Tanzania Journal of Engineering and Technology 2024, **43**(2):92 - 107
OPEN ACCESS articles distributed under Creative Commons Attribution Licence [CC BY-ND]
Websites: https://ajol.org/tjet; https://tjet.udsm.ac.tz

Copyright © 2024 College of Engineering and Technology, University of Dar es Salaam
ISSN 1821-536X (**print**); ISSN 2619-8789 (**electronic**)
https://doi.org/10.52339/tjet.v43i2.925

Regular Research Manuscript

Assessment of Web Security Vulnerabilities for Common Open Source Virtualization Software

Said Ally

ICT Department, The Open University of Tanzania, P.O. Box 23409, Dar es Salaam, Tanzania
Corresponding Author: said.ally@out.ac.tz, saidallymasomaso@gmail.com
[†]ORCID: https://orcid.org/0000-0002-4419-6004

ABSTRACT

Open-source hypervisors have emerged as an integral technology for virtualizing server resources in cloud and data center computing. Hypervisor security efficiency is determined by virtual machine isolation, which is a de facto adoption factor in the selection process, as well as its ability to respond to web attacks. This paper assesses the security performance of Proxmox VE and XenServer for type 1 hypervisors, and Kernel Virtual Machine and Oracle Virtual Box for type 2 hypervisors. Security analysis was conducted using common exposures extracted from vulnerability databases and mapped against the OWASP 2013 and 2017 projects. For clarity, experiments were carried out on a testbed with prebuilt virtual machines, each hosting one hypervisor installed as an attack target. Kali Linux was configured in one virtual machine to run recursive penetration testing for information gathering, vulnerability detection, penetration attempts, and exploitation of weak spots. The infrastructure was set in both homogeneous and heterogeneous execution environments, with a series of tests nested with each other. All four hypervisors are vulnerable to physical kernel isolation, as unprivileged users can gain root access and launch guest-to-guest and host-to-guest attacks. Among the two, guest-to-guest attacks were found to be more common than host-to-guest attacks, indicating that virtual machine isolation is weaker than the underlying host. Type 1 hypervisors have a lower rate of host-to-guest attacks than guest-to-guest attacks, implying that XenServer and Proxmox VE provide better isolation than KVM and OVB due to the near-native speed, security, and efficiency of their virtual machines. All four hypervisors were found to be vulnerable to buffer overflow exploits and error-triggering sensitive information leaks, which were primarily caused by adopter default misconfigurations in the deployment process rather than software design flaws. This implies that greater efforts are required by open-source adopters when shifting from physical to virtual computing.

ARTICLE INFO
Submitted: **May 14, 2023**

Revised 1: **Nov. 27, 2023**

Revised 2: **May 15, 2024**

Accepted: **July 15, 2024**

Published: **Aug. 2024**

Keywords: *Virtualization, Open source, Hypervisor, Web security vulnerabilities, and Virtual machine*

INTRODUCTION

Virtualization has transformed how computing infrastructure is managed in the IT industry, particularly in processing, memory, storage, and networking. Through this technique, a physical server is partitioned into several partly or completely isolated virtual machines (VMs) using software known as a hypervisor. A hypervisor is an apparatus that creates, controls, and manages VMs. It is available as type 1 (native or bare metal), which runs directly on hardware with a shorter kernel compilation time, or type 2 (hosted), which runs as an application guest (guest OS) on top of the host operating system (Popek & Goldberg, 1974; Morabito et al., 2015; Mishra and Mishra, 2016), as indicated in Figure 1.

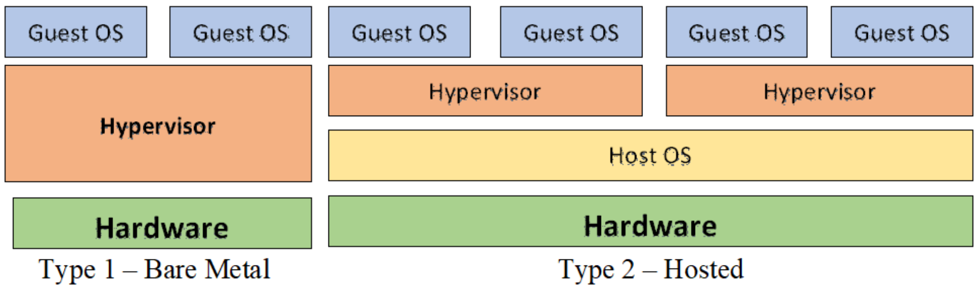

Figure 1: Structure of hypervisor type 1 and 2 (Popek, & Goldberg, 1974).

Virtualizing server resources is popular because of its various benefits, including significant cost savings in personnel, space, power, and cooling. Using open-source software (OSS) solutions in the virtualization process can lead to additional reductions for adopters (Ghapanchi et al., 2011; Bridge, 2018). According to Tiemann (2009), the OSS can save the ICT industry about $1 trillion each year, taking advantage of their General Public License (GPL-GNU) which permits free software access and source code modification.

The hypervisor, as an integral virtualization component, has been vulnerable to security attacks since its inception (Anumukonda et al., 2021). Historically, attacks have been technologically advanced because of the proliferation of Internet services, tablets, IoT devices, cloud computing, and social media platforms, as indicated in Table 1.

Table 1: The classification of cybercrimes over the decades (Jimmy, 2024)

SN	Period	Type of Cybercrimes
1	1940s	Years without computer crime
2	1950s	Phone phreaking decade
3	1960s	Hacking and vulnerability terms appear
4	1970s	Born of computer security
5	1980s	The years of ARPANET to Internet
6	1990s	Computer viruses and worms have become popular
7	2000s	The Internet grows excessively
8	2010s	Cybercriminals discover several security breaches in computer systems
9	2020s	Cybercrimes have become and industry

Cybercrimes targeting virtual platforms deny access to mission-critical infrastructure. In cloud computing, security threats are real (Shaikh and Meshram, 2021; Montasari et al., 2021), as they have expanded dramatically since 2020, with the main cause being hypervisor vulnerabilities (Jimmy, 2024). In open-source domain, critical vulnerabilities exploited as attack targets have been accelerating at an exponential rate (Hong et al., 2022). Depending on how the adoption process is managed and the design and configuration of the hypervisor, virtualization in an open-source environment can be viewed as both an opportunity and a threat to adopters (Ally et al., 2018).

In terms of design, all open-source virtual systems are considered vulnerable to fault-error attacks (Cerveira et al., 2020), with around 29% having at least zero-day exploits (Sonatype, 2021). Most design vulnerabilities originate from third-party libraries and dependencies (Imtiaz et al., 2021). For misconfigurations, the problem has become very common in recent years, with reports showing an increase of 650% by 2021 (Sonatype, 2021), mainly due to insider attacks (Jartelius, 2020; Krishnan et al., 2023). Since hypervisors provide a single point of failure in a virtual environment, all VMs become vulnerable to a variety of internal and external sources, including other VMs, the underlying hypervisor, the server administrator, or the Internet (Arif and Shakeel, 2015). The most common vulnerabilities in VMs include mobility, hypervisor intrusion, and Denial of Service (DoS) attacks (Hyde, 2009); unauthorized VM communication; VM alteration and hopping; hypervisor hyper-jacking; unsecure VM migration; malicious code injection; side channel attacks; and virtualization sprawl (Ramana et al., 2015; Mahjani, 2015). Other hypervisor attacks include modification, isolation breakage, multi-tenancy issues in a VM environment, and data compliance issues across multiple VMs (Rachana and Guruprasad, 2014; Wueest, 2014; Obasuyi and Sari, 2015; Nazir and Lazarides, 2016), trojanized prebuilt VMs, improperly configured virtual firewalls and hypervisors, and data leakage through offline images (Yauri and Abah, 2016).

Since the conception of virtualization, about 60% of businesses have relied on vulnerability detection and management (Ghelani, 2022), depending on the severity of the attack and hypervisor isolation (Chen et al., 2023). While one-third of users on global virtual servers store sensitive data (Singh et al., 2016), about 75% of attacks occur remotely (Verizon, 2017). Despite the availability of cutting-edge security tools and techniques, hypervisor attacks and cloud data breaches continue to occur (Thales, 2018). Thus, assessing the security design of the most common open-source hypervisors and their configuration features is crucial, given that VMs are created by different virtualization software with varying levels of isolation and strength when responding to web attacks.

METHODS AND MATERIALS

Selected Hypervisors

The study is designed to assess the security vulnerabilities of four open-source hypervisors: Proxmox VE (Proxmox VE, 2016; 2021; Goldman, 2016) and XenServer (XenServer, 2017; 2018) are type 1 hypervisors, while KVM (KVM, 2021; Hirt, 2010; Kiszka, 2010; Chirammal et al., 2016) and OVB (OVB, 2021) are type 2 hypervisors. These are the most popular and commonly used open-source virtualization solutions (Anwer et al., 2010; Kulkarni et al., 2012; Obasuyi and Sari, 2015). The platforms are freely distributed under the GNU-GPL open-source license, with source code and bugs available and fixed by the user community. Table 2 shows the hypervisor profiles indicating name, version, and release date for each type.

Table 2: Hypervisor profiles

SN	Hypervisor	Version	Year	Type
1	Kernel Virtual Machine	2.6.20	2007	2
2	Oracle VirtualBox	6.1.20	2021	
3	Proxmox VE	7.1	2021	1
4	XenServer	7.0	2016	

Vulnerability Analysis

The vulnerability analysis of the four hypervisors was carried out in two stages. The first stage applied secondary data retrieved from a classical web-based national vulnerability database (NVD), a publicly available U.S. government repository (NVD, 2021). The database uses a unique identifier system (Mitre, 2024) referenced as Common Vulnerabilities and Exposure (CVE). An application programming interface (API) was used to search for vulnerabilities using relevant keywords such as hypervisor names (*Proxmox VE, XenServer, KVM, and Oracle Virtual Box*), and the top ten web attacks reported in the Open Web Application Security Project (OWASP) lists for 2013 and 2017. Only vulnerabilities that matched all keywords were accepted as correct CVEs.

Furthermore, the detected vulnerabilities were rated based on their severity level, with a focus on those with the most critical impact (Walkowski et al., 2021). For clarity, vulnerabilities that are not directly related to hypervisors were ignored, regardless of how they affected other layers of the virtual execution environment (Parast et al., 2022). Table 3 shows 16 thematic areas selected as the most prevalent vulnerabilities based on the defined criteria, such as those that overlap in both OWASP 2013 and 2017 lists, those deemed common web attacks, and design and configuration flaws in hypervisors or VMs.

Table 3: List of web security vulnerabilities [OWASP, 2013; 2017]

WSV 01-16	**Web Security Vulnerabilities (WSV)**	**OWASP Top 10 Lists**	
		2013	**2017**
WSV01	Buffer Overflow Exploits	A6	-
WSV02	CGI-BIN Parameter Manipulation	A6	A2
WSV03	Form/hidden Field Manipulation	-	-
WSV04	Forceful Browsing	A8, A10	-
WSV05	Cookie/Session Poisoning	A2	A2
WSV06	Broken Access Control Lists (ACLs)/Weak Passwords	A2, A7	A5
WSV07	Cross-Site Scripting (XSS)	A3	A7
WSV08	Command (SQL) Injection/Worm Attack	A1	A1
WSV09	Error Triggering Sensitive Information Leaks	A6	A3
WSV10	Insecure Use of Crypto	A5	-
WSV11	Server Misconfiguration	A5	A6
WSV12	Back Doors and Debug Options	A7	-
WSV13	Website Defacement	A7, A8, A10	-
WSV14	Well-Known Platform Vulnerabilities	A9	A9
WSV15	Zero-Day Exploits	-	-
WSV16	Man in the Middle Attack	-	-

According to OWASP (2013, 2017), the selected vulnerabilities are considered the most prominent web security issues that ideally affect VM performance in cloud systems. Buffer overflow exploits, form/hidden field manipulation, insecure use of crypto, back doors and debug options, zero-day exploits, and man in the middle attacks are among the vulnerabilities covered in the analysis, despite not being on the OWASP top ten lists for 2013 and 2017. The sonatype software composition analysis tool was used to detect and retrieve vulnerabilities from the database (Sonatype, 2024).

Design of the Test Lab

The second stage involved a series of experiments in which all CVE records were screened to determine how each vulnerability was associated with the hypervisor. Experiments were conducted through running penetration tests to identify the vulnerable source files, the source of attacks, and the access methods. This is a high-coverage approach, considering hidden vulnerabilities that may not be captured using matching keywords alone (Hong et al., 2022).

A virtual test lab was used as the optimum test environment since vulnerability analysis of open-source code can be executed dynamically in real-time (Ghelani et al., 2022). A physical testbed was configured with an Intel® CoreTM i7-8565 CPU@1.80GHz, a 1.99GHz x64-based processor, 16 GB of usable RAM, and a 64-bit operating system. Hypervisors were installed as pre-built VMs to serve as attack targets. One VM was left for Kali Linux, configured to run penetration tests that determine isolation strength among VMs and between VMs and their underlying hypervisors.

Kali Linux is cross-platform, well-suited to virtual execution environments, and capable of simultaneously detecting and attacking weak spots (Nazir and Lazarides, 2016). Hypervisor and VM profiles at various stages of penetration testing were created using NMAP (2023), OpenVAS (2023), OWASP-ZAP (2023), and Metasploit (2023) for information gathering, vulnerability detection, penetration attempts, and exploitation of weak points. These Kali Linux security tools are popular and freely available for vulnerability analysis. The selected tools are useful for dynamic vulnerability analysis as they are of an open-source nature (Mogicato & Zermin, 2023).

Penetration tests were conducted in both homogeneous and heterogeneous infrastructures. In each scenario, tests were recursive and repeated three times as specified in the algorithm. Kali Linux was configured as an attacking machine with an IP address of 10.10.10.10. In each hypervisor, a maximum of 10 VMs were created and assigned IP addresses as follows:

- Proxmox VE was assigned an IP address of 10.10.10.50, and all its VMs ranged from 10.10.10.51 to 10.10.10.59.
- XenServer was assigned 10.10.10.60, with all its VMs ranging from 10.10.10.61 to 10.10.10.69.
- KVM with 10.10.10.20 and all its VMs ranging from 10.10.10.21 to 10.10.10.29.
- OVB at 10.10.10.30 and all its VMs range from 10.10.10.31 to 10.10.10.39.

Because guest machines differ in the installed OS in a typical computing architecture, all security aspects linked to the OS were viewed as extraneous factors and thus disregarded in the analysis. Figure 2 shows the structural design of the test lab.

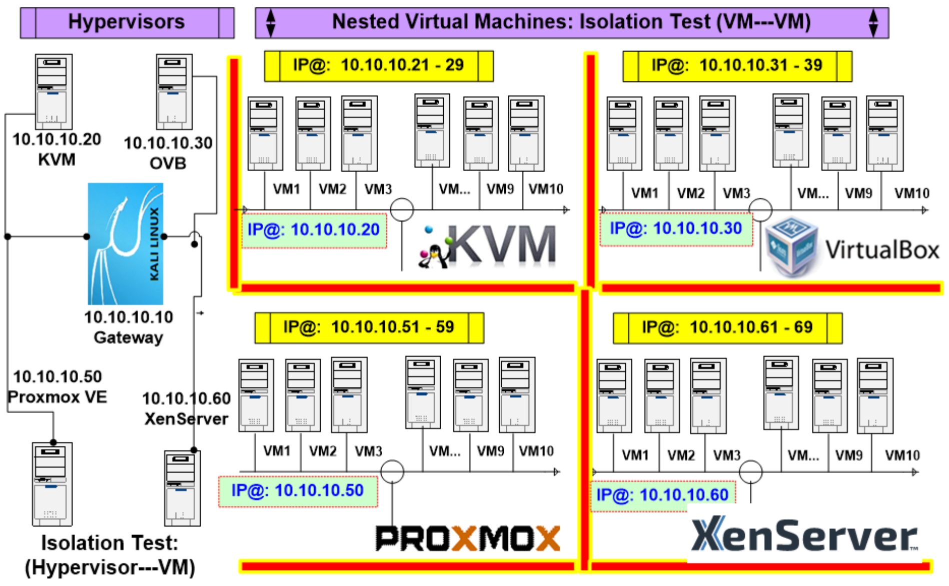

Figure 2. Penetration testing design virtual lab.

A testing algorithm with 13 sequential steps guided the vulnerability analysis of the four hypervisors.

1. *Select **open source-based type 1 and type 2 hypervisors**: **HPV$_1$, HPV$_2$, HPV$_3$, HPV$_4$.***
2. *Design a **penetration test lab** using an open-source virtual execution environment.*
3. *Create **N** virtual machines in each **HPV$_j$***
4. *Analyze potential web vulnerabilities from **online databases** and the **OWASP** Top 10.*

5. *Rank and prioritize the most prominent vulnerabilities on VMs based on severity levels.*
6. *Run recursive penetration tests (**PT_1, PT_2, PT_3**) for each **HPV_j**.*
7. *Collect data results for each **PT_j**.*
8. *Change the nesting level for each **HPV_j**.*
9. *Repeat steps 6, 7, and 8 for **PT_j** and **HPV_j**.*
10. *Perform vulnerability analysis for data from all **PT_j** for each **HPV_j***
11. *Rate and rank the penetration level between **G2G** and **H2G** for each **HPV_j**.*
12. *Analyze hypervisor design faults and security misconfigurations.*
13. *Suggest an acceptable degree of isolation among VMs for each **HPV_j**.*

RESULTS AND DISCUSSION

Hypervisor and Web Security Vulnerability

Essentially, the findings show that security vulnerabilities differ between hypervisors. XenServer was found vulnerable to eight (8) attacks (53.3%), followed by Proxmox VE and KVM, each with seven (7) attacks (46.7%), and OVB, with two (2) attacks (13.3%).
Further analysis revealed that buffer overflow exploits and error-triggering sensitive information leaks are common web vulnerabilities across all hypervisors. Except for OVB, all other hypervisors are vulnerable to CGI-BIN parameter manipulation, XSS, cookie/session poisoning, and form/hidden field manipulation. Table 4 summarizes the vulnerabilities that arise in each hypervisor.

Table 4: **Web Security Vulnerabilities for OSS based hypervisors**

SN	Vulnerability	Open Source Hypervisors				Total (Yes)
		Type 1		Type 2		
		Proxmox VE	XenServer	KVM	OVB	
1	WSV01	Yes	Yes	Yes	Yes	**4**
2	WSV09	Yes	Yes	Yes	Yes	**4**
3	WSV02	Yes	Yes	Yes	No	**3**
4	WSV07	Yes	Yes	Yes	No	**3**
5	WSV05	Yes	Yes	Yes	No	**3**
6	WSV03	Yes	Yes	Yes	No	**3**
7	WSV06	Yes	Yes	No	No	**2**
8	WSV08	No	Yes	No	No	**1**
9	WSV11	No	No	Yes	No	**1**
10	WSV04	No	No	No	No	**0**
11	WSV10	No	No	No	No	**0**
12	WSV12	No	No	No	No	**0**
13	WSV13	No	No	No	No	**0**
14	WSV14	No	No	No	No	**0**
15	WSV15	No	No	No	No	**0**
	Total	**7**	**8**	**7**	**2**	**24**

Proxmox VE

According to the analysis, Proxmox VE is vulnerable to buffer overflow exploits, error-triggering sensitive information leaks, CGI-BIN parameter manipulation, XSS, cookie/session and secure socket layer (SSL) poisoning, form/hidden field manipulation, and broken ACLs/weak passwords. Although the buffer overflow exploit is not considered a critical vulnerability, if the function *exitcode_proc_write* of file *arch/um/kernel/exitcode.c* in the Linux kernel is not properly configured, local users can gain root privileges and perform write operations on the written data to overrun the buffer size and overwrite adjacent memory locations, resulting in data loss and DOS attacks.
The software provides for insecure hostname checks as it does not validate the

user-supplied POST parameter. It allows authenticated remote users to overwrite configuration file settings using a form created in the VM. This implies that a hacker can easily gain access to the host filesystem and login credentials to execute commands that expose root privileges. The vulnerability is basically a CGI-BIN parameter manipulation of data exchanged between a browser and a web application (tampering with cookies, form fields, URL strings, and HTTP headers), which allows for a high-risk remote attack. Insecure hostname checking also leads to an XSS vulnerability, especially when the LXC.pm and OpenVZ.pm files are poorly configured. Through this vulnerability, remote attackers can inject arbitrary web and client-side (malicious/payloads) scripts executable in a visitor's browser via multiple parameter file extensions (.htm) using social engineering techniques.

Proxmox VE is also vulnerable to broken ACLs and weak passwords during the authentication process, primarily caused by a login error. When the username and password are sent to the server as an AJAX request, the information can be leaked through message system feedback. Through login failure, server responses such as "Username does not exist" for an incorrect username or "Authentication failed" for the wrong password reveal sensitive information that an attacker can use to deduce whether a username exists or not. Other possible vulnerabilities in Proxmox VE include cookie/session poisoning despite the usage of SSL and TLS for network encryption, as well as secure API implementation in the file *net/sched/act_api.c* of the Linux kernel. While cookie poisoning results in identity theft, SSL spoofing or poisoning allows for man-in-the-middle and DOS attacks via uninitialized memory access and system halts or crashes.

XenServer

Analysis shows that all vulnerabilities found in Proxmox VE also affect XenServer. The command/SQL injection was found as an additional vulnerability that affects XenServer, specifically the login and configuration files like login.php, config/writeconfig.php, and include/config.ini.php.

Through this vulnerability, a remote attacker can execute arbitrary SQL commands and PHP code via the username parameter. Attackers can also use the hypervisor to run arbitrary code in HVM graphics console support to cause a buffer overflow exploit and DOS attack. Through cookie and session poisoning, network connections can be established to execute arbitrary commands, craft certificates, and impersonate servers to bypass authentication. This happens when the hypervisor fails to enforce access policies because of broken ACLs and weak passwords. The password can also be changed through a cross-site request forgery attack in the file *config/changepw.php* triggered by the XenServer resource kit. Changing server-side policies can also be caused by CGI-BIN parameter manipulation when local users execute unspecified API calls to bypass authentication, modify the guest virtual hard disk, and stop VMs from functioning.

For XSS attacks, the vulnerable part was the http interface of the API file, through which remote attackers can inject arbitrary web scripts via unspecified vectors. The potential vulnerable web scripts where username and location-related parameters can be manipulated include config/edituser.php, sessionid, vmname, console.php, forcerestart.php, vmrefid, and forcesd.php. Remote attackers can hijack the authentication of privileged users by injecting arbitrary and static PHP code into files *include/config.ini.php* and *config/writeconfig.php*.

KVM

Analysis shows that KVM and Proxmox VE have nearly similar vulnerabilities except that Proxmox VE is subject to broken ACLs and weak passwords, whereas KVM is sensitive to server misconfiguration. Server misconfiguration becomes critical when KVM is nested in levels 0 and 1 and obtains read and write access to the hardware, especially when the file *arch/x86/kvm/vmx.c* fails to control registers.
A buffer overflow exploit was spotted in the *arch/x86/kvm/x86.c* and *virt/kvm/irq_comm.c* files when a privileged guest user can call the kvm_set_irq function on the host OS. This vulnerability causes DOS attacks, memory corruption, invalid memory copies, large memory allocations, and out-of-bounds reads and writes. The *kvm.cgi* file is vulnerable to arbitrary web script injection caused by the modification of the CGI-BIN parameter and XSS attacks. A remote attacker can decrypt https sessions via SSL poisoning due to a failure in the verification of server certificates, the presence of a hardcoded SSL private key, and the usage of stolen user keys in the KVM subsystem. Attackers can sniff network traffic, spoof servers, and read and modify data.
Other vulnerable KVM files that can cause error-triggering sensitive information leaks include *arch/x86/kvm/emulate.c*, *arch/x86/kvm/x86.c*, and *virt/kvm/kvm_main.c*. Information leakage in KVM is mainly a timing error caused by a failure to check if kernel addresses are set at the time of memory allocation in the physical address space. This results in a DOS attack that primarily affects service initialization and virtual CPU resources.

OVB

Major vulnerabilities found in OVB are buffer overflow exploits and errors triggering sensitive information leaks. Buffer overflow is caused by boundary errors in the core hypervisor subcomponent, especially when the length of user-supplied data is not properly validated. In the process of OVB installation, a local attacker was able to escalate privileges on VMs by executing arbitrary low-privileged code. This can be interpreted to imply that the attacker can execute code at the hypervisor level, which is far more serious.
Nesting OVB at level 1 revealed vulnerabilities in both VMs and core hypervisor subcomponents as it allows read and write operations to inaccessible local files. An attacker can gain remote access and affect the http protocol, share folders, and GUI components. Through this vulnerability, malicious local users can get access to the system and launch DoS attacks.

Attack Methods and Sources

Vulnerability sources and access methods for web attacks are crucial in determining the security of VMs and their underlying hypervisors. According to the analysis, the hypervisor design flaws and the adopter misconfigurations are the major vulnerability sources. The study also shows that the number of vulnerable source files varies between hypervisors. Table 5 shows critical insights about each hypervisor, including web vulnerabilities, access points, and sources of attacks.
Software design faults were found in 11 attacks, and adopter misconfigurations in 13 attacks. Along with the attack point, 12 attacks were local and 14 were remote based. Analysis shows that the Proxmox VE, XenServer, and KVM have nearly equal numbers of vulnerabilities and the same average for attack point and attack source, indicating slightly similar security strengths as shown in Figure 3.

Table 5: Summary of web security vulnerabilities for hypervisors

Hypervisor	Web Vulnerabilities						Total
	Overall Attack Point			Weak Point (Source)			
	Local	Remote	Both	Configuration	Design	Both	
Proxmox VE	2	4	1	4	3	0	14
XenServer	2	5	1	5	3	0	16
KVM	4	3	0	4	3	0	14
OVB	2	0	0	0	2	0	4
Total	10	12	2	13	11	0	48

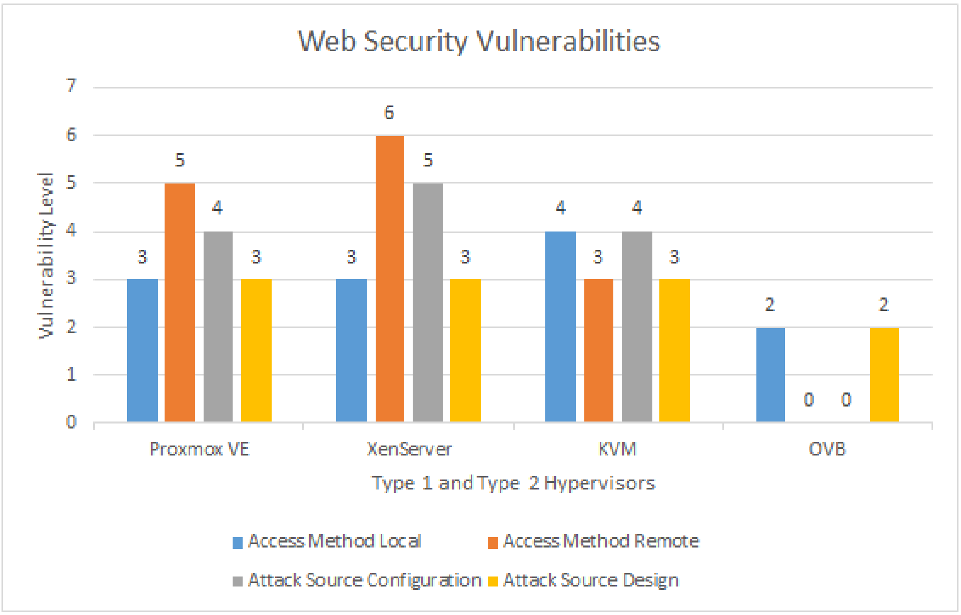

Figure 3: Impact level of web security vulnerability for each hypervisor.

Specifically, OVB has the fewest vulnerabilities, KVM has the most local attacks, and XenServer has the most remote attacks. Although OVB appears to be less vulnerable, being a type 2 hypervisor makes it less secure than Proxmox VE and XenServer, which are both type 1 hypervisors. For web vulnerabilities, OVB outperforms KVM, while Proxmox VE is slightly more secure than XenServer. The findings also reveal that some of the attacks on type 1 hypervisors can be launched both locally and remotely. Major hypervisor issues stem slightly equally between misconfigurations and design flaws. All hypervisors were found equally vulnerable to CGI-BIN parameter manipulation, form/hidden field manipulation, cookie/session/SSL/TLS poisoning, and XSS attacks.

Guest-to-Guest and Host-to-Guest Attacks

Analysis shows that in all hypervisors, there is a possibility of guest-to-guest (G2G) and host-to-guest (H2G) being infection layers. The isolation level as a key determinant factor in establishing the level of G2G and H2G attacks varies among hypervisors. In all hypervisors, there is a possibility of a host attack from the guest machine, resulting in the breakout of physical kernel isolation, as shown in Figure 4.

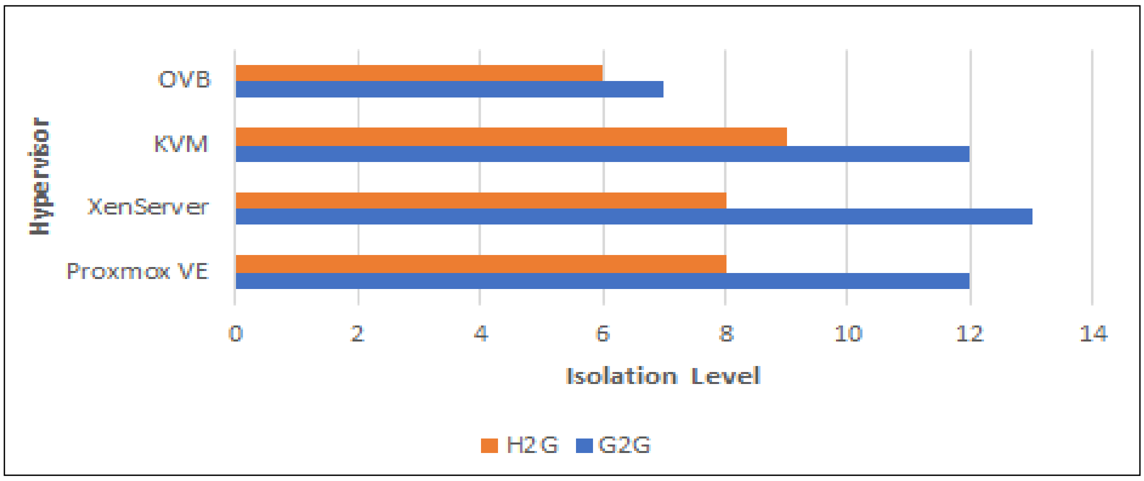

Figure 4: Isolation level for G2G and H2G in four hypervisors.

The buffer overflow exploit was found to be the most prevalent across all the hypervisors, with a noticeable impact on both G2G and H2G attacks. Error-triggering sensitive information leaks also affect all hypervisors except Proxmox VE for H2G penetrations. Vulnerabilities such as forceful browsing, insecure use of crypto, backdoors and debug options, website defacement, well-known platform vulnerabilities, and zero-day exploits have no significant impact on G2G and H2G attacks. This explains why these vulnerabilities are not included in the OWASP top 10 lists for 2013 and 2017.

Analysis also shows that the rate of G2G attacks is higher than H2G attacks in all four hypervisors, indicating that penetration attacks between VMs occur more frequently than between VMs and their underlying hosts, mainly caused by adopter misconfiguration. This is clearly demonstrated in Figures 5 and 6.

In terms of type 1 hypervisors, Proxmox VE had 46.7% G2G penetration and 20% H2G penetration. For XenServer, G2G penetration was 53.3% and H2G penetration was 26.7%, indicating that both hypervisors are more secure on H2G than G2G, making them more resistant to host penetration from any guest machine. For type 2 hypervisors, G2G penetration in KVM is 20% higher than H2G penetration. The biggest concern with KVM is server misconfiguration, implying that KVM adopters pay little attention to hypervisor setups; otherwise, KVM can offer the same level of security as a type 1 hypervisor, taking advantage of the Linux kernel despite being a type 2 hypervisor. For OVB, both G2G and H2G attacks had a penetration rate of 13.3%.

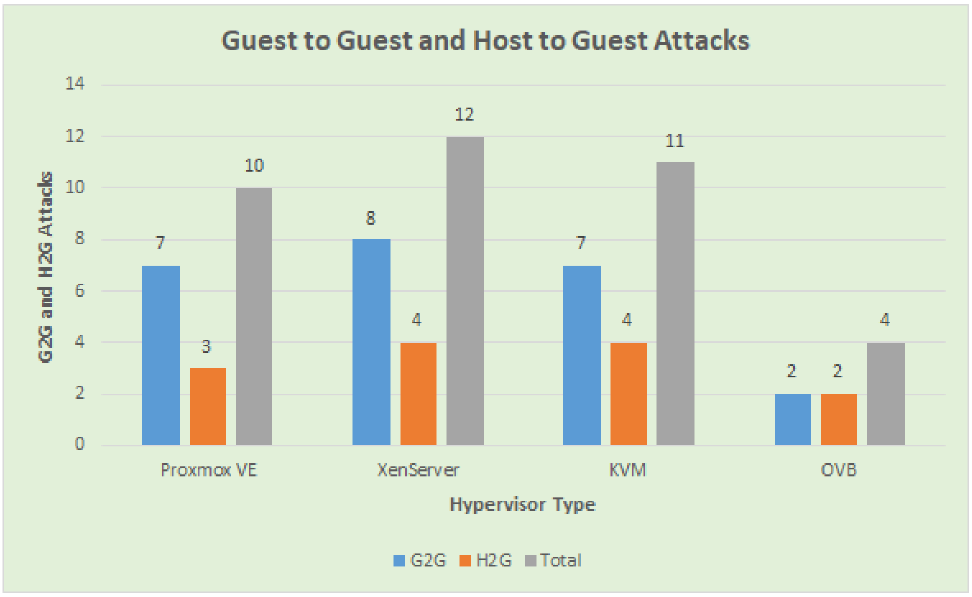

Figure 5: Overall impact level for web security vulnerability.

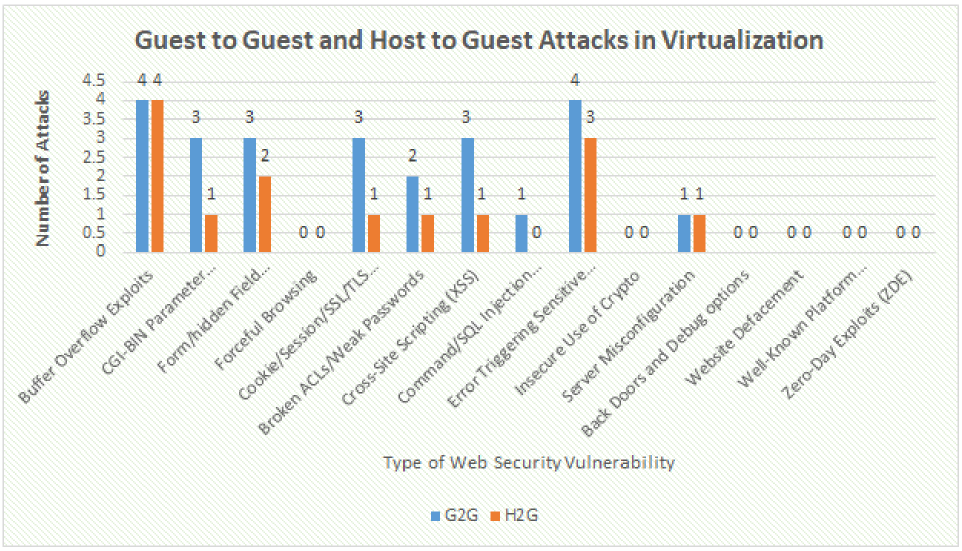

Figure 6: Guest to guest and host to guest attacks based on web security vulnerability.

Discussion

Since they are not created equal, the security of hypervisors varies. For example, code injection inside one guest VM in XenServer can hardly propagate to other nearby VMs or hypervisors due to the high level of isolation. The possibility of H2G attacks in all four hypervisors conforms to the study by Bazm et al. (2017) and Cheng et al. (2018), which focused on distributed side-channel attacks and the breakout of physical kernel isolation. Isolation violations in virtualized environments occur intra-process, inter-process, user-kernel, inter-VM, and VM-hypervisor, with major issues being hypervisor design flaws or adopter misconfigurations.

Type 1 hypervisors are more secure than type 2 hypervisors, which depend solely on the security status of the host OS. Type 2 hypervisors suffer from the effect of the host OS overhead when accessing computing resources. Regardless of the security settings and the type of hypervisor, if major design flaws are not well handled by software vendors, the software can be at high risk of attack. Common challenges to address throughout the deployment process include the choice of hypervisor type, software maturity status, number of supported VMs, size of vCPU, vRAM, vHDD, and vNIC, and compatibility with host and guest OS. Hypervisor security is also affected by other critical extraneous factors, including the technology compatibility (hardware, network, and system software resources), license type, adoption strategy to guide and enforce policies for deployed VMs, and homogeneous and heterogeneous infrastructure setup. When a virtualized server is loaded with unnecessary hardware-specific drivers or prebuilt open-source applications, the risk of compromising VM security increases.

Hypervisor security performance is also affected by design parameters such as virtualization method, size of computing workloads, virtual resource allocation, and the order in which VMs access resources. Placing all high-CPU-demand VMs on the same host can jeopardize the performance of other VMs by preventing them from getting resources. For instance, dynamic resource allocation is a good practice for balancing processing speed, memory, and storage requirements, but it can also be detrimental to VMs running mission-critical applications by leaving them resource-starved. Control of virtual resources can also be looked at in terms of how the default configurations are administered. For example, a hypervisor can experience less resource consumption, which is vital for VM performance, when non-GUI server cores are used and can disable the connection of peripheral devices to the host.

Security configurations also depend on the level of VM nesting, VM states (dormant or active), and how the hypervisor can disable, remove, or hide unwanted interfaces, windows, ports, devices, and services for guest machines. For example, in some cases, for security reasons, the adopter may be forced to power off the VM when copying a VM image or performing a backup and create separate volumes for each VM to reduce the number of disk I/O operations. Any misconfiguration of VMs causes processing delays, especially when they are registered for DHCP on wireless networks and their files are stored on drives used by the OS. In this scenario, system processes run continually on the drives where the system files are kept. Adopters need to be alert when performing the undo or reverse process in virtual systems to avoid re-exposing previously patched vulnerabilities.

Security misconfiguration can cause guest unavailability, particularly when there is a mismatch between the hypervisor and guest operating systems, VM migration is not controlled, authorizations are incorrect, and failover and scale-up strategies are not implemented. Proper mapping between virtual and physical devices is vital to ensuring that guest operating systems are associated with the appropriate host system and that an access path between VMs is not created.

As with any other web system, general security considerations are vital for hypervisor performance, especially when it comes to physical and logical security, hardware and network security, system security, and application security. Given that open-source technology is reliant on patching, adopters should ensure that the source code is obtained from reliable and trusted sources at all phases of development. This allows for the separation of privileged and administrative functions in backup and crash plans, the choice of static and unique IP addresses, communication encryption for HTTPS, TLS, and SSH protocols, and VPN services for the host OS. This is in line with the recommendations of Cheng et al. (2018), which describe a strategic process for resolving server configuration issues.

CONCLUSION

Despite the security threats, attacks, and vulnerabilities connected with hypervisor virtualization, it is evident that the technology is gaining popularity around the world. This study serves as a valuable resource for businesses considering the shift from physical to virtual computing. The choice of an appropriate and secure hypervisor that is free of vulnerabilities and capable of responding to web attacks is crucial throughout the virtualization process.

Analysis of the four hypervisors reveals a considerable level of security breach among VMs due to guest-to-guest attacks rather than host-to-guest attacks between VMs and their underlying hypervisor. While most attacks are remotely executed, there is a likelihood that the security loopholes are mainly the result of the adopter's misconfiguration rather than design faults. Thus, for companies to benefit from virtualized open-source computing infrastructure, a security assessment is critical during the adoption process. The study is valuable for open-source adopters of hypervisor virtualization since it considers all design and configuration factors.

Recommendations

While server virtualization is an undeniable technology due to its use in cloud and data center computing, adopters need to be conscious of hypervisor security given its open-source nature and widespread use. For a smooth and secure adoption and use of open-source hypervisor virtualization, the following are key security recommendations based on technology and management dimensions:

- Adopters should focus on software design attributes such as hypervisor type, maturity level, future technology trends, virtualization method, license type, market ranking, usability and ease of use, virtual resource limits, patch management, external libraries and support, relationships with vendors, security threats, exploits, and innovation risk, as well as isolation level among VMs and between VMs and their underlying host.
- Adopters should focus on configuration attributes, especially computing infrastructure and workloads, resource balancing, prioritization, allocation, utilization, and limits through controlling preconfigured default settings and trusted grouping based on risk level.
- Adopters should consider industry-accepted best practices and security guidelines for the virtualization technology adoption framework, with a focus on adopter profile, external compliance, laws and regulations, virtualization roles and functions, technology compatibility for host and guest OS and VM image formats, internal and external reviews by performing SWOC analysis, business process reviews, feasibility studies, and a secure migration plan for a physical-to-virtual conversion.

REFERENCES

Ally, S., Jiwaji, N. T., & Tarimo, C. (2018). A Review of Adopter’s Common Misconfigurations of Virtual Machines: The Case of Tanzania. *Huria: Journal of the Open University of Tanzania*, **25**(2): 158-180.

Anwer, M. B., and Nayak, A., Feamster, N., & Liu, L. (2010). Network I/O Fairness in Virtual Machines, ACM journal. In Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures, pp. 73-80. ACM.

Anumukonda, N. S. K., Yadav, R. K., & NS, R. (2021). A Painstaking Analysis of Attacks on Hypervisors in Cloud Environment. In *2021 6th International Conference on Machine Learning Technologies* (pp. 150-157).

Arif, M. & Shakeel, H. (2015). Virtualization security: Analysis and open challenges. International Journal of Hybrid Information Technology (IJHIT). ISSN: 1738-9968. **8**(2):237-246. http://dx.doi.org/10.14257/ijhit.2015.8.2.22

Bazm, M. M., Lacoste, M., Südholt, M., & Menaud, J. M. (2017). Side-Channels Beyond the Cloud Edge: New Isolation Threats and Solutions. In *IEEE International Conference on Cyber Security in Networking (CSNet),* October *2017*.

Bridge, R., (2018). Open source software: Advantages & disadvantages. *Enterpreneur*

Handbook. https://entrepreneurhandbook.co.uk/open-source-software/, Date Accessed: 13-July-2018

Cerveira, F., Barbosa, R., Madeira, H., & Araujo, F. (2020). The effects of soft errors and mitigation strategies for virtualization servers. *IEEE Transactions on Cloud Computing*, **10**(2): 1065-1081.

Chen, J., Li, D., Mi, Z., Liu, Y., Zang, B., Guan, H., & Chen, H. (2023). Security and Performance in the Delegated User-level Virtualization. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)* (pp. 209-226).

Cheng, Y., Zhang, Z., & Nepal, S. (2018). Still Hammerable and Exploitable: on the Effectiveness of Software-only Physical Kernel Isolation. *arXiv preprint arXiv:1802.07060*.

Chirammal, H. D., Mukhedkar, P., & Vettathu, A. (2016). *Mastering KVM virtualization*. Packt Publishing Ltd.

Ghapanchi A. H., Aurum A., Low G. (2011). A taxonomy for measuring the success of open source software projects, *First Monday*, **16**(8): https://doi.org/10.5210/fm.v16i8.3558

Ghelani, D., Hua, T. K., & Koduru, S. K. R. (2022). Cyber security threats, vulnerabilities, and security solutions models in banking. *Authorea Preprints*.

Ghelani, D. (2022). Cyber security, cyber threats, implications, and future perspectives: A Review. *Authorea Preprints*.

Goldman, R. (2016). *Learning Proxmox VE*. Packt Publishing Ltd.

Hirt T., (2010). KVM–the kernel-based virtual machine, 2010 *Red Hat Inc*.

Hong, H., Woo, S., Choi, E., Choi, J., & Lee, H. (2022). xVDB: A high-coverage approach for constructing a vulnerability database. *IEEE Access*, **10**: 85050-85063.

Hyde, D. A Survey on the Security of Virtual Machines. 2009. Available online: http://www.cse.wustl.edu/~jain/cse571-09/ftp/vmsec/index.html.

Imtiaz, N., Thorn, S., & Williams, L. (2021). A comparative study of vulnerability reporting by software composition analysis tools. In *Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pp. 1-11.

Jartelius, M. (2020). The 2020 Data Breach Investigations Report–a CSO's perspective. *Network Security*, **2020**(7): 9-12.

Jimmy, F. N. U. (2024). Cyber security Vulnerabilities and Remediation Through Cloud Security Tools. *Journal of Artificial Intelligence General Science*, **2**(1):129-171.

Kali (2022). Kali Linux. Website: https://www.kali.org/, Date Accessed: 11-05-2022.

Kiszka, J., T DE IT, C. T., & Linux, C. C. C. E. (2010). Architecture of the kernel-based virtual machine (kvm). In *Siemens Availability: http://www. linux-kongress. org/2010/slides/KVM-Architecture-LK2010.pdf*.

Krishnan, P., Jain, K., Aldweesh, A., Prabu, P., & Buyya, R. (2023). OpenStackDP: a scalable network security framework for SDN-based OpenStack cloud infrastructure. *Journal of Cloud Computing*, **12**(1):26.

Kulkarni, O., Bagul, S., Gawali, D., & Swamy, P. (2012). Virtualization technology: A leading edge. *International Journal of Computer Application*, **2**(2): ISSN: 2250-1797.

KVM., (2021). Kernel Virtual Machine - The Open Source Type 2 Hypervisor" https://www.linux-kvm.org/, Date Accessed: 11-02-2021.

Mahjani, M. (2015). Security issues of virtualization in cloud computing environments. Master Thesis for award of Master of Science in Information Security degree at Lulea University of Technology, Sweden. 62pp.

Metasploit (2022). Metasploit penetration testing framework. Website: https://www.metasploit.com/, Date Accessed: 11-05-2023.

Mishra, A. & Mishra, R. (2016). Virtualization security. *Global Research and Development Journal for Engineering*, **1**(12): 20 – 24.

Mitre (2024). Mitre CVE Database, https://cve.mitre.org/. Date Accessed:11-03-2024.

Mogicato, R., & Zermin, A. (2023). Design and Implementation of a Collaborative, Lightweight Malware Analysis Sandbox using Container Virtualization. MSc. Dissertation, University of Zurich, Switzerland.

Montasari, R., Macdonald, S., Hosseinian-Far, A., Carroll, F., & Daneshkhah, A. (2021). Network and hypervisor-based attacks in cloud computing environments. *International Journal of Electronic Security and Digital Forensics*, **13**(6): 630-651.

Morabito, R., Cozzolino, V., Ding, A. Y., Beijar, N., & Ott, J. (2018). Consolidate IoT edge computing with lightweight virtualization. *IEEE Network*, **32**(1): 102-111.

Nazir, S. & Lazarides, M. (2016). Securing industrial control systems on a virtual platform: how to best protect the vital virtual business assets. *White Paper*, FIRSTCo

NMAP (2023). Network Mapper. Website: https://nmap.org/, Date accessed: 11-03-2023.

NVD (2021). "National vulnerability database", https://nvd.nist.org, Date Accessed: 10-03-2021

Obasuyi, G. C & Sari, A. (2015). Security challenges of virtualization hypervisors in virtualized hardware environment. International Journal of Communications, Network and System Sciences, **8**(7): 260-273. http://dx.doi.org/10.4236/ijcns.2015.87026

OpenVAS (2022). Open Vulnerability Assessment Scanner. Website: https://www.openvas.org/, Date Accessed: 11-05-2023.

OVB, (2021). Oracle Virtual Box - The Open Source Type 2 Hypervisor. https://www.virtualbox.org, Date Accessed: 13-05-2021

OWASP (2013). "The *Open Web Application Security Project*" Top 10 of 2013, https://owasp.org/, https://owasp.org/www-project-top-ten/, Date Accessed: 10-03-2021

OWASP (2017). "The *Open Web Application Security Project*" Top 10 of 2017, https://owasp.org/ and https://owasp.org/www-project-top-ten/, Date Accessed: 10-03-2021

OWASP-ZAP (2023). OWASP Zed Attack Proxy. Website: https://www.kali.org/tools/zaproxy/, Date Accessed: 14-04-2023

Parast, F. K., Sindhav, C., Nikam, S., Yekta, H. I., Kent, K. B., & Hakak, S. (2022). Cloud computing security: A survey of service-based models. *Computers & Security*, **114**: 102580.

Popek, G. J. & Goldberg, R. P. (1974). Formal requirements for virtualizable third generation architectures Communications of ACM, **17**(7): 412-421.

Proxmox VE Server Solutions (2016), https://www.proxmox.com/, software version 4.4, Date Accessed: 03-01-2017

Proxmox VE, (2021). The Open Source Type 1 Hypervisor, https://www.proxmox.com/en/proxmox-ve, Date Accessed: 16-05-2021

Rachana, S. C. & Guruprasad, H. S. (2014). Securing the virtual machines. *International Journal of Computer Technology & Applications* (IJCTA), ISSN: 2229 – 6093, **5**(3): 1012-1019

Ramana, V. V., Reddy, Y. S., Reddy, G. R. S., and Ravi, P. (2015). An assessment of virtual machineassails. *International Journal of Advanced Technology in Engineering and Science*, www.ijates.com, ISSN: 2348 – 7550, **3**(1): 315–320.

Shaikh, A. H., & Meshram, B. B. (2021). Security issues in cloud computing. In *Intelligent Computing and Networking* (pp. 63-77). Springer, Singapore.

Singh, H., Manhas, P., Maan, D., & Sethi, N. (2016). Cloud computing security and privacy issues–A systematic review. *International Science Press*, **9**(11): 4979–4992.

Sonatype (2024). Sonatype OSS Index Software Composition Analysis tool. https://ossindex.sonatype.org/, Date Accessed: 18-04-2024.

Sonatype. (2021). *State of the Software Supply Chain*. [Online]. Available: https://www.sonatype.com/resources/state-of-the-software-supply-chain-2021

Thales, (2018). 2018 Global threat report, *451 Group for Thales, 2018 Thales Data Threat Report - Global Edition*, https://dtr.thalesesecurity.com/, Date Accessed: 13-07-2018

Tiemann M., (2009). How open source software can save the ICT industry one trillion dollars per year. http://www.opensource.org/files/OSS-2010.pdf, Date Accessed: 13-07-2018.

Verizon, (2017). 2017 Data breach investigations report, 10th Edition, https://www.ictsecuritymagazine.com/wpcontent/uploads/2017-Data-Breach-Investigations-Report.pdf, Date Accessed: 13-07-2018

Walkowski, M., Oko, J., & Sujecki, S. (2021). Vulnerability management models using a common vulnerability scoring system. *Applied Sciences*, **11**(18):8735.

Wueest, C. (2014). Threats to virtual environments. *Security Response, Symantec*, version 1, pp. 18

XenServer Hypervisor, (2017). Xen Hypervisor-Software version 4.8, https://xenproject.org/, Date Accessed: 04-01-2017

XenServer, (2018). The Open Source Type 1 Hypervisor https://xenserver.org, 2018

Yauri, B. A., & Abah, J. (2016). Mitigating security threats in virtualized environments. *International Journal of Computer Science and Network Security*, **16**(1): 101.