UNI Roll No: 6893549
Name: Mukesh Kumar

Exercise 1: Threat Modelling

1.1 Prepare these items for this task: [1]

1. User authentication and Data Encryption:

User authentication and data encryption are essential for cloud security. Authentication makes sure that only authorized people can access cloud services, using tools like multi-factor authentication (MFA), role-based access control (RBAC), and single sign on (SSO) for extra protection. On the other hand, data encryption keeps sensitive information safe by making it unreadable to anyone. There are two types encryption at rest, which protects stored data, and encryption in transit, which secures data while it's being sent over the network. Proper key management and end-to-end encryption help ensure that only the right people can decrypt and access the data.

2. Threat Detection and Response:

Threat detection in cloud systems identifies potential risks using tools like IDS, SIEM, and UBA. Once detected, response involves executing an Incident Response Plan (IRP), containing the threat, conducting forensic analysis, and applying patches to prevent future attacks.

3. Client-Side Integrity Verification for Uploaded Files:

Client-side integrity verification ensures files remain secure. Users generate a hash before uploading and compare it with the downloaded file's hash to confirm integrity. Cloud servers can also verify encrypted files without compromising privacy.

4. Limited Data Retention Policy and Data Centre compliance:

A Limited Data Retention Policy helps minimize the risk of exposing unnecessary data by setting clear timeframes for how long different types of data are kept. Once the retention period ends, the data is either deleted or anonymized. This policy supports privacy compliance. Data centre compliance ensures security standards like ISO 27001 are met, but navigating varying legal requirements across regions can be challenging.

5. Security Audits and Penetration Testing:

Security audits and penetration testing are essential for spotting and fixing system vulnerabilities. Audits examine security policies and infrastructure to ensure they meet standards and uncover any weaknesses. Penetration testing, on the other hand, simulates attacks to find flaws before hackers can exploit them. Both methods are proactive and vital, but they need to be done thoroughly and by experts to be effective

UNI Roll No: 6893549
Name: Mukesh Kumar

1.2 Risk Assessment

Distributed Denial of Service (DDoS) Attack

A DDoS attack involves overwhelming a system with excessive traffic, making it inaccessible to genuine users. ACME Clouds' reliance on cloud infrastructure and its exclusive membership model make it susceptible to such disruptions.

Likelihood

DDoS attacks are highly likely as they are a common method used to target cloud services.

Impact

The consequences would be significant, including service downtime, financial losses, customer dissatisfaction, and damage to reputation.

Vulnerability

Weaknesses include the absence of advanced traffic filtering tools and risks related to the exclusive access model. A non-distributed server setup further exacerbates the threat.

Encryption Key Compromise

Description

Encryption key compromise occurs when attackers gain unauthorized access to keys, potentially exposing sensitive data. This risk is heightened if users store keys insecurely or fall victim to phishing schemes.

Likelihood

The probability is moderate, as it depends on user behaviour and susceptibility to social engineering.

Impact

The outcome would be devastating, leading to data breaches, privacy violations, and loss of trust from customers.

Vulnerability

Key risks include insecure storage methods, outdated cryptography, and weak protections in client-side software, making keys easier to exploit.

Countermeasure: To mitigate DDoS attacks, ACME Clouds should deploy advanced traffic filtering tools, such as AWS Shield or Cloudflare, to detect and block malicious activity. Additional measures include rate limiting, using content delivery networks (CDNs) to distribute traffic across servers, and implementing a redundant server architecture to ensure system availability during attacks. Regular penetration testing is also crucial to identify and address potential vulnerabilities.

To prevent encryption key compromise, secure storage solutions like hardware security modules (HSMs) and encrypted keychains should be prioritized. Two-factor authentication (2FA) can provide an extra security layer, while regular key rotation minimizes the impact of compromised keys. Educating users on secure key management and phishing risks, along with enhancing client-side application security, will further reduce threats.

UNI Roll No: 6893549
Name: Mukesh Kumar

Exercise 2: Access Control

2.1 Access Control Matrices, Lists and Capability Lists

a) Access control matrix

Subjects/Objects	FileA.txt	RunMe.out	Test.sh
UserA	{R}	{W,X}	{R,W,X}
UserB	{X}	{R,W}	{R}

b) Access control lists (ACL):

Subjects/Objects	File	Permissions
FileA.txt	UserA	{R}
FileA.txt	UserB	{X}
RunMe.out	UserA	{W,X}
RunMe.out	UserB	{R,W}
Test.sh	UserA	{R,W,X}
Test.sh	UserB	{R}

c) Set of capability lists:

Subjects/Objects	File	Permissions
UserA	FileA.txt	{R}
UserA	RunMe.out	{W,X}
UserA	Test.sh	{R,W,X}
UserB	FileA.txt	{X}
UserB	RunMe.out	{R,W}
UserB	Test.sh	{R}

d) UserB cannot run the script Test.sh because he lack the necessary execute permission. The ability to run a script requires the execute (X) permission, which UserB does not have for this file. While UserB can read the contents of the script, allowing them to view or analyze it.

2.2 Multilevel Security – Confidentiality

Subjects/Objects	Location
User1 (Level4)	File2, File6
User2 (Level3)	-
User3 (Level2)	File1, File3, File4
User4 (Level1)	File5

UNI Roll No: 6893549
Name: Mukesh Kumar

Giving reasons and answer are based on the Bell La Padula model

a) Based upon the instructions, User1 is in the top position and has level 4 access. It means that he can read any file based upon the Bell LaPadula model (No Read Up) and only write File 2 and File 6 (No Write Down).

b) User3 has granted Level2 access so that he can read both File5 and File3, but cannot modify File5, which is a part of Level1 (lowest) and follows the No Write Down rule. On the other hand, User3 can modify File1, File3 and File4 in his directory.

c) User2 is able to read both the files because he has Level3 access, and both the files are placed below Level3.

d) Subject (User 4) belongs to the lowest level of the group and has been given level 1 access, so he cannot compute File2 (having Level 4 security) and File1 (having Level 2 security).

e) This classification is based upon the famous Bell LaPadula model, having a no-read-up and no-write-down approach. It means that the user cannot read the files above their level, and he cannot modify the files below his level. This model is perfect for protecting all the information, except data leaks from the file itself.

f) The main drawback of the Bell LaPadula model is that it cannot protect the information leak (indirectly), though it can restrict the write and read data for users. For example, File 3 contains multiplied values from File 2 and File 4 in this case, so User 3 can get the data by doing some basic calculation.

2.3 Role based Access Control

a) Role Based Access Control (RBAC) in MS SQL Server ensure security management by assigning users specific permissions based on their roles. This approach enforces separation of duties, ensuring that users receive only the access they need for their functions. Administrators can quickly change users' roles and modify their permissions as required.

b) The Principle of Least Privilege ensures that users are given only the necessary access to perform their tasks. This minimizes security risks, protects sensitive information, and streamlines permission management by restricting unnecessary access

c)

Subjects/(Objects, access right)	Printer, View (P0)	Network, View (P1)	Network, SendFile (P2)	Directory, View (P3)	Printer, SendFile (P4)
Alice (Manager)	yes	yes	yes	yes	yes
Bob (Admin)	yes	yes	yes	-	-
Claire (Admin)	yes	yes	yes	-	-
Dom (Operator)	yes	yes	-	-	-

Eve (User)	yes	-	-	-	-

2.4 Unix Access Control [4]

a) The /tmp directory in Linux is an important shared space where all users can create temporary files. Its permissions, represented as 1777 in octal, combine broad accessibility with a key security feature such as add, read and modify. The "1" at the start refers to the sticky bit, which ensures that only the file's owner or the root user can delete or modify a file, even though others may have write access to the directory. This prevents users from accidentally or intentionally interfering with each other's files. The rest of the permissions, "777," allow everyone to read, write, and execute within the directory, making it highly accessible while the sticky bit maintains file specific security.

b) The passwd command lets users change their passwords, which involves updating the /etc/shadow file where account information is securely stored. Since this file is highly protected and only accessible to the root user, the passwd command needs a way to work with elevated privileges. This is achieved through the setuid bit, which allows the command to run with root permissions even when executed by a regular user. This setup ensures that users can safely update their passwords without being given direct access to critical system files, maintaining both the security of sensitive data and the functionality needed for password management

Exercise 3 Software Vulnerabilities

3.1 Briefly answer the following questions

a) Even with advanced penetration testing tools like Burp Suite and Metasploit, security vulnerabilities persist due to technical, human, and organizational challenges.
   1. **Complex Systems and Human Error:** Modern software systems are highly complex, increasing the likelihood of mistakes. Developers may introduce vulnerabilities due to oversights, tight deadlines, or limited knowledge of secure coding practices. Tools like Burp Suite are effective for identifying known issues but often struggle with detecting logical flaws, which require deeper manual analysis.

   2. **Misuse of Tools:** Penetration testing tools require expertise to use effectively. Inexperienced users may misconfigure these tools or misinterpret their findings, leading to missed vulnerabilities. Furthermore, not all organizations invest in skilled security professionals, increasing the risk of undetected flaws

   3. **Evolving Threat Landscape:** Cyber threats evolve faster than tools can adapt. While tools like Metasploit focus on known vulnerabilities, new attack methods and zero day exploits often fall outside their scope. This dynamic environment requires continuous updates and threat intelligence integration, which many organizations struggle to maintain.

   4. **Legacy Systems and Technical Debt:** Many organizations rely on outdated systems that were not designed for modern security practices. These legacy systems often contain vulnerabilities that are hard to patch without significant disruptions. Additionally, technical

debt accumulated quick fixes and deferred maintenance compounds these security challenges.

5. **Slow Patch Management:** Even when vulnerabilities are identified, timely patching remains a challenge. Large organizations with complex infrastructures face logistical hurdles in applying updates, leaving known vulnerabilities exposed. Attackers often exploit these delays, increasing the risk of breaches.

6. **Limited Resources and Budget:** Smaller organizations, in particular, face budgetary and resource constraints that hinder their ability to implement comprehensive security measures. Outdated tools, limited testing, and insufficient staff result in undetected vulnerabilities and greater exposure to attacks.

b) Most stack protection mechanisms don't protect every buffer by default due to practical challenges like performance, complexity, and compatibility issues. Several key factors contribute to this limitation[5]

1. **Performance and Memory Constraints**: Protecting all stack buffers can slow down program execution because of the extra checks required to ensure data integrity. This performance hit is particularly problematic in high performance applications. Additionally, methods like canaries or shadow stacks require extra memory for verification data, making universal protection inefficient or even unfeasible in memory constrained environments.

2. **Compatibility and Complexity:** Applying protection to every buffer often creates compatibility challenges and can require significant changes to the code structure. Some buffers don't fit easily into these protection schemes, increasing the risk of implementation errors. Stack protection also relies on compiler generated checks, and applying these checks universally can complicate the process, especially for languages with complex memory management.

3. **Legacy Code Compatibility:** Many older systems and applications were designed without modern protections in mind. Retrofitting these protections can lead to crashes, unpredictable behaviour, or compatibility issues with existing software, making universal adoption impractical.

4. **Balancing Security and Developer Control:** Developers often need the flexibility to selectively enable or disable protections based on their application's needs. Universal protection can negatively impact performance or memory use in critical systems, so giving developers control allows them to strike the right balance.

5. **Layered Security Approach:** Stack protection is just one part of a broader security strategy. Mechanisms like address space layout randomization (ASLR), data execution prevention (DEP), and control flow integrity (CFI) work alongside stack protection to create a layered defence, reducing the need for universal stack buffer protection.

c) A prompt injection attack occurs when someone tricks an AI by adding harmful instructions to its input, causing it to act in unintended ways, like bypassing safety measures or revealing sensitive data. These attacks can lead to data leaks or security breaches. For example, in this case, it leaked personal data, including name, email, website, phone number, cell phone, and fax number[6].

3.2 Consider the following program [7]

a) Entering 13 'A's in the password input triggers a buffer overflow, as the buff array is designed to hold only 12 characters. This overflow modifies adjacent memory, such as the attempts variable, potentially allowing more than the intended number of logins attempts and causing unpredictable behaviour in the program.

b) When someone attempts to enter 25 A's as a password, it leads to a segmentation fault. This happens because the program tries to access memory it shouldn't, such as going beyond the limits of an array. As a result, the program crashes. The operating system detects the invalid memory access and usually stops the program. It often displays an error message like "segmentation fault" or "core dumped" to let someone know something went wrong.

c) Typing 18 A's causes the program to overflow the input buffer, accidentally changing the correct variable 0 to 1, which tricks it into granting access. To get that number, start increasing the number of A's from 13.

d) When someone enter 17 A's, it's not enough to overflow the buffer and reach the memory where the correct variable is stored. Since the correct variable isn't changed, it stays as 0, and the authentication fails.

e) A segmentation fault doesn't happen with 18 A's because the overflow stays within the program's allocated memory. It changes nearby variables like correct, but doesn't access restricted memory, which would trigger a segmentation fault.

3.3 Buffer overflow example in the real world [8]

CVE 2019 13726 was a serious vulnerability in Google Chrome's password manager that could have compromised user credentials stored in the browser. The issue stemmed from Chrome's autofill feature, which failed to properly verify the origin of websites requesting autofill data. As a result, malicious websites could potentially exploit this flaw to access saved passwords, posing a significant security risk. Exploiting this vulnerability would allow attackers to retrieve sensitive login information, bypass security protections, and gain unauthorized access to user accounts.

The primary risk of this vulnerability was credential theft, which could lead to account hijacking. Once attackers obtained login details, they could carry out further malicious activities such as identity theft, financial fraud, or unauthorized access to sensitive personal or corporate information, depending on the type of exposed accounts.

Google addressed this vulnerability in Chrome version 77.0.3865.75 by implementing stricter origin checks for the autofill feature. This update ensured that passwords would only be filled in on trusted and legitimate websites, effectively blocking unauthorized access to user credentials. The fix significantly improved the security of Chrome's password manager, highlighting the importance of

properly validating browser security features to protect against exploitation. Users were urged to update to the latest version of Chrome to safeguard their accounts and prevent password leakage.

Exercise 4 Web Security

4.1 Briefly answer the following questions

a) Cookies are small files stored by browsers to help websites remember information. They are divided into three types first party, second party, and third party cookies.
First party cookies are created by the website a user visits and store data like login details or shopping cart items. They improve the user experience by keeping session information.

Second party cookies come from trusted partners of the website. These are used in cases like affiliate marketing or data sharing, helping users interact smoothly across partnered websites.

Third party cookies are set by external organizations, such as advertisers, to track users across multiple websites. They are often used for targeted ads but raise privacy concerns due to extensive tracking.

b) The "same origin policy" (SOP) is a security feature in web browsers designed to limit interactions between resources from different origins. It ensures that documents or scripts from one origin cannot directly access content from another origin without proper permissions. However, third party cookies can bypass this restriction due to how browsers handle cookies from external domains.
When a user visits a website, it may load resources such as advertisements or social media features from third party domains. These elements can set cookies in the user's browser, even though SOP restricts direct access between the website and the external content. As long as the browser requests resources from the third-party domain, that domain can create cookies. These cookies are then accessible to the third-party domain on subsequent visits to any website that integrates content from the same domain, enabling cross site tracking and targeted advertising.

4.2 Demonstrate a CSRF attack [9]

a) A CSRF (Cross-Site Request Forgery) token is a security feature used in web applications to ensure that requests made to a server are intentionally initiated by an authenticated user. The server generates a unique token and associates it with the user's session. This token must be included in sensitive actions, such as form submissions or account modifications. When the server receives a request, it verifies the token to confirm its authenticity. By validating the presence of this token, CSRF protection prevents unauthorized third parties from exploiting authenticated sessions to execute unintended commands on behalf of the user [10].

b) CSRF tokens are a crucial security measure, effectively preventing unauthorized actions by embedding a unique, unpredictable token into forms or requests. This token ensures that the action comes from an authenticated user, mitigating CSRF risks. Requiring authentication for sensitive actions further strengthens security, limiting operations like account updates to logged-in users only.

UNI Roll No: 6893549
Name: Mukesh Kumar

Exercise 5: The future of malware detection [11]

Anti-Viruses under the microscope- A hands-on perspective

a) Modern antivirus (AV) solutions use a mix of techniques to identify and neutralize threats, relying on both static and dynamic detection methods. Static methods, like signature-based matching, focus on predefined patterns to detect known malware, while dynamic techniques, such as sandboxing and behaviour analysis, evaluate activities in real time to identify suspicious behaviour. These systems target multiple attack surfaces, including files, processes, memory, network traffic, and browser activities. The architecture of AV systems typically follows a client server model, with the Graphical User Interface (GUI) serving as the user facing component for configuration and reporting, and the AV Cor a background service handling the actual detection and protection processes.

   While significant advancements have been made, signature-based detection remains a cornerstone of AV systems. This method is highly effective for identifying known threats but comes with limitations. Shorter signatures can lead to false positives, while longer ones can slow down performance and require substantial storage. Additionally, many AV products share underlying detection engines, which complicates evaluations as identical detections may be counted across multiple solutions. To address these challenges, modern AVs increasingly rely on cloud-based intelligence and telemetry. Cloud services provide real time updates to signatures and behavioural models, while telemetry aggregates data from users to refine threat detection. However, this reliance on external systems introduces risks, such as potential data leakage or unauthorized access, which must be mitigated to ensure user trust.

   Detecting advanced threats like rootkits remains particularly challenging. Rootkits often operate at the kernel level, using advanced techniques to hide their presence from traditional detection methods. Tools like memory scanning and kernel callbacks have been developed to uncover these threats, but the evolving sophistication of rootkits continues to outpace many detection strategies. This ongoing arms race between malware developers and AV solutions underscores the importance of innovation.

b) Malware detection software is facing increasing challenges as cyber threats grow more sophisticated and adaptive. Techniques like polymorphism and encryption allow malware to constantly change its structure, bypassing static detection methods that rely on fixed signatures. Rootkits, which operate deep in the kernel, manipulate core system processes to hide from antivirus software, further complicating detection and removal. While signature-based methods remain essential for identifying known threats, their dependency on regular updates leaves systems vulnerable to novel malware, especially given the overwhelming volume of new threats emerging daily.

   Cloud-based intelligence has improved detection by enabling real time updates and global telemetry sharing. However, this reliance on cloud services introduces latency issues, risks of data breaches, and potential vulnerabilities to attacks targeting cloud infrastructure. The rise of zero-day threats has amplified these concerns, as detecting these previously unknown vulnerabilities requires advanced behavioural modelling that current methods often fail to achieve in time.
   Adding to the complexity, attackers are now leveraging AI to create malware capable of adaptive and unpredictable behaviour, outpacing traditional defences. Although more sophisticated detection algorithms aim to counter these developments, they often come at the cost of increased resource usage, which can degrade system performance.

UNI Roll No: 6893549
Name: Mukesh Kumar

References:

[1] S. Pearson and A. Benameur, "Privacy, Security, and Trust Issues Arising from Cloud Computing," in *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, Indianapolis, IN, USA, 2010, pp. 693-702. [Online]. Available: https://doi.org/10.1109/CloudCom.2010.66

[2] Cloudflare, “What is a DDoS Attack?,” *Cloudflare Learning Center*, 2023. [Online]. Available: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/

[3] **OWASP**, “Cryptographic Storage Cheat Sheet,” *OWASP Cheat Sheet Series*, 2023. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html

[4] Covers file permissions, including sticky bits and setuid bits, with examples related to common directories and commands.
[Online]. Available: https://tldp.org/LDP/intro-linux/html/

[5] Microsoft Documentation, "Buffer Security Check," Visual Studio Documentation, https://msdn.microsoft.com/en-us/library/8dbf701c.aspx (accessed: February 2, 2024).

[6] OWASP Foundation. (2023). LLM01: Prompt Injection. In OWASP Top 10 for Large Language Model Applications, from https://genai.owasp.org/llmrisk/llm01-prompt-injection/(Retrieved November 19, 2024).

[7] Wikipedia Contributors. (2024). Buffer Overflow. In Wikipedia, The Free Encyclopedia. Retrieved November 19, 2024, from https://en.wikipedia.org/wiki/Buffer_overrun

[8] National Institute of Standards and Technology. (n.d.). CVE-2019-13726 Detail. National Vulnerability Database. Retrieved February 2, 2024, from https://nvd.nist.gov/vuln/detail/CVE-2019-13726

[9] PHP Tutorial. (n.d.). PHP CSRF - Prevent Cross-Site Request Forgery Attacks. Retrieved February 2, 2024, from https://www.phptutorial.net/php-tutorial/php-csrf/

[10] DateMill. (2024). CSRF Token in PHP: A Comprehensive Guide to Fortifying Web Defenses. Retrieved from: https://www.datemill.com/csrf-token-php/

[11] Botacin, M., Domingues, F. D., Ceschin, F., Machnicki, R., Alves, M. A. Z., de Geus, P. L., & Grégio, A. (2021). AntiViruses under the microscope: A hands-on perspective. Computers & Security, 102500