is not shown here. The code is self-explanatory and follows chronological order.

| No. | Code Snippet |
|---|---|
| 1 | //** Get the function address to create a task **//<br>// This function returns the address of a C++ member function which is then<br>// used to create a task in the BMC system.<br>long *getFunPtrOthers(pmfdOthers abc2) {<br>// This uses xyz as a stack pointer to locate the function address<br>long *ab1;<br>long *xyz; // Stack pointer<br>pmfdOthers abc1; // Instance of a member pointer<br>apptask first; // Instance of a class<br>xyz = ( long * ) &first;<br>abc1 = abc2; // Initialize the pointer with a function<br>xyz--;<br>ab1 = ( long * ) *xyz;<br>return( long * ) ab1; }; |
| 2 | //** Capture the address of the task **/<br>// HttpTask is a C++ member function for a task Initialize pointer with HttpTask<br>// function address.<br>FunPtrX = &apptask::HttpTask;<br>// Store function pointer in an array.<br>FunAddrArray[ 7 ] = ( long )getFunPtrOthers( FunPtrX ); |
| 3 | //** Create the task and push task ID onto the stack **//<br>// The following two lines can be repeated for as many tasks as needed.<br>// Task IDs are pushed onto stack and then popped to be inserted into a circular list.<br>// This circular list acts as a "ready queue" in the BMC system.<br>// Create a task using a function address and return a unique task ID.<br>Task_ID = createTask( ( long ) FunAddrArray[ 7 ], 0 );<br>stk.push( Task_ID ); // Push this task ID onto a stack. |
| 4 | //** Insert the task into the Circular List **//<br>// When HTTP request arrive in TCP the following insert function is called.<br>tsk.insertHttpTask( TCBRecordNumber, io.AOAgetTimer ( ) );<br>// This function inserts the task into a circular list.<br>int apptask::insertHttpTask( int tcbrno, long timerc ) {<br>int taskid;<br>long timer;<br>// Code not shown ...<br>if( stk.empty() == 0 ) {<br>stk.pop( &taskid ); // Get task ID from the stack.<br>// Code not shown ...<br>cir.insert( taskid, 0, 0x05, tcbrno, timerc );<br>// Code not shown ...<br>} return 0; }; |
| 5 | //** Run a task from a circular list **//<br>// In main loop, a task ID "n" from the circular list is used to run the task.<br>task.AOArunTask( n ); // C++ member function |
| 6 | //** Run task call in C **//<br>// The above C++ function calls the following C funtion<br>runTask( Task_ID ); |
| 7 | //** Run task in assembly **//<br>// The above C call invokes the following assembly function<br>runTaskasm32( Task_ID ); |
| 8 | //** Run task call implementation in assembly**//<br>; The assembly call for runTask32 is shown below.<br>; Notice that it invokes an interrupt 0efh to run a task.<br>runTaskasm32 PROC C public Task_ID:DWORD<br>; Code not shown ...<br>int 0efh<br>; Code not shown ...<br>ret<br>runTaskasm32 ENDP |

**Figure 8.** Task creation.

### 3.3.2. Reading and Processing an Ethernet Packet

In the BMC paradigm, device drivers are part of an application suite. They are written to work without using any OS system calls or API. All device drivers must be written from

scratch and are known as bare device drivers. There are two data structures in the Ethernet driver, known as the transmit and receive descriptor lists. These are circular lists with 4096 entries. When a packet arrives, the hardware sets a bit known as descriptor done (DD). Application programmers can read this bit to check if a packet is received. Similarly, when a packet is transmitted, the hardware sets the transmitter DD bit. An application program can examine these DD bits to communicate with the Ethernet control hardware.

For receiving a packet, if a given slot's DD bit is set, then this packet will be read and passed onto the application, as shown in Figure 9. The function isRDescDone0() in Code Snippet 1 shows how an application can peek into the Ethernet receive buffer for an HTTP GET request and copy the GET data into another buffer. After copying the data, it calls discardPacket() to reset the DD bit. Code Snippet 2 shows how a received packet invokes the RcvCall1() function to process a request, and Code Snippet 3 shows how it calls the IPHandler() function. Code Snippet 4 illustrates how it calls the UDPHandler() function, and Code Snippet 5 shows the processing of the UDP packet with comments, as this step has a large amount of code, which is not shown here. In essence, this code demonstrates that all the Ethernet packet processing is done as a single thread of execution without any layers. In addition, this is application code designed and controlled by an application programmer without the intervention of any middleware. It is event-driven with polling to receive packets. Send is not shown as it is like receive.

| No. | Code Snippet |
|---|---|
| 1 | `// Reading and processing an Ethernet Packet.`<br>`retcode = EO.isRDescDone0( bcoreid, 1, &pktsize, &preqid );`<br><br>`// Peeking at an Ethernet packet in an application code.`<br>`int EtherObjRcv::isRDescDone0( int cid, int takeget, long long *pktlen, long long *reqid ){`<br><br>`WStack stk0;`<br>`// Other variables are not included.`<br>`unsigned long long *ul2;`<br>`*pktlen = 0;`<br>`*reqid = 0;`<br><br>`// Each descriptor is 16 bytes in size.`<br>`lptr = ( long long * ) ( RDLPointer + ReceiveOutPtr * 16 );`<br>`dptr = *lptr;  // Data pointer for the packet arrived.`<br>`cptr = ( char *) ( dptr );`<br>`iptr = ( long long *) ( cptr );`<br>`lptr++;`<br>`temp = *lptr;  // Status, checking for the D bit.`<br><br>`// Reading receive descriptor and checking if the packet arrived.`<br>`temp = temp & 0x0000000100000000;`<br><br>`if ( temp != 0 ) {`<br>`*pktlen = ( *lptr ) & 0x000000000000ffff; // Get the packet length.`<br>`retcode = 0;`<br><br>`// Check if it is a GET packet.`<br>`if ( cptr[ 0+42 ] == 'G' && cptr[ 1+42 ] == 'E' && cptr[ 2+42 ] == 'T' ){`<br>`lcounter++;`<br><br>`// Checking if the stack has any indexes.`<br>`if ( stk0.empty() == 0 ){`<br>`preqid = stk0.pop();`<br><br>`*reqid = preqid;`<br>`rtype = 1;`<br><br>`// A round-robin algorithm is used to load balance cores.`<br>`for ( i = 0; i < ( *pktlen ); i++ )`<br>`dataptr[ i ] = cptr[ i ];  // Copying received data into a buffer.`<br><br>`// Inserting packet into a circular list for a given core (code not shown)`<br><br>`discardPacket();`<br><br>`}  // End of if statement for stk0 has the indexs.`<br>`}  // End of if statement for GET.`<br>`}  // End of if statement for temp.`<br>`} // End of isRDescDone0 function.` |

**Figure 9.** *Cont*.

| No. | Code Snippet |
|---|---|
| 2 | // Processing a received packet<br><br>// Core 1 processing requests<br>// Check if there are any pending requests for Core 1 in circular list<br>**if** (cir1.**getCount() > 0) {**<br>// Get a packet from its circular list.<br>// Code not shown...<br>// Process the arrived request.<br>**RcvCall1(** preqid, pktsize, type **);**<br>**}**<br>// (The rest of the code is not shown) |
| 3 | // Implementation of RCV call1.<br>**void** AOATask1**::RcvCall1( int** reqid, **int** pktsize, **int** type **){**<br>// Code not shown...<br>retcode = ip.**IPHandlerICMP(** pktsize **-** 14, starttime, type **);** // Call IP Handler.<br>// Code not shown...<br>**}** // End of RCV call1. |
| 4 | // Implementation of IP Handler.<br>**int** IPObj1**::IPHandler(int** reqid, **char** *IPPack, **int** size,<br>**char** *macaddr, **long** starttime, **int** currenttask) **{**<br>// Check IP header formats and errors. Remove header.<br>// Code not shown...<br>**if (** protocol **==** UDP **) {**<br>// Code not shown...<br>// Call UDP Handler.<br>retcode **=** udp.**UDPHandler(** reqid, **&**IPPack**[** HeaderLength **],**<br>**(** TotalLength **-** HeaderLength**), &**IPPack**[** 12 **],**<br>**&**IPPack**[** 16 **],** protocol, macaddr, currenttask **);**<br>**}** // End of if statement.<br>**}** // End of IPHandler function. |
| 5 | // Implementation of UDP Handler.<br>**int** UDPObj1**::UDPHandler(int** reqid, **char** *UDPPack, **int** size,<br>**char** *SourceIP, **char** *TargetIP,<br>**int** Protocol, **char** *dmac, **int** currenttask**) {**<br><br>// UDP header processing, checking errors.<br>// Code not shown...<br>// Process UDP request.<br>// Send header and data; putting the data in an output circular list for the core.<br>// Core 0 takes this data and puts it into the Ethernet buffer.<br>// Code not shown...<br><br>**}** // End of UDPHandler function. |

**Figure 9.** UDP-based protocol.

### *3.4. Summary of the Above Code Snippets*

In the above Sections 3.1–3.3, we discussed the use of direct hardware interfaces, the BMC development methodology, and some application code snippets. It is evident from these discussions that this approach is different from conventional methods. When a machine is made bare, the programmer controls software development at compile time and run time. The programmer also controls run-time behavior and does not allow any unintended functions to interfere with a given application suite. Hardware interfaces are included at development time, and flow control cannot be altered at runtime by other applications. This is truly an application-oriented model with a security-by-design approach. The task creation code shows how it is intertwined with an application suite, thus allowing the system to perform only intended tasks. The Ethernet peeking demonstrates that all device drivers are also seamlessly integrated with a given application suite. The packet processing shows how Ethernet, IP, and UDP are integrated as a single thread of execution instead of going through different layers. In this approach, packet processing goes through a simple invocation of calls from one protocol to another. Overall, a given application suite is custom-designed and simple; there is only one user mode; the machine is bare and has no valuable resources; and the programmer has control at all stages.

### *3.5. Properties of the BMC Paradigm*

There have been many real-world applications built to demonstrate the feasibility of bare systems using the BMC paradigm, including chat [43], Web servers [53], USB

drivers [42], and Web mail servers with TLS [54]. These real systems are used to identify system guidelines and system characteristics.

#### 3.5.1. BMC System Guidelines

1. System Approach: It is a revolutionary approach that was invented to address obsolescence and security problems. It makes applications independent from the execution environment, and all computing devices are bare.
2. Open/Closed System: It is a closed system. Injecting an attacker's code is not possible. Open source must not be confused with open systems. The BMC is a closed system. The computing box is bare, and the application running in the bare box is not accessible to hackers. In case hackers create a similar system, they can only hack their own system, not others. If we make an open system, security vulnerabilities are easy to fix, but hackers and developers are at the same knowledge level.
3. Global Focus: It has a local focus. It is architected, designed, and implemented to avoid global users.
4. User USBs: There are dual USBs (the first for booting and the second for an application suite). Both USBs must be physically secured.
5. Equal Access: All bare users have equal access.
6. Education and Knowledge: Restricted to authorized bare users.
7. Layers: There are no layers at any level.
8. Bare Internet: To make BMC viable, a bare Internet concept [43] is introduced. In a bare Internet, all intermediate nodes, such as routers and gateways, must be bare, and they must be physically secured. At present, a bare Internet is overlaid on the existing Internet.
9. Heterogeneity: No heterogeneity is allowed in programming languages, hardware, and software. The security of systems and applications is more controllable.
10. User/Developer Convenience: User convenience takes lower priority over other design issues in defining system guidelines. For example, a system administrator must physically distribute user accounts to guarantee the authentication of users. This is not convenient, but all other electronic means may not be secure. The BMC systems are not global, and the users are limited within a given domain-specific application. These systems are designed for secure domains, not for a global world. Non-bare users can use conventional systems if BMC systems do not fit their needs. BMC systems provide an alternative approach to current computing systems.
11. Training: All authorized bare users must have consistent training and education.
12. Software Installation: Online installation is not allowed. There is nothing to install in a bare machine. The user carries secure USBs. These USBs are distributed to authorized bare users by physical means.
13. Wi-Fi: Wi-Fi is currently not supported due to its security issues.
14. Script and Batch Files: These are not allowed.
15. Attachments and Links: Attachments and links are not allowed in emails.
16. Social Media Platforms: There is no support for social media applications. Social media can use the conventional Internet, which is isolated from a bare Internet. This approach will reduce the number of users on a bare Internet.
17. Advertisements: Advertisements are not allowed. All domain-specific applications and their users are properly authorized and tracked on a bare Internet.
18. Unsolicited Websites: Only access to authenticated bare websites is allowed.
19. Automated Tools: Automated tools are designed to work with only bare computing devices and applications. Their use is restricted to bare users only.
20. Cookies: No cookies are allowed.

#### 3.5.2. BMC Characteristics

1. OS/Kernel/Embedded: The OS/kernel/embedded concept is eliminated. There is no such centralized program; each domain-specific application is a self-controlled,

self-managed, and self-executed entity. There is no interaction with entities outside an application suite. As there is no OS/kernel, a user carries a flash drive with boot code and a domain-specific application suite. Figure 6 shows a memory map for a flash drive containing a client's UDP-based chat program.

2. Applications: All computer applications are polarized as domain-specific entities. They are independent of any platform. In conventional systems, they are dependent on platforms and execution environments.
   We have developed and demonstrated numerous domain-specific applications, including web sites, webmail, email, VoIP, text-only browsers, editors, file systems, database applications, and chat. In these systems, there are bare servers and clients that run on bare machines. For example, in the chat domain, there are physically vetted users who communicate in their own domain. All these users are vetted and use bare machines. We use context-based authentication to validate users. This is one example of a domain. All the above domain applications were tested on the Internet with bare servers and clients. On the Internet, domain-specific applications are used to communicate securely within a domain using bare devices.
3. Programming Languages: A single programming language (C/C++) is used to write applications, thus avoiding all heterogeneity in writing applications.
4. Executable: It is a single monolithic executable, which implies a single address space. Only one executable format is allowed.
5. Linking: Uses static linking, which prevents expanding the code segment to load foreign code. This provides ultimate security for the BMC applications, as malware code cannot be linked.
6. Loading: Uses static loading, which prevents extending the code segment to load malware code.
7. Multi-tasking: Multi-tasking is offered within an application suite controlled by an application programmer through events and its control flow.
8. System Calls/API: There are no system calls or APIs available to the outside world (outside an application suite). It uses a direct hardware API, integrated within an application suite.
9. Sockets: No sockets exist in the BMC paradigm, as there is no OS. Remote computer communication is implemented within a process and hidden inside an application suite.

For example, attackers use sockets to create man-in-the-middle attacks. In the BMC paradigm, sockets are not used as we build domain-specific applications with no open ports, not allowing users to create their own threads or processes. That is, items such as sockets are not needed in implementing BMC applications. The BMC paradigm inherently does not need such security-prone facilities as it does not have an OS. This approach is different from the "security through obscurity" concept. There are many such facilities that are not needed in the BMC paradigm, which offers inherent security in BMC applications. The following quote also supports your concern.

10. Open Ports: There are no open ports in BMC programs. As it is a domain-specific application suite, IP addresses and port numbers are managed within each application code in a hardcoded manner. This is not visible outside the application suite. When a packet arrives, its corresponding event (pre-defined) will call an appropriate method to process it.
11. During Execution: During execution, only one application suite runs. There is no interaction with other application suites or external modules in a bare computing device. No exploits are possible for an attacker.
12. Event/Interrupt Driven: The application suite is event-driven. Limited user interrupts are used for input.
13. Shared Memory/Message Passing: Only a shared memory approach is used within an application suite.
14. Concurrency Control: Concurrency control is avoided by using circular lists. The code is simpler and more directly accessible to the program.

15. I/O: There are direct hardware APIs that are hidden within an application suite.
16. Third-Party Software: There is no third-party software used in applications.
17. Network Interfaces: All network interfaces are hidden from the outside world. Attackers cannot use this direct hardware API.
18. Internet Communication: Communication on the Internet is restricted to a bare Internet and its bare users only.
19. Internet Downloads: No downloads are allowed.
20. IoT: All IoT devices must be bare computing devices and follow the BMC paradigm and its characteristics as described in this paper.
21. Application Program Control: A given application has its own control flow as intended at design time.
22. Computing Device: A computing device (PC, laptop, smartphone, or other) is bare, meaning that it has no intelligence, no OS, and no mass storage. It cannot boot or run until external media runs the boot code and loads an application suite controlled by the owner. The bare device has no valuable resources. Thus, a bare computing device can be used by any user at any time. When one application suite is running, another one cannot run; therefore, there is no intrusion from other applications. There is nothing in the bare computing device to be attacked, as there are no valuable resources when it is not running. When a bare device is running, it must be physically secured by the owner. Physical security at the bare box is not convenient, but it is required to guarantee security of the device and the running application suite.
23. Users: Only bare users can communicate with each other. For a given domain-specific application, there are a limited number of bare users. They must be physically authorized, authenticated, and controllable. We have not defined any formal methods for authentication at this point; however, we can use existing models in banking, driver's license, etc.
    Our BMC system is totally controlled by its owner, with physical security. If a trusted user changes roles and tries to exploit other bare systems, this user can only damage his/her own system and not those of others. This is because an owner's valuable resources can only be accessed by the owner's application suite. When a BMC application suite is running, it is a closed system that performs only intended functions. In addition, when the roles of trusted users change, their privileges are changed in the system.

In the bare machine computing model, each bare box and an application suite is owned by an owner, and it is physically secured. If an outsider or insider gained the same knowledge as the original developer, then they could reconstruct a similar system of their own. However, they will be using their own system at their own location. All valuable resources are accessible only by the owner and stored in an owner's detachable mass storage. Moreover, if they come through the network, they will not be able to access resources, as only trusted users are allowed to communicate within the domain. Users within the domain are vetted and authenticated to communicate within the group for each transaction. In this model, all users in the group must be physically trusted. This system cannot solve insider attacks and physical security violations. All the bare nodes in the system as well as on the network must be bare and physically secure. This is a harsh requirement, but it is necessary to build highly secure systems.

24. Messages: Each message is encrypted, integrity-protected, and authenticated using credentials physically given to authorized users by an administrator.
25. User-Secured USBs: Uses two USBs, one bootable and one with an encrypted application suite. These USBs must be physically secured by a user.
26. BIOS/Firmware: Bundled with a domain-specific application suite (not implemented currently).
27. Protocols: A bare machine connects to a network via wired Ethernet. Network protocols are limited to IP and UDP (TCP/TLS is used for demonstration only) implemented within the bare application. Other protocols are not available outside the application

running on the machine. Each application communicates only with its peers to ensure reliability and proper functionality. Furthermore, protocols are integrated within the application, and there are no protocol layers as in a conventional system. The attacker does not have the path to invoke these protocols. Although an attacker can spoof IP addresses, such packets will fail user authentication and be dropped. Every packet has bare user authentication, encryption, an integrity check, and replay protection.

28. Passwords: No password files are stored on bare machines. Context-based authentication is used in bare systems.

The above BMC system guidelines and characteristics impose many restrictions for user options. This does not mean that the BMC applications have limited their user functionality. Any features or functions offered by conventional systems can also be provided by BMC applications. The functional requirements, design, and implementation are conducted in many ways to achieve the highest security possible by design. In many cases, a given functionality, protocols, and interfaces are hidden in an application. This is referred to as an A-to-A (application-to-application) approach.

For example, email attachments and downloads are not allowed in BMC characteristics. Users can still receive an attachment as a set of packets without explicitly providing attachment features. A sender and a receiver of an email application handle the disassembly and assembly of packets in their respective applications with a hidden knowledge of file characteristics. All emails and additional packets are encrypted when transmitted on a network. At the receiving site, it is ensured that this file data cannot be executed. Thus, attachments are handled differently than in conventional systems. Many other convenient features in conventional systems are implemented in BMC applications in a hidden manner.

Similarly, in BMC, software downloads are not allowed. All software that runs on a bare machine must be physically installed on a bootable USB by an authorized vendor for an authorized bare user. It is not convenient for bare users, but it is the most secure way to design BMC applications.

As per the following quotation, "In recent years, more advanced versions of "security through obscurity" have gained support as a methodology in cybersecurity through Moving Target Defense and cyber deception. NIST's cyber resiliency framework, 800–160 Volume 2, recommends the usage of security through obscurity as a complementary part of a resilient and secure computing environment" [55].

BMC code can run on older and new Intel processors without any (or just a few) changes. The variety and breadth of the BMC work [43,53,54,56] show that it is possible to build similar systems for secure applications in computer and information systems. The BMC paradigm can thus be an alternative to existing OS-based platforms and applications with security by design.

#### 3.5.3. BMC Limitations

As described before, the bare machine computing paradigm has unique system guidelines and characteristics to reduce obsolescence and provide inherent security in computer and information systems. However, this approach has its own limitations, as listed below:

1. It is not a general-purpose global system.
2. It requires physical security.
3. It needs all users to be authenticated and vetted.
4. It uses a domain-specific application approach (by dividing applications into domains).
5. Only users within a domain can communicate.
6. When domain-specific users change roles, their authentication privileges are revoked.
7. All nodes in the network must be bare to guarantee high security.
8. Uses the bare Internet, where all nodes are bare and physical security is assumed.

## 4. Comparison of Conventional and BMC Systems

This section compares system guidelines in Table 1 (20 items) and system characteristics in Table 2 (28 items). The system guidelines are informal requirements or policies needed to build a system. The system characteristics are inherent properties of a system. Some system guidelines and characteristics may be prone to security attacks. Some system guidelines and characteristics may be resilient to security attacks. Thus, we define two terms to compare conventional and BMC systems: PECSV indicates possible exposure to cybersecurity vulnerabilities, and PRCSV designates possible resilience to cybersecurity vulnerabilities. The references obtained from the literature review are shown in the tables to determine possible vulnerabilities and resiliency in BMC systems.

**Table 1.** Comparison of System Guidelines.

| Seq. | System Guidelines | Conventional Systems | PECSV | BMC Systems | PRCSV |
|---|---|---|---|---|---|
| 1 | System Approach | Evolutionary | | Revolutionary | |
| 2 | Open/Closed | Open | ⚠ Conventional Guideline 2 | Closed | ✔ BMC Guideline 2 |
| 3 | Global Focus | Yes | | Local Focus | |
| 4 | Global Users | Yes | ⚠ [1,2] | Local Users | |
| 5 | Equal Access | Yes | | Restricted | ✔ BMC Guideline 5 |
| 6 | Free Learning | Yes | ⚠ [57] | Restricted | ✔ BMC Guideline 6 |
| 7 | Free Internet | Yes | | Bare Internet | ✔ BMC Guideline 7 |
| 8 | Layered Systems | Yes | ⚠ [58] | No | ✔ BMC Guideline 8 |
| 9 | Heterogeneity | Yes | ⚠ [59] | No | ✔ BMC Guideline 9 |
| 10 | User/Developer Convenience | Yes | ⚠ [37] | Not Top Priority | ✔ BMC Guideline 10 |
| 11 | Training | Yes | | Yes | ✔ BMC Guideline 11 |
| 12 | Software Installation Online | Yes | ⚠ [60] | No | ✔ BMC Guideline 12 |
| 13 | Wi-Fi | Yes | ⚠ [61,62] | Not Supported Yet | ✔ BMC Guideline 13 |
| 14 | Scripts and Batch Files | Yes | ⚠ [63] | No | ✔ BMC Guideline 14 |
| 15 | Attachments and Links | Yes | ⚠ [64,65] | No | ✔ BMC Guideline 15 |
| 16 | Social Media | Yes | ⚠ [66,67] | No | ✔ BMC Guideline 16 |
| 17 | Advertisements | Yes | ⚠ [68,69] | No | ✔ BMC Guideline 17 |
| 18 | Unsolicited Websites | Yes | ⚠ [70,71] | No | ✔ BMC Guideline 18 |
| 19 | Automated Tools | Yes | ⚠ [72,73] | Restricted | ✔ BMC Guideline 19 |
| 20 | Cookies | Yes | ⚠ [74] | No | ✔ BMC Guideline 20 |

⚠ PECSV: Possible Exposure to Cybersecurity Vulnerabilities. ✔ PRCSV: Possible Resilience to Cybersecurity Vulnerabilities. N/A: Not Applicable.

**Table 2.** Comparison of System Characteristics.

| Seq. | System Characteristics | Conventional Systems | PECSV | BMC Systems | PRCSV |
|---|---|---|---|---|---|
| 1 | OS/Kernel/Embedded | Yes | ⚠ [75] | No | ✔ BMC Characteristic 1 |
| 2 | Applications | Environment-centric | | Domain-specific | ✔ BMC Characteristic 2 |
| 3 | Programming Languages | Many | | ASM/C/C++ | BMC Characteristic 3 |
| 4 | Executable | Format varies depending on OS | | Single monolithic executable | ✔ BMC Characteristic 4 |
| 5 | Linking | Dynamic (DLLs) | ⚠ [76] | Static (No DLLs) | ✔ BMC Characteristic 5 |
| 6 | Loading | Dynamic | | Static | ✔ BMC Characteristic 6 |
| 7 | Multi-tasking | Yes | | Yes (Only within a domain-specific application) | ✔ BMC Characteristic 7 |
| 8 | System Calls/API | Yes | ⚠ [77] | No (Direct Hardware Interfaces) | ✔ BMC Characteristic 8 |
| 9 | Sockets | Yes | ⚠ [78] | No | ✔ BMC Characteristic 9 |
| 10 | Open Ports | Yes | ⚠ [79] | No | ✔ BMC Characteristic 10 |
| 11 | During execution | A given application, OS, and other applications | | Only a given domain-specific application-suite | BMC Characteristic 11 |
| 12 | Event/Interrupt driven | Both | | Event-driven | ✔ BMC Characteristic 12 |
| 13 | Shared memory/Message Passing | Both | | Shared Memory (Single Address Space) | ✔ BMC Characteristic 13 |
| 14 | Concurrency control | Semaphores and other | | Uses circular lists as buffers, avoids concurrency controls | ✔ BMC Characteristic 14 |
| 15 | I/O | Interrupt driven | ⚠ [80] | Direct Hardware Interfaces | ✔ BMC Characteristic 15 |
| 16 | Third-Party Software | Yes | ⚠ [81,82] | No | ✔ BMC Characteristic 16 |
| 17 | Network interfaces | Many interactions during the session between a client and a server | ⚠ [83,84] | Only a few interactions (Short data sessions) | ✔ BMC Characteristic 17 |
| 18 | Communication on the Internet | Global Internet | ⚠ [85] | Bare Internet | ✔ BMC Characteristic 18 |
| 19 | Downloads on the Internet | Yes | ⚠ [86–89] | No | ✔ BMC Characteristic 19 |
| 20 | IoTs | Small OS, Embedded Nodes | ⚠ [90] | Must be bare nodes | ✔ BMC Characteristic 20 |

**Table 2.** *Cont.*

| Seq. | System Characteristics | Conventional Systems | PECSV | BMC Systems | PRCSV |
|---|---|---|---|---|---|
| 21 | Computing Device | Valuable Resources (Storage, OS, Other) | ⚠ Obvious | No valuable resources (Bare) | ✔ BMC Characteristic 21 |
| 22 | Application-program Control | No, OS controls it | | Yes, Domain-specific application suite controls the control flow as designed | ✔ BMC Characteristic 22 |
| 23 | Users | Global, All | ⚠ [2] | Only Bare Users, Authorized | ✔ BMC Characteristic 23 |
| 24 | Messages | May not have valid authentication | ⚠ [2] | Each message contains encrypted bare user authentication | ✔ BMC Characteristic 24 |
| 25 | User-Secured USBs | N/A | | Uses two USBs, boots, and an encrypted application suite | ✔ BMC Characteristic 25 |
| 26 | BIOS/Firmware | Not bundled with OS | ⚠ [91] | Bundled with the domain-specific application suite | ✔ BMC Characteristic 26 |
| 27 | Protocols | Layered protocols | ⚠ [92] | Integrated with the application-suite | ✔ BMC Characteristic 27 |
| 28 | Passwords | Password files part of OS structures | ⚠ [13,93] | No password files (authentication stored in program structures) | ✔ BMC Characteristic 28 |

⚠ PECSV: Possible Exposure to Cybersecurity Vulnerabilities. ✔ PRCSV: Possible Resilience to Cybersecurity Vulnerabilities. N/A: Not Applicable.

Using these two tables, we make the following observations. In Table 1, in conventional system guidelines, 15 out of 20 have possible exposure to cybersecurity vulnerabilities, whereas in BMC systems, 17 out of 20 have possible resilience to cybersecurity vulnerabilities. Similarly, in Table 2, in conventional system characteristics, 17 out of 28 have possible exposure to cybersecurity vulnerabilities, whereas in BMC systems, 26 out of 28 have possible resilience to cybersecurity vulnerabilities. Overall, BMC systems show possible resiliency, and conventional systems indicate possible exposure to cybersecurity vulnerabilities. This indicates that BMC systems are inherently secure by design.

## 5. Cyberattacks and Analysis

An attack can have many stages, such as reconnaissance, weaponization, delivery, exploitation, installation, command, and control (C2), and actions on objectives [94]. An attacker may select the stages in their attack methodology based on suitability for their execution plan. Each cyberattack can be traced to one or more root causes in an information system: hardware, software, firmware, OS, protocols, layers, complexity, programming language, humans, and more. Most of the literature on cybersecurity indicate that after an attack, it is typical to focus on the symptoms of an attack and fix the problem as needed. Many times, the roots of an attack may still exist and manifest as a new attack. This process goes on as a "cat and mouse" game in a never-ending cycle. It is essential to find and remove the root causes of cyberattacks to eliminate them. In current information systems, cyberattack roots are inherently hidden in the architecture, protocols, policies, procedures, objectives, requirements, design, and implementation of these systems. Thus, it is harder to remove them as it assumes global markets and users. The cyberattack space is infinitely large, as shown in Figure 1, and it is difficult to comprehend and address all possibilities for a given cyberattack. Eliminating the roots of cyberattacks may require revolutionary and non-evolutionary approaches, which are not the focus of current cybersecurity research.

### 5.1. Overview of Selected Cyberattacks

In this section, we analyzed the chosen 22 cyberattacks. The related work and the definitions of this cyberattack are provided in Section 2.1. We identified each attack's preventive mechanisms and some root causes using the existing literature. These root causes will help to compare the security strengths and weaknesses of conventional and BMC approaches.

#### 5.1.1. Buffer Overflow

The following root causes and preventive methods are derived from [8,95].
Root Causes:

1. Lack of bounds checking in functions.
2. Lack of input validation and sanitization by programmers.

Preventive Mechanisms:

1. Consistently update OSs, programming languages, and compilers to their latest versions.
2. Implement code protection mechanisms.
3. Use safe functions; validate and sanitize input data.
4. Employ static code analysis tools.
5. Provide consistent training and reviews.

#### 5.1.2. Phishing Attack

The following root causes and preventive methods are derived from [9,96].
Root Causes:

1. Clicking links can trigger malware downloads or redirect to malicious websites.
2. Downloading files can contain malicious code.
3. Running downloaded code can execute malware and cause attacks.
4. Messages from unauthenticated users may trigger attacks.
5. Email addresses are easily obtained, aiding targeted attacks.
6. Email attachments are a common phishing attack vector.
7. Website phishing can mimic legitimate sites for credential theft.

Preventive Mechanisms:

1. Check website URLs.
2. Scrutinize website design.
3. Beware of urgency-based tactics.
4. Enable two-factor authentication (2FA).
5. Report suspicious activity.
6. Utilize access control lists (ACLs).
7. Employ email filtering.
8. Use machine learning-based detection.
9. Maintain regular backups.
10. Prioritize strong passwords.
11. Educate employees on cybersecurity awareness.
12. Develop incident response plans.
13. Provide consistent training and reviews.

#### 5.1.3. Ransomware

The following root causes and preventive methods are derived from [10,77,91,97,98].
Root Causes:

1. Downloading email attachments.
2. Downloading files from untrusted websites.
3. Allowing downloaded code to run automatically.
4. Attacker accessing OS APIs.

5. Attacker exploiting hardware vulnerabilities (TPM: Trusted Platform Model, BIOS, firmware).
6. Freely available educational resources on cybersecurity vulnerabilities and attack techniques.

Preventive Mechanisms:

1. Deploy IDS (Intrusion Detection System) and IPS (Intrusion Prevention System).
2. Utilize TPM. (Note: it was also found that TPM has security vulnerabilities).
3. Maintain regular backups.
4. Implement blacklist-based detection: block known malicious domains and IP addresses.
5. Employ advanced ransomware detection: Use rule-based, statistical, machine learning-based, or hybrid techniques.
6. Change the file extensions randomly.
7. Provide consistent training and reviews.

#### 5.1.4. Denial of Service (DoS) and Distributed Denial of Service (DDoS)

The following root causes and preventive methods are derived from [11,99].
Root Causes:

1. Allowing attackers to use a legitimate machine and infect it.
2. Allowing attackers to flood requests.
3. Freely available educational resources on cybersecurity vulnerabilities and attack techniques.

Preventive Mechanisms:

1. Information entropy.
2. Machine learning-based methods.
3. Artificial neural networks (ANN).
4. Statistical analysis.
5. Flow statistics.
6. Rate Limiting.
7. TCP Proxies.
8. Provide consistent training and reviews.

#### 5.1.5. Man-in-the-Middle (MITM)

The following root causes and preventive methods are derived from [12,78,100].
Root Causes:

1. Attacker using public Wi-Fi access points.
2. Using a secure socket layer provided by the OS.
3. Exploiting protocol vulnerabilities in ARP, DHCP, DNS, ICMP, and IP.
4. Using open-source automation tools.
5. Using open-source OSs.
6. Lack of proper authentication measures to validate users.
7. Freely available educational resources on cybersecurity vulnerabilities and attack techniques.

Preventive Mechanisms:

1. Enable two-factor authentication (2FA).
2. ARP Spoofing Detection: cryptographic, voting-based, hardware, server-based, host-based solutions.
3. DNS Spoofing Detection: entropy-based, cryptographic, artificial neural network (ANN) solutions.
4. IP Spoofing Defense: router-based, host-based, hybrid solutions.
5. SSL/TLS Solutions: detecting forged certificates, certificate pinning, multipath probing, forcing SSL/TLS connections, friendly MITM, TLS extensions.
6. Provide consistent training and reviews.

#### 5.1.6. Password Attack

The following root causes and preventive methods are derived from [13,101].
Root Causes:

1. Exploiting OS vulnerabilities.
2. Attacker modifying the number of password entry limits.
3. Attacker accessing password files.
4. Attacker accessing OS APIs.
5. Attacker accessing system calls.
6. Weak or predictable passwords.

Preventive Mechanisms:

1. Conduct penetration testing.
2. Enable two-factor authentication (2FA).
3. Enforce and manage strong password policies.
4. Monitor activity for suspicious behavior.
5. Employ a layered defense for a strong security posture.
6. Limit attempts to enter a correct password.
7. Change passwords frequently.
8. Avoid using the same password for multiple accounts.
9. Avoid passwords that are vulnerable to dictionary attacks.
10. Use a password manager.
11. Do not write down passwords.
12. Consider password-less authentication techniques.
13. Provide consistent training and reviews.

#### 5.1.7. Trojan Horse

The following root causes and preventive methods are derived from [14,102,103].
Root Causes:

1. Users download either files or software.
2. Running the downloaded code automatically.
3. Exploiting OS vulnerabilities.
4. Attacker accessing system calls.
5. Attacker accessing OS APIs.
6. An attacker using auto-run-in script files.

Preventive Mechanisms:

1. Avoid opening suspicious emails.
2. Download software only from verified publishers.
3. Scan URLs before clicking.
4. Use antivirus software.
5. Deploy honeypots.
6. Provide consistent training and reviews.

#### 5.1.8. Virus

The following root causes and preventive methods are derived from [15,104,105].
Root Causes:

1. User downloading email attachments.
2. User downloading software.
3. Users accessing fake websites.
4. User using an infected USB or other mass storage device.
5. Allowing script files in emails.
6. Attacker using batch files.
7. Exploiting OS vulnerabilities.

Preventive Mechanisms:

1. Antivirus software.
2. Firewalls.
3. Keeping OSs up to date.
4. Regularly backing up digital records.
5. Scanning systems and USBs.
6. Provide consistent training and reviews.

#### 5.1.9. Worms

The following root causes and preventive methods are derived from [16,106].
Root Causes:

1. Downloading software.
2. Downloading email attachments.
3. Accessing fake websites.
4. Using an infected USB or other mass storage device.
5. Allowing script files in emails.
6. Attackers using batch files.
7. Exploiting OS vulnerabilities.

Preventive Mechanisms:

1. Antivirus software.
2. Firewalls.
3. Keeping OSs up to date.
4. Frequent backups of digital records.
5. Scanning systems and USBs for malware.
6. Provide consistent training and reviews.

#### 5.1.10. Spyware

The following root causes and preventive methods are derived from [17,107].
Root Causes:

1. Downloading software.
2. Downloading email attachments.
3. Accessing fake websites.
4. Using an infected USB or other mass storage device.
5. Allowing script files in emails.
6. Attackers using batch files.
7. Exploiting OS vulnerabilities.

Preventive Mechanisms:

1. Spyware detection algorithms.
2. Antivirus software.
3. Firewalls.
4. Keeping OSs up to date.
5. Regularly backing up digital records.
6. Scanning systems and USBs for malware.
7. Provide consistent training and reviews.

#### 5.1.11. Adware

The following root causes and preventive methods are derived from [18,108].
Root Causes:

1. Marketing.

Preventive Mechanisms:

1. Uninstall adware.
2. Reset web browser settings.
3. Delete web browser caches and cookies.
4. Use antivirus software.
5. Provide consistent training and reviews.

#### 5.1.12. Rootkit

The following root causes and preventive methods are derived from [19,109].
Root Causes:

1. Installing software online.
2. Downloading software.
3. Exploiting OS vulnerabilities.
4. Accessing fake websites.
5. Using an infected USB or other mass storage device.
6. Using devices with infected firmware.
7. Firmware vulnerabilities.
8. Freely available educational resources on cybersecurity vulnerabilities and attack techniques.

Preventive Mechanisms:

1. Antivirus software.
2. Firewalls.
3. Rootkit scanners.
4. Avoiding phishing scams.
5. Keeping OSs up to date.
6. Provide consistent training and reviews.

#### 5.1.13. Botnet

The following root causes and preventive methods are derived from [20,110].
Root Causes:

1. Software vulnerabilities.
2. Downloading software.
3. Using an infected USB or other mass storage device.
4. Open ports.

Preventive Mechanisms:

1. Using an IDS (Intrusion Detection System) and an IPS (Intrusion Prevention System).
2. Firewalls.
3. Enforce and manage strong password policies.
4. Access Control Lists (ACLs).
5. AI and automation tools for security.
6. Using bot managers.
7. Enable two-factor authentication (2FA).
8. Provide consistent training and reviews.

#### 5.1.14. Data Breach

The following root causes and preventive methods are derived from [21,111].
Root Causes:

1. User mistakes or negligence.
2. Malicious insiders.
3. Lack of physical security.
4. Downloading software.

Preventive Mechanisms:

1. Regularly back up digital records.
2. Cryptography.
3. Identity and access management (IAM), such as strong password policies.
4. Incident Response Plan (IRP).
5. Enable two-factor authentication.
6. AI and automation tools for security.
7. Provide consistent training and reviews.

#### 5.1.15. Advanced Persistent Threats (APT)

The following root causes and preventive methods are derived from [22,112].
Root Causes:

1. Downloading software.
2. Clicking on email attachments.
3. Using an infected USB or other mass storage device.
4. Accessing fake websites.
5. Exploiting OS vulnerabilities.
6. Freely available educational resources on cybersecurity vulnerabilities and attack techniques.

Preventive Mechanisms:

1. Access Control Lists (ACLs).
2. Controlling external media use.
3. Protecting valuable data.
4. Managing endpoint security.
5. Implementing Network Access Control (NAC).
6. Blocking high-risk applications.
7. Blocking known malware servers.
8. Analyzing security breaches for prevention.
9. Network and host hardening.
10. Provide consistent training and reviews.

#### 5.1.16. SQL Injection

The following root causes and preventive methods are derived from [23,113].
Root Causes:

1. System privileges are granted to the DBMS.
2. DBMS bypassing OS controls.
3. Failure to validate and authenticate user-entered data.

Preventive Mechanisms:

1. Input validation and sanitization.
2. Using prepared statements.
3. Firewalls.
4. Controlling database permissions.
5. Scanning code for SQL vulnerabilities.
6. Using a secure ORM framework.
7. Using properly constructed stored procedures.
8. Applying the principle of least privilege to database accounts.
9. Prohibiting default root or admin access to applications.
10. Changing DBMS accounts from defaults to something else.
11. Provide consistent training and reviews.

#### 5.1.17. Supply Chain

The following root causes and preventive methods are derived from [24,114].
Root Causes:

1. Providing backdoors in software and hardware.
2. Exploiting OS vulnerabilities.
3. Open-source code.
4. Downloading software.
5. Using an infected USB or other mass storage device.
6. Users accessing fake websites.

Preventive Mechanisms:

1. Implement honey tokens.
2. Secure privileged access management (PAM).
3. Implement a Zero Trust Architecture (ZTA).
4. Identify potential insider threats.
5. Identify and protect vulnerable resources.
6. Minimize access to sensitive data.
7. Implement strict shadow IT rules.
8. Conduct regular third-party risk assessments.
9. Monitor vendor networks for vulnerabilities.
10. Identify all third-party data leaks.
11. Disable backdoors.
12. Provide consistent training and reviews.

#### 5.1.18. URL Interpretation

The following root causes and preventive methods are derived from [25,115].
Root Causes:

1. Attacker accessing private files through a URL link.

Preventive Mechanisms:

1. Input validation and sanitization.
2. URL encoding to prevent malicious characters.
3. Avoid using user input directly in code.
4. Implement strict access permissions.
5. Enable two-factor authentication (2FA).
6. Provide consistent training and reviews.

#### 5.1.19. Insider Threats

The following root causes and preventive methods are derived from [26,116].
Root Causes:

1. Failure to apply strong security policies to private data.
2. User mistakes or negligence.
3. Freely available educational resources on cybersecurity vulnerabilities and attack techniques.

Preventive Mechanisms:

1. Implement proper access management.
2. Employ user behavior analytics to access private data.
3. Use offensive security measures.
4. Provide consistent training and reviews.

#### 5.1.20. Eavesdropping

The following root causes and preventive methods are derived from [27,117].
Root Causes:

1. Freely available online automation tools.
2. Open Wi-Fi access at public places.

Preventive Mechanisms:

1. Cryptography.
2. Provide consistent training and reviews.

#### 5.1.21. Cookies

The following root causes and preventive methods are derived from [28,118].
Root Causes:

1. Marketing tools and techniques online.
2. Attackers intruding into machines.
3. Exploiting vulnerable protocols to steal cookies.

Preventive Mechanisms:

1. Do not enable cookies and disable options in browser settings.
2. Provide consistent training and reviews.

#### 5.1.22. Social Engineering

The following root causes and preventive methods are derived from [29,119].
Root Causes:

1. Marketing tools and techniques online.
2. Attackers are intruding into machines.
3. Social platforms and networks.

Preventive Mechanisms:

1. Do not click on malicious links.
2. Do not download malicious software.
3. Do not enable cookies and disable options in browser settings.
4. Do not engage in conversations with unknown users.
5. Provide consistent training and reviews.

### *5.2. Description and Analysis of Selected Cyberattacks*

This section analyzes the data collected on the 22 selected cyberattacks described in Section 5.1. Each cyberattack has root causes and preventive mechanisms, as shown in the cybersecurity literature review.

#### 5.2.1. Root Causes for the 22 Cyberattacks

This section provides all elements of analysis and evaluation of BMC for cybersecurity. The 22 cyberattacks have some common root causes, as described in Section 5.1. All these root causes were collected in a list, and duplicates were eliminated. The new list resulted in the 53 root causes listed in Table 3. As these root causes are derived from the 22 chosen cyberattacks, this list may not be a complete list of all root causes that may exist in the field of cyberattacks. The names of the root causes are slightly modified to make them more readable while preserving the original meaning.

**Table 3.** Root Causes.

| | |
|---|---|
| 1. | Allowing attackers to flood requests. |
| 2. | Allowing script files in emails. |
| 3. | Attacker accessing API. |
| 4. | Attacker accessing password files. |
| 5. | Attacker accessing private files from a URL link. |
| 6. | Attacker accessing system calls. |
| 7. | Attacker intruding into machines. |
| 8. | Attacker can modify the number of password entry limits. |
| 9. | Attacker using auto-run in script files. |
| 10. | Attacker using batch files. |
| 11. | Attacker using cookies. |
| 12. | Attacker using public Wi-Fi access point. |
| 13. | Attacker using website phishing. |
| 14. | DBMS by-passing OS. |
| 15. | Email addresses of users are easy to obtain. |
| 16. | Enabling adware to use browser apps. |
| 17. | Freely available educational resources on cybersecurity vulnerabilities and attack techniques. |
| 18. | Freely available online automation tools. |
| 19. | Attacker can use a legitimate machine and infect. |
| 20. | Hardware vulnerabilities. |
| 21. | Including attachments in emails. |
| 22. | Lack of bounds checking in functions. |
| 23. | Lack of data validation by developers. |
| 24. | Lack of physical security. |
| 25. | Malicious insiders. |
| 26. | Marketing tools and techniques online. |
| 27. | Not validating and authenticating user-entered data. |
| 28. | Open ports. |
| 29. | Open-source automation tools. |
| 30. | Open-source code. |
| 31. | Open-source OSs. |
| 32. | Open Wi-Fi access at public places. |
| 33. | OS vulnerabilities. |
| 34. | Protocol vulnerabilities in ARP, DHCP, DNS, ICMP, and IP. |
| 35. | Providing backdoors in software and hardware. |
| 36. | Receiving messages from unauthenticated users. |
| 37. | Running downloaded code automatically. |
| 38. | Secure sockets layer provided by the OS. |
| 39. | Software vulnerabilities. |
| 40. | Strong security policies are not applying to private data. |
| 41. | System privileges given to DBMS. |
| 42. | There is no proper authentication measure to validate users. |
| 43. | User accessing fake websites. |
| 44. | User clicking an unsolicited link. |
| 45. | User downloading a file from an unidentified website. |
| 46. | User downloading an unsolicited file. |
| 47. | User downloading email attachments. |
| 48. | User downloading software. |
| 49. | User installing software online. |
| 50. | User mistakes or negligence. |
| 51. | User using an infected USB. |
| 52. | User using infected firmware. |
| 53. | Firmware vulnerabilities. |

#### 5.2.2. Preventive Mechanisms for the 22 Selected Cyberattacks

The 22 chosen cyberattacks have some common preventive mechanisms, as described in Section 5.1. All these preventive mechanisms were collected in a list, and duplicates were eliminated. The new list has 97 individual preventive mechanisms, which are listed

in Table 4. As these preventive mechanisms are derived from the 22 chosen cyberattacks, this list may not be a complete list of all preventive mechanisms that may exist in the field of cyberattacks. The names of the preventive mechanisms are modified to make them more readable while preserving the original meaning.

**Table 4.** Preventive Mechanisms.

| |
|---|
| 1. Access control lists (ACL). |
| 2. AI and automation tools. |
| 3. Analyzing security breaches. |
| 4. Anti-virus software. |
| 5. Artificial neural networks. |
| 6. Avoid opening suspicious emails. |
| 7. Avoid using user input directly. |
| 8. Avoiding phishing. |
| 9. Beware of urgency. |
| 10. Blacklist-based. |
| 11. Blocking high-risk applications. |
| 12. Blocking known malware servers. |
| 13. Change file extensions randomly. |
| 14. Change password frequently. |
| 15. Changing DBMS accounts to something else. |
| 16. Check bounds on string functions. |
| 17. Check Website URLs. |
| 18. Provide consistent training and reviews. |
| 19. Controlling database permissions. |
| 20. Controlling external media. |
| 21. Cryptography. |
| 22. Delete web browser caches and cookies. |
| 23. Detection of ARP spoofing. |
| 24. Detection of DNS spoofing. |
| 25. Disable backdoor. |
| 26. Do not click on malicious links. |
| 27. Do not download malicious software. |
| 28. Do not enable cookies and disable options in browser settings. |
| 29. Do not write down passwords. |
| 30. Download software from verified publishers. |
| 31. Email filtering. |
| 32. Enable two-factor authentication. |
| 33. Enforce and manage strong passwords. |
| 34. Firewalls. |
| 35. Flow statistics. |
| 36. Identify all potential insider threats. |
| 37. Identify all third-party data leaks. |
| 38. Identify and protect vulnerable resources. |
| 39. Identity and access management (IAM), such as strong passwords. |
| 40. Implement a Zero Trust Architecture (ZTA). |
| 41. Implement honey tokens. |
| 42. Implement proper access management. |
| 43. Implement strict shadow IT rules. |
| 44. Implementing NAC (Network Access Control). |
| 45. Incident response plan (IRP). |
| 46. Information entropy. |
| 47. Input validation. |
| 48. IP spoofing defense. |
| 49. Keep operating systems up to date. |
| 50. Layered defense for a strong security posture. |

**Table 4.** *Cont.*

| |
|---|
| 51. Limit access permissions. |
| 52. Limit the number of attempts to enter a correct password. |
| 53. Machine learning-based method. |
| 54. Managing endpoint security. |
| 55. Minimize access to sensitive data. |
| 56. Minimizing the privileges that are given to all database accounts. |
| 57. Monitor activity. |
| 58. Monitor vendor networks for vulnerabilities. |
| 59. Network and host hardening. |
| 60. No same password for all accounts. |
| 61. Penetration testing. |
| 62. Perform static code analysis. |
| 63. Plan ahead of security attacks. |
| 64. Prohibiting DBA or admin access to applications. |
| 65. Protect code segment. |
| 66. Protecting valuable data. |
| 67. Ransomware detection techniques. |
| 68. Rate limiting. |
| 69. Regularly backup digital records. |
| 70. Report suspicious activity. |
| 71. Reset web browser settings. |
| 72. Rootkit scanners. |
| 73. Scan URLs. |
| 74. Scanning code for SQL injection vulnerabilities. |
| 75. Scanning system and USBs. |
| 76. Scrutinize website design. |
| 77. Secure privileged access management. |
| 78. Send regular third-party risk assessments. |
| 79. Spyware algorithm detection. |
| 80. SSL/TLS solutions. |
| 81. Statistical analysis. |
| 82. Take password protection seriously. |
| 83. TCP proxies. |
| 84. TPM (Trusted Platform Module). |
| 85. Uninstall adware. |
| 86. URL encoding. |
| 87. Use honeypot. |
| 88. Use the latest OS, programming languages, and compilers. |
| 89. Use offensive security measures. |
| 90. Use password-less authentication. |
| 91. Use password managers. |
| 92. Use user behavior analytics for accessing private data. |
| 93. Using an ORM framework. |
| 94. Using bot manager. |
| 95. Using IDS and IPS. |
| 96. Using prepared statements. |
| 97. Using properly constructed stored procedures. |

### *5.3. Analysis of Cyberattacks*

This section provides a discussion and analysis of 22 cyberattacks, root causes, preventive mechanisms, and an evaluation of the BMC paradigm regarding cybersecurity vulnerabilities.

#### 5.3.1. Root Causes vs. Cyberattacks

There are 22 chosen cyberattacks and 53 identified root causes. Figure 10 shows a graph between the root causes and their corresponding cyberattacks. This figure shows how many times a given root cause appears in the number of cyberattacks.

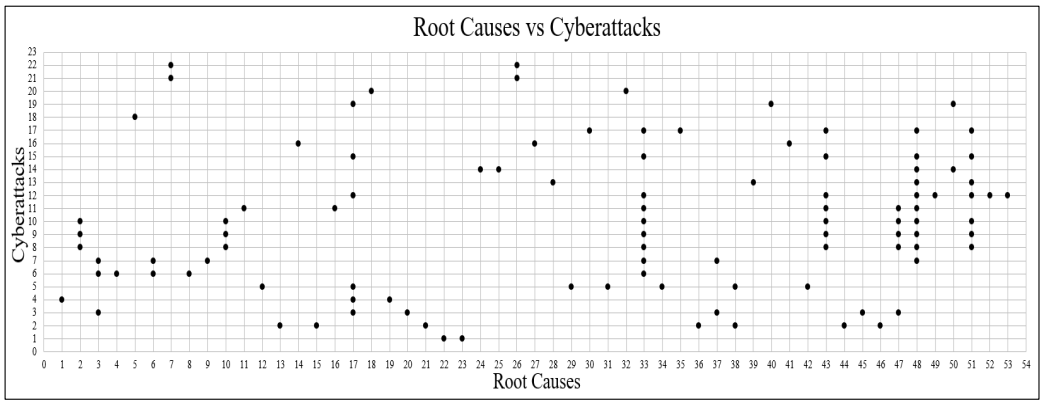


**Figure 10.** Root causes vs. cyberattacks.

Observations: When a root cause occurs in many cyberattacks, it needs more attention to address it. That does not mean less frequency of root causes is not important. The following discussions refer to Figure 10, where more frequent occurrences of root causes are the focus.

OS vulnerabilities (#33) are identified as a root cause of cyberattacks at 9/22, or 40%. These vulnerabilities are a common source of attacks due to their complexity and their role as middleware for all applications. Many OS design features are offered due to convenience and evolution, which may be exploited by an attacker. Also, OS versions and releases change often [2], while users continue to use older versions. Attackers have more OS vulnerability space by mixing old and new version releases.

System calls are part of an OS that is provided to build user applications and does not allow users to access the hardware directly. The OS middleware handles hardware control using a mode bit. An attacker may escalate the mode bit privilege and take over the system. An API is provided for convenience to system programmers and helps to port the code across many platforms. APIs can be used by attackers to exploit OS vulnerabilities. There are many other examples that illustrate OS vulnerabilities at many levels of an OS.

Fake websites (#43) are identified as a root cause of cyberattacks in 7/22, or 31%. Fake websites can be created easily with the existing procedures. These links can be put into emails and lure users to access a fake website. There are many phishing attacks that cause users to access fake websites.

Downloads (#48) are identified as a root cause of cyberattacks in 10/22, or 45%. This indicates that many attacks can be triggered by downloads. Attackers can attach malware to a download file, thus causing a cybersecurity vulnerability. Downloads are essential and used for convenience, which is a vulnerable feature.

Infected USBs (#51) are identified as a root cause of cyberattacks in 7/22, or 31%. It is easy to infect removable devices, and thus it contributes to more vulnerabilities. Once an infected USB is plugged in, it may spread malicious software and make a file system inaccessible or crash a system.

Other root causes can be read from the graph, and it can be observed that their frequency of occurrence in cyberattacks is smaller. Some root causes with smaller frequencies may also be important based on their effectiveness in a cyberattack. All root causes must be addressed and removed from the system to achieve a higher level of security.

Many other root causes, such as downloads, script files, and sockets/open ports, depend upon the underlying OS. This is an indirect involvement of the OS, which is not directly counted in OS vulnerabilities. Otherwise, OS vulnerabilities may be higher than what is shown in this section (40%).

#### 5.3.2. Preventive Mechanisms vs. Cyberattacks

There are 22 chosen cyberattacks and 97 preventive mechanisms. Figure 11 shows a graph between the preventive mechanisms and their corresponding cyberattacks. This figure shows preventive mechanism usage in different types of cyberattacks.

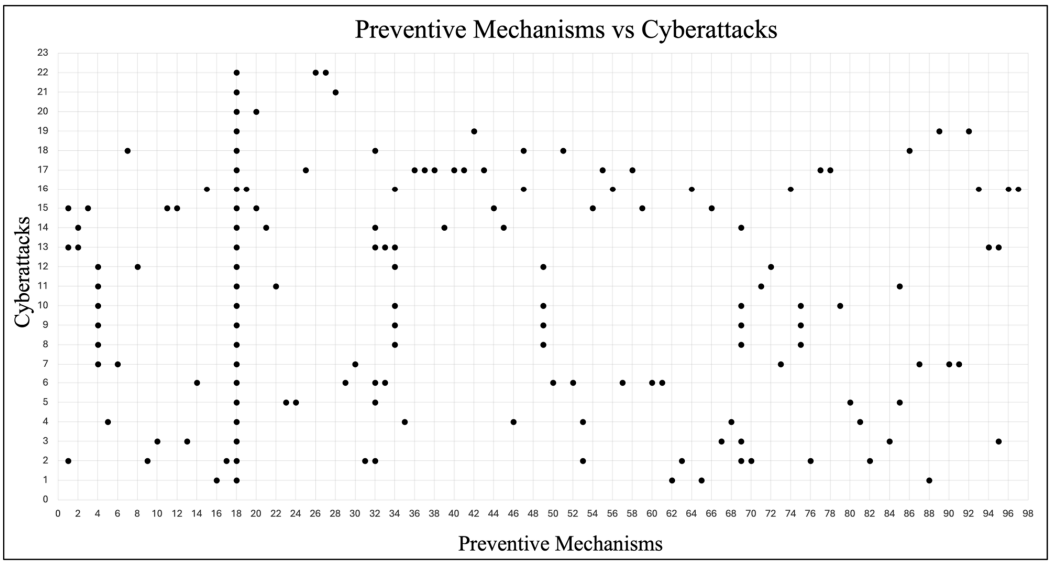


**Figure 11.** Frequency of preventive mechanisms.

Observations: Some observations from this graph are the following: If a preventive mechanism is used in more attacks, it indicates that it is more important to implement it. The preventive mechanism of consistent training and reviews (#18) is used for all cyberattacks. The preventive mechanisms, including antivirus software (#4), two-factor authentication (#32), firewalls (#34), and regularly backing up digital records (#69), have a frequency of occurrence of 6/22 (or 27%). The preventive mechanism "keep OS up to date" (#49) has a frequency of occurrence of 4/22 (or 18%). The preventive mechanism "scanning system and USB" has a frequency of occurrence of 3/22 (or 13.6%). Other preventive measures have a frequency of occurrence of less than 9%. Notice that none of these preventive mechanisms guarantee 100% protection from any cyberattack.

#### 5.3.3. Conventional Root Causes Applicable to BMC Systems

Table 3 shows a list of root causes collected in Section 5.1. As these root causes are derived from conventional systems, we need to validate their applicability to BMC systems. Table 5 illustrates the applicability of the root causes to BMC systems with relevant references and justifications. Figure 12 shows that 8/53 = 15% of root causes are applicable to BMC systems.

**Table 5.** The conventional root causes applicable to BMC systems.

| Seq. | Conventional Root Cause | Applicability to BMC<br>BMC Preventive Mechanism | Reference |
|---|---|---|---|
| 1 | Allowing attackers to flood requests | NO: uses stateless server and lean UDP protocol | [56] |
| 2 | Allowing script files in emails | NO: script files are not allowed in emails | Section 3.5.1<br>Item #14 |
| 3 | Attacker accessing API | NO: Direct Hardware API (HAPI) is not available externally (NO OS) | Section 3.5.2<br>Item #8 |
| 4 | Attacker accessing password files | NO: A single bare machine at a time is used by a single user; no passwords are stored in the bare machine. | Section 3.5.2<br>Item #28 |
| 5 | Attacker accessing private files from a URL link | YES: Uses URL encoding and limits access permissions such as conventional systems. | Table 4<br>Items #51, #86 |
| 6 | Attacker accessing system calls | NO: No system calls are available externally; only HAPI is used by applications | Section 3.5.2<br>Item #8 |
| 7 | Attackers intruding into machines | NO: A machine is bare; while running one user, another user cannot run an application | Section 3.5.2<br>Item #2 |
| 8 | Attacker is able to modify the number of password entry limits | NO: Not Applicable | Section 3.5.2<br>Item #28 |

**Table 5.** *Cont.*

| Seq. | Conventional Root Cause | Applicability to BMC<br>BMC Preventive Mechanism | Reference |
|---|---|---|---|
| 9 | An attacker using auto-run in script files | NO: script files are not allowed | Section 3.5.1<br>Item #14 |
| 10 | Attacker using batch files | NO: A batch file is not allowed; only bare applications run | Section 3.5.1<br>Item #14 |
| 11 | Attacker using cookies | NO: Cookies are not allowed | Section 3.5.1<br>Item #20 |
| 12 | Attacker using public Wi-Fi access point | NO: Wireless is not allowed at this point. | Section 3.5.1<br>Item #13 |
| 13 | Attacker using Website phishing | NO: (1) In emails, website links are not allowed.<br>(2) Only bare-to-bare users communicate on the bare Internet. Although it is applicable but prevented in both of the above two cases. | Section 3.5.1<br>Item #18 |
| 14 | DBMS bypassing OS | NO: Not possible (No OS) | Section 3.5.2<br>Item #1 |
| 15 | Email addresses are easily obtained | NO: Issuing of email addresses for bare users is restricted and physically controlled by a bare email administrator. | Section 3.5.2<br>Item #23 |
| 16 | Enabling Adware to use Browser Apps | NO: No advertisements are allowed. | Section 3.5.1<br>Item #17 |
| 17 | Freely available educational resources on cybersecurity vulnerabilities and attack techniques. | NO: no free education. One of the main reasons for speedy cyberattacks is due to the fastest way to learn attacking techniques freely using the Internet. | Section 3.5.1<br>Item #6 |
| 18 | Freely available online automation tools | NO: A closed system, no freely available automation tools. | Section 3.5.1<br>Item #19 |
| 19 | Allowing attackers to use a legitimate machine and infect | NO: All legitimate machines are bare; nothing to infect them. | Section 3.5.2<br>Item #21 |
| 20 | Hardware vulnerabilities | NO: Cannot get to hardware, as HAPI is not available outside running applications. As the machine is bare, physical attacks can only damage the machine. | Section 3.5.2<br>Item #21 |
| 21 | Including attachments in emails | NO: Attachments are not allowed. | Section 3.5.1<br>Item #15 |
| 22 | Lack of bounds checking in functions | NO: All string functions are strictly enforced for bound checking. | Table 4<br>Item #16 |
| 23 | Lack of data validation by programmers | YES: No exposure of attacks; all data entered by users is checked for valid data and data types. Strict data checking is conducted at a programming level. All bare code is developed as one homogeneous entity; there is no third-party software, no external models. | Table 4<br>Item #47 |
| 24 | Lack of physical security | YES: Physical security is required when a user is running an application suite on a bare machine (this is a mandatory requirement in the BMC paradigm). Otherwise, there is no need for physical security because the machine is bare. | Section 3.5.2<br>Item #21 |
| 25 | Malicious insiders | YES: All insiders must be properly authorized and trusted; otherwise, there is no security for any system. | Section 3.5.2<br>Item #23 |
| 26 | Marketing tools and techniques online | NO: Marketing tools and techniques are not online. | Section 3.5.1<br>Item #17 |
| 27 | Not validating and authenticating user entered data | YES: User entered data must be validated and authenticated. | Table 4<br>Item #47 |
| 28 | Open ports | NO: There are no open ports. | Section 3.5.2<br>Item #10 |
| 29 | Open-source automation tools | NO: There are no open-source automation tools. | Section 3.5.1<br>Item #19 |
| 30 | Open-source code | NO: Not allowed. | Section 3.5.1<br>Item #2 |
| 31 | Open-source OSs | NO: No OS. | Section 3.5.2<br>Item #1 |
| 32 | Open Wi-Fi access at public places | NO: Wi-Fi is not allowed at this point. | Section 3.5.1<br>Item #13 |

**Table 5.** *Cont.*

| Seq. | Conventional Root Cause | Applicability to BMC<br>BMC Preventive Mechanism | Reference |
|---|---|---|---|
| 33 | OS vulnerabilities | NO: No OS. | Section 3.5.2 Item #1 |
| 34 | Protocol vulnerabilities in ARP, DHCP, DNS, ICMP, and IP | NO: We limit our protocols to Ethernet and IP. The other protocols are not available outside the application running on the machine. Furthermore, there are no protocol layers. These protocols are implemented within the bare application. The attacker does not have the path to invoke these protocols. Although IP spoofing can be conducted, the attacker does not have bare user authentication. Bare user authentication is used in every packet, encrypted and validated for each message transmission. | Section 3.5.2 Item #27 |
| 35 | Providing backdoors in software and hardware | NO: Do not allow any backdoors. | Section 3.5.1 Item #2 |
| 36 | Receiving messages from unauthenticated users | NO: Only communicate with the authentication users. | Section 3.5.2 Item #23 |
| 37 | Running a downloaded code automatically | NO: There is no downloading code, although there is no dynamic linking and loading to run a code. The bare code is statically compiled. | Section 3.5.2 Item #5, #6 |
| 38 | Secure socket layer provided by OS | NO: No socket concept. | Section 3.5.2 Item #9 |
| 39 | Software vulnerabilities | YES: Buffer overflow was discussed in items 22 and 23. There could be possible programming errors; however, the intruder has no access to modify or exploit injecting new code due to static binding. There are no deserialization issues as intruder has no access to the flow control of application. Overall, any software vulnerabilities cannot cause any harm as the application code is statically bound. | Section 3.5.2 Item #5, #6 |
| 40 | Strong security policies not applying to private data | YES: Only if the attacker is an insider. | |
| 41 | System privileges given to DBMS | NO: No OS. DBMS is part of an application suite. | Section 3.5.2 Item #1 |
| 42 | There is no proper authentication measure to validate users | NO: All users are properly authorized and authenticated. | Section 3.5.2 Item #23 |
| 43 | User accessing fake websites | NO: Not applicable (Only access legitimate and authenticated bare websites) | Section 3.5.1 Item #18 |
| 44 | User clicking an unsolicited link | NO: No downloads in emails; no website links in emails (only access legitimate and authenticated bare websites using the Bare Internet) | Section 3.5.2 Item #19 |
| 45 | User downloading a file from an unidentified website | NO: No online downloading (only access legitimate and authenticated bare websites using the Bare Internet) | Section 3.5.2 Item #19 |
| 46 | User downloading an unsolicited file | NO: No online downloading (only access legitimate and authenticated bare websites using the Bare Internet) | Section 3.5.1 Item #18 |
| 47 | User downloading email attachments | NO: No attachments in emails. | Section 3.5.1 Item #15 |
| 48 | User downloading software | NO: No online downloading for software (Use CDs/USBs from authenticated bare application providers) | Section 3.5.2 Item #19 |
| 49 | User installing software online | NO: No installation software online. | Section 3.5.1 Item #12 |
| 50 | User mistakes or negligence | YES: All bare users must be properly trained to handle sensitive data. | Section 3.5.1 Item #11 |
| 51 | User using an infected USB | NO: Dual USBs (one for booting and second for application). Both USBs must be physically secure. | Section 3.5.2 Item #25 |
| 52 | User using infected firmware | NO: Bundled with a domain-specific application suite | Section 3.5.2 Item #26 |
| 53 | Firmware vulnerabilities | NO: Bundled with a domain-specific application suite | Section 3.5.2 Item #26 |

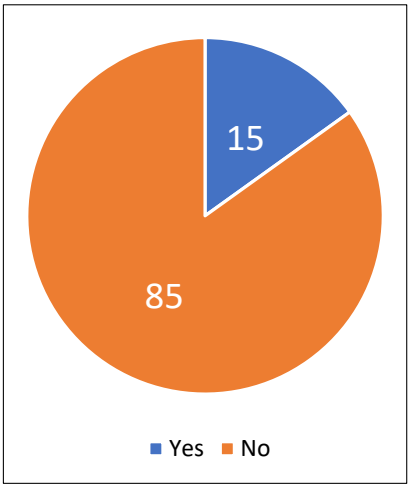


**Figure 12.** Root causes applicability to BMC.

For BMC systems, using Table 1, system guidelines, 17 out of 20 are resilient to cybersecurity vulnerabilities. Similarly, using Table 2, system characteristics: 26 out of 28 are resilient to cybersecurity vulnerabilities. This makes it likely that BMC systems are resilient to cyberattacks by design.

#### 5.3.4. Conventional Preventive Mechanisms Applicable to BMC Systems

The 97 preventive mechanisms, listed in Table 4, are studied in this section. The applicability of these preventive mechanisms to BMC systems is listed in Table 6, with some justifications. As shown in Figure 13, about 63% of these preventive mechanisms are applicable to BMC systems, as these mechanisms are very general in nature.

**Table 6.** The conventional preventive mechanisms applicable to the BMC system.

| Seq. | Conventional Preventive Mechanisms | Applicability to BMC |
|---|---|---|
| 1 | Access Control Lists (ACL) | YES |
| 2 | AI and automation tools | YES |
| 3 | Analyzing security breaches | YES |
| 4 | Anti-Virus software | NO: No OS. |
| 5 | Artificial neural networks | YES |
| 6 | Avoid opening suspicious emails | YES |
| 7 | Avoid using user input directly | YES |
| 8 | Avoiding phishing | YES |
| 9 | Beware of urgency | NO: Not applicable |
| 10 | Blacklist-Based | NO: Not applicable, not needed since only communicates with trusted bare users. |
| 11 | Blocking high-risk applications | NO: only runs intended domain-specific application suite |
| 12 | Blocking known malware servers | NO: No OS; malware cannot be downloaded and it cannot communicate with domain-specific application suite. |
| 13 | Change file extensions randomly | NO: Not applicable. Attackers Do not have access to file Hardware API (HAPI) |
| 14 | Change password frequently | YES |
| 15 | Changing DBMS accounts to something else | NO: Not applicable. No OS, database application is part of bare domain-specific application suite. |
| 16 | Check bounds on string functions | YES. Uses only functions with bounds checking. |
| 17 | Check Website URLs | YES |
| 18 | Consistent trainings and reviews | YES |
| 19 | Controlling database permissions | YES |
| 20 | Controlling external media | YES |

**Table 6.** *Cont.*

| Seq. | Conventional Preventive Mechanisms | Applicability to BMC |
|---|---|---|
| 21 | Cryptography | YES |
| 22 | Delete web browser caches and cookies | NO: Not applicable. Cookies are not allowed. |
| 23 | Detection of ARP Spoofing | YES |
| 24 | Detection of DNS Spoofing | YES |
| 25 | Disable backdoor | YES |
| 26 | Do not click on malicious links | YES |
| 27 | Do not download malicious software | NO: Not applicable. Downloads are not allowed. |
| 28 | Do not enable cookies and disable options in browser settings | NO: cookies are not allowed. |
| 29 | Do not write down passwords | YES |
| 30 | Download software from verified publishers | NO: Not applicable. |
| 31 | Email filtering | YES |
| 32 | Enable two-factor authentication | NO: Not applicable. |
| 33 | Enforce and manage strong passwords | YES |
| 34 | Firewalls | NO: No OS. |
| 35 | Flow statistics | YES |
| 36 | Identify all potential insider threats | YES |
| 37 | Identify all third-party data leaks | YES |
| 38 | Identify and protect vulnerable resources | YES |
| 39 | Identity and access management (IAM), such as strong passwords | YES |
| 40 | Implement a Zero Trust Architecture (ZTA) | NO: By default, the BMC system is built based on the Zero Trust concept. |
| 41 | Implement Honey tokens | NO: No OS. |
| 42 | Implement proper access management | YES |
| 43 | Implement strict shadow IT rules | NO: Not applicable. |
| 44 | Implementing NAC (Network Access Control) | YES |
| 45 | Incident Response Plan (IRP) | NO: Not applicable. |
| 46 | Information entropy | YES |
| 47 | Input validation | YES: Checks for data type and size. |
| 48 | IP Spoofing Defense | YES |
| 49 | Keep operating systems up to date | NO: Not applicable since there is no OS. |
| 50 | Layered defense for a strong security posture | YES |
| 51 | Limit access permissions | YES |
| 52 | Limit the number of attempts to enter the correct password | YES |
| 53 | Machine Learning-Based Method | YES |
| 54 | Managing endpoint security | YES |
| 55 | Minimize access to sensitive data | YES |
| 56 | Minimizing the privileges that are given to all database accounts | NO: Not applicable; no OS, database application is part of bare domain-specific application suite. |
| 57 | Monitor activity | YES |
| 58 | Monitor vendor networks for vulnerabilities | YES |
| 59 | Network and host hardening | YES |
| 60 | No same password for all accounts | YES |
| 61 | Penetration testing | YES: Applicable to applications only, as there is no OS |
| 62 | Perform static code analysis | YES |
| 63 | Plan ahead of security attacks | YES |
| 64 | Prohibiting DBA or Admin access to applications | NO: Not applicable |
| 65 | Protect code segment | YES |
| 66 | Protecting valuable data | YES |
| 67 | Ransomware Detection Techniques | NO: Not applicable; no OS. |
| 68 | Rate Limiting | YES |
| 69 | Regularly backup digital records | YES |
| 70 | Report suspicious activity | YES |

**Table 6.** *Cont.*

| Seq. | Conventional Preventive Mechanisms | Applicability to BMC |
|---|---|---|
| 71 | Reset web browser settings | NO: Not applicable |
| 72 | Rootkit scanners | NO: Not applicable; no OS. |
| 73 | Scan URLs | YES |
| 74 | Scanning code for SQLI vulnerabilities | YES |
| 75 | Scanning system and USBs | YES |
| 76 | Scrutinize website design | YES |
| 77 | Secure Privileged Access Management | NO: No OS. All user accesses are part of domain-specific suite. |
| 78 | Send regular third-party risk assessments | NO: No third-party software allowed. |
| 79 | Spyware algorithm detection | NO: This algorithm is part of a domain-specific application. |
| 80 | SSL/TLS Solutions | NO: Not sockets. TLS is part of a domain-specific application. |
| 81 | Statistical analysis | YES |
| 82 | Take password protection seriously | YES |
| 83 | TCP Proxies | NO: Proxies are not allowed. |
| 84 | TPM (Trusted Platform Module) | NO: The BIOS is bundled with application suites. |
| 85 | Uninstall adware | NO: Not applicable. Adware is not allowed. |
| 86 | URL encoding | YES |
| 87 | Use Honeypot | NO: Not applicable; no OS. |
| 88 | Use the latest OS, Programming languages, and Compilers | NO: Not applicable; no OS. Need to use the latest programming languages and compilers. |
| 89 | Use offensive security measures | YES |
| 90 | Use Password-less authentication | YES |
| 91 | Use password manager | NO: Not applicable; no OS. |
| 92 | Use user behavior analytics for accessing private data | YES |
| 93 | Using an ORM Framework | NO: Not applicable |
| 94 | Using Bot Manager | NO: No OS |
| 95 | Using IDS and IPS | NO: No OS |
| 96 | Using Prepared Statements | YES |
| 97 | Using properly constructed stored procedures | YES |

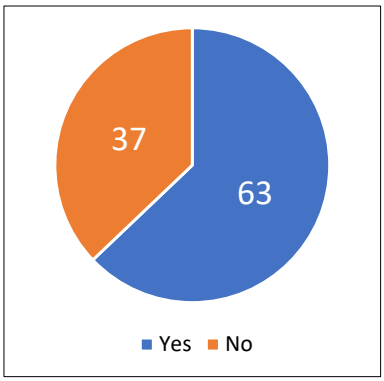


**Figure 13.** Preventive mechanisms applicability to BMC.

The remaining preventive mechanisms are not applicable, as most of them are related to the OS. Tables 1 and 2 illustrate the resiliency to cybersecurity vulnerabilities by design in BMC systems. Thus, additional preventive methods do not apply.

#### 5.3.5. The Cyberattacks vs. the BMC Paradigm

Table 7 provides a final evaluation of the security strengths of the BMC paradigm and its applications using BMC system guidelines, characteristics, and conventional preventive measures as discussed before. Using the BMC approach, out of 22 cyberattacks studied, only two of them do not have protection. These two cyberattacks are related to physical security and the trust of insiders, which are applicable to all systems.

**Table 7.** Evaluation of Bare Machine Computing (BMC) for cyberattacks.

| # | Type of Attack | Root Cause | BMC (Guidelines, Characteristics, and Preventive Mechanisms) | Prevents Root Cause (Yes/No) | Prevents Attack (Yes/No) |
|---|---|---|---|---|---|
| 1 | Buffer Overflow | 1. Lack of bounds checking in functions | Uses only functions with bounds checking. Table 4, Item 16 | Yes | Yes |
| | | 2. Lack of input validation and sanitization by programmers | Checks for data type and size. Table 4, Item 47 | Yes | |
| 2 | Phishing | 1. User clicking an unsolicited link | No Attachments and Links in emails BMC Guidelines (15) | Yes | Yes |
| | | 2. User downloading an unsolicited file | No downloads are allowed. BMC Characteristics (19) | Yes | |
| | | 3. Running a downloaded code | | | |
| | | 4. Receiving messages from unauthenticated users | Each message contains encrypted bare user authentication, which is given in person. BMC Characteristics (24) | Yes | |
| | | 5. Email addresses of users are easy to obtain | Only bare users can communicate with each other. All bare users must be physically authorized and authenticated. BMC Characteristics (23) | Yes | |
| | | 6. Including attachments in emails | No Attachments and Links in emails BMC Guidelines (15) | Yes | |
| | | 7. Attacker using Website phishing | Only bare users can communicate with each other. All bare users must be physically authorized and authenticated. BMC Characteristics (23) | | |
| 3 | Ransomware | 1. User downloading email attachments | No downloads are allowed. BMC Characteristics (19) | Yes | Yes |
| | | 2. User downloading a file from an unidentified Website | | | |
| | | 3. Running a downloaded code automatically | | | |
| | | 4. Attacker accessing OS API | There are no system calls or APIs available to the outside world (outside an application suite). BMC Characteristics (8) | Yes | |
| | | 5. Attacker exploiting hardware vulnerabilities | A computing device (PC, Laptop, Smartphone, Server, Client, etc.) is bare BMC Characteristics (21) | Yes | |
| | | 6. Freely available educational resources on cybersecurity vulnerabilities and attack techniques | Education and Knowledge: Restricted to authorized bare users. BMC Guidelines (6) | Yes | |

**Table 7.** *Cont.*

| # | Type of Attack | Root Cause | BMC (Guidelines, Characteristics, and Preventive Mechanisms) | Prevents Root Cause (Yes/No) | Prevents Attack (Yes/No) |
|---|---|---|---|---|---|
| 4 | DOS & DDOS | 1. Allowing attackers to use a legitimate machine and infect | A computing device (PC, Laptop, Smartphone, Server, Client, etc.) is bare. BMC Characteristics (21) | Yes | Yes |
| | | 2. Allowing attackers to flood requests | Rate Limiting. Table 4, Item 68 | Yes | |
| | | 3. Freely available educational resources on cybersecurity vulnerabilities and attack techniques | Free Education: is not allowed. BMC Guidelines (6) | Yes | |
| 5 | MitM | 1. Attacker using public Wi-Fi access point | Currently Wi-Fi is not supported. BMC Guidelines (13) | Yes | Yes |
| | | 2. Secure socket layer provided by OS | No sockets exist in the BMC paradigm as there is no OS. BMC Characteristics (1 and 9) | Yes | |
| | | 3. Protocol vulnerabilities in ARP, DHCP, DNS, ICMP, and IP | Network Interfaces and Protocol Vulnerabilities. BMC Characteristics (17 and 27) | Yes | |
| | | 4. Open-source automation tools | Automated tools are designed to work with only bare computing devices and applications. BMC System Guidelines (19) | Yes | |
| | | 5. Open-source OSs | No operating system. BMC Characteristics (1) | Yes | |
| | | 6. There is no proper authentication measure to validate users | Only bare users can communicate with each other. All bare users must be physically authorized and authenticated. BMC Characteristics (23) | Yes | |
| | | 7. Freely available educational resources on cybersecurity vulnerabilities and attack techniques | Education and Knowledge: Restricted to authorized bare users. BMC Guidelines (6) | Yes | |
| 6 | Password | 1. OS vulnerabilities | No operating system. BMC Characteristics (1) | Yes | Yes |
| | | 2. Attacker modifying the number of password entry limits | No password files are stored in bare machines. BMC Characteristics (28) | Yes | |
| | | 3. Attacker accessing password files | | | |
| | | 4. Attacker accessing OS API | There are no system calls or APIs available to the outside world (outside an application suite). BMC Characteristics (8) | Yes | |
| | | 5. Attacker accessing system calls | | | |

**Table 7.** *Cont.*

| # | Type of Attack | Root Cause | BMC (Guidelines, Characteristics, and Preventive Mechanisms) | Prevents Root Cause (Yes/No) | Prevents Attack (Yes/No) |
|---|---|---|---|---|---|
| 7 | Trojan Horse | 1. User downloading software | No downloads are allowed. BMC Characteristics (19) | Yes | Yes |
| | | 2. Running a downloaded code automatically | | | |
| | | 3. OS vulnerabilities | No operating system. BMC Characteristics (1) | Yes | |
| | | 4. Attacker accessing system calls | There are no system calls or APIs available to the outside world (outside an application suite). BMC Characteristics (8) | Yes | |
| | | 5. Attacker accessing OS API | | | |
| | | 6. An attacker using autorun in script files | Script files are not allowed. BMC Guidelines (14) | Yes | |
| 8 | Viruses | 1. User downloading email attachments | No downloads are allowed. BMC Characteristics (19) | Yes | Yes |
| | | 2. User downloading software | | | |
| | | 3. User accessing fake websites | Only access to authenticated bare websites. BMC Guidelines (18) | Yes | |
| | | 4. User using an infected USB | Dual USBs (one for booting and a second for application). Both USBs must be physically secure. BMC Characteristics (25) | Yes | |
| | | 5. Allowing script files in emails | Scripts and batch files are not allowed. BMC Guidelines (14) | Yes | |
| | | 6. Attacker using batch files | | | |
| | | 7. OS Vulnerabilities | No operating system. BMC Characteristics (1) | Yes | |
| 9 | Worms | 1. User downloading software | No downloads are allowed. BMC Characteristics (19) | Yes | Yes |
| | | 2. User downloading email attachments | No Attachments and Links in emails BMC Guidelines (15) | Yes | |
| | | 3. User accessing fake Websites | Only access to authenticated bare websites. BMC Guidelines (18) | Yes | |
| | | 4. User using an infected USB | Dual USBs (one for booting and a second for application). Both USBs must be physically secure. BMC Characteristics (25) | Yes | |
| | | 5. Allowing script files in emails | Script and batch files are not allowed. BMC Guidelines (14) | Yes | |
| | | 6. Attacker using batch files | | | |
| | | 7. OS Vulnerabilities | No operating system. BMC Characteristics (1) | Yes | |

**Table 7.** *Cont.*

| # | Type of Attack | Root Cause | BMC (Guidelines, Characteristics, and Preventive Mechanisms) | Prevents Root Cause (Yes/No) | Prevents Attack (Yes/No) |
|---|---|---|---|---|---|
| 10 | Spyware | 1. User downloading software | No downloads are allowed. BMC Characteristics (19) | Yes | Yes |
| | | 2. User downloading email attachments | No Attachments and Links in emails BMC Guidelines (15) | Yes | |
| | | 3. User accessing fake Websites | Only access to authenticated bare websites. BMC Guidelines (18) | Yes | |
| | | 4. User using an infected USB | Dual USBs (one for booting and a second for application). Both USBs must be physically secure. BMC Characteristics (25) | Yes | |
| | | 5. Allowing script files in emails | Script and batch files are not allowed. BMC Guidelines (14) | Yes | |
| | | 6. Attacker using batch files | Script and batch files are not allowed. BMC Guidelines (14) | Yes | |
| | | 7. OS Vulnerabilities | No operating system. BMC Characteristics (1) | Yes | |
| 11 | Adware | 1. User downloading software | No downloads are allowed. BMC Characteristics (19) | Yes | Yes |
| | | 2. User downloading email attachments | No Attachments and Links in emails BMC Guidelines (15) | Yes | |
| | | 3. User accessing fake Websites | Only access to authenticated bare websites. BMC Guidelines (18) | Yes | |
| | | 4. Enabling adware to use browser apps | Advertisements are not allowed. BMC Guidelines (17) | Yes | |
| | | 5. OS Vulnerabilities | No operating system. BMC Characteristics (1) | Yes | |
| | | 6. Attacker using cookies | Cookies are not allowed. BMC Guidelines (20) | Yes | |
| 12 | Rootkits | 1. User installing software online | Online installation is not allowed. BMC Guidelines (12) | | Yes |
| | | 2. User downloading software | No downloads are allowed. BMC Characteristics (19) | Yes | |
| | | 3. OS Vulnerabilities | No operating system. BMC Characteristics (1) | Yes | |
| | | 4. User accessing fake websites | Only access to authenticated bare websites. BMC Guidelines (18) | Yes | |
| | | 5. User using an infected USB | Dual USBs (one for booting and a second for application). Both USBs must be physically secure. BMC Characteristics (25) | Yes | |
| | | 6. User using infected firmware | Bundled with the domain-specific application suite. BMC Characteristics (26) | Yes | |
| | | 7. Firmware vulnerabilities | | | |
| | | 8. Freely available educational resources on cybersecurity vulnerabilities and attack techniques | Education and Knowledge: Restricted to authorized bare users. BMC Guidelines (6) | Yes | |

**Table 7.** *Cont.*

| # | Type of Attack | Root Cause | BMC (Guidelines, Characteristics, and Preventive Mechanisms) | Prevents Root Cause (Yes/No) | Prevents Attack (Yes/No) |
|---|---|---|---|---|---|
| 13 | Botnets | 1. Software vulnerabilities | Yes, however, these software vulnerabilities cannot be exploited by the attacker as the domain-specific application suite is statically bounded. BMC Characteristics (2 and 5) | Yes | Yes |
| | | 2. User downloading software | No downloads are allowed. BMC Characteristics (19) | Yes | |
| | | 3. User using an infected USB | Dual USBs (one for booting and a second for application). Both USBs must be physically secure. BMC Characteristics (25) | Yes | |
| | | 4. Open ports | There are no open ports in BMC applications. BMC Characteristics (10) | Yes | |
| 14 | Data Breaches | 1. User mistakes or negligence | Yes: BMC Guidelines (11). Table 4, Item 18 | No | No, these threats are related to physical security abuse. |
| | | 2. Malicious insiders | Yes, all insiders must be properly authorized and trusted. Otherwise, there is no security for any system. BMC Characteristics (23) | No | |
| | | 3. Lack of physical security | Yes: Physical security is required when a user is running an application suite on a bare machine (this is a mandatory requirement in the BMC paradigm). Otherwise, there is no need for physical security because the machine is bare. BMC Characteristics (21) | No | |
| | | 4. User downloading software | No downloads are allowed. BMC Characteristics (19) | Yes | |
| 15 | Advanced Persistent Threats | 1. User downloading software | No downloads are allowed. BMC Characteristics (19) | Yes | |
| | | 2. User using an infected USB | Dual USBs (one for booting and a second for application). Both USBs must be physically secure. BMC Characteristics (25) | Yes | |
| | | 3. User accessing fake websites | Only access to authenticated bare websites. BMC Guidelines (18) | Yes | |
| | | 4. OS Vulnerabilities | No operating system. BMC Characteristics (1) | Yes | |
| | | 5. Freely available educational resources on cybersecurity vulnerabilities and attack techniques | Education and Knowledge: Restricted to authorized bare users. BMC Guidelines (6) | Yes | |
| 16 | SQL Injection | 1. System privileges given to DBMS | No operating system. BMC Characteristics (1) | Yes | Yes |
| | | 2. DBMS by-passing OS | The DBMS is part of the specific-domain application suite. BMC Characteristics (2) | Yes | |
| | | 3. Not validating and authenticating user-entered data | Each message contains encrypted bare user authentication, which is given in person. BMC Characteristics (24) | Yes | |

**Table 7.** *Cont.*

| # | Type of Attack | Root Cause | BMC (Guidelines, Characteristics, and Preventive Mechanisms) | Prevents Root Cause (Yes/No) | Prevents Attack (Yes/No) |
|---|---|---|---|---|---|
| 17 | Supply Chain | 1. Providing backdoors in software and hardware | There are no hardware backdoors, as the devices are bare and physically secured. There are no software backdoors, as the domain-specific application suite can only perform intended functions. BMC Guidelines (2) and BMC Characteristics (2) | Yes | Yes |
| | | 2. OS Vulnerabilities | No operating system. BMC Characteristics (1) | Yes | |
| | | 3. Open-source code | It is a closed system. BMC Guidelines (2) | Yes | |
| | | 4. User downloading software | No downloads are allowed. BMC Characteristics (19) | Yes | |
| | | 5. User using an infected USB | Dual USBs (one for booting and a second for application). Both USBs must be physically secure. BMC Characteristics (25) | Yes | |
| | | 6. User accessing fake Websites | Only access to authenticated bare websites. BMC Guidelines (18) | Yes | |
| 18 | URL Interpretation | 1. Attacker accessing private files through a URL link | Uses URL encoding and limited access permissions. Table 4, Items 51, 86 | Yes | Yes |
| 19 | Insider Threats | 1. Strong security policies not applying to private data | Yes, if the attacker is an insider | No | No |
| | | 2. User mistakes or negligence | Yes: BMC Guidelines (11). Table 4, Item 18 | No | |
| | | 3. Freely available educational resources on cybersecurity vulnerabilities and attack techniques | Education and Knowledge: Restricted to authorized bare users. BMC Guidelines (6) | Yes | |
| 20 | Eavesdropping | 1. Freely available online automation tools | Automated tools are designed to work with only bare computing devices and applications. BMC System Guidelines (19) | Yes | Yes |
| | | 2. Open Wi-Fi access at public places | Currently Wi-Fi not supported. BMC Guidelines (13) | Yes | |
| 21 | Cookies | 1. Marketing tools and techniques online | Advertisements are not allowed. BMC Guidelines (17) | Yes | Yes |
| | | 2. Attackers are intruding into machines | A computing device (PC, Laptop, Smartphone, Server, Client, etc.) is bare. When one application suite is running, another one cannot run, thus there is no intrusion from other applications. BMC Characteristics (21) | Yes | |
| 22 | Social Engineering | 1. Marketing tools and techniques online | Advertisements are not allowed. BMC Guidelines (17) | Yes | Yes |
| | | 2. Attackers are intruding into machines | A computing device (PC, Laptop, Smartphone, Server, Client, etc.) is bare. When one application suite is running, another one cannot run, thus there is no intrusion from other applications. BMC Characteristics (21) | Yes | |

## 6. Significant Contributions

The BMC paradigm results in a new approach to developing computer applications. If all computing devices are bare, including PCs, laptops, and smartphones, there are no resources to protect other than the hardware. Bare machines are ownerless and usable by any user at any time by loading their own bare applications. There is nothing to compromise on a bare machine other than a running application. A bare machine and bare machine applications can survive for a long period of time without obsolescence. A single bare machine can serve many users in a time-shared manner, one user at a time, without dedicating an individual machine to each user. The BMC paradigm forces applications to use abstractions and extensions instead of obsoleting them before their normal life span. The focus on end-user applications instead of computer platforms will reduce waste, dumping, and the obsolescence of hardware, software, and people skills.

When the OS is eliminated, all OS vulnerabilities are eliminated. While a given BMC application suite is running, other applications cannot interfere with it since they cannot run at the same time. A BMC application suite runs only intended functions and nothing else. BMC systems are inherently secure by design and are likely to prevent many cybersecurity attacks. This paper will put a seed in the ground for building future systems that are highly secure. This paper also provides an alternative to conventional systems to build highly secure systems.

Conventional systems are centralized, based on some flavor of an OS, and are potentially exposed to cybersecurity vulnerabilities by their design. Since BMC systems are closed systems, application information on system internals and security mechanisms is not readily available except through reverse engineering. Such information may only be useful for devising attacks against one BMC application, as other applications may have completely different characteristics. The absence of an OS enables applications and their hardware interfaces to be customized to improve security. Only a port currently used by an application can be targeted by a DoS or DDoS attack. There are no OS interfaces or protocols to easily compromise since they are hidden and integrated with the application. BMC systems are not accessible by unauthorized global users unless they steal the credentials of legitimate bare users. The inability to download software and open email attachments and the unavailability of free and open specifications limit cybersecurity vulnerabilities in BMC systems.

In a bare Internet, all devices are bare, and all authorized users are authenticated. This offers an alternative to the current Internet, which is designed for a global market and global users. The Internet is large, complex, and global in nature, whereas a bare Internet is small and only serves a limited population of bare users. Solving cybersecurity problems on the current Internet using global solutions is not practical. In a bare Internet, it is possible to divide and conquer security problems by isolating bare users from the rest of the Internet.

This work serves as a foundation for building secure computer and information systems based on the BMC paradigm. BMC systems are simple, secure by design, and easy to build. The BMC approach enables novel computer architectures and innovative information system design to support domain-specific applications that are independent of computer platforms.

## 7. Conclusions

We evaluated the BMC paradigm for its resilience against 22 popular cyberattacks. We collected data for these attacks and identified their root causes and preventive methods. As most of these attacks target conventional systems with some form of an OS, we identified conventional system guidelines and characteristics and compared them with BMC system guidelines and characteristics. In addition, we provided some background on BMC applications and examined code snippets to give more insight into the BMC paradigm.

We found that in conventional system guidelines, 15 out of 20 have possible exposure to cybersecurity vulnerabilities, whereas in BMC systems, 17 out of 20 have possible

resilience to cybersecurity vulnerabilities. Similarly, in conventional system characteristics, 17 out of 28 have possible exposure to cybersecurity vulnerabilities, whereas in BMC systems, 26 out of 28 have possible resilience to cybersecurity vulnerabilities. Overall, BMC systems show possible resiliency, and conventional systems indicate possible exposure to cybersecurity vulnerabilities.

The following observations are made on root causes versus cyberattacks. Root cause contributions for OS vulnerabilities are 40%, fake websites are 31%, downloads are 45%, and infected USBs are 31%. This clearly indicates the dominance of operating systems in cyberattacks. Similarly, the following observations are made on preventive mechanisms versus cyberattacks. The preventive mechanism contribution for consistent training and reviews is 100%. The preventive mechanism contributions for antivirus software, two-factor authentication, firewalls, and regularly backing up digital records are 27% for each. The preventive mechanism contribution for "keeping OS up to date" is 18%. The preventive mechanism contributions for "scanning system and USB" are 13.6%. Other preventive mechanisms have contributions less than 9%. Notice that none of these preventive mechanisms guarantee 100% protection from any cyberattack.

We also noted that for the BMC systems, the applicability of the root causes is 15%, and the applicability of preventive mechanisms is 63%.

We conducted a comprehensive evaluation of the BMC paradigm using the selected attacks and their root causes, which showed that 20 out of 22 attacks are preventable by using this paradigm. The other two attacks, namely data breaches and insider threats, are not preventable as they are caused by insiders who misuse their trust. We described significant contributions of this research, which justify use of the BMC paradigm for building real-world applications with intrinsic security. Future studies should further investigate the inherent security strengths of the BMC paradigm and the resilience of BMC applications against possible cyberattacks.

**Author Contributions:** Conceptualization, F.A., R.K.K., A.L.W., N.S. and A.R.; methodology, F.A., R.K.K., A.L.W., N.S. and A.R.; validation, F.A., R.K.K., A.L.W. and N.S.; investigation, F.A., R.K.K., A.L.W., N.S. and A.R.; resources, F.A., R.K.K., A.L.W., N.S. and A.R.; writing—original draft, R.K.K.; writing—review and editing, F.A., R.K.K., A.L.W. and N.S.; supervision, R.K.K. and A.L.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** Data are contained within the article.

**Acknowledgments:** The authors acknowledge that the Grammarly AI tool was used to check and correct English grammar and mistakes.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Statista. Number of Internet and Social Media Users Worldwide as of January 2024. Available online: https://www.statista.com/statistics/617136/digital-population-worldwide/ (accessed on 27 March 2024).
2. Aslan, Ö.; Aktuğ, S.S.; Ozkan-Okay, M.; Yilmaz, A.A.; Akin, E. A Comprehensive Review of Cyber Security Vulnerabilities, Threats, Attacks, and Solutions. *Electronics* **2023**, *12*, 1333. [CrossRef]
3. Alenezi, M.; Zarour, M. On the Relationship between Software Complexity and Security. *IJSEA* **2020**, *11*, 51–60. [CrossRef]
4. Mellal, M.A. Obsolescence—A review of the literature. *Technol. Soc.* **2020**, *63*, 101347. [CrossRef]
5. Zallio, M.; Berry, D. Design and Planned Obsolescence. Theories and Approaches for Designing Enabling Technologies. *Des. J.* **2017**, *20*, S3749–S3761. [CrossRef]
6. Aladeojebi, T.K. Planned Obsolescence. *IRJSE* **2013**, *4*, 1504–1508.
7. Malinauskaite, J.; Erdem, F.B. Planned Obsolescence in the Context of a Holistic Legal Sphere and the Circular Economy. *Oxf. J. Leg. Stud.* **2021**, *41*, 719–749. [CrossRef]
8. Drozd, M.; Barabas, M.; Gregr, M.; Chmelar, P. Buffer overflow attacks data acquisition. In Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems, Prague, Czech Republic, 15–17 September 2011; pp. 775–779. [CrossRef]

9. Zieni, R.; Massari, L.; Calzarossa, M.C. Phishing or Not Phishing? A Survey on the Detection of Phishing Websites. *IEEE Access* **2023**, *11*, 18499–18519. [CrossRef]
10. Razaulla, S.; Fachkha, C.; Markarian, C.; Gawanmeh, A.; Mansoor, W.; Fung, B.C.M.; Assi, C. The Age of Ransomware: A Survey on the Evolution, Taxonomy, and Research Directions. *IEEE Access* **2023**, *11*, 40698–40723. [CrossRef]
11. Tripathi, N.; Hubballi, N. Application Layer Denial-of-Service Attacks and Defense Mechanisms: A Survey. *ACM Comput. Surv.* **2021**, *54*, 86. [CrossRef]
12. Conti, M.; Dragoni, N.; Lesyk, V. A Survey of Man In The Middle Attacks. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 3. [CrossRef]
13. Alkhwaja, I.; Albugami, M.; Alkhwaja, A.; Alghamdi, M.; Abahussain, H.; Alfawaz, F.; Almurayh, A.; Min-Allah, N. Password Cracking with Brute Force Algorithm and Dictionary Attack Using Parallel Programming. *Appl. Sci.* **2023**, *13*, 5979. [CrossRef]
14. National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce. *Guide to Malware Incident Prevention and Handling for Desktops and Laptops*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2013. Available online: https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-83r1.pdf (accessed on 27 January 2024).
15. Khan, H.A.; Syed, A.; Mohammad, A.; Halgamuge, M.N. Computer virus and protection methods using lab analysis. In Proceedings of the IEEE 2nd International Conference on Big Data Analysis (ICBDA), Beijing, China, 10–12 March 2017; pp. 882–886. [CrossRef]
16. Saudi, M.M.; Cullen, A.J.; Woodward, M.E. STAKCERT Framework in Eradicating Worms Attack. In Proceedings of the International Conference on CyberWorlds, Bradford, UK, 7–11 September 2009; pp. 257–264. [CrossRef]
17. Naser, M.; Abu Al-Haija, Q. Spyware Identification for Android Systems Using Fine Trees. *Information* **2023**, *14*, 102. [CrossRef]
18. Umar, F.; Khurana, S.S.; Singh, P.; Kumar, M. An Empirical Study on Detection of Android Adware Using Machine Learning Techniques. *Multimed Tools Appl.* **2024**, *83*, 38753–38792. [CrossRef]
19. Kühnhauser, W.E. Root Kits—An operating systems viewpoint. *SIGOPS Oper. Syst. Rev.* **2004**, *38*, 12–23. [CrossRef]
20. Owen, H.; Zarrin, J.; Pour, S.M. A Survey on Botnets, Issues, Threats, Methods, Detection and Prevention. *J. Cybersecur. Priv.* **2022**, *2*, 74–88. [CrossRef]
21. Fleury-Charles, A.; Chowdhury, M.M.; Rifat, N. Data Breaches: Vulnerable Privacy. In Proceedings of the IEEE International Conference on Electro Information Technology (eIT), Mankato, MN, USA, 19–21 May 2022; pp. 538–543. [CrossRef]
22. Gan, C.; Lin, J.; Huang, D.-W.; Zhu, Q.; Tian, L. Advanced Persistent Threats and Their Defense Methods in Industrial Internet of Things: A Survey. *Mathematics* **2023**, *11*, 3115. [CrossRef]
23. Alghawazi, M.; Alghazzawi, D.; Alarifi, S. Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review. *J. Cybersecur. Priv.* **2022**, *2*, 764–777. [CrossRef]
24. Sobb, T.; Turnbull, B.; Moustafa, N. Supply Chain 4.0: A Survey of Cyber Security Challenges, Solutions and Future Directions. *Electronics* **2020**, *9*, 1864. [CrossRef]
25. Sharma, P.; Nagpal, B. A Study on URL Manipulation Attack Methods and Their Countermeasures. *IJETCSE* **2015**, *15*, 116–119.
26. Saxena, N.; Hayes, E.; Bertino, E.; Ojo, P.; Choo, K.-K.R.; Burnap, P. Impact and Key Challenges of Insider Threats on Organizations and Critical Businesses. *Electronics* **2020**, *9*, 1460. [CrossRef]
27. Kim, M.; Suh, T. Eavesdropping Vulnerability and Countermeasure in Infrared Communication for IoT Devices. *Sensors* **2021**, *21*, 8207. [CrossRef] [PubMed]
28. Sivakorn, S.; Polakis, I.; Keromytis, A.D. The Cracked Cookie Jar: HTTP Cookie Hijacking and the Exposure of Private Information. In Proceedings of the IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2016; pp. 724–742. [CrossRef]
29. Salahdine, F.; Kaabouch, N. Social Engineering Attacks: A Survey. *Future Internet* **2019**, *11*, 89. [CrossRef]
30. CVE—Common Vulnerabilities and Exposures. MITRE Corporation. Available online: https://cve.mitre.org/ (accessed on 17 July 2024).
31. CWE—Common Weakness Enumeration. MITRE Corporation. Available online: https://cwe.mitre.org/ (accessed on 17 July 2024).
32. MITRE ATT&CK®. MITRE Corporation. Available online: https://attack.mitre.org/ (accessed on 17 July 2024).
33. IoT Business News. State of IoT 2023: Number of Connected IoT Devices Growing 16% to 16.0 Billion Globally—Wi-Fi, Bluetooth, and Cellular Driving the Market. Available online: https://iotbusinessnews.com/2023/05/25/34645-state-of-iot-2023-number-of-connected-iot-devices-growing-16-to-16-0-billion-globally-wi-fi-bluetooth-and-cellular-driving-the-market/ (accessed on 27 January 2024).
34. Zhang, Y.; Frank, R.; Warkentin, N.; Zakimi, N. Accessible from the open web: A qualitative analysis of the available open-source information involving cyber security and critical infrastructure. *J. Cybersecur.* **2022**, *8*, tyac003. [CrossRef]
35. Mafamane, R.; Ouadou, M.; Hassani, A.T.J.; Minaoui, K. Study of the heterogeneity problem in the Internet of Things and Cloud Computing integration. In Proceedings of the 2020 10th International Symposium on Signal, Image, Video and Communications (ISIVC), Saint-Etienne, France, 7–9 April 2021; pp. 1–6. [CrossRef]
36. Evolution of Computing. The Problem of Growing Complexity in the Evolution of Computing. Available online: https://evolutionofcomputing.org/Multicellular/ProblemStatement.html (accessed on 27 January 2024).
37. Umejiaku, A.P.; Dhakal, P.; Sheng, V.S. Balancing Password Security and User Convenience: Exploring the Potential of Prompt Models for Password Generation. *Electronics* **2023**, *12*, 2159. [CrossRef]
38. Statista. Number of Internet of Things (IoT) Connected Devices Worldwide from 2019 to 2023, with Forecasts from 2022 to 2030. Available online: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/ (accessed on 27 March 2024).

39. Okafor, U.; Karne, R.K.; Wijesinha, A.L.; Appiah-Kubi, P. Eliminating the Operating System via the Bare Machine Computing Paradigm. In Proceedings of the Fifth International Conference on Future Computational Technologies and Applications, Valencia, Spain, 27 May–1 June 2013; pp. 1–6.
40. MisCircuitos. Difference between Bare Metal vs. Embedded Linux. Available online: https://miscircuitos.com/difference-between-bare-metal-vs-embedded-linux/ (accessed on 27 January 2024).
41. IBM. What is a Bare Metal Server? Available online: https://www.ibm.com/topics/bare-metal-dedicated-servers (accessed on 27 January 2024).
42. Karne, R.K.; Wijesinha, A.L.; Liang, S.; Appiah-Kubi, P. A Bare PC Mass Storage USB Driver. *Int. J. Comput. Appl.* **2013**, *21*, 32.
43. Alotaibi, F.; Karne, R.K.; Wijesinha, A.; Soundararajan, N.; Rangi, A. A Chat Application on a Bare Internet. In Proceedings of the 2024 IEEE 48th Annual Computers, Software, and Applications (COMPSAC), Osaka, Japan, 2–4 July 2024; pp. 2411–2416.
44. Engler, D.R. The Exokernel Operating System Architecture. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1998.
45. Levis, P. Experiences from a decade of TinyOS development. In Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation, Hollywood, CA, USA, 8–10 October 2012; pp. 207–220.
46. Lange, J.; Pedretti, K.; Hudson, T.; Dinda, P.; Cui, Z.; Xia, L.; Bridges, P.; Gocke, A.; Jaconette, S.; Levenhagen, M.; et al. Palacios and Kitten: New High Performance Operating Systems For Scalable Virtualized and Native Supercomputing. In Proceedings of the 2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS), Atlanta, GA, USA, 19–23 April 2010; pp. 1–12. [CrossRef]
47. Isaac, O.; Okokpujie, K.; Akinwumi, H.; Juwe1, J.; Otunuya, H.; Alagbe, O. An Overview of Microkernel Based Operating Systems. *IOP Conf. Ser. Mater. Sci. Eng.* **2021**, *1107*, 012052. [CrossRef]
48. Kong, X.; Chen, J.; Bai, W.; Xu, Y.; Elhaddad, M.; Raindel, S.; Padhye, J.; Lebeck, A.R.; Zhuo, D. Understanding RDMA Microarchitecture Resources for Performance Isolation. In Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation, Boston, MA, USA, 17–19 April 2023; pp. 31–48.
49. Pai, V.S.; Druschel, P.; Zwaenepoel, W. IO-Lite: A Unified I/O Buffering and Caching System. *ACM Trans. Comput. Syst.* **2000**, *18*, 37–66. [CrossRef]
50. Zhang, I.; Liu, J.; Austin, A.; Roberts, M.L.; Badam, A. I'm Not Dead Yet! The Role of the Operating System in a Kernel-Bypass Era. In Proceedings of the Workshop on Hot Topics in Operating Systems, Bertinoro, Italy, 13–15 May 2019; pp. 73–80. [CrossRef]
51. Baccelli, E.; Gündogan, C.; Hahm, O.; Kietzmann, P.; Lenders, M.S.; Petersen, H.; Schleiser, K.; Schmidt, T.C.; Wählisch, M. RIOT: An Open Source Operating System for Low-End Embedded Devices in the IoT. *IEEE Internet Things J.* **2018**, *5*, 6. [CrossRef]
52. Sen, S.; Guérin, R.; Hosanagar, K. Functionality-rich Versus Minimalist Platforms: A Two-sided Market Analysis. *ACM SIGCOMM Comput. Commun. Rev.* **2011**, *41*, 36–43. [CrossRef]
53. Soundararajan, N.; Karne, R.; Wijesinha, A.; Ordouie, N.; Chang, H. Design Issues in Running a Webserver on Bare PC Multi-Core Architecture. In Proceedings of the 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, Spain, 13–17 July 2020; pp. 555–564. [CrossRef]
54. Appiah-Kubi, P.; Karne, R.K.; Wijesinha, A.L. A Bare PC TLS Webmail Server. In Proceedings of the 2012 International Conference on Computing, Networking and Communications (ICNC), Maui, HI, USA, 30 January–2 February 2012; pp. 149–153. [CrossRef]
55. Wikipedia. Available online: https://en.wikipedia.org/wiki/Security_through_obscurity (accessed on 20 August 2024).
56. Alotaibi, F.; Karne, R.K.; Wijesinha, A. A Stateless Bare PC Web Server. In Proceedings of the 19th International Conference on Web Information Systems and Technologies (WEBIST 2023), Rome, Italy, 15–17 November 2023; pp. 406–413. [CrossRef]
57. The SSL Store. Executing a Man-in-the-Middle Attack in Just 15 Minutes. Available online: https://www.thesslstore.com/blog/man-in-the-middle-attack-2 (accessed on 27 March 2024).
58. Alwis, C.D.; Porambage, P.; Dev, K.; Gadekallu, T.R.; Liyanage, M. A Survey on Network Slicing Security: Attacks, Challenges, Solutions and Research Directions. *IEEE Commun. Surv. Tutor.* **2024**, *26*, 534–570. [CrossRef]
59. Harrison, R. Reducing complexity in securing heterogeneous networks. *Netw. Secur.* **2015**, *10*, 11–13. [CrossRef]
60. Li, L.; Daoyuan, L.; Bissyandé, T.F.; Klein, J.; Trao, Y.L.; Lo, D.; Cavallaro, L. Understanding Android app piggybacking: A systematic study of malicious code grafting. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 1269–1284. [CrossRef]
61. Alhamry, M.; Elmedany, W. Exploring Wi-Fi WPA2 KRACK Vulnerability: A Review Paper. In Proceedings of the 2022 International Conference on Data Analytics for Business and Industry (ICDABI), Sakhir, Bahrain, 25–26 October 2022; pp. 766–772.
62. Vondráček, M.; Pluskal, J.; Ryšavý, O. Automated Man-in-the-Middle Attack Against Wi-Fi Networks. *J. Digit. Forensic. Secur. Law* **2018**, *13*, 9. [CrossRef]
63. Pan, Z.; Shen, W.; Wang, X.; Yang, Y.; Chang, R.; Liu, Y.; Liu, C.; Liu, Y.; Ren, K. Ambush From All Sides: Understanding Security Threats in Open-Source Software CI/CD Pipelines. *IEEE Trans. Dependable Secur. Comput.* **2024**, *21*, 403–418. [CrossRef]
64. Duman, S.; Büchler, M.; Egele, M.; Kirda, E. Pellucid Attachment: Protecting Users from Attacks via E-mail Attachments. *IEEE Trans. Dependable Secure Comput.* **2023**, *21*, 1342–1354. [CrossRef]
65. Hakak, S.; Khan, W.Z.; Imran, M.; Choo, K.R.; Shoaib, M. Have You Been a Victim of COVID-19-Related Cyber Incidents? Survey, Taxonomy, and Mitigation Strategies. *IEEE Access* **2020**, *8*, 124134–124144. [CrossRef]
66. Cengiz, A.B.; Kalem, G.; Boluk, P.S. The Effect of Social Media User Behaviors on Security and Privacy Threats. *IEEE Access* **2022**, *10*, 57674–57684. [CrossRef]

67. Chang, V.; Golightly, L.; Xu, Q.A.; Boonmee, T.; Liu, B.S. Cybersecurity for children: An investigation into the application of social media. *Enterp. Inf. Syst.* **2023**, *17*, 2188122. [CrossRef]
68. Masri, R.; Aldwairi, M. Automated malicious advertisement detection using VirusTotal, URLVoid, and TrendMicro. In Proceedings of the 2017 8th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 4–6 April 2017; pp. 336–341. [CrossRef]
69. Pooranian, Z.; Conti, M.; Haddadi, H.; Tafazolli, R. Online Advertising Security: Issues, Taxonomy, and Future Directions. *IEEE Commun. Surv. Tut.* **2020**, *23*, 2494–2524. [CrossRef]
70. Shantanu, B.; Janet, J.; Arul Kumar, R.J. Malicious URL Detection: A Comparative Study. In Proceedings of the 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), Coimbatore, India, 25–27 March 2021; pp. 1147–1151. Available online: https://ieeexplore.ieee.org/document/9396014 (accessed on 20 August 2024).
71. Aljabri, M.; Altamimi, H.S.; Albelali, S.A.; Al-Harbi, M.; Alhuraib, H.T.; Alotaibi, N.K.; Alahmadi, A.A.; Alhaidari, F.; Mohammad, R.M.A.; Salah, K. Detecting Malicious URLs Using Machine Learning Techniques: Review and Research Directions. *IEEE Access* **2022**, *10*, 121395–121417. [CrossRef]
72. Cunningham, B.; Fuller, E.; Little, C.; Schack, T.; Dykstra, T.; Hoagberg, M.; Miles, G.; Rogers, R. *Network Security Evaluation Using the NSA IEM*; Syngress: Rockland, MA, USA, 2005; ISBN 978-1-59749-035-1.
73. Gao, Z.; Ansari, N. Tracing cyber attacks from the practical perspective. *IEEE Commun. Mag.* **2005**, *43*, 123–131. [CrossRef]
74. Yang, J. Analysis on cookies and cybersecurity. In Proceedings of the Third International Symposium on Computer Engineering and Intelligent Communications (ISCEIC 2022), Xi'an, China, 16–18 September 2022; Volume 12462, pp. 217–224.
75. Bhurtel, M.; Rawat, D.B. Unveiling the Landscape of Operating System Vulnerabilities. *Future Internet* **2023**, *15*, 248. [CrossRef]
76. Jang, M.; Kim, H.; Yun, Y. Detection of DLL Inserted by Windows Malicious Code. In Proceedings of the 2007 International Conference on Convergence Information Technology (ICCIT 2007), Gwangju, Republic of Korea, 21–23 November 2007; pp. 1059–1064. [CrossRef]
77. Alzahrani, S.; Xiao, Y.; Sun, W. An Analysis of Conti Ransomware Leaked Source Codes. *IEEE Access* **2022**, *10*, 100178–100193. [CrossRef]
78. Chordiya, A.R.; Majumder, S.; Javaid, A.Y. Man-in-the-Middle (MITM) Attack Based Hijacking of HTTP Traffic Using Open Source Tools. In Proceedings of the 2018 IEEE International Conference on Electro/Information Technology (EIT), Rochester, MI, USA, 3–5 May 2018; pp. 438–443. [CrossRef]
79. Sang, F.L.; Nicomette, V.; Deswarte, Y. I/O Attacks in Intel PC-based Architectures and Countermeasures. In Proceedings of the First SysSec Workshop, Amsterdam, The Netherlands, 6 July 2011; pp. 19–26. [CrossRef]
80. Gozman, D.; Willcocks, L. The emerging Cloud Dilemma: Balancing innovation with cross-border privacy and outsourcing regulations. *J. Bus. Res.* **2019**, *97*, 235–256. [CrossRef]
81. Benaroch, M. Third-party induced cyber incidents—Much ado about nothing? *J. Cybersecur.* **2021**, *7*, tyab020. [CrossRef]
82. Shah, M.; Soni, V.; Shah, H.; Desai, M. TCP/IP network protocols—Security threats, flaws and defense methods. In Proceedings of the 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 16–18 March 2016; pp. 2693–2699.
83. Liu, R.; Yu, B.; Wang, B.; Ye, J.; Huang, J.; Kong, X. SEEKER: A Root Cause Analysis Method Based on Deterministic Replay for Multi-Type Network Protocol Vulnerabilities. In Proceedings of the 2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Wuhan, China, 9–11 December 2022; pp. 131–138. [CrossRef]
84. Geetha, K.; Sreenath, N. SYN flooding attack—Identification and analysis. In Proceedings of the International Conference on Information Communication and Embedded Systems (ICICES2014), Chennai, India, 27–28 February 2014; pp. 1–7.
85. AbdAllah, E.G.; Hassanein, H.S.; Zulkernine, M. A Survey of Security Attacks in Information-Centric Networking. *IEEE Commun. Surv. Tut.* **2015**, *17*, 1441–1454. [CrossRef]
86. Kalafut, A.; Acharya, A.; Gupta, M. A study of malware in peer-to-peer networks. In Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, Rio de Janeriro, Brazil, 25–27 October 2006; pp. 327–332.
87. Lalonde Lévesque, F.; Chiasson, S.; Somayaji, A.; Fernandez, J.M. Technological and Human Factors of Malware Attacks: A Computer Security Clinical Trial Approach. *ACM Trans. Priv. Secur.* **2018**, *21*, 18. [CrossRef]
88. Faruk, M.J.H.; Shahriar, H.; Valero, M.; Barsha, F.L.; Sobhan, S.; Khan, A.; Whitman, M.; Cuzzocrea, A.; Lo, D.; Rahman, A.; et al. Malware Detection and Prevention using Artificial Intelligence Techniques. In Proceedings of the 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 15–18 December 2021; pp. 5369–5377. [CrossRef]
89. Syafitri, W.; Shukur, Z.; Mokhtar, U.A.; Sulaiman, R.; Ibrahim, M.A. Social Engineering Attacks Prevention: A Systematic Literature Review. *IEEE Access* **2022**, *10*, 39325–39343. [CrossRef]
90. Shokeen, R.; Shanmugam, B.; Kannoorpatti, K.; Azam, S.; Jonkman, M.; Alazab, M. Vulnerabilities Analysis and Security Assessment Framework for the Internet of Things. In Proceedings of the 2019 Cybersecurity and Cyberforensics Conference (CCC), Melbourne, Australia, 8–9 May 2019; pp. 22–29. [CrossRef]
91. Winter, J.; Dietrich, K. A hijacker's guide to communication interfaces of the trusted platform module. *Comput. Math. Appl.* **2013**, *65*, 748–761. [CrossRef]
92. Ylli, E.; Fejzaj, J. Man in the Middle: Attack and Protection. In Proceedings of the 4th International Conference on Recent Trends and Applications in Computer Science and Information Technology, Tirana, Albania, 21–22 May 2021; pp. 198–204.

93. Otta, S.P.; Panda, S.; Gupta, M.; Hota, C. A Systematic Survey of Multi-Factor Authentication for Cloud Infrastructure. *Future Internet* **2023**, *15*, 146. [CrossRef]
94. Lockheed Martin. Gaining the Advantage: Cyber Kill Chain®. Available online: https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/Gaining_the_Advantage_Cyber_Kill_Chain.pdf (accessed on 27 January 2024).
95. Pirry, C.; Marco-Gisbert, H.; Begg, C. A Review of Memory Errors Exploitation in x86-64. *Computers* **2020**, *9*, 48. [CrossRef]
96. Alabdan, R. Phishing Attacks Survey: Types, Vectors, and Technical Approaches. *Future Internet* **2020**, *12*, 168. [CrossRef]
97. Oz, H.; Aris, A.; Levi, A.; Uluagac, A.S. A Survey on Ransomware: Evolution, Taxonomy, and Defense Solutions. *ACM Comput. Surv.* **2022**, *54*, 238. [CrossRef]
98. Yamany, B.; Elsayed, M.S.; Jurcut, A.D.; Abdelbaki, N.; Azer, M.A. A Holistic Approach to Ransomware Classification: Leveraging Static and Dynamic Analysis with Visualization. *Information* **2024**, *15*, 46. [CrossRef]
99. Saghezchi, F.B.; Mantas, G.; Violas, M.A.; de Oliveira Duarte, A.M.; Rodriguez, J. Machine Learning for DDoS Attack Detection in Industry 4.0 CPPSs. *Electronics* **2022**, *11*, 602. [CrossRef]
100. Morsy, S.M.; Nashat, D. D-ARP: An Efficient Scheme to Detect and Prevent ARP Spoofing. *IEEE Access* **2022**, *10*, 49142–49153. [CrossRef]
101. Събев, П.; Petrov, M. Android Password Managers and Vault Applications: Data Storage Security Issues Identification. *J. Inf. Secur. Appl.* **2022**, *67*, 103152. [CrossRef]
102. Gudipati, V.K.; Vetwal, A.; Kumar, V.; Adeniyi, A.; Abuzneid, A. Detection of Trojan Horses by the analysis of system behavior and data packets. In Proceedings of the 2015 Long Island Systems, Applications and Technology, Farmingdale, NY, USA, 1 May 2015; pp. 1–4. [CrossRef]
103. Chen, N.; Chen, B. Defending against OS-Level Malware in Mobile Devices via Real-Time Malware Detection and Storage Restoration. *J. Cybersecur. Priv.* **2022**, *2*, 311–328. [CrossRef]
104. Djenna, A.; Bouridane, A.; Rubab, S.; Marou, I.M. Artificial Intelligence-Based Malware Detection, Analysis, and Mitigation. *Symmetry* **2023**, *15*, 677. [CrossRef]
105. Vander–Pallen, M.A.; Addai, P.; Isteefanos, S.; Mohd, T.K. Survey on Types of Cyber Attacks on Operating System Vulnerabilities since 2018 onwards. In Proceedings of the 2022 IEEE World AI IoT Congress (AIIoT), Seattle, WA, USA, 6–9 June 2022; pp. 01–07. [CrossRef]
106. Syeda, D.Z.; Asghar, M.N. Dynamic Malware Classification and API Categorisation of Windows Portable Executable Files Using Machine Learning. *Appl. Sci.* **2024**, *14*, 1015. [CrossRef]
107. U.S. Cybersecurity and Infrastructure Security Agency (CISA). Protecting Your Home Computer from Spyware, U.S. Cybersecurity and Infrastructure Security Agency (CISA). 2005. Available online: https://www.cisa.gov/sites/default/files/publications/spywarehome_0905.pdf (accessed on 27 January 2024).
108. Vasani, V.; Bairwa, A.K.; Joshi, S.; Pljonkin, A.; Kaur, M.; Amoon, M. Comprehensive Analysis of Advanced Techniques and Vital Tools for Detecting Malware Intrusion. *Electronics* **2023**, *12*, 4299. [CrossRef]
109. Kumar, S.S.; Valavan, A.P.; Prathiksha, V. Prevention of Kernel Rootkit in Cloud Computing. In Proceedings of the 2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 17–19 May 2023; pp. 732–739. [CrossRef]
110. Thanh Vu, S.N.; Stege, M.; El-Habr, P.I.; Bang, J.; Dragoni, N. A Survey on Botnets: Incentives, Evolution, Detection and Current Trends. *Future Internet* **2021**, *13*, 198. [CrossRef]
111. Molitor, D.; Raghupathi, W.; Saharia, A.; Raghupathi, V. Exploring Key Issues in Cybersecurity Data Breaches: Analyzing Data Breach Litigation with ML-Based Text Analytics. *Information* **2023**, *14*, 600. [CrossRef]
112. Alshamrani, A.; Myneni, S.; Chowdhary, A.; Huang, D. A Survey on Advanced Persistent Threats: Techniques, Solutions, Challenges, and Research Opportunities. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 1851–1877. [CrossRef]
113. OWASP Foundation. SQL Injection Prevention Cheat Sheet. Available online: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html (accessed on 27 January 2024).
114. Fan, L.; Zhang, B.; Xiong, S.; Li, Q. Secure Change Control for Supply Chain Systems via Dynamic Event Triggered Using Reinforcement Learning under DoS Attacks. *Electronics* **2024**, *13*, 1136. [CrossRef]
115. S. M. Christey. Chapter 11: Preventing Common Problems. Available online: https://www.cgisecurity.com/owasp/html/ch11s04.html (accessed on 17 July 2024).
116. Lee, I. Analysis of Insider Threats in the Healthcare Industry: A Text Mining Approach. *Information* **2022**, *13*, 404. [CrossRef]
117. Chang, X.; Peng, L.; Zhang, S. Allocation of Eavesdropping Attacks for Multi-System Remote State Estimation. *Sensors* **2024**, *24*, 850. [CrossRef]
118. Alharbi, J.A.; Albesher, A.S.; Wahsheh, H.A. An Empirical Analysis of E-Governments' Cookie Interfaces in 50 Countries. *Sustainability* **2023**, *15*, 1231. [CrossRef]
119. Airehrour, D.; Vasudevan Nair, N.; Madanian, S. Social Engineering Attacks and Countermeasures in the New Zealand Banking System: Advancing a User-Reflective Mitigation Model. *Information* **2018**, *9*, 110. [CrossRef]