**Week 3: Workshop Session**

**Title: Integrity**

**Expected Learning Outcomes:**

Understand the concept of data integrity.

Understand the concept of data checksum.

Data integrity measurement using hash functions, their limitations and mitigations.

Q1: Explain in your own terms the concept of data integrity.

Data integrity is ensuring that data has not been modified knowingly or unknowingly throughout each stage of its lifecycle (e.g., at rest, in transit or in process)

For example, in Law Enforcement, ensuring the integrity of the chain of custody is essential. This chain of custody ensures that the evidence obtained from an investigation has not been tampered with or modified in any way. In digital forensics this is typically enforced via cryptographic hash functions. Typically, when a hard drive is seized, a checksum of the computer’s hard drive is taken to ensure it has not been modified after being seized.

Q2: What are in your own words threat to data integrity?

There are 2 primary types of threats to Data Integrity, which are **active** (on purpose), or **passive** (by accident).

Passive threats, are essentially **accidental** changes to data.

For example, the TCP protocol has a mechanism to detect packet losses (via sequence numbers) and errors by utilising per-segment checksums. Retransmission then occurs to ensure the integrity of the end-data.

Another example is RAID (Redundant Array of Independent Disks) levels, which are typically used in SAN (Storage Area Networks). In RAID levels 2-6, a **parity** bit, byte or disk is used to detect errors across the disks.

Active threats are purposeful changes to data.

This can also happen at each stage in the data’s lifecycle, but the primary risk comes when data is in transit. For example, when downloading an application from the internet, the data could be modified in transit by a threat actor and the original application could be replaced with a malicious one. (This is slightly contrived, as the data is likely to be encrypted. Data encryption does not protect against data integrity as encrypted data can still be modified, but an attacker would have to break the encryption in order to introduce a malicious program)

Q3. What are cryptographic hash functions?

Cryptographic hash functions act as a form of “fingerprint” for data.

They can be applied to data of arbitrary length in order to produce a fixed-length and (arguably) unique output.

Strong hash functions **must** produce unique outputs for each unique input, but some algorithms (still widely used) such as SHA1 and MD5 have been proven insecure as 2 different inputs have produced the same output. Additionally NIST (National Institute of Standards and Technology) have formally retired SHA1 [NIST Retires SHA-1 Cryptographic Algorithm | NIST](https://www.nist.gov/news-events/news/2022/12/nist-retires-sha-1-cryptographic-algorithm)

Q4: What are properties of hash functions?

Source [Randomized hashing for digital signatures (nist.gov)](https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-106.pdf)

**Secure hash functions:**

* Are one-way only (Preimage resistance)
* Produce unique checksums when using 2 distinct inputs of any length(Collision Resistance)
* Computationally infeasible to find a message (V1) that generates the same checksum as a known message (V2) (Second preimage resistance)
* Generate a fixed length output (which varies on the algorithm used)

Q5. How are hash functions applied in real-life?

Passwords:

* To avoid storing passwords in plaintext, passwords are hashed (and typically salted) before being stored.

Digital Signatures:

* HTTPS certificates, Software certificates, etc all utilise hashing functions as part of the overall digital signature. Typically, a hash of the original document or file is generated and then encrypted with the **signer’s** private key. The signer then presents their **public** key, which can be used to decrypt the encrypted checksum, and compare it with the document.

Checksums:

* In the chain of custody (especially in digital forensics) hash functions are used to ensure that files have not been modified by investigators (ensuring the integrity of the data).
* File-integrity monitoring (typically used to detect changes to system critical / sensitive files)

Q6. What is the use of salts in generating hash functions?

Salts in cryptographic hashes function similarly to IVs (Initialisation Vectors) in encryption. Salts are typically a random value appended to the given input in order to generate different outputs when the same input is given.

e.g. When given 2 **identical** inputs they will generate different checksums.

These are typically used for passwords, as introducing a salt reduces the effectiveness of rainbow table or lookup attacks.

For example the **unsalted** MD5 hash of password is 5f4dcc3b5aa765d61d8327deb882cf99, if this hash is found in a database then there’s no need for an attacker to waste resources brute-forcing it (using tools like Hashcat or John the Ripper).

Practice exercises One:

1. Go to: https://gchq.github.io/CyberChef/
2. Type your full name and student ID.
3. Generate different hash values (message digest) of the input.
4. Count the MD and explain why the length of the MD differs in different hash functions.

Different hash functions have different lengths due to the need to balance security vs efficiency.

For example, the CRC32 “hash” function is typically fast, but produces a smaller digest (32 bits) which leads to collisions. This typically isn’t an issue, as CRC32 is not designed to be cryptographically secure, and detect **active** modifications by a threat actor. Instead it’s typically used for detecting **random** (passive) errors introduced during data transmission or storage.

Another example, is the SHA1 hash (producing a 160 bit checksum/40 hex chars). This hash function **was** widely used as a checksum, in digital signatures, digital time stamps and other used, until it’s insecurity was highlighted. Producing a collision in February 2017 (<https://eprint.iacr.org/2017/190.pdf>) by Marc Stevens et al. Due to this, a new family of hashes were created. SHA-2, the most widely used version being SHA-256, producing a checksum of 256 bits / 64 hex chars. This family is now recommended to be used for Digital signatures and checksums, although usage in passwords are generally discouraged. This is due to the speed of the hashing function, making it much easier for an attacker to brute force the hash.

For passwords, salted hashing algorithms (like yescrypt, the new default algorithm used in Debian systems) that are computationally expensive to attack are typically recommended by bodies like NIST.

Practice exercise Two: Immersive labs

1. Introduction to Hashing
2. Cyber Million: Hashing and Encoding
3. Linux CLI: Ep. 15 – Generating File Hashes
4. OWASP 2021: Ep.8 – Software and Data Integrity Failures
5. NIST 800-53: Ep.19 – System and Information Integrity
6. Secure Fundamentals: The CIA Triad

**Workshop Reflective Writing**

Reflect on your learning this week and put them down across these subheadings:

1. Depth and breadth of understanding. This is all about putting down some details of what you learnt and did not understand well in this week’s session. What will you do going forward on areas you did not comprehend during the session?
2. Critical information, contextualising the session’s learning components to assignment development.
3. Personal thoughts, subjective analysis of personal development during this week’s session.

Eddy, W.M. (n.d.). *RFC 9293: Transmission Control Protocol (TCP)*. [online] www.ietf.org. Available at: https://www.ietf.org/rfc/rfc9293.html.

‌