Name: Giridhar Benkipur
URN: 6898345

EXERCISE-1: Threat Modelling

1.1 a) Further assumptions about the system:

- **1. Data Redundancy**:

As ACME is a cloud-based file-sharing platform, it is very much necessary that data is never lost for any reason. To protect customer data, we could set up Geo-redundant system, where data is stored in at least one other additional location. This also helps ensure that if one site experiences traffic overload and fails [1], the microservices/software can automatically switch to another geographic location.

- **2. Defined file size and file type for the users**:

ACME customers will have a total of 15GB of free storage available for themselves. If the user exceeds more than 15 GB of data, then he has to opt for a better plan. And with respect to type of files, we would be allowing only major/famous file formats [2] to reduce the risk of injection using berserk type of files.

- **3. Delete recommendations for duplicate files**:

In case our ACME encounters duplicate files, ACME creates a temporary folder that holds all these duplicate entries. With respect to security, if there are duplicate versions of a single file and the employee uses the old file for instance, this action leads to multiple security vulnerably. The user has an option to delete all the duplicate files by deleting this folder. This is an efficient way to reduce the storage consumed by garbage entries.

- **4. Integrity assurance for users**:

One of the feature of ACME is that it uses advanced cryptographic primitives and algorithms to maintain security over the process. This means every file is hashed and potentially stored in our MongoDB. On demand, we can provide our users with their hashes and signatures of the files, and they can recompute the hash and verify them. This will surely build their trust, knowing files are safe and untampered in the whole process.

- **5. Every move being logged**:

We need to make use of separate servers to store logs. Be it server logs, client logs, network logs, LTM logs [3]. These sets of logs are the only resources that will come handy when we face an issue or a data breach. We need to create an RLS (Remote Log Server) and RLS_Bkp Server. After a certain point of time, the logs are pushed to RLS_Bkp. After 30 days, RLS_Bkp clears and stores a new set of logs with the help of a CRON Job.

Name: Giridhar Benkipur
URN: 6898345

1.1 b) I used draw.io to systematically draw the attack tree, it is attached in next page.

The references with respect headings are given below:

[4] Data Redundancy Exploits

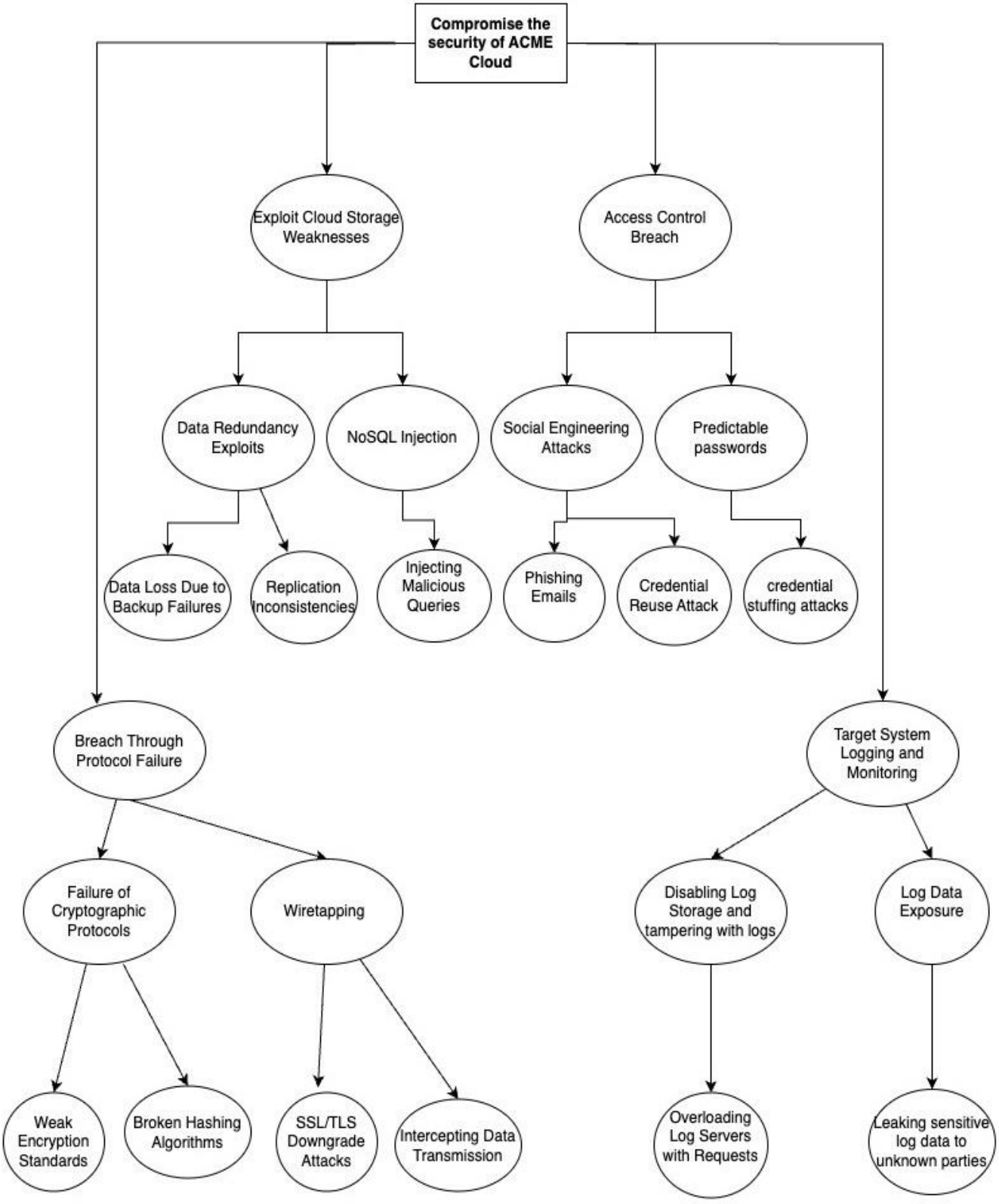
[5] Breach through Protocol failure

[6] Failure of Cryptographic Protocols

[7] Wiretapping

[8] Target system logging and monitoring

Name: Giridhar Benkipur
URN: 6898345



Name: Giridhar Benkipur
URN: 6898345

1.2) Risk Assessment

***Threat 1**: Data Redundancy Exploits:*

Description:

As described in the assumptions, ACME implements geo-redundancy to avoid data loss and ensure availability. But the problem is, redundancy mechanisms might expose the system to data corruption and synchronization attacks if misconfigured. Strict consistency is lacking in most NoSQL databases [9], such as MongoDB, which can lead to exploitation of these issues.

Vulnerabilities:

Data Desynchronization: There is a chance that geographic redundancy may lead to data desynchronization between the primary site and the redundant site due to network latency or any form of system failure, thus leaving the backup site stale or inconsistent.
This vulnerability is valid as is listed with CVE number: **CVE-2024-8305.**

Redundancy Attack Surface: An attacker could use failover mechanisms to cause data loss or disruption of service.
Attack is listed with number: **CVE-2021-32036.**

NoSQL Database Exposure: Poor configuration / Misconfiguration of MongoDB will result in unauthorized access to redundant copies of data to unauthorized 3rd parties.
This is a valid vulnerability listed in **CVE-2021-20333.**

Impact and Justification of the Likelihood:

Risk of Data Desynchronization: Impact: High, Likelihood: Medium – It's a common vulnerability in geo-redundant systems, the likelihood depends on the redundancy setup.

Redundancy Attack Surface: Impact: High, Likelihood: Likely – Failover mechanisms are usually targeted in redundancy setups and may be easier to exploit if not strictly controlled and checked regularly.

NoSQL Database Exposure: Impact: High, Likelihood: Medium to Likely – Poorly configured NoSQL databases are often targets of unauthorized access attempts.

Name: Giridhar Benkipur
URN: 6898345

Countermeasure:

Impose Consistency: Can possibly deploy hash verifications to detect any tampering of data prior to synchronization.

Enforce Redundancy Access Control: Block unauthorized failover access by allowing access to identified authorities only.

Hardening of Database Configuration: Enforce strong access controls to secure MongoDB so that unauthorized access to data does not occur.

Threat 2: Insecure Logging Practices

Description:

Having poor logging practices have been known to reveal valuable data and even weaken the security of a system as a whole [10]. Logs contain valuable information about usernames, sessionID, and potential customer info. In case of negligence, this may become available to attackers, and security of the company may be compromised.

Vulnerabilities:

Lack of Log access control: Apps with poor permission on log files or log management systems may give unauthorized access to users allowing them to read, modify, or delete logs. A similar case is discussed in **CWE-732.**

Unsecure Log Movement: Logs transmitted over unsecured channels are vulnerable to interception by attackers for disclosure of data. Listed with Database **CVE-2020-3452.**

Impact and Justification of the Likelihood:

Risk of data loss or data exposure: Impact: Medium to High, Likelihood: Medium to likely – It is common that the logs are stored insecurely. Which ultimately makes it an easy point for the attackers to get info from a system.

Risk of unsecured Log Transmission: Impact: High, Likelihood: Medium- Unsecured log transmission can lead to sensitive data being intercepted, altered, or even leaked to the attackers.

Name: Giridhar Benkipur
URN: 6898345

Countermeasure:

Rate Limiting for Log Entries: We can attempt to implement rate limiting mechanisms to prevent overflows.

Automated Review of Audit Logs: Conduct periodic logs review and identification of patterns, unauthorized access attempts, or system errors.

EXERCISE-2: Access Control

2.1 a) Access control matrix:

	FileA.txt	**RunMe.out**	**Test.sh**
User A	R	W, X	R, W, X
User B	X	R, W	R

2.1 b) ACL's:

- **FileA.txt:**
  - User A: R
  - User B: X
- **RunMe.out:**
  - User A: W, X
  - User B: R, W
- **Test.sh:**
  - User A: R, W, X
  - User B: R

2.1 c) Capacity Lists:

- **User A's Capabilities**:
  - FileA.txt: R
  - RunMe.out: W, X
  - Test.sh: R, W, X
- **User B's Capabilities**:
  - FileA.txt: X

- RunMe.out: R, W
- Test.sh: R

2.1 d)

User B has only Read permission for Test.sh file (as seen in the Capability list) and does not have permission to Write or Execute. This is the reason, User B will not be able to execute the file Test.sh.

2.2 a)

According to Bell-La Padula model, there are 2 base rules [11]:
Simple Security Property: “no read up”
Security Property: “no write down”

Read Access: User1 has Level4 access (highest level in this case), he can read all files at level4 or even below Level4, which includes File1, File2, File3, File4, File5, and File6. So, User1 can read all the files.
Write Access: User1 cannot write down to any lower level. Therefore, User1 can only write to objects at Level4, in this case File2 and File6 (as both are at level4).

2.2 b)

User3 has level2 access.

- Read: User3 can read files at or below Level2. This includes File1 (Level2), File3 (Level2), File4 (Level2), and File5 (Level1).
- Write: User3 cannot write to any file below Level2, so he cannot write to File5.

Since User3 can read both File5 and File3, he has sufficient access to compute File5 + File3.

2.2 c)

User2 has Level3 access.

- Read: User2 can read files at or below Level3, which includes File1 (Level2), File3 (Level2), File4 (Level2), and File5 (Level1).
- Write: User2 cannot write to any file below Level3, so he cannot write to File5.

User2 has read access to both File5 and File3, which means he can perform the computation.

Name: Giridhar Benkipur
URN: 6898345

2.2 d)

User4 has Level1 access

- Read: User4 can only read files at Level1, which includes File5.
- Write: User4 can write only to files at Level1.

User4 cannot access File1 (Level2) or File2 (Level4) due to the no-read-up rule in the Bell-La Padula model. Therefore, User4 cannot compute anything involving File1 and File2.

2.2 e)

According to the Bell-La Padula model, users with lower security levels should not be able to access Level4 data. Since File3's value depends on File2 (Level4) and File4 (Level2), there is a potential risk of leaking Level4 information indirectly through computations involving File3, which is at Level2. This indirect access could allow users at Level2, or even lower, to refer to information at Level4/File2, which is a violation of the confidentiality of data at Level4.

Therefore, classification as it is does not protect Level4 information because File3, being a level2 object, contains data from an object at a Level4(File2), hence causing information leakage of Level4.

2.2 f)

The problem here is confidentiality breach. Since File3 contains data derived from File2 (which is a Level4 file), the information at Level4 can be used by users with access to Level2 data, which violates the confidentiality properties of the Bell-La Padula structure. This will lead to low-level users indirectly accessing sensitive information from higher security levels.
The problem raised is information leakage, where Level4 data might be used by users with Level2 access through File3, thus compromising the confidentiality of Level4 information.

2.3 a)

The policies/rules in Microsoft SQL Server use the RBAC model to standardize permission management across different user roles, which makes it easier to enforce the Principle of Least Privilege. By grouping users into server-level and database-level roles, administrators can efficiently assign and manage permissions without needing to individually configure access for each user, which reduces the chance of human error.

RBAC model also has a few defined requirements, where roles like sysadmin and

securityadmin have different levels of responsibilities and capabilities. By making use of RBAC, it helps to make access control safe and easier to manage SQL Server, which ends up being beneficial for admins and also general security of the system is improvised.

2.3 b)

The security-related principle that security architects use in order to set permissions for predefined roles in SQL Server is the Principle of Least Privilege. This ensures that each role will only be granted those permissions that are really needed to perform its particular tasks, therefore helping to minimize the probability of unauthorized access or just accidental misuse of the permissions in question. Referred [27] to understand better about Least Privilege.

2.3 c)

In the case of RBAC model, employees are only allowed to access the information necessary to effectively perform their duties [32]. Read [33] this NIST document to understand the concept better.

- **User**: P0
- **Operator**: P1
- **Admin:** P2
- **Manager**: P3, P4

Mapping of Permissions to (Object, Access-Right)

- **P0** = (Printer, View)
- **P1** = (Network, View)
- **P2** = (Network, SendFile)
- **P3** = (Directory, View)
- **P4** = (Printer, SendFile)

User	Printer (View) P0	Network (View) P1	Network (SendFile) P2	Directory (View) P3	Printer (SendFile) P4
Alice	✓	✓	✓	✓	✓
Bob	✓	✓	✓		
Claire	✓	✓	✓		

Name: Giridhar Benkipur
URN: 6898345

Dom	✓	✓			
Eve	✓				

2.4 a)

/**tmp** directory on linux is usually set to permissions of **1777** in octal numbers.

Explanation of 1777:
1 is called the sticky bit [12] which prevents the users from deleting files they don't own in the /tmp directory.
7 for the owner (7 means the file has read, write, execute permissions).
7 for the group (7 means the file has read, write, execute permissions).
7 for others (7 means the file has the read, write, and execute permissions).

The /tmp directory is used to store temporary files and is accessible by all users on the system. The sticky bit ensures that no user can remove or change files created by another user, since /tmp is a shared directory [28].

Checked on Open Nebula as well:
user@Ubuntu-Nebula:~$ ls -ld /tmp/
drwxrwxrwt 16 root root 4096 Nov 14 16:10 /tmp

user@Ubuntu-Nebula:~$ stat /tmp/

Access: (**1777**/drwxrwxrwt) Uid: (0/ root) Gid: (0/ root)

[31] Referred to this website to understand about stat command.

2.4 b)

The **passwd** file has permissions set to **4755** in octal notation.

Open Nebula Snippet:
user@Ubuntu-Nebula:/usr/bin$ stat /usr/bin/passwd

Access: (4755/-rwsr-xr-x) Uid: (0/ root) Gid: (0/ root)

- 4 is the special bits. 4 is used for setuid. It's used to execute the program with privileges of its owner, rather than the user who runs it.
- 7 for the owner (7 means the file has read, write, execute permissions).
- 5 for the group (5 means read and execute).

The passwd command is set so, that it allows the users to change their passwords with root privileges. The setuid as special bit, makes sure that the command runs with root's permissions and the user should be able to modify the /etc/passwd and the shadow file. So, the owner has full permissions over command, while the group and others have read and execute permissions for the command. Referred to [29] to understand better on file permissions.

EXERCISE-3: Software Vulnerabilities

3.1 a)

Human Miscalculations: Tools like Burp Suite and Metasploit are highly dependent on the skills and decisions of the user. Misconfigurations or lack of training, or not following best practices, can lead to gaps in coverage and cause vulnerabilities to go undetected. [13] Here Burp Suite itself mentions the significance of having proper setups to avoid problems like software/security misconfiguration.

Illusion of Safety: There's a chance that companies may become overdependent on automated tools like Burp Suite or Metasploit and mistakenly think their system is fully secure. This false sense of security can lead to problems with improper follow-up testing or patching of the issues. [14] referred to this website for better understanding. [15] This website describes the darks side of automated tools, and its harm caused.

3.1 b)

- **System burden**: Protecting every type of buffer may not even be necessary. Incase if it were to be done, it would lead to a lot of system lagging and overload. Stack protection processed work by implementing additional checks (like stack canaries) [16] to detect overflows. Performing all these processes would eventually slow down the execution.

- **Complexity and Vulnerabilities**: System complexity and human oversight are some of the major contributors to the persistence of vulnerabilities. As modern systems increase in complexity, misconfigurations and coding mistakes become very frequent. Even with the best pen-testing tools available, human aspects of oversight and the large scope of today's systems create gaps.

3.1 c)

Prompt Injection [17] is the vulnerability that was exploited to end up with such a response. It's a type of vulnerability that manipulates LLM's response by repeating certain commands or specific patterns [18]. This injection led to a response including sensitive information which was not supposed to show up.

There is name, designation, email, web link, phone number, fax number, and cell phone number. This information is being taken from training data or system prompts revealing information.

3.2 a)

When I enter "AAAAAAAAAAAAA" (13 A's), this does not cause any issues, the buffer in the program can hold the provided input, because the buffer size defined in the program is 13. Referred [19] to understand better. But when I provide more than 13 A's, this leads to a buffer overflow. This process can potentially change the memory locations located beside the buffer, which in-turn might change the memory allocations of other variables.

3.2 b)

The output when I give 25 A's:

user@Ubuntu-Nebula:~/Desktop/CW1/Code$ ./authenticate

Please enter your password:AAAAAAAAAAAAAAAAAAAAAAAAA

AAAAAAAAAAAAAAAAAAAAAAAAA

Try again

Segmentation fault (core dumped)

This happens when buffer overflow extends beyond the memory allocation of local variable to other regions that are important to run the program properly. Usually, a segmentation fault occurs when a program tries to access memory it does not have permission to use [30]. The excessive input overwrites control structures or memory addresses on the stack, leading to a program crash.

3.2 c)

18 is the least number of A's required to get authentication. I determined this by gradually incrementing the A's starting from 13. When I provide 18A's, buffer overflow affects the variable, setting it to '1', which leads to bypass the checking of password and directly authenticates the user.

3.2 d)

If n is the number of A's required as a minimum to authenticate, then n-1 A's are insufficient to authenticate since the overflow isn't enough to reach and change the right variable in memory. Either the password is continually requested, or after a set number of tries, the program simply terminates.

3.2 e)

When I input exactly n number of A's (18 here), it does not extend far enough to corrupt critical data or even access invalid memory regions, so the program does not encounter a segmentation fault. Instead, it modifies the correct variable to a non-zero value, which leads to successful authentication.

3.3)

Description of the flow of CVE-2019-13726:

Use-after-free Vulnerability (UAF) is observed when a specific program continues to access memory that has already been deallocated. This will lead to a program not handling the lifecycle of an object in memory. CVE-2019-13726 describes UAF in a Chrome browser's password manager. This misconduct would be exploited by anomalies using web content, enabling them to execute arbitrary code which leads to compromising on the security of the system.

Its Impact:

The main problem pertaining to this is Unintended Code Execution. An anomaly will be able to inject a code snippet into the system which might lead to data breach (in this case the data is list of passwords, which is a high impact attack).

Execution of this code would possibly lead to a higher level of attacks like downloading and installing malwares into the local system.

Fix:

This vulnerability was addressed on patch release in Chrome version Chrome version 78.0.3904.87, by Google. This fix made sure there was proper memory management and no use of arbitrary memory. Refined testing procedures were built to prevent similar fore coming attacks.

EXERCISE-4: Web Security

4.1 a)

Cookie is a text file containing data. The information about the user visiting the website gets stored in it. This way, even when the user closes the browser and opens it again, the user doesn't need to put in his username and pwd on that website.

First Party Cookies:

First-party cookies are created and stored by the website when the user visits the particular website [20]. Every website the user visits stores information about his visit. They store information about current sessions, pages visited, or items in cart, etc. First-party cookies are set by the web server from the website's own webserver.

Second Party Cookies:

A second-party cookie would mean that the data being gathered on the site the user is on, will be sent to another site [21] because of a pre-arranged agreement between the two sites to share data.

Second-party cookies capture the same info as first-party cookies do.

Name: Giridhar Benkipur
URN: 6898345

Third party Cookies:

Third-party cookies are bits of information not created by and not stored by the website the user visits. However, they do store info regarding the user's behaviors. That is put into use by a third party. They normally include providers of advertising, analytics, and tracking services—like first-party cookies.

4.1 b)

Same Origin Policy: SOP is a security feature in web browser that will help prevent malicious websites from attacking other websites [22].

When a user opens a site, the site may have content like ad's loaded from a third-party domain. That's where the third-party cookies come in.

When the third-party content is loaded in the main website, those cookies can be set in the user's browser. The cookies get stored with the third-party's domain and not the website's domain.

Even though SOP prevents direct interaction between websites from different domains, browsers do allow third-party cookies to be set and read provided they are bound to the third-party's domain. This is what enables cross-site tracking and ad personalization.

So, third-party cookies bypass SOP by being embedded in the third-party's domain, and browsers automatically store and send them with applicable requests to the same domain. [23]

Examples:

User goes to my_website.com (first party). my_website.com embeds an ad from random_ads.com (third-party). random_ads.com sets a cookie in the user's browser that it can then access later when the user hits a different site that also requests ads from random_ads.com.

4.2 a)

In this task I created a basic website using PHP, SQLite, and HTML. Here the CSRF_Attacker.php simulates a malicious form submission to the application index.php and validate.php. When I try to insert values using the valid index.php page, the values are

successfully being added to the db. But when I use the CSRF_Attacker.php page to input the values, it gives me a token error, which is working as designed. So, this blocks any type of CSRF attack to legitimate applications. README.txt is attached with supporting files to execute the same.

4.2 b)

The code provided is with CSRF token validation. So, in this case, the code iterates and validates if it is a true CSRF token, and only then the values are added to the db. In case the token does not match with the generated token, the attempt will be unsuccessful and our application db will be untampered. The README.txt file has details on what to comment for a successful SCRF attack. However, the provided code detects and blocks the invalid token.

EXERCISE-5: The future of malware protection

I chose the paper to be: **SoK: Cryptojacking Malware**

5a)

Summary:

This paper provides an in depth overview of Cryptojacking, a variation of malware that specifically tries to exploit user's computational resources for cryptocurrency mining purposes, without the permission of the user. The authors stresses on different types of cryptojacking malwares, explaining about their mechanisms and various platforms the attacker tries to exploit. The authors also emphasize the need for more in depth understanding of the processes as the current detection mechanisms and technologies are lagging with respect to the complexities of attacks that we foresee in the near future.

The study also highlights about the increasing number of incidents related to cryptojacking and many mining scripts/programs available from different service providers. The paper concludes by identifying good research directions, checking the possibility of coming up with very advanced detection mechanisms and better understanding of the challenges of cryptojacking.

Future challenges for malware detection:

- Dynamic behavior of cryptojacking:

The big challenge is the dynamic nature of cryptojacking, where malware might change its operational parameters quite frequently. Accordingly, detection systems must adapt to such fluidity in real time. Challenges were faced while identifying cryptojacking websites as mentioned in [24].

- Challenges in resource consumption and malware detection:

The problem is the malware detection software's/algorithms must be able to differentiate between these two, if a high resource computing valid application is running or it's a potential cryptojack attack. Justification for these points is mentioned in [25].

- Cross Environment threats:

As Cryptojacking attack can be done on multiple platforms like Windows, Linux, MacOS and even of IoT devices, there has be systematic applications/software's on all varieties of OS to successfully encounter the reduce the effect of Cryptojacking. [26] talks about multiplatform attacks.

5c)

References for the entire coursework:

[1] Jordyn Calderon, “Understanding Geo-Redundant Storage Benefits”, https://nfina.com/geo-redundant-storage/ , assessed: Oct 2024

[2] “File Upload Protection – 10 Best Practices for Preventing Cyber Attacks”, https://emtmeta.com/file-upload-protection-10-best-practices-for-preventing-cyber-attacks/ assessed: Jan 2022

[3] David Bunting, “The Top 5 Security Logging Best Practices to Follow Now”, https://www.chaossearch.io/blog/cyber-security-logging assessed: Jul 2024

[4] Data Redundancy Exploits, “How to Avoid the Risks of Data Redundancy” https://dotsecurity.com/insights/blog-data-redundancy-risks accessed: April 2022

[5] Breach through Protocol failure “Rethink Security of Your Cloud File Transfer Platform”, https://www.goanywhere.com/blog/rethink-security-your-cloud-file-transfer-platform accessed: Nov 2024

[6] Failure of Cryptographic Protocols, “OWASP Top 10 Cryptographic Failures Explained”, https://www.securityjourney.com/post/owasp-top-10-cryptographic-failures-explained accessed: Aug 2023

[7] Wiretapping, “Leading Security Experts Say FBI Wiretapping Proposal Would Undermine Cybersecurity”, https://cdt.org/insights/leading-security-experts-say-fbi-wiretapping-proposal-would-undermine-cybersecurity/ , accessed: May 2013

[8] Target system logging and monitoring, “Top seven logging and monitoring best practices”, https://www.blackduck.com/blog/logging-and-monitoring-best-practices.html , accessed: Nov 2021

[9] David Kirkpatrick, “Mongodb - Security Weaknesses in a typical NoSQL database”, https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/mongodb-security-weaknesses-in-a-typical-nosql-database/ , accessed: March 2013

[10] “Enhancing Application Security: Key Insights into Insufficient Logging & Monitoring”, https://cyberrisk-countermeasures.info/2024/06/04/enhancing-application-security-key-insights-into-insufficient-logging-monitoring/ accessed: June2024

[11] “Lapadula Model”, https://www.sciencedirect.com/topics/computer-science/lapadula-model , accessed: 2011

[12] “Understanding UNIX permissions and file types”, https://unix.stackexchange.com/questions/183994/understanding-unix-permissions-and-file-types

[13] “Using Burp to Test for Security Misconfiguration Issues”, https://portswigger.net/support/using-burp-to-test-for-security-misconfiguration-issues

[14] “Cybersecurity Myths: Why Complacency is Not an Option”, https://www.arcticsecurity.com/resources/cybersecurity-myths-why-complacency-is-not-an-option

[15] “The Dark Side of Burp Suite: How Misusing Penetration Testing Tools Can Harm Your Web App”, https://blog.poespas.me/posts/2024/08/13/penetration-testing-misusing-burp-suite-for-web-app-scanning/ , accessed: Aug 2024

[16] “Stack Canaries – Gingerly Sidestepping the Cage”, https://www.sans.org/blog/stack-canaries-gingerly-sidestepping-the-cage/ , accessed: Feb 2021

[17] “What is a prompt injection attack?” https://www.ibm.com/topics/prompt-injection

[18] “Extracting Training Data from ChatGPT”, https://not-just-memorization.github.io/extracting-training-data-from-chatgpt.html , Nov 2023

[19] “Buffer Overflow Attacks Explained”, https://bluegoatcyber.com/blog/buffer-overflow-attacks-what-you-need-to-know/

[20] Bas Gosewisch “What are first, second and third party cookies?”, https://basgosewisch.com/first-second-and-third-party-cookies-what-it-all-means/ , accessed: April 2024

[21] “Second-Party Cookies:”, https://coschedule.com/marketing-terms-definitions/second-party-cookies

[22] “Same-origin policy (SOP)”, https://portswigger.net/web-security/cors/same-origin-policy

[23] “SameSite cookie recipes”, https://web.dev/articles/samesite-cookie-recipes

[24] Hadeel A. Almurshid, “A Holistic Intelligent Cryptojacking Malware Detection System”, https://ieeexplore.ieee.org/document/10738793 , assessed: Oct 2024

[25] Josh Lake, “What is cryptojacking (with examples) and how do you stop it?”, https://www.comparitech.com/blog/information-security/cryptojacking/ , assessed: March 2020

[26] Miguel Hernández, “LABRAT: Stealthy Cryptojacking and Proxyjacking Campaign Targeting GitLab” https://sysdig.com/blog/labrat-cryptojacking-proxyjacking-campaign/ , assessed: Aug 2023

[27] “SQL Server, Part 3: Adopting the principle of least privilege”, https://blogs.manageengine.com/it-security/it-security-passwordmanagerpro/2020/05/11/sql-server-part-3-adopting-the-principle-of-least-privilege.html , accessed: May 2020

[28] Sagar Sharma, “Everything Essential About the tmp Directory in Linux”, https://linuxhandbook.com/tmp-directory/ , May 2024

[29] Kai Yuan, “Advanced File Permissions in Linux”, https://www.baeldung.com/linux/advanced-file-permissions , March 2024

[30] “Lecture Notes: Basics of Buffer Overflows”, https://cs4401.walls.ninja/notes/lecture/basics_bufferoverflows.html

[31] “Stat Command in Linux”, https://linuxize.com/post/stat-command-in-linux/ , accessed: Dec 2020

[32] “What is Role-Based Access Control (RBAC)? Examples, Benefits, and More”, https://www.digitalguardian.com/blog/what-role-based-access-control-rbac-examples-benefits-and-more , Aug 2018

[33] “Role-Based AC”, https://csrc.nist.gov/CSRC/media/Presentations/Role-based-Access-Control-an-Overview/images-media/alvarez.pdf