Distributed Operating System Security and Protection: A Short Survey

20

Ghada Abdelmoumin and Noha Hazzazi

Abstract

In this paper, we investigate several modern distributed operating systems (DiOSs) and their security policies and mechanisms. We survey the various security and protection issues present in DiOSs and review strategies and techniques used by DiOSs to control access to system resources and protect the integrity of the information stored in the system from accidental events and malicious activities. Further, we distinguish between network security and DiOSs security and explore the attack surface of DiOSs compared to traditional operating systems. We concentrate on a class of distributed operating systems known as cloud operating systems (COSs).

Keywords

Attack surface · Cloud system · Distributed system · Operating system · Security · Threats · Vulnerabilities · Protection

20.1 Introduction

Distributed systems have been in existence since the advent of the Internet. These systems are reliable, high-performing, fault-tolerant, modular, and scalable computing platforms in which various computers or nodes access shared remote resources and perform distributed computations collaboratively. A distributed system is an assemblage of autonomous and heterogeneous processors that are connected, loosely-coupled, and geographically dispersed. These processors don't share a memory or use a common clock. Instead, they communicate over the network using messages and execute asynchronously to provide services and compute distributed tasks jointly, each with its discrete notion of time. Each computer or node (i.e., a hardware device or software process) in a distributed system runs its local operating system, a network protocol stack, and middleware that implements the distributed software [1]. Further, nodes with their separate clock can join and leave the system, thus leading to a highly dynamic system and an underlying network with an unceasingly changing topology and performance [2].

Various operating system classes can run on systems with multiple computers or processors. Kshemkalyani and Singhal in [1] specify three classes of operating systems: network operating system, distributed operating system, and multiprocessor operating system. The amount of coupling, i.e., tightly-coupled or loosely-coupled, between the software and hardware components in a distributed system determines the class of operating system running on the system. A system in which the hardware (processors) is loosely-coupled and the software (middleware and distributed software) is tightly-coupled runs a distributed operating system. To that extent, loosely-coupled hardware is heterogeneous (different speeds and possibly operating systems) and geographically dispersed, whereas a tightly-coupled software is homogeneous. Conversely, systems running a network operating system have heterogeneous software and hardware, and those running a multiprocessor operating system have homogeneous software and hardware. This paper focuses on the distributed operating system class and surveys various distributed operating systems implementations and their security implications.

The primary objective of this paper is to explore DiOSs internal and external threats, define their attack surface, and distinguish between the various DiOSsimplementations in

G. Abdelmoumin (✉) · N. Hazzazi
Howard University, Washington, DC, USA
e-mail: ghada.abdelmoumin@bison.howard.edu; noha.hazzazi@howard.edu

S. Latifi (ed.), *17th International Conference on Information Technology–New Generations (ITNG 2020)*, Advances in Intelligent Systems and Computing 1134,
https://doi.org/10.1007/978-3-030-43020-7_20

terms of their security and protection strategies and mechanisms.

The organization of the rest of this paper is as follows: In Sect. 20.2, we describe distributed systems architecture and components and define the distributed operating system (DiOS) and explain its functions. Next, in Sect. 20.3, we explore various DiOS implementations. In Sect. 20.4, we introduce cloud computing and discuss the cloud operating system (COS) and describes its functions. Following in Sect. 20.5, we present traditional DiOSs security issues, address security issues pertinent to COSs, and explore various DiOSs security and protection mechanisms with a particular focus on COSs. Finally, we present our conclusions and future work in Sect. 20.6.

20.2 Distributed Systems

20.2.1 Definition and Characteristics

A distributed system is an assemblage of independent and networked computers or processors, often dispersed. They jointly collaborate over a communication network to provide services, access to shared resources, and solve computationally complex, but modular problems. Several characteristics distinguish distributed systems from other networked or centralized systems. Distributed systems are highly dynamic autonomous and geographically dispersed systems, often characterized as heterogeneous, loosely-coupled, high-performing, reliable, fault-tolerant, scalable, modular, transparent, and cost-effective systems [1, 3, 4]. Other characteristics include concurrency of nodes, lack of a global clock and shared memory, distributed execution, and independent failure of nodes.

Distributed systems are inherently complex systems. To simplify their complexity, they use a layered architecture in which the system components stack in layers. In addition to the hardware components (i.e., processor and local memory), each node in a distributed system has three interacting main software components: an operating system, a network protocol stack, and middleware. Figure 20.1 [1] shows the interaction of distributed system software components. The middleware which drives the distributed system enables the system to be transparent by concealing the system independent hardware and software components from users and applications [1, 2, 4].

Transparency is an important characteristic of distributed systems and an essential design goal. A transparent distributed system appears to the user and application as a "single coherent system" or "virtual uniprocessor" rather than a collection of distinct nodes [4, 5]. It hides the systems resources and renders them autonomous. For example, a distributed system in which access to remote resources is transparent, users access the remote resources using the same operations they use to access the local resources. A transparent system, therefore, hides the location and identity of its resources from the user.

In general, the main goal of distributed systems is to make distribution transparent such that processes running on multiple processors and resources residing on multiple nodes are invisible to the users. Several aspects of a distributed system are transparent to the users and applications. Table 20.1 shows the different transparent aspects of a distributed system. These aspects include access, location, relocation, migration, replication, concurrency, mobility, scaling, performance, and failure [3–5]. Access and location transparencies, also known as network transparency, have a direct impact on distributed resources utilization. While transparency is a desired aspect of a distributed system, there are situations where hiding the system resources has an undesirable effect. For example, location-based and context-aware systems require that distributed resources be visible.

Although distributed systems offer many benefits to users and applications, such as resource sharing and computation speedup, their main advantage is their infrastructure. The distributed resources and services support a view of a system in which a resource or a service represents a utility - one of the most important aspects of cloud computing. We discuss cloud-based systems and COSs in Sect. 20.4.

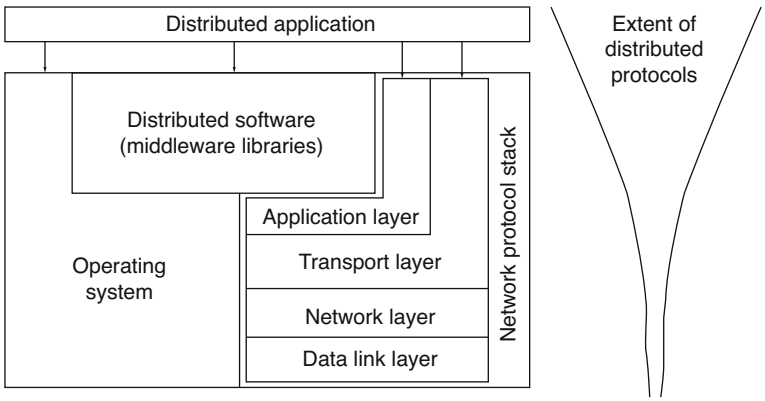

Fig. 20.1 Distributed operating software components (Source: Distributed Systems Principles, Algorithms, and Systems, p. 3)

Table 20.1 Transparent aspects of distributed systems

Transparent aspect	Description
Acess	Accessing a distributed resource is the same as accessing a local resource
Location	The distributed resource namespace (identifier) remains the same irrespective of location
Relocation	The user is not aware of the movement of the data, processes, and or computations
Migration	The user is not aware when the resource location changes
Mobility	The user is not aware when the system ports resources
Concurrency	The user is not aware of the concurrent access to resources by others
Scaling	The user and applications are not aware when the system adds new nodes, resource, or both
Performance	System configuration can improve its performance
Failure	The system conceals the failure from the user and their applications and allows them to continue their tasks

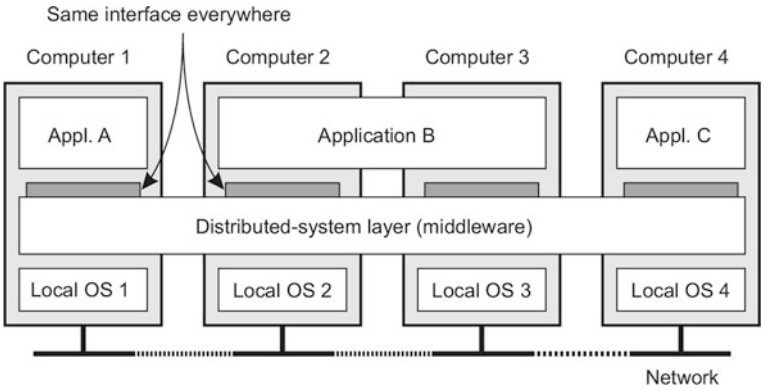

Fig. 20.2 A distributed system organized as middleware. The middleware layer extends over multiple machines and offers each application the same interface (Source: DISTRIBUTED-SYSTEMS.NETS, used with permission)

20.2.2 Distributed Operating Systems (DiOSs)

A distributed operating system manages all distributed resources and services of the distributed system. In a distributed system environment, the operating systems running on the various nodes appear to the user and applications as a single operating system. More importantly, the distributed nature of the system is unknown to the processes that are running on the various nodes. Distributed operating systems allow users to access remote resources and request distributed services, in the same manner, they access local resources and request local services. Users use the same set of operations to access remote resources, as well as local resources.

It is imperative to state that distributed operating systems differ from network operating systems in many ways. In a network operating system, the user is aware of the multiplicity of nodes, resources, and services, and each node is aware of the network. Conversely, a distributed operating system hides this multiplicity of nodes, resources, and services from the user, as well as the network. Typically, a distributed operating system aggregates the distributed system resources to produces a single-system image that hides the heterogeneous and distributed nature of the system and gives the illusion that a single operating system controls the network [3, 4, 6]. According to Khapre et al. [7], the extent to which the user is aware of the multiple nodes or computers that make up a distributed system is what distinguishes distributed operating systems from network operating systems [7]. Figure 20.2 [8, 9] shows the general structure of a distributed operating system in which the middleware layer provides the same interface for the applications running on the various nodes.

Unlike a network operating system, a distributed operating system can transfer data and computation across the various nodes while hiding the transfer details from users and applications. A process transfer is also possible when there is a need to balance the load in response to high-demand computing. Although distributed systems offer many advantages, the heterogeneous and sprawling nature of these systems has a direct impact on their security [4, 10]. Additionally, the distributed operating system must consider three primary areas of transparency. These areas include execution, file system, and protection [4]. Hence, securing and protecting distributed systems require a pervasive, yet transparent, approach to security and protection. In such an approach, the distributed operating system plays a critical role in protecting distributed resources from unauthorized access and securing them against malicious or accidental destruction or alteration.

20.3 DiOSs Implementations

There are several implementations of distributed operating systems, some of which date back to the late 1970s [11]. In 1970, IBM introduced the VM distributed operating system, which enabled administrators to create several virtual

machines on its System/370 mainframe [12]. Some of these early DiOSs supported homogeneous environments, while others supported heterogeneous environments. Some provided a varying degree of transparency, while others offered full transparency. Some supported distributed applications, while others executed large distributed computations. For example, Roscoe (1970s) ran on homogeneous processors, while Cronus (1980s) supported heterogeneous environments. Saguaro (the mid-1980s) supported varying degrees of transparency while MOSIX (1980s–1990s), a UNIX based operating system, supported full transparency and dynamic process migration for load balancing. SODA (the mid-1980s) ran distributed applications while CONDOR (1980s) distributed computations among several processors.

20.4 Cloud-Based Systems

20.4.1 Cloud Operating System (COS)

Advances in computing and virtualization technology, coupled with the need to design a distributed operating system that presents a single system image and consumes fewer system resources have led to a new class of distributed operating systems that are lightweight but yet effective. Hence, COS is a new class of operating systems that aims at improving resource utilization. It provides a unified view of distributed computing resources while maintaining a transparent cloud-based system. A cloud-based system is a collection of distributed, interconnected, and virtualized nodes dynamically provisioned and presented as one or more unified computing resources by a service provider [13]. Virtualization is the main characteristic of a cloud-based system. A typical cloud-based system consists of "compute" servers, data servers, and workstations where the workstations, servers, networks, and storage are virtual entities running on single physical nodes [14, 15].

COSs are conceptually-centralized, browser-based operating systems that make use of the single system image (SSI) paradigm approach to aggregate data and compute resources and present a transparent but yet unified view of the system [5, 14]. COS SSI implements a hypervisor-level of abstraction to aggregate virtual resources to gives an illusion that the user is interacting with a single system. Unlike traditional OS that multiplex multiple processes or threads, COS is fully multiplexed by the virtualization [16]. Modern DiOSs or COSs include Chromium OS, CloudLinux OS, Azure, Amazon EC2, Eucalyptus, Qubes OS, SlapOS, OSv, MeghaOS, and Harvey OS, to name a few. There are several COSs currently available that enable the data and application to exist and run on the Internet, such as Amoeba, Glide, Kohive, Cloudo, myGoya, Zimdesk, Ghost, Mirage, XOS, eyeOS and OpenStack Cloud [14, 17, 18]. While some of these COSs are proprietary, others are either open source or experimental prototypes. In both cases, some COSs are easily discernible as operating systems; others are difficult to separate from the service platform, i.e., the line between the cloud as an operating system and a cloud as a service is scrupulously unclear.

In addition to the traditional OS features and functions, COS specific features and functions include managing of the network, compute and storage, management of virtual machine (VM) life-cycle, management of VM image, management of security, management of remote cloud capacity, pure programming abstractions, robust isolation techniques between users and applications, robust integration with network resources, dynamic placement of multitier services on distributed infrastructure, definition of security policy on the users, dynamic creation of and movement of VM and associated storage, management of workload placement, and smooth execution of VM [14, 18, 19]. Figure 20.3 [19] shows a logical model of COS.

COSs extend the function of security and protection of traditional OS to provide a multi-level distributed security

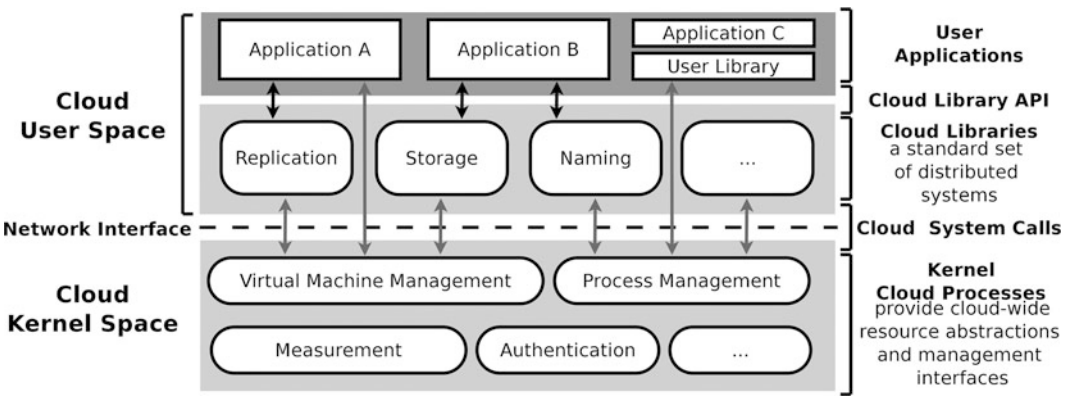

Fig. 20.3 A logical model of Cloud OS, featuring the division between Cloud kernel/Cloud user space and the system call and library API interfaces (Source: IEEE/IFIP, p. 339)

and protection function. Although COSs have many functions and offer several benefits, the security and protection of COSs remain a significant concern. The distributed nature and openness of cloud-based systems, virtualization, and existing vulnerabilities at the hardware, database, applications, and operating system levels are invariably threats to cloud-based systems, hence COSs security.

20.5 DiOS Security Issues

20.5.1 Traditional DiOS

Typically, a secure distributed operating system is one that secures the distributed system information and resources from accidental and malicious activities and protects them against unauthorized access. Therefore, a secure operating system must employ security and protection mechanisms to defend from internal and external attacks, control access to information and resources, and distinguish between authorized and unauthorized use [3]. Distributed systems are conceptually centralized but physically decentralized.

Decentralization raises serious security, privacy, and trust issues. A secure operating system must preserve user privacy and ensure user authenticity, provide robust authorization, and secure communication between distributed services [13]. In addition to openness, the heterogeneity of nodes in a distributed system is another concern. The desire to have a set of heterogeneous nodes with various degree of assurances of their security features can lead to some security problems [20]. In particular, it constrains the DiOS security features at the expense of the individual nodes' security features.

Since a distributed system is built on top of a set of single nodes connected by a network, it is essential to distinguish between the network and distributed operating system security. In general, a distributed operating system supports distributed applications by identifying and implementing functions that are common to most distributed applications and providing these functions to applications during runtime as services. These runtime services are available at the higher layers of the OSI model, typically above the transport layer.

A common practice among software developers is to write a covert method (trapdoor) that allows them to bypass authentication mechanisms. A DiOS likely has several trapdoors. Inevitably, covert methods induce vulnerabilities in the operating system, hence, exposing it to intentional, as well as unintentional exploitation. Vulnerabilities are not necessarily exclusive to operating systems; they also exist in hardware, network, database systems, and applications [21]. While some vulnerabilities may cause a potential for an attack, others cause no harm. In all cases, an adversary must not know of vulnerabilities that are unknown to the user.

20.5.2 Contemporary DiOS

A COS is a web-based software stack that manages virtual resources of a cloud-based system created over physical nodes based on service-level agreements between the user and service provider. In a multi-tenant model, a virtual server generates virtual resources as isolated virtual nodes for users to use, and virtual storage enables these virtual nodes to store their data. While virtualization is one of the essential characteristics of a cloud-based system, it poses serious security challenges as it can be exploited by attackers whose goal is to compromise the virtual node [22–26]. Therefore, a secure COS must employ security mechanisms that ensure strong resource isolation, mediated sharing, and secure communication, and further, help protect administrator credentials and the virtualization layer [15, 27]. Stealing identify and administrator credentials, launching malicious software, and compromising the virtualization layer are some of the most common attack vectors. Potential attack vectors include denial of service attacks, cloud malware injection attack, side-channel attacks, authentication attacks, and man-in-the-middle cryptographic attacks [22]. Other attacks include a control-flow attack on tenant OS, a virtualization-based attack on the hypervisor, kernel rootkits attack, and cross VM attacks [23]. The attack surface of a COS is the sum of all attack vectors and existing vulnerabilities.

In a typical cloud-based system, there are three different classes of participants: service instance, cloud user, and cloud provider and six different attack surfaces: service-to user, user-to-service, cloud-to-service, service-to-cloud, cloud-to-user, and user-to-cloud [22, 23, 28]. In general, virtualizing physical nodes, pooling shared resources, allowing multi-tenancy, and enabling access to the same physical resources by multiple services increase the attack surface. A vulnerability exploits in a cloud-based system can cause data-leakage, denial of service (DoS), or violation of privacy [25, 29]. Some examples of vulnerability exploits reported by Common Vulnerabilities and Exposure system (CEV) include a KVM (kernel-based virtual machine) hypervisor flaw on how it handles the guest machine specific registers, a vulnerability in the Oracle VM VirtualBox component of Oracle virtualization, and a memory leak in Xen 4.2 hypervisor. The former and latter exploits can cause a denial of service attack (DoS), while the Oracle virtualization vulnerability exploit can allow a low privileged attacker to compromise the VM [30]. The CVE system contains a database of the most recent cloud vulnerabilities.

20.5.3 DiOS Security and Protection

Whether it is for enterprises or end-users, a secure COS is one that protects the virtualization layer or fabric of the

cloud-based system, isolates virtual nodes running on the same physical node, control access to physical resources by multiple services, and secure communication between services and virtual nodes. Also, a secure COS has well-defined security policies on users, as well as workload or services isolation and service-level-agreements, to control the use of the configurable pool of cloud resources [14, 15, 19]. To minimize the attack surface and prevent possible exploits, the COS must patch zero-day vulnerabilities, as well as other measures used by administrators to secure on-premise OS. According to [31], the same practices used to secure OSs running in customers' data centers can be used to secure the OS running in the cloud utilizing a securely configured templates of OS known as gold images to simplify the process. Such practices include hardening OS, applying the latest patches, installing endpoint-based antivirus, and deploying IDS/IPS and firewalls.

Depending on the cloud platform developer and the specific COS, several security and protection mechanisms exist. To ensure the security of the virtualization layer and overcome the vulnerabilities of a VM, [25] proposes an approach that distinguishes malicious VMs from valid ones. The mechanism, called security supervisor, requires each virtual node in the virtualization environment to send requests to receive virtual resources. It then validates each node identity to ensure the node is non-malicious before it grants the requested resource.

To minimize the OS attack surface, the Google Cloud Platform uses a minimal operating system footprint approach by trimming unnecessary packages [32]. The container-optimized OS running on its cloud platform has a minimal footprint, immutable root filesystem, verified boot, stateless configuration, security-hardened kernel, security-centric default, and automatic updates. To manage access to VM images, the container-optimized OS provides a mechanism called instance access that allows for fine-grained access control instead of user accounts.

To eliminate the hypervisor attack surface, Szefer et al. [26] propose the elimination of the hypervisor layer by allowing VMs to run directly on the underlying hardware instead of hardening or minimizing the virtualization software. In their proposed NoHype architecture, the VMs are allowed to run directly on the underlying hardware without a hypervisor. The ITRI COS implementation in [15] has built-in properties to provide security protection. These built-in properties provide multi-tenant isolation, role-based access control, distributed protection, and DDoS mitigation, among others. Microsoft Azure delivers integrated security policies to control access and a Just-In-Time (JIT) VM to lockdown in-bound traffic and to reduce the VM exposure to attacks [33]. Additionally, It uses a machine-learning adaptive application control mechanism to control which application can run on the VM. Further, it provides a file integrity monitoring (FIM) mechanism to track and identify any changes to the virtualization environment that might indicate an attack.

The Center for Internet Security developed CIS hardened images on shielded virtual machines; a preconfigured virtual machine images based on the CIS benchmark security recommendations [34]. CIS images protect the VMs against advanced threats. Ensuring workloads are trusted and verifiable, protecting secrets against exfiltration and reply, and offering live migration and patching.

TrustZone [35], is a hardware-based solution that provides hardware isolated execution domain. It allows the isolation of system resources such that they are only accessible via the trusted environment without the need to include the hypervisor in a trusted computing base. TrustZone enables the operation of two entirely separate threads: a secure thread and a general thread. The secure thread has full privileges to access all the system resources. However, the general thread cannot access the resources accessed by the secure thread. In addition to those mentioned above, there are other strategies and techniques used to address security and protection at the various levels (application, kernel, hypervisor, virtualization layers) of a cloud-based system that we are planning to consider in the future.

While it is intuitive to implement a secure and reliable COS, highly secure COSs by themselves are insufficient for securing and protecting the cloud-based system. Marinescu [16] argues that a highly secure COS and application-specific security are both necessary, and a better approach to security is to implement security above the operating system.

20.6 Conclusion and Future Work

COS, a class of distributed operating systems, provides a computing environment for dynamically creating, provisioning, and managing virtual resources and workload over the web. It aggregates and pools configurable virtual resources and presents them to multiple virtual nodes using a unified view while maintaining transparency and efficient resource utilization. Many nodes can run on one physical machine, and multiple services can access a single shared resource. Virtualization, transparency, openness, and heterogeneity are the main characteristics of COSs.

Virtualization abstracts and controls access to the physical resources to allow the same services to run in parallel on multiple heterogeneous and connected but yet isolated nodes irrespective of their affinity. Nonetheless, it introduces many vulnerabilities and gives rise to several security threats at the level of application, kernel, virtualization, and hypervisor layers. Consequently, a COS must extend traditional security and protection functions to offer a global, multi-level security and protection strategies and mechanisms.

In this paper, we discussed distributed operating systems and their aspects, as well as issues related to their overall security and protection. We focused on a class of modern distributed systems known as cloud operating systems or COSs. Subsequently, we identified various COS implementations, discussed their functions and features, defined their attack surfaces, and highlighted several security and protection issues. Further, we discussed several strategies and mechanisms used by various COSs to secure and protect cloud-based systems from unauthorized access, as well as internal and external threats. This survey is not exhaustive and remains a work in progress. In the future, we are planning to expand our current review and use the framework in the article [23] to classify, define, and evaluate current security and protection mechanisms for COSs.

References

1. Kshemkalyani, A., Singhal, M.: Distributed Systems Principles, Algorithms, and Systems. Cambridge University Press, New York (2008)
2. Steen, M.V., Tanenbaum, A.S.: A brief introduction to distributed system. Computing. **98**, 967–1009 (2016)
3. Silberschatz, A., Galvin, P.B., Gagne, G.: Operating System Concepts, 8th edn. Wiley, Hoboken (2009)
4. The Univerity of Edinburgh: https://www.inf.ed.ac.uk/teaching/courses/ds/slides1516/OS.pdf
5. Healy, P., Lynn, T., Barrett, E., Morrison, J.P.: Single System Image. J. Parallel Distrib. Comput. **90–91**, 35–51 (2016)
6. Puder, A., Römer, K., Pilhofer, F.: Distributed Systems Architecture: A Middleware Approach. Morgan Kaufmann, Burlington (2006)
7. Khapre, S., Jean, J., Amudhavel, J., Chandramohan, D., Sujatha, P., Narasimhulu, V.: Survey on distributed operating systems: a real time approach. Int. J. Comput. Sci. Emerg. Technol. **1**(2), 109–123 (2010)
8. Steen, M.V., Tanenbaum, A.S.: Distributed Systems, 3rd ed. Maarten van Steen. Distibuted-systems.net (2017)
9. DISTRIBUTED-SYTEMS.NETS: Maarten Van Steen. https://www.distributed-systems.net/
10. Coulouris, G.F., Dollimore, J., Kindberg, T., Blair, G.: Distributed Systems: Concepts and Design. Addison-Wesley, New York (2012)
11. Wichita State University: http://www.cs.wichita.edu/chang/lecture/cs843/homework/dist-os.html
12. IBM: https://www.ibm.com/cloud/blog/cloud-computing-history
13. Pathan, A.S.K., Pathan, M., Lee, H.Y.: Advancements in Distributed Computing and Internet Technologies. Trends and Issues. IGI Global, Hershey (2012)
14. Kumar, O., Goel, V., Rai, D.: Cloud as an evolutionary operating system. IJCA J. ICNICT. **6**, 11–14 (2012)
15. Chiueh, T., Chang, E.J., Huang, R., Lee, H., Sung, V., Chiang, M.H.: Security considerations in ITRI cloud OS. In: International Carnahan Conference on Security Technology (ICCST), pp. 107–112. IEEE, New York (2015)
16. Marinescu, D.: Cloud Computing Theory and Practice. Morgan Kaufmann, Waltham (2013)
17. Bardhan, N., Singh, P.: Operating system used in cloud computing. Int. J. Comput. Sci. Inform. Technol. **6**(1), 542–544 (2015)
18. Chandra, D.G., Malaya, D.B.: A study on cloud OS. In: International Conference on Communication Systems and Network Technologies (CSNT), pp. 692–697. IEEE, New York (2012)
19. Pianese, F., Bosch, P., Duminuco, A., Janssens, N., Stathopoulos, T., Steiner, M.: Toward a cloud operating system. In: IEEE/IFIP-Network Operations and Management Symposium Workshops, pp. 335–342. IEEE, New York (2010)
20. Casey, T.A., Vinter, S.T., Weber, D.G., Varadarajan, R., Rosenthal, D.: A secure distributed operating system. In: IEEE Symposium on Security and Privacy, pp. 27–38. IEEE, New York (1988)
21. Bhargava, B., Lilien, L.: Vulnerabilities and threats in distributed systems. In: 1st International Conference on Distributed Computing and Internet Technology, ser. ICDCIT, pp. 146–157. Springer-Verlag, New York (2004)
22. Singh, A., Shrivastava, D.M.: Overview of attacks on cloud computing. Int. J. Eng. Innov. Technol. **1**(4), 321–323 (2012)
23. Sgandurra, D., Lupu, E.: Evolution of attacks, threat models, and solutions for virtualized systems. J. CSUR. **48**(3), 1–38 (2016)
24. Pék, G., Buttyán, L., Bencsáth, B.: A survey of security issues in hardware virtualization. J. CSUR. **45**(3), 1–34 (2013)
25. Uday Kumar, N.L., Siddappa, M.: Ensuring security for virtualization in cloud services. In: International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT), pp. 248–251. IEEE, New York (2016)
26. Szefer, J., Keller, E., Lee, R.B., Rexford, J.: Eliminating the hypervisor attack surface for a more secure cloud. In: 18th ACM Conference on Computer and Communications Security (CCS), pp. 401–412, New York. ACM (2011)
27. Microsoft: https://download.microsoft.com/download/6/7/3/673E651E-C5B3-4C93-A69A-94042EB6DE22/Windows_Server_2016_Security_Better_protection_begins_at_the_OS_Whitepaper_EN_US.pdf
28. Gruschka, N., Jensen, M.: Attack surfaces: a taxonomy for attacks on cloud services. In: 3rd International Conference on Cloud Computing, pp. 276–279. IEEE, New York (2010)
29. Gkortzis, A., Rizou, S., Spinellis, D.: An empirical analysis of vulnerabilities in virtualization technologies. In: IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pp. 533–538. IEEE, New York (2016)
30. Common vulnerabilities and exposure. https://cve.mitre.org/
31. VMware and SAVVIS: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/whitepaper/cloud/vmware-savvis-cloud-white-paper-en.pdf
32. Goole Cloud: https://cloud.google.com/container-optimized-os/docs/concepts/security
33. Microsoft Azure: https://docs.microsoft.com/en-us/azure/security-center/tutorial-protect-resources
34. Center for Internet Security: https://www.cisecurity.org/cis-hardened-image-list/
35. Pettersen, R., Johansen, H.D., Johansen, D.: Secure edge computing with ARM TrustZone. In: 2nd International Conference on Internet of Things, Big Data and Security, pp. 102–109. SCITEPRESS-Science and Technology Publications, Setúbal (2017)