THE UNIVERSITY OF SYDNEY

Assignment 1

INFO3616: Principles of Security and Security Eng

510415022

Problem 1. Fundamentals of Security Engineering (25 marks)

a) **Definitions** **(5 marks)**

Safety engineering is the engineering of systems in such a way as to minimise risk and harm *when failures do occur.* It involves hazard identification, risk analysis, and fail-safe design mechanisms. For example, ensuring that a nuclear power plant shuts down safely in the case of an accident falls under safety engineering.

While safety engineering is focused on what occurs after a failure, reliability engineering is about minimising system failures in the first place. Its aim is to allow a system to perform its intended function consistently over time. For example, ensuring that a server has 99.9% uptime is a task of reliability engineering.

Security engineering aims to ensure that systems behave as intended even in the face of malicious actors, accidents, or unpredictable events. Unlike safety and reliability engineering, which primarily address unintentional failures, security engineering explicitly addresses *deliberate harm.*

b) **Conceptual Framework** **(10 marks)**

i. The four components of the conceptual framework for security are:

A. Policy: Defines the security goals of the system, or in other words, what it means for the system to be secure.

B. Mechanisms: Outlines the technical machinery available in order to enforce the policies. For example, cryptographic hashes provide data integrity.

C. Assurance: Describes the limits of each mechanism and the extent to which we can rely on it. For example, estimating how long it would take to brute-force a 10-character password.

D. Incentives: Outlines the motives one may have for protecting or attacking the system. This helps us understand the degree of security our system needs.

ii. A. Policy: The system must uphold key security goals:

- Confidentiality: only authorised employees with multi-factor authentication can access client health data in plaintext form.
- Integrity: data and access logs must remain unmodified except by authorised actions.
- Authenticity and Authorisation: access requests must be verified as genuine and only permitted for employees with the right privileges.
- Accountability and Non-repudiation: access logs must reliably record which user performed each action, preventing denial of responsibility.

- Availability: the system should provide secure access for employees when required, even in the face of attempted attacks.

B. Mechanisms: These policies are enforced by multi-factor authentication combining passwords and biometric scans, along with regular security audits and monitoring of access logs.

C. Assurance: The biometric system has a false acceptance rate of 0.001% and a false rejection rate of 0.5%. The combination of biometric scans and passwords improves security by at least 100 times compared to using passwords alone.

D. Incentives:

- For defending the system: Employees are motivated to maintain strong security practices such as regularly updating passwords and enabling MFA through job protection and bonuses. The company itself is incentivised to protect its reputation and contracts.
- For attacking the system: There are strong financial motives to compromise the system, such as selling sensitive healthcare data on black markets or using it for ransom.

c) **Security Goals** **(10 marks)**

i. MOVEit Breach, 2023

- **Compromised security goal:** Confidentiality
- **Explanation:** Attackers exploited a vulnerability in Progress Software's *MOVEit* software, a widely-used file transfer system (Kapko & Himmel, 2024). This allowed them access to the personal data of over 90 million people. The MOVEit software failed to ensure data confidentiality because attackers were able to read and steal private data that should only have been accessible to authorised users and systems.

ii. Okta Support System Breach, 2023

- **Compromised security goal:** Authorisation
- **Explanation:** A service account credential was accidentally stored in an employee's personal Google account, which was later compromised. The attacker used these credentials to access Okta's customer support system and obtain files containing active session tokens. With these tokens, the attacker was able to perform actions reserved for legitimate customers (Asaff, 2024; Burns, 2023). Authorisation was compromised because the system incorrectly allowed an unauthorised party to access sensitive resources by abusing valid but stolen credentials.

iii. MGM Resorts Cyberattack, 2023

- **Compromised security goal:** Availability
- **Explanation:** Attackers used social engineering to impersonate an MGM IT support employee and gain administrative privileges. They then deployed ransomware across MGM's servers, disabling reservation systems, digital room keys, slot machines, parking systems, and more (Braithwaite, 2023; Schrader, 2025). Availability was compromised because critical MGM systems were taken offline, preventing hotel guests from accessing essential services.

iv. Medibank Breach, 2022

- **Compromised security goal:** Confidentiality
- **Explanation:** Attackers used stolen privileged credentials to gain access to Medibank's internal systems and download about 200GB of sensitive customer data. The breach exposed the personal and medical information of almost 10 million customers, including names, birth dates, passport numbers, and Medicare claim details (Kost, 2024; Robertson, 2024). Confidentiality was compromised because highly sensitive records, which should only have been accessible to Medibank and its authorised users, were stolen and later leaked on the dark web.

v. SolarWinds Supply Chain Attack, 2020

- **Compromised security goal:** Integrity
- **Explanation:** Attackers compromised SolarWinds' *Orion* software updates by inserting malicious code into legitimate signed updates. When customers, including U.S. government agencies and global organisations, installed these updates, the malware created a backdoor for remote access (Oladimeji & Kerner, 2023; "SolarWinds", 2020). Integrity was compromised because trusted software updates were tampered with, and digital signatures gave users false assurance that the software was authentic and unmodified.

Problem 2. Social Engineering in Practice (25 marks)

- **Email Address:** `ojxfslgc@example.com`
- **Plaintext Password:** `archiesprings2019@`
- **MD5 Hash:** `0ef450f543d7f77b2b7a92100b0c2107`

My script `hack.py` conducts a dictionary-style password search using keywords manually extracted from the target's Twitter profile.

1. **Keyword selection:** A set of candidate words and tokens are placed in the text file `keywords.txt`. These have been manually chosen from the target's Twitter profile.
2. **Candidate generation:** The script generates all possible concatenations of the kewords (with repetition) using the Python standard library function `itertools.product`. Each candidate is validated to ensure it is fewer than 20 characters and contains at most one special character.
3. **Hashing:** The candidate password is hashed using the MD5 algorithm (sourced from `pycryptodome`).
4. **Dataset lookup:** The candidate's MD5 hash is compared against the breach dataset. On finding a match, the program outputs the corresponding email address, plaintext password, and hash, then terminates.

The program is executed from the command line with:

```
python hack.py
```

Problem 3. Access Control (25 marks)

a) **Basics** **(15 marks)**

i. Access control is often categorised into two forms: Discretionary Access Control (DAC) and Mandatory Access Control (MAC). Both are based on the principle that *subjects* (such as users or processes) request access to *objects* (such as files, databases, or devices).

- **DAC:** Responsibility for access is placed on the owner of the object. The owner can decide who else may read, write, or execute the object, giving DAC a high degree of flexibility. However, this flexibility introduces risk: owners may forget to revoke permissions, accidentally grant access to the wrong user, or simply lack the technical expertise to manage permissions securely.
- **MAC:** Access decisions are governed by a central authority, usually defined by a security policy or administrator, rather than by individual object owners. A "security kernel" enforces the rules by checking whether each subject is allowed to interact with each object. This model is less flexible but much more rigorous, as it removes human error from day-to-day access decisions and ensures that all access follows a consistent policy.

ii. Cloud-based storage solutions use elements of both DAC and MAC for defining access control policies.

For example, the owner of a file in OneDrive or Google Drive can decide who to share it with and at what permission level (edit, review-only, read-only, etc.). They have fine-grained control on things such as the sharing mechanism (link sharing vs email-only) and can revoke access at any time. This behaviour is characteristic of *discretionary* access control.

However, in enterprise settings, administrators can enforce organisation-wide security policies, such as limiting external sharing or setting expiry dates for shared links. In these cases, access decisions are governed by centralised policies rather than by individual owners, indicative of *mandatory* access control.

iii. **Privileged execution:** The x86 architecture enforces access control using *protection rings*, which define privilege levels from 0 (most privileged) to 3 (least privileged). In practice, the OS kernel executes in ring 0, while user applications run in ring 3. The CPU prevents lower-privileged code from directly executing higher-privileged instructions or accessing protected memory, embodying the principle of least privilege. Any such attempt triggers a general protection fault, which the OS handles to maintain strict separation between user processes and the kernel.

Isolated cryptographic components: In x86 architectures, sensitive cryptographic material such as encryption keys are stored in dedicated hardware components such as the Trusted Platform Module instead of main memory. This restricts access to the keys so that only a few authorised functionalities can use them through a dedicated interface. This design ensures that even if the operating system is compromised, attackers cannot simply extract cryptographic secrets from main memory.

iv. a. **Rule-based access control (RuBAC):** In rule-based access control, access is granted or denied according to a set of predefined rules, rather than a user's role. These rules are often expressed in "if-then" logic, such as allowing or blocking requests based on IP address, time of day, or type of action attempted. Unlike RBAC, RuBAC does not consider a user's security clearance if a rule is violated, making it more rigid but also more precise. *Example:* Network firewalls commonly use rule-based control, e.g., "if the source IP is outside the corporate network and the port is 22, then deny the connection."

b. **Attribute-based access control (ABAC):** Attribute-based access control makes access decisions based on attributes of the subject (e.g., job title, clearance level), the object (e.g., document classification), and environmental conditions (e.g., time, location, device security posture). Policies are defined in terms of these attributes, which allows highly dynamic and context-aware access decisions. ABAC provides greater flexibility than RBAC, since permissions are not tied to static roles but to conditions at the time of access. *Example:* A healthcare system may allow doctors to access patient records only if they are currently on duty and physically located within the hospital's network.

b) **Security Policy Models – Definitions** **(2 marks)**

The Bell-LaPadula model aims to address the security goal of *confidentiality*. It enforces:

(a) Simple Security Property ("no read up"): no principal may read data at a higher security level

(b) Star (*) Security Property ("no write down"): no principal may write data to a lower security level

This stops information from leaking downwards in the multilevel security policy.

In comparison, the Biba model addresses *integrity*. It does this through a reverse of Bell-LaPadula:

(a) Simple Integrity Property ("no read down"): a subject may not read data from a lower integrity level.

(b) Star (*) Integrity Property ("no write up"): a subject may not write information to a higher integrity level.

This prevents untrustworthy data from flowing upwards in the hierarchy.

c) **Security Policy Models – Example** **(8 marks)**

i. **False:** In a Bell-LaPadula model, Sarah cannot read the file `financial_report.txt` because she has an *Internal* clearance level, while the file requires *Confidential*, which is higher. Bell-LaPadula enforces the *no read up* rule, so access is denied.

ii. **True:** In a Biba model, Michael, who has a *Confidential* clearance, can edit `company_memo.txt`, which only requires *Internal.* Biba enforces no write up but permits write down, so writing to a lower integrity level is allowed.

iii. **False:** In a Bell-LaPadula model, Thomas, who has a *Top Secret* clearance, cannot copy the contents of `strategic_plan.txt` (Top Secret) into `company_memo.txt` (Internal). This would be writing down to a lower security level, which Bell-LaPadula forbids to prevent information leakage.

iv. **False:** In a Biba model, Emma, who has a *Secret* clearance, cannot modify `strategic_plan.txt`, which requires *Top Secret.* Biba enforces no write up, so subjects cannot alter objects at a higher integrity level.

Problem 4. Linux Access Control (25 marks)

a) **Basic Access Control** **(6 marks)**

i. UID: 1003.[1]

```
id -u sheppard
```

ii. GID: 1006.[2]

```
getent group scientists
```

iii. The user `carter` belongs to groups `carter`, `sudo`, `humans`, `military`, `scientists`, `pegasus`.[3]

```
groups carter
```

iv. The users in group `humans` are `carter`, `sheppard`, `weir`, and `mckay`.

```
getent group humans
```

v. No, `ronan` does not have sudo access.

```
sudo -l -U ronan
```

vi. Yes, `carter` has sudo access.

```
sudo -l -U carter
```

b) **File Permissions** **(10 marks)**

i. Non-hidden files owned by `teyla`:

```
find / -type f -user teyla ! -name ".*" 2>/dev/null

> /home/teyla/gate_addresses.txt
> /home/teyla/this_year_trade.txt
> /opt/training_manual.txt
> /var/tmp/herbal_recipes.txt
> /var/log/hidden_gate.txt
> /etc/ancient_knowledge.txt
```

ii. All the files owned by `teyla` and associated with the group `ancients`:

```
find / -type f -user teyla -group ancients 2>/dev/null

> /home/teyla/gate_addresses.txt
> /etc/ancient_knowledge.txt
```

[1]With assistance from AskUbuntu
[2]With assistance from StackOverflow
[3]With assistance from AskUbuntu

iii. First we find a file owned by `mckay` and check its permissions:

```
F=$(find / -type f -user mckay 2>/dev/null | head -n 1)
ls -l $F

> -rwxrw-r-- 1 mckay scientists 0 Aug 11 01:32 /lab/
    research/energy_readings.txt
```

This output tells us that:

- The owner `mckay` has read, write and execute permissions
- The group `scientists` has read and write permissions
- Everybody else only has read permissions

Therefore, `carter` and `ladon` can only write to the file if they are in the group `scientists` OR if they have `sudo` access. From Problem 4 Part a) we know that `carter` satisfies these conditions. Let's check if `ladon` fulfils these as well:

```
groups ladon

> ladon : ladon geniis

sudo -l -U ladon

> User ladon is not allowed to run sudo on lab22YQ9P.
```

As `ladon` is not in the `scientists` group and does not have `sudo` access, only `carter` can write to the file.

iv. Examining the file permissions in `kolya`'s home directory:

```
cd ../kolya
ls -l

> --wx---r-x 1 kolya geniis 29 Aug  8 05:32 genii_launch.
    sh
> --wx-----x 1 kolya geniis 29 Aug  8 05:36
    genii_pre_launch.sh
```

This shows that:

- For file `genii_launch.sh`, `kolya` can write and execute, the group `geniis` has no permissions, and everyone else can read and execute.
- For file `genii_pre_launch.sh`, `kolya` can write and execute, the group `geniis` has no permissions, and everyone else can only execute.

Hence, `kolya` can execute both files. As `ladon` is in the group `geniis` (see Part iii. above), they cannot execute either of the files. Finally, we need to check if `todd` is in the `geniis` group:

```
groups todd

> todd : todd
```

So, as `todd` falls into the “other” category, he can execute the file.

c) **Directory Permissions** **(9 marks)**

i. `/mission_reports` is owned by `root` and belongs to the group `root`.

ii. Yes, I was allowed to delete a file.

iii. No, it says `Operation not permitted`.

iv. The directory `/military_reports` has permissions `drwxrwxrwx` meaning that any user can create or delete files in the directory. In contrast, the directory `/science_reports` has permissions `drwxrwxrwt` – the difference being that the sticky-bit is set. This sticky bit restricts modifications to files to their owner. Since the files inside `/science_reports` are owned by `ford`, `ladon` cannot delete them.

v. The `/tmp` directory also has permissions `drwxrwxrwt` with the sticky bit set. This allows any user or process to create and store temporary files in this directory while simultaneously preventing processes from interfering with each other's files. If the sticky bit was not set, any user could delete or overwrite another user's temporary data, leading to a breach of key security goals such as data integrity.

References

Asaff, K. (2024). Unpacking the okta data breach. *Portnox*. https://www.portnox.com/blog/cyber-attacks/unpacking-the-okta-data-breach/

Braithwaite, S. (2023). Alphv: Hackers reveal details of mgm cyber attack – westoahu cybersecurity. *University of Hawaii*. https://westoahu.hawaii.edu/cyber/global-weekly-exec-summary/alphv-hackers-reveal-details-of-mgm-cyber-attack/

Burns, N. (2023). The 2023 okta data breach. *www.metacompliance.com*. https://www.metacompliance.com/blog/cyber-security-awareness/okta-data-breach

Kapko, M., & Himmel, J. (2024). Progress software's *MOVEit* meltdown: Uncovering the fallout. https://www.cybersecuritydive.com/news/progress-software-moveit-meltdown/703659/

Kost, E. (2024). What caused the medibank data breach? *UpGuard*. https://www.upguard.com/blog/what-caused-the-medibank-data-breach

Oladimeji, S., & Kerner, S. M. (2023). Solarwinds hack explained: Everything you need to know. *TechTarget*. https://www.techtarget.com/whatis/feature/SolarWinds-hack-explained-Everything-you-need-to-know

Robertson, J. (2024). Medibank allegedly failed to heed it warnings before one of australia's worst cybersecurity breaches. *ABC News*. https://www.abc.net.au/news/2024-06-22/medibank-alerts-australia-cybersecurity-breach/104003576

Schrader, D. (2025). An overview of the mgm cyber attack. *Netwrix.com*. https://blog.netwrix.com/mgm-cyber-attack

Solarwinds. (2020). *Wikipedia*. https://en.wikipedia.org/wiki/SolarWinds