**Abstract**

Cloud Computing is increasingly becoming popular as many enterprise applications and data are moving into cloud platforms. However, a major barrier for cloud adoption is real and perceived lack of security. In order to better define these threats to which a cloud hypervisor is exposed, we conducted an in-depth analysis and highlighted the security concerns of the cloud. We basically focused on the two particular issues, i.e., (a) data breaches and (b) weak authentication. For in-depth analysis, we have successfully demonstrated a fully functional private cloud infrastructure running on Cloud Stack for the software management and orchestrated a valid hack. We analysed the popular open-source hypervisors, followed by an extensive study of the vulnerability reports associated with them. Based on our findings, we propose the characterization and countermeasures of hypervisor’s vulnerabilities.

|  |  |  |
| --- | --- | --- |
| **1.** | **INTRODUCTION** |  |
|  | * 1. **Overview**   Cloud computing is fast becoming a popular option for renting of computing and storage infrastructure services (called Infrastructure as a Service or IaaS) ; for remote platform building and customization for business processes (called Platform as a Service or PaaS) ; and for renting of business applications as a whole (called Software as a Service or SaaS). There is no doubt that the convenience and low cost of cloud computing services have changed our daily lives; however, the security issues associated with cloud computing make us vulnerable to cybercrimes that happen every day. Hackers employ a variety of techniques to gain access to clouds without legal authorization or disrupt services on clouds in order to achieve specific objectives. Hackers could trick a cloud into treating their illegal activity as a valid instance, therefore, gaining unauthorized access to the information stored in the cloud. Once the exact location of data is located, hackers steal private and sensitive information for criminal activities. According to DataLossDB, there were 1,047 data breach incidents during the first nine months of 2012, compared to 1,041 incidents during the entire year of 2011. Epsilon and Stratfor were two data breach victims. In the data leakage accident, Epsilon leaked millions of names and email addresses from the customer databases. Stratfor’s 75,000 credit card numbers and 860,000 user names and passwords were stolen . Hackers could also take advantage of the massive computing power of clouds to fire attacks to users who are in the same or different networks. For instance, hackers rented a server through Amazon’s EC2 service and carried out an attack to Sony's PlayStation Network . Therefore, a good understanding of cloud security threats is necessary in order to provide more secure services to cloud users. |  |
|  | * 1. **Problem Definition**   In the past, hackers used multiple computers or a botnet to produce a great amount of computing power in order to conduct cyber-attacks on computer systems. This process is complicated and can take months to complete. Nowadays, a powerful computing infrastructure, including both software and hardware components, could be easily created using a simple registration process in a cloud computing service provider. By taking advantage of the prevailing computing power of cloud networks, hackers can fire attacks in a very short time. For example, brute force attacks and DoS attacks can be launched by abusing the power of cloud computing. A brute force attack is a technique used to break passwords. The success of this attack is greatly reliant on powerful computing capability because thousands of possible passwords are needed to be sent to a target user’s account until it finds the correct one to access. Cloud computing system provides a perfect platform for hackers to launch this type of attack. Thomas Roth, a German researcher, demonstrated a brute force attack in the Black Hat Technical Security Conference. He managed to crack a WPA-PSK protected network by renting a server from Amazon’s EC2. In approximately 20 minutes, Roth fired 400,000 passwords per second into the system and the cost of using EC2 service was only 28 cents per minute. DoS attacks attempt to disrupt a host or network resource in order to make legitimate users unable to access the computer service. They come in a variety of forms and aim at a variety of services. Generally, they are categorized into three basic types: consumption of scarce, limited, or non-renewable resources, destruction or alteration of configuration information, and physical destruction or alteration of network components . Among them, flooding is the most common way in which hackers crumble the victim’s system with the use of an overwhelming number of bogus requests; therefore, the services to legitimate users are blocked. When the flooding attack is applied to cloud services, two types of DoS could happen in cloud computing systems: direct DoS and indirect DoS . When a cloud server receives a large volume of flooded requests, it will provide more computational resources to cope with the malicious requests. Finally, the server exhausts its full capability and a direct DoS is occurred to all requests from legitimate users. Moreover, the flood attack could possibly cause indirect DoS to other servers in the same cloud when the servers share the workload of the victim server, which results a full lack of availability on all of the services. |  |
| **2.** | **LITERATURE SURVEY 1**  **Securing cloud hypervisors: a survey of the threats, vulnerabilities, and countermeasures ,**  Barrowclough, J. P., & Asif, R.  *Security and Communication Networks,* (2018)  The exponential rise of the cloud computing paradigm has led to the cybersecurity concerns, taking into account the fact that the resources are shared and mediated by a ‘hypervisor’ that may be attacked and user data can be compromised or hacked. In order to better define these threats to which a cloud hypervisor is exposed, we conducted an in-depth analysis and highlighted the security concerns of the cloud. We basically focused on the two particular issues, i.e., (a) data breaches and (b) weak authentication. For in-depth analysis, we have successfully demonstrated a fully functional private cloud infrastructure running on CloudStack for the software management and orchestrated a valid hack. We analyzed the popular open-source hypervisors, followed by an extensive study of the vulnerability reports associated with them. Based on our findings, we propose the characterization and countermeasures of hypervisor’s vulnerabilities. These investigations can be used to understand the potential attack paths on cloud computing and Cloud-of-Things (CoT) applications and identify the vulnerabilities that enabled them  **LITERATURE SURVEY 2**  **Secure virtualization for cloud environment using hypervisor-based technology**  Sabahi, F, *International Journal of Machine Learning and Computing,* (2012)  In this paper, I propose virtualization architecture to secure cloud. In the proposed architecture, I try to reduce the workload, decentralize security-related tasks between hypervisor and VMs, and convert the centralized security system to a distributed one. The distributed security system is a very good way to reduce the workload from hypervisor-based virtualization, but this distribution may inject vulnerabilities to cloud. In addition, distributed security systems have more complexity than centralized ones. Because of several benefits, such as the fault-tolerant International Journal of Machine Learning and Computing, Vol. 2, No. 1, February 2012 capability, of distributed security management, it is not possible to ignore it and persist on centralized managing, but it is important to use a distributed management unit with care warily. Actually, in cloud there are lot users and their application that are running but security is important for all of them. The cloud must work properly and creates an immune environment against attacks, no matter what application is running on the cloud. In the computer world, anything makeable is breakable, however. In addition, cloud is an Internet-based technology, and but building root-of-trust cloud systems seemed impossible. Therefore, it seems main area of concern in cloud is security and cloud providers will face innumerable vicissitudes when their cloud become bigger than now.  **LITERATURE SURVEY 3**  **Eliminating the hypervisor attack surface for a more secure cloud**  Szefer, J., Keller, E., Lee, R. B., & Rexford, J. (2011, October)  In *Proceedings of the 18th ACM conference on Computer and communications security* (pp. 401-412).  Cloud computing is quickly becoming the platform of choice for many web services. Virtualization is the key underlying technology enabling cloud providers to host services for a large number of customers. Unfortunately, virtualization software is large, complex, and has a considerable attack surface. As such, it is prone to bugs and vulnerabilities that a malicious virtual machine (VM) can exploit to attack or obstruct other VMs -- a major concern for organizations wishing to move to the cloud. In contrast to previous work on hardening or minimizing the virtualization software, we eliminate the hypervisor attack surface by enabling the guest VMs to run natively on the underlying hardware while maintaining the ability to run multiple VMs concurrently. Our NoHype system embodies four key ideas: (i) pre-allocation of processor cores and memory resources, (ii) use of virtualized I/O devices, (iii) minor modifications to the guest OS to perform all system discovery during bootup, and (iv) avoiding indirection by bringing the guest virtual machine in more direct contact with the underlying hardware. Hence, no hypervisor is needed to allocate resources dynamically, emulate I/O devices, support system discovery after bootup, or map interrupts and other identifiers. NoHype capitalizes on the unique use model in cloud computing, where customers specify resource requirements ahead of time and providers offer a suite of guest OS kernels. Our system supports multiple tenants and capabilities commonly found in hosted cloud infrastructures. Our prototype utilizes Xen 4.0 to prepare the environment for guest VMs, and a slightly modified version of Linux 2.6 for the guest OS. Our evaluation with both SPEC and Apache benchmarks shows a roughly 1% performance gain when running applications on NoHype compared to running them on top of Xen 4.0. Our security analysis shows that, while there are some minor limitations with cur- rent commodity hardware, NoHype is a significant advance in the security of cloud computing.  **LITERATURE SURVEY 4** Architectural support for secure virtualization under a vulnerable hypervisor Jin, S., Ahn, J., Cha, S., & Huh, J. (2011, December)  In *2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (pp. 272-283). IEEE.  Although cloud computing has emerged as a promising future computing model, security concerns due to malicious tenants have been deterring its fast adoption. In cloud computing, multiple tenants may share physical systems by using virtualization techniques. In such a virtualized system, a software hypervisor creates virtual machines (VMs) from the physical system, and provides each user with an isolated VM. However, the hypervisor, with a full control over hardware resources, can access the memory pages of guest VMs without any restriction. By compromising the hypervisor, a malicious user can access the memory contents of the VMs used by other users. In this paper, we propose a hardware-based mechanism to protect the memory of guest VMs from unauthorized accesses, even with an untrusted hypervisor. With this mechanism, memory isolation is provided by the secure hardware, which is much less vulnerable than the software hypervisor. The proposed mechanism extends the current hardware support for memory virtualization with a small extra hardware cost. The hypervisor can still flexibly allocate physical memory pages to virtual machines for efficient resource management. However, the hypervisor can update nested page tables only through the secure hardware mechanism, which verifies each mapping change. Using the hardware-oriented mechanism in each system securing guest VMs under a vulnerable hypervisor, this paper also proposes a cloud system architecture, which supports the authenticated launch and migration of guest VMs.  **LITERATURE SURVEY 5**  **Securing cloud computing systems**  Gurkok, Cem   In *Computer and Information Security Handbook*, pp. 897-922. Morgan Kaufmann, 2017  Cloud computing is a method of delivering computing resources. Cloud computing services, ranging from data storage and processing to software such as customer relationship management systems, are now available instantly and on demand. In times of financial and economic hardship, this new low-cost ownership model for computing has received lots of attention and is seeing increasing global investment. Generally speaking, cloud computing provides implementation agility, lower capital expenditure, location independence, resource pooling, broad network access, reliability, scalability, elasticity, and ease of maintenance. While in most cases cloud computing can improve security due to ease of management, the lack of knowledge and experience of the provider can jeopardize customer environments. This chapter discusses various cloud computing environments and methods to make them more secure for hosting companies and their customers. |  |
| **3.** | **SYSTEM ANALYSIS** |  |
|  | **2.1 Existing System**  Cloud service models not only provide different types of services to users but they also reveal information which adds to security issues and risks of cloud computing systems. IaaS which is located in the bottom layer, which directly provides the most powerful functionality of an entire cloud. IaaS also enables hackers to perform attacks, e.g. brute-forcing cracking, that need high computing power. Multiple virtual machines are supported by IaaS, gives an ideal platform for hackers to launch attacks that require a large number of attacking instances. Loss of data is another security risk of cloud models. Data in cloud models can be easily accessed by unauthorized internal employees, as  well as external hackers. The internal employees can easily access data intentionally or accidently. External hackers may gain access to databases in  such environments using hacking techniques like session hijacking and network channel eavesdropping. Virus and Trojan can be uploaded to  cloud systems and can cause damage .It is important to identify the possible cloud threats in order to implement a system which has better security mechanisms to protect cloud computing environments. |  |
|  | **2.2 Proposed system**  A cloud computing infrastructure includes a cloud service provider, which provides computing resources to cloud end users who consume those resources. In order to assure the best quality of service, the providers are responsible for ensuring the cloud environment is secure. This can be done by defining stringent security policies and by applying advanced security technologies.  **Security Policy Enhancement**  With a valid credit card, anyone can register to utilize resources offered by cloud service providers. This causes hackers to take advantage of the powerful computing power of clouds to conduct malicious activities, such as spamming and attacking other computing systems. By mitigating such abuse behavior caused by weak registration systems, credit card fraud monitoring and block of public black lists could be applied [31]. Also, implementation of security policies can reduce the risk of abuse use of cloud computational power [32]. Well-established rules and regulations can help network administrators manage the clouds more effectively. For example, Amazon has defined a clear user’s policy and isolates (or even terminates) any offending instances whenever they receive a complaint of spam or malware coming through Amazon EC2  **Access Management**  The end users’ data stored in the cloud is sensitive and private; and access control mechanisms could be applied to ensure only authorized users can have access to their data. Not only do the physical computing systems (where data is stored) have to be continuously monitored, the traffic access to the data should be restricted by security techniques. Firewalls and intrusion detection systems are common tools that are used to restrict access from untrusted resources and to monitor malicious activities. In addition, authentication standards, Security Assertion Markup Language (SAML) and eXtensible Access Control Markup Language (XACML), can be used to control access to cloud applications and data. SAML focuses on the means for transferring authentication and authorization decisions between cooperating entities, while XACML focuses on the mechanism for arriving at authorization decisions  **Data Protection**  Data breaches caused by insiders could be either accidental or intentional. Since it is difficult to identify the insiders’ behavior, it is better to apply proper security tools to deal with insider threats. The tools include: data loss prevention systems, anomalous behavior pattern detection tools, format preserving and encryption tools, user behavior profiling, decoy technology, and authentication and authorization technologies. These tools provide functions such as real-time detection on monitoring traffic, audit trails recording for future forensics, and trapping malicious activity into decoy documents. |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | **2.3 Hardware Environment**  **System Configuration Required:**  Processor: Core i7 or later  RAM: 16GB  HDD: 30GB  **System Configuration Minimum:**  Processor: Core i3 or later  RAM: 8GB  HDD: 30GB |  |
|  | **2.5 Software Environment**   * VMWare Workstation 14 * Kali Linux (Attacking Machine) * Ubuntu (Victim Machine) hosted with Web Application * Windows or Any Unix based Host Machine |  |
| **4.** | **SYSTEM DESIGN** |  |
|  | **Architecture Diagram** |  |
|  |  |  |
|  | **Hypervisor Flow Diagram** |  |
|  | **Data Flow Diagram**  Armitage Tutorial - Cyber Attack Management for Metasploit |  |
|  | **UML Diagrams** |  |
| **5.** | **SYSTEM ARCHITECTURE** |  |
|  | **5.1 Architecture Overview**  The hypervisor is hosted on the windows base machine and the hypervisor which we use is VMWare Workstation. The hypervisor handles two virtual machines Ubuntu based machine which hosts a LAMP Stack Web server and Kali linux which acts as a attacking machine. They both are connected through a private network. The kali linux OS has various penetration testing tools built in it. Metasploit is a penetration testing framework that makes hacking simple. It's an essential tool for many attackers and defenders. Point Metasploit at your target, pick an exploit, what payload to drop, and hit Enter. During the information gathering phase of a pentest, Metasploit integrates seamlessly with Nmap, SNMP scanning and Windows patch enumeration, among others. There's even a bridge to Nessus, Tenable's vulnerability scanner. Pretty much every reconnaissance tool you can think of integrates with Metasploit, making it possible to find the chink in the armor you're looking for. Once on a target machine, Metasploit's quiver contains a full suite of post-exploitation tools, including privilege escalation, pass the hash, packet sniffing, screen capture, keyloggers, and pivoting tools. You can also set up a persistent backdoor in case the machine in question gets rebooted. Wireshark is very similar to [tcpdump](https://en.wikipedia.org/wiki/Tcpdump" \o "Tcpdump), but has a [graphical](https://en.wikipedia.org/wiki/Graphical_user_interface) [front-end](https://en.wikipedia.org/wiki/Front-end_and_back-end) and integrated sorting and filtering options. Wireshark lets the user put [network interface controllers](https://en.wikipedia.org/wiki/Network_interface_controller) into [promiscuous mode](https://en.wikipedia.org/wiki/Promiscuous_mode) (if supported by the [network interface controller](https://en.wikipedia.org/wiki/Network_interface_controller)), so they can see all the traffic visible on that interface including unicast traffic not sent to that network interface controller's [MAC address](https://en.wikipedia.org/wiki/MAC_address). However, when capturing with a [packet analyzer](https://en.wikipedia.org/wiki/Packet_analyzer) in promiscuous mode on a port on a [network switch](https://en.wikipedia.org/wiki/Network_switch), not all traffic through the switch is necessarily sent to the port where the capture is done, so capturing in promiscuous mode is not necessarily sufficient to see all network traffic. [Port mirroring](https://en.wikipedia.org/wiki/Port_mirroring) or various [network taps](https://en.wikipedia.org/wiki/Network_tap) extend capture to any point on the network. Simple passive taps are extremely resistant to tampering. |  |
|  | **5.1 Module Design Specification (Description of the Modules)**  Most people assume that all a penetration tester, or hacker, needs to do is sit down in front of a computer and begin typing an obscure string of code and voila any computer in the world is instantly opened. This stereotype based in Hollywood legend is far from the truth. Professionals in this field are very meticulous in the approach used when to uncovering and exploiting vulnerabilities in computer systems. Over time a proven framework has emerged that is used by professional ethical hackers. The four phases of this framework guide the penetration tester through the process of empirically exploiting information systems in a way that results in a well-documented report that can be used if needed to repeat portions of the testing engagement. This process not only provides a structure for the tester but also is used to develop high-level plans for penetration testing activities.    **Reconnaissance**  This phase starts with the test team knowing little about the target. The level of detail provided to the team can range from knowing only the organizations name and possibly a website address to detailed and specific system information including IP address space and technologies used defined in the ROE to limit or scope the test event. The ROE may also limit the test team’s ability to conduct activities including bans on social engineering and destructive activities like denial of service (DoS) and distributed denial of service (DDoS) attacks. The targets own website holds vast information for developing the profile for the engagement. For example, many sites proudly display organizational charts and key leader’s profiles. These should be used as a basis for developing a target profile and information about key leaders in the organization can be used for further harvesting of information on social media sites and for social engineering, if allowed in the stated ROE.  **Scanning**  After the penetration tester has completed the reconnaissance phase of an organization, they will move into the scanning phase. In this phase, the penetration tester can take the information learned about the employees, contractors, and information systems to begin expanding the view of physical and logical information system structures within the organization. Like any of the other phases in the penetration testing lifecycle, the penetration tester can return to earlier phases as needed to gain more information to enhance information gathered in the scanning phase. The main focus of the scanning phase is to determine specific information about the computers and other devices that are connected to the targeted network of the organization. Throughout this phase, the focus is on finding live hosts, determining node type (desktop, laptop, server, network device, or mobile computing platform), operating system, public services offered (web applications, SMTP, FTP, etc.), and even possible vulnerabilities. Vulnerabilities at this level are often referred to as, “low hanging fruit.” Scanning is done with a number of different tools; however, this chapter will focus on some of the best known and most effective tools including Nmap, Hping, and Nessus. The goal of this phase is to have a listing of possible targets for the next phase of the penetration testing lifecycle: exploitation.  **Exploitation**  As defined by the National Institute of Science and Technology (NIST), Special Publication 80030, Appendix, B, page B-13, a vulnerability is a “weakness in an information system, system security procedures, internal controls, or implementation that could be exploited by a threat source;” however, this definition is too broadly scoped for use when discussing exploitation and requires further explanation. A vulnerability is caused by an “error.” The error can exist in multiple places throughout the information system AND through the humans that either use or administer the networks and computers on a daily basis. Vulnerabilities with the information system can exist inside or outside of the network, lay dormant in poorly coded and unchecked software, generated through improper security controls (more pecifically, through haphazardly configured applications and network devices), or outside of the technical network through various social means that exploit the users of the information system. Consider for a moment that the word vulnerability is synonymous with the word weakness. Exploitation is simply using a weakness to leverage access into an information system or render is useless via a denial of service. The only limit of the exploitation from an attacker is the breakdown of pure drive and willpower to continue fighting against the security measures in place protecting the information system. The best tool a penetration tester has is his or her brain. Remember that there are many doors, or points of entry, into a system. If you find that one door is closed, move on to the next. Exploitation is one of the hardest and most coveted talents of a penetration tester. It takes time, knowledge, and great persistence to learn all of the attack types for a single attack vector.  **Maintaining Access**  Exploiting a computer, networking device, or web service is great; however, the goal of most penetration tests is to maintain access to the compromised system. There are a number of methodologies for maintaining access to exploited victim systems; however, the overarching conclusion of every methodology is not to steal information but to reduce the time-consuming and exhaustive efforts required to keep attacking the same machine over and over after it’s already been compromised. If a security tester is working with a team, remote collocated servers or is in need of a secondary access point for a later access to the computer system, then efforts and expectation can be easily managed and further attacks can be more precise. Maintaining access is a secondary art form that involves just as much, if not more, thought than the exploitation of a system. This chapter covers the basic concepts of security testers and hackers alike use to maintain access and keep the compromised session going. Some of the concepts presented are very advanced. The reader should not get discouraged if reading this chapter doesn’t make sense the first time though. This chapter ends with a section designed to keep the reader’s attention focused and help reenforce the advanced methodologies presented. |  |
|  |  |  |
| **6.** | **SYSTEM IMPLEMENTATION** |  |
|  | **6.1 Exploitation Implementation**  Metasploit is a modular system. To better understand the framework, it will help view the Metasploit Framework as if it were a vehicle. The framework, much like the chassis of James Bond’s well maintained Aston Martin, provides a housing for all of modules that actually fuel the car. HD Moore, much like “Q” from the James Bond films, has stocked the nooks and crannies around the engine with an arsenal of goodies. If one of the modules within the framework becomes damaged or is removed, the vehicle can still function and continue to unleash wave after wave of attack.  The framework breaks down into the module types:  1. Exploit Modules  2. Auxiliary Modules An Overview of Metasploit 137  3. Payloads  4. Listeners  5. Shellcode  Applications that interface with the Metasploit framework could be considered a sixth category, such as Armitage; however, these are not part of the actual framework itself. Just because James Bond can control his vehicle from his watch doesn’t mean the vehicle needs the owner to wear the wrist watch to operate it.  **Exploit Modules**  Exploit modules are prepackaged pieces of code within the database that when run against a victim computer will attempt to leverage a vulnerability on the local or remote system compromising the system and allowing for DoS, disclosure of sensitive information, or the upload of a specially crafted payload module such as Meterpreter shell or other type of call back shell.  **Auxiliary Modules**  Auxiliary modules, unlike exploit modules, do not require the use of a payload to run. These types of modules include useful programs such as scanners, fuzzers, and SQL injection tools. Some of the tools within the auxiliary directory are extremely powerful and should be used with caution. Penetration testers use the plethora of scanners in the auxiliary directory to gather a deep understanding of the system to be attacked and then transition to exploit modules. Payloads If James Bond’s Aston Martin is a reference for the Metasploit Framework itself, the exploit and auxiliary modules would be akin to the rocket launchers and flame throwers under the hood. In this model, payloads would be the specialized communications equipment that can be attached to the target to maintain covert communications and tracking. While using an exploit against a vulnerable machine, a payload is generally attached to the exploit before its execution. This payload contains the set of instructions that the victim’s computer is to carry out after compromise. Payloads come in many different flavors and can range from a few lines of code to small applications such as the Meterpreter shell. One should not just automatically jump to the Meterpreter shell. Metasploit contains over 200 different payloads. There are payloads for NetCat, dynamic link library (DLL) injection, user management, shells, and more. Thinking like a spy might give the security tester a proper mindset when it comes to payload selection. |  |
|  | **6.2 Packet Sniffing Implementation**  The HTPP protocol is very insecure and should never be used for sites containing sensitive data. Surprisingly, the default installation of CloudStack configures the Apache web-server to use HTTP and with the administrator using this web application to perform live migrations; any malicious user capable of capturing the communication from the client browser to the web-server would have the ability to obtain the full permissions of this privileged user. This exercise installed a Kali laptop on the network, to simplify the evaluation the network switch which was configured to SPAN all traffic from the web-server to the Kali network port. With WireShark running, a full network trace was captured, while the administrator logged into the management interface. Figure 23 shows the WireShark filtering parameters used to display the username and password as it was transferred over the network in clear text. Most hackers have a good understanding of network protocols; however, if diving through a WireShark output is too difficult, Netresec have developed Network Miner a tool that analyses and formats the output of WireShark file for you. Running NetworkMiner and opening the capture pcap fle display the interesting data in tabular format clearly shows the cloud username and password. This type of attack could have been prevented if the CloudStack application was configured to use HTTPS for all web traffic; this is a secure and encrypted communication channel. Therefore, if the traffic between the client browser and web-server were encrypted using HTTP, WireShark would not have been able to compromise this password. |  |
| **7.** | **SYSTEM TESTING** |  |
|  | * **Black Box Penetration Testing**: In this approach, the tester assesses the target system, network or process without the knowledge of its details. They just have a very high level of inputs like URL or company name using which they penetrate the target environment. No code is being examined in this method. * **White Box Penetration Testing**: In this approach, the tester is equipped with complete details about the target environment – Systems, network, OS, IP address, source code, schema, etc. It examines the code and finds out design & development errors. It is a simulation of an internal security attack. * **Grey Box Penetration Testing**: In this approach, the tester has limited details about the target environment. It is a simulation of external security attacks. |  |
|  |  |  |
|  | **7.3 Test Cases & Reports / Performance Analysis**   1. Check if the web application is able to identify spam attacks on contact forms used on the website. 2. Proxy server – Check if network traffic is monitored by proxy appliances. The proxy server makes it difficult for hackers to get internal details of the network thus protecting the system from external attacks. 3. Spam email filters – Verify if incoming and outgoing email traffic is filtered and unsolicited emails are blocked. 4. Many email clients come with inbuilt spam filters that need to be configured as per your needs. These configuration rules can be applied to email headers, subject or body. 5. Firewall – Make sure the entire network or computers are protected with firewalls. A Firewall can be software or hardware to block unauthorized access to a system. A Firewall can prevent sending data outside the network without your permission. 6. Try to exploit all servers, desktop systems, printers, and network devices. 7. Verify that all usernames and passwords are encrypted and transferred over secure connections like https. 8. Verify information stored in [website cookies](https://www.softwaretestinghelp.com/website-cookie-testing-test-cases/). It should not be in a readable format. 9. Verify previously found vulnerabilities to check if the fix is working. 10. Verify if there is no open port in the network. 11. Verify all telephone devices. 12. Verify WIFI network security. 13. Verify all HTTP methods. PUT and Delete methods should not be enabled on a web server. 14. Verify if the password meets the required standards. The password should be at least 8 characters long containing at least one number and one special character. 15. Username should not be like “admin” or “administrator”. 16. The application login page should be locked upon a few unsuccessful login attempts. 17. Error messages should be generic and should not mention specific error details like “Invalid username” or “Invalid password”. 18. Verify if special characters, HTML tags, and scripts are handled properly as an input value. 19. Internal system details should not be revealed in any of the error or alert messages. 20. Custom error messages should be displayed to end-users in case of a web page crash. 21. Verify the use of registry entries. Sensitive information should not be kept in the registry. 22. All files must be scanned before uploading them to the server. 23. Sensitive data should not be passed in URLs while communicating with different internal modules of the web application. 24. There should not be any hardcoded username or password in the system. 25. Verify all input fields with long input string with and without spaces. 26. Verify if reset password functionality is secure. 27. Verify application for[SQL Injection](https://www.softwaretestinghelp.com/sql-injection-%E2%80%93-how-to-test-application-for-sql-injection-attacks/). 28. Verify application for [Cross-Site Scripting](https://www.softwaretestinghelp.com/security-testing-of-web-applications/). 29. Important input validations should be done at the server-side instead of JavaScript checks at the client-side. 30. Critical resources in the system should be available to authorized persons and services only. 31. All access logs should be maintained with proper access permissions. 32. Verify user session ends upon log off. 33. Verify that directory browsing is disabled on the server. 34. Verify that all applications and database versions are up to date. 35. Verify URL manipulation to check if a web application is not showing any unwanted information. 36. Verify memory leak and buffer overflow. 37. Verify if incoming network traffic is scanned to find Trojan attacks. 38. Verify if the system is safe from Brute Force Attacks – a trial and error method to find sensitive information like passwords. 39. Verify if the system or network is secured from DoS (denial-of-service) attacks. Hacker can target network or a single computer with continuous requests due to which resources on the target system gets overloaded resulting in the denial of service for legit requests. 40. Verify application for HTML script injection attacks. 41. Verify against COM & ActiveX attacks. 42. Verify against spoofing attacks. Spoofing can be of multiple types – IP address spoofing, Email ID spoofing, 43. ARP spoofing, Referrer spoofing, Caller ID spoofing, Poisoning of file-sharing networks, GPS spoofing. 44. Check for an uncontrolled format string attack – a security attack that can cause the application to crash or execute the harmful script on it. 45. Verify XML injection attack – used to alter the intended logic of the application. 46. Verify against canonicalization attacks. 47. Verify if the error pages are displaying any information that can be helpful for a hacker to enter into the system. 48. Verify if any critical data like the password is stored in secret files on the system. 49. Verify if the application is returning more data than it is required. |  |
| **8.** | **CONCLUSION** |  |
|  | **8.1 Conclusion and Future Enhancements**  Cloud computing is in continual development in order to make different levels of on-demand services available to customers. While people enjoy benefits cloud computing brings, security in clouds is a key challenge. Much vulnerability in clouds still exists and hackers continue to exploit these security holes. In order to provide better quality of service to cloud users, security flaws must be identified. In this paper, we examined the security vulnerabilities in clouds from three perspectives (abuse use of cloud computational resources, data breaches, and cloud security attacks), included related real-world exploits, and introduced countermeasures to those security breaches. In the future, we will continue to contribute to the efforts in studying cloud security risks and the countermeasures to cloud security breaches. |  |
|  | **APPENDICES** |  |
|  | **A.1 Sample Screens** |  |
|  | **Figure : Linux Test Machine**    **Figure : Nmap Scan Results – SQL**      **Figure: SQL Exploit**    **Figure : Exploited Database**    **Figure : Nmap Scan - Samba**    **Figure : Samba Exploitation – Gaining Root Access**    **Figure : Cracking Hashes with John the Ripper**  **Figure: WireShark – Capturing Packets**    **Figure: PHP Admin Login**    **Figure: After Login PHPMyAdmin**    **Figure: Sniffed Credentials**  **A.2 Publications** |  |
|  | **REFERENCES** |  |

[1] S. Singh, Y. Jeong, and J. H. Park, “A survey on cloud computing security: Issues, threats, and solutions,” Journal of Network and Computer Applications, vol. 75, pp. 200–222, 2016.

[2] NIST, Final Version of NIST Cloud Computing Defnition Published, 2011.

[3] L. Turnbull and J. Shropshire, “Breakpoints: An analysis of potential hypervisor attack vectors,” in Proceedings of the IEEE SoutheastCon 2013: Moving America into the Future, 2013.

[4] S. Manavi and S. Mohammadalian, “Secure model for virtualization layer in cloud infrastructure,” International Journal of Cybersecurity and Digital Forensics, vol. 1, no. 1, pp. 32–40, 2012.

[5] L. Almutair and H. Zaghloul, in Proceedings of the Te Tird International Conference on Digital Information Processing and Communications (ICDIPC ’13), pp. 676–686, UAE, 2013.

[6] D. Zissis and D. Lekkas, “Addressing cloud computing security issues,” Future Generation Computer Systems, vol. 28, no. 3, pp. 583–592, 2012.

[7] Kamesh and N. Sakthi Priya, “Security enhancement of authenticated RFID generation,” International Journal of Applied Engineering Research, vol. 9, no. 22, pp. 5968–5974, 2014.

[8] M. Ali, S. U. Khan, and A. V. Vasilakos, “Security in cloud computing: opportunities and challenges,” Information Sciences, vol. 305, pp. 357–383, 2015.

[9] O. Harfoushi, B. Alfawwaz, N. a. Ghatasheh et al., “Data security issues and challenges in cloud computing: A conceptual analysis and review,” Communications and Network, vol. 6, no. 1, pp. 15– 21, 2014.

[10] N. Arya, M. Gidwani, and S. K. Gupta, “Hypervisor security - A major concern,” International Journal of Information and Computation Technology, vol. 3, no. 6, pp. 533–538, 2013.