SD Card Forensics on ESP32

RAS Minor Mini-Project - 1 PC408

Phase 3 : Forensic Protection using SHA256

Mentored by : Dr. Tapas Kumar Maiti

Jhil Patel

11.12.2025
SID : 202301090

INTRODUCTION

Phase 3 extends the basic ESP32–SD card data logger developed earlier by introducing forensic security mechanisms that protect the integrity of stored data.

In Phase 2, the system was able to log, view, append, and overwrite data, but any manual or external tampering went undetected, as demonstrated earlier. On re-opening the CSV file after altering values manually, the system treated tampered and authentic entries identically.

To overcome this vulnerability, Phase 3 integrates cryptographic hashing (SHA-256), chain-based integrity verification, NVS-based final hash storage, and HTTP-based tampering simulation, transforming the simple logger into a forensically robust IoT evidence recorder.

OBJECTIVES

- Phase 3 aims to introduce a complete forensic layer on top of the Phase-2 hardware system:
- Detects any modification, deletion, insertion, or cloning of SD card entries.
- Secure each log entry using a hash chain where every row depends on the previous one.
- Store final hash securely in ESP32's NVS to detect SD card replacement.
- Allow controlled tampering simulation through HTTP endpoints for demonstration.
- Use tools like Wireshark to inspect packet flow and show how network-layer attacks or unauthorized HTTP tampering can be detected.
- Maintain full backward compatibility with logging features implemented in earlier phases.

PREVALENT DATA TAMPERING METHODS

Digital logging systems using removable SD cards are vulnerable to several forms of manipulation. Common tampering methods include:

1. Manual File Editing (Physical Tampering)

If someone gains physical access to the SD card, they can open `data.csv` on a computer and change, delete, or replace entries. This includes modifying sensor values, inserting fake rows, or rewriting the entire file.

2. Unauthorized Serial Command Injection

Attackers with temporary USB access can send fake serial commands such as `append 0` to insert false readings or disrupt normal logging. Phase-2 had no mechanism to detect this.

3. SD Card Swapping or Cloning

A cloned or altered SD card can be inserted into the device. Without a verification mechanism, the system cannot distinguish genuine logs from forged ones.

4. Network-Based Tampering (When HTTP Access Exists)

If HTTP endpoints are exposed without authentication, attackers may modify logs remotely (e.g., editing lines, appending fake entries, or overwriting files). These attacks simulate real-world IoT vulnerabilities.

5. Firmware Manipulation

If the firmware is not protected, an attacker can reflash modified code that alters logging behavior, disables checks, or writes fabricated values.

6. Timestamp Interference

Altering system time—either through RTC manipulation or NTP spoofing—can create misleading or out-of-order log sequences, affecting forensic accuracy.

Overall, these risks show why cryptographic protections such as **SHA-256 hash chaining** are necessary to make tampering detectable and maintain the integrity of digital evidence.

EXISTING INDUSTRY METHODS FOR DATA INTEGRITY

To prevent tampering and ensure legal admissibility, several standard forensic practices are widely used:

1. Chain of Custody (CoC)

Every transfer of digital evidence is recorded with date, time, handler identity, and purpose. Evidence is stored in tamper-evident packaging or secure lockers to prevent unauthorized access.

2. Forensically Sound Collection

Investigators create exact bit-level copies of original media using write-blockers to avoid modifying the source. The original device is handled as little as possible to preserve integrity.

3. Data Integrity Verification

Cryptographic hash functions (such as MD5 or SHA-256) are used to generate a unique fingerprint for each file. Any later alteration results in a completely different hash, making tampering immediately detectable.

4. Secure Storage and Access Control

Digital evidence is stored in encrypted form and protected with strict access controls. Only authorized personnel can access the data, often enforced using multi-factor authentication.

5. Comprehensive Documentation and Logging

Every action—collection, examination, copying, and transfer—is documented thoroughly. Audit logs allow investigators to trace any interaction with the digital evidence.

METHOD IMPLEMENTED IN THIS WORK - SHA256

In this project, a cryptographic hash-chaining method was implemented on the ESP32–SD card logging system to detect any form of data tampering. Every new log entry stores not only the timestamp and temperature value but also the hash of the previous entry. This creates a continuous chain where each record depends on the one before it.

Any modification—whether editing a value, deleting a line, inserting fake readings, or swapping the SD card—breaks this chain and is immediately detected during verification. The final hash is also stored securely in the ESP32's NVS memory, enabling detection of SD card cloning or replacement.

This approach transforms a simple data logger into a tamper-evident recording device, aligning it with forensic practices where integrity, traceability, and verifiability are essential.

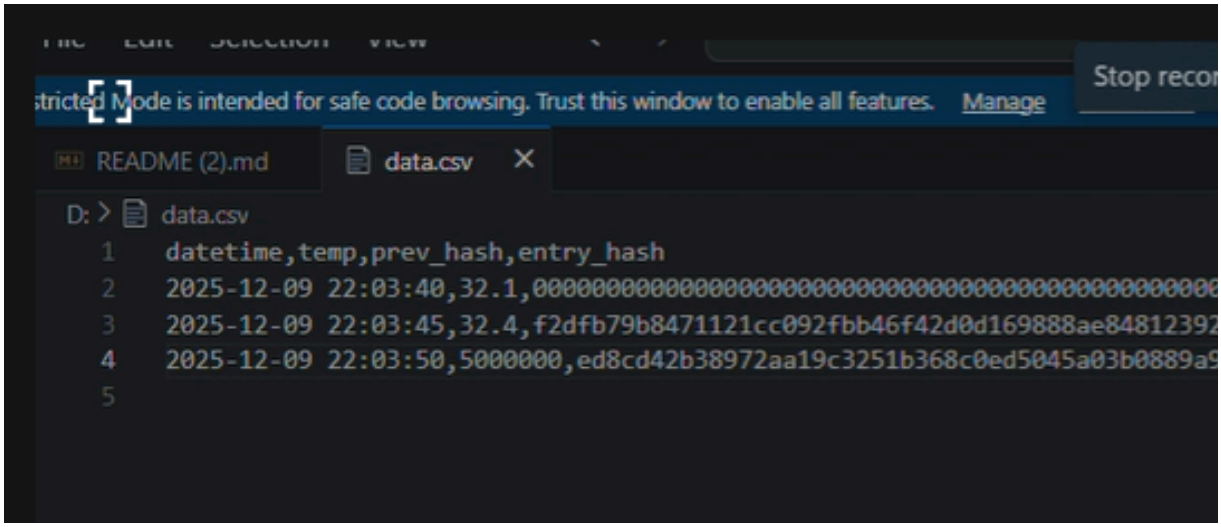
IMPLEMENTATION WALKTHROUGH

The following screenshots illustrate how the ESP32 processes entries, stores chained hashes, and reacts when tampering is attempted.

In normal flow the temperature sensor data is logged every 5 sec with Date-Time stamp.

```
22:02:44.578 -> ----- FILE CONTENT -----
22:02:44.578 -> datetime,temp,prev_hash,entry_hash
22:02:44.578 ->
22:02:44.578 -> ------------------------
22:02:44.578 ->
22:02:44.578 -> Chain OK ✓ No tampering detected.
22:02:44.578 ->
22:02:44.578 -> === INIT MENU ===
22:02:44.578 -> show | append | overwrite | reset
22:02:51.507 ->
22:02:51.507 -> APPEND MODE STARTED
22:02:51.554 ->
22:02:51.554 -> Runtime:
22:02:51.554 -> show | append <value> | stop
22:02:56.545 -> Logged: 2025-12-09 22:02:56,30.3
22:03:01.537 -> Logged: 2025-12-09 22:03:01,24.6
22:03:06.529 -> Logged: 2025-12-09 22:03:06,34.1
22:03:11.535 -> Logged: 2025-12-09 22:03:11,23.8
22:03:16.577 -> Logged: 2025-12-09 22:03:16,32.5
```

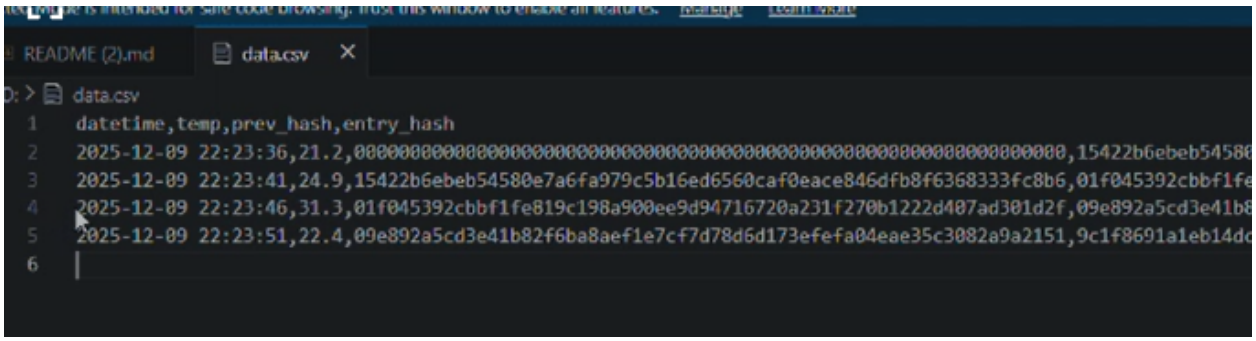
Manually altering data file

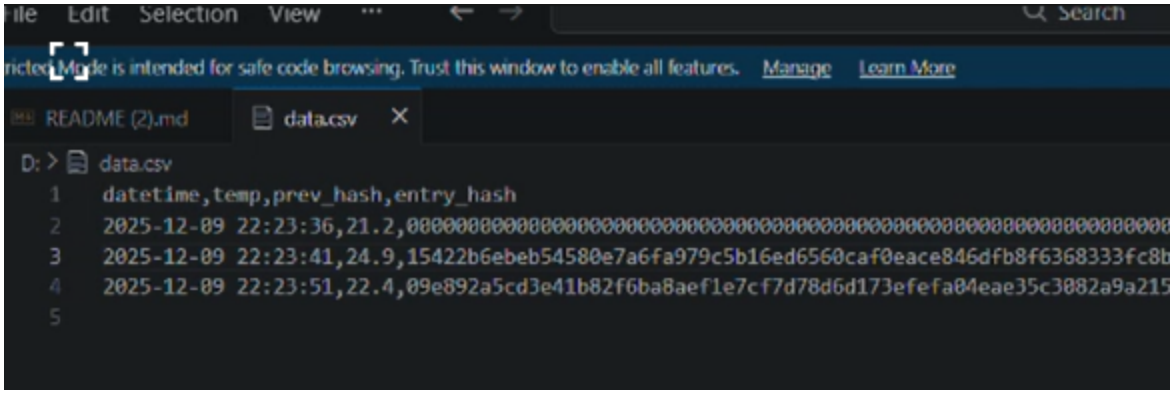


Tampering detected

```
22:21:16.456 -> === INIT MENU ===
22:21:16.456 -> show | append | overwrite | reset
22:21:21.695 ->
22:21:21.695 -> ----- FILE CONTENT -----
22:21:21.695 -> datetime,temp,prev_hash,entry_hash
22:21:21.695 -> 2025-12-09 22:03:40,32.1,0000000000000000000000000000000000000000000000000000000000000000
22:21:21.695 -> 2025-12-09 22:03:45,32.4,f2dfb79b8471121cc092fbb46f42d0d169888ae848123925
22:21:21.743 -> 2025-12-09 22:03:50,5000000,ed8cd42b38972aa19c3251b368c0ed5045a03b0889a98
22:21:21.743 ->
22:21:21.743 -> ------------------------
22:21:21.743 ->
22:21:21.743 -> TAMPERED at line 3 (hash mismatch)
22:21:21.743 ->
```

Deleting data line

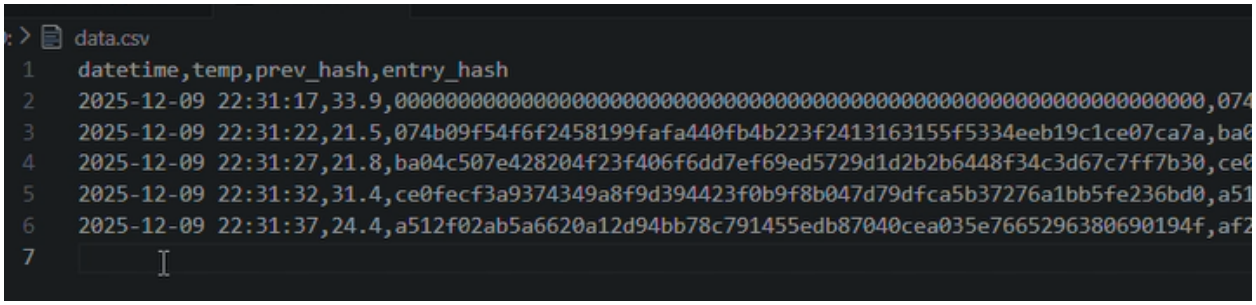


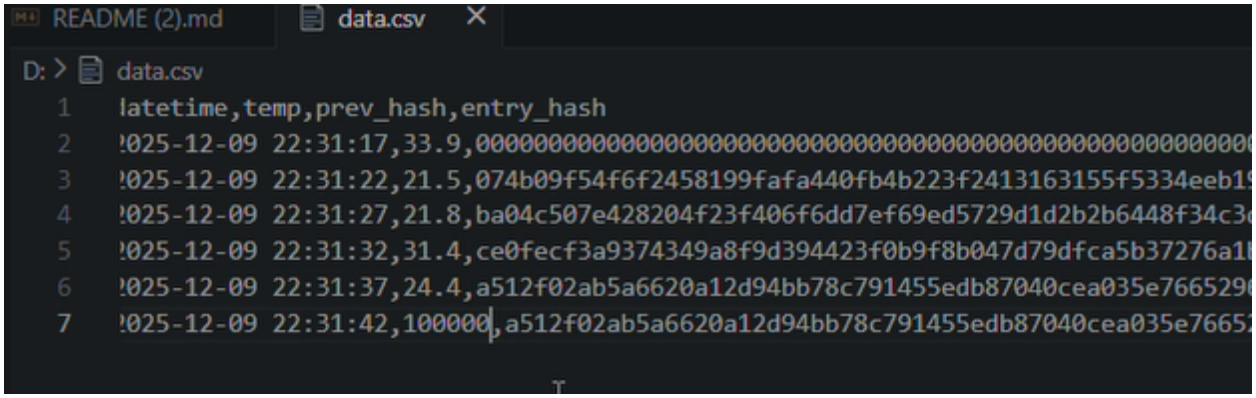


Tampering detected

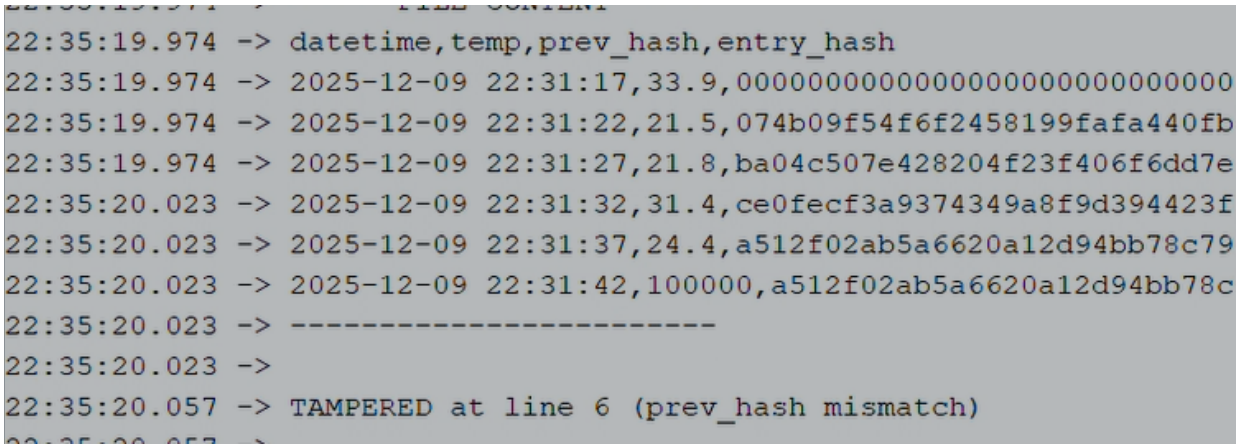
```
22:29:18.421 -> ----- FILE CONTENT -----
22:29:18.421 -> datetime,temp,prev_hash,entry_hash
22:29:18.421 -> 2025-12-09 22:23:36,21.2,0000000000000000000000000000000000
22:29:18.421 -> 2025-12-09 22:23:41,24.9,15422b6ebeb54580e7a6fa979c5b16ed6
22:29:18.470 -> 2025-12-09 22:23:51,22.4,09e892a5cd3e41b82f6ba8aef1e7cf7d7
22:29:18.470 ->
22:29:18.470 -> ------------------------
22:29:18.470 ->
22:29:18.470 -> TAMPERED at line 3 (prev_hash mismatch)
22:29:18.470 ->
```

Adding duplicate line

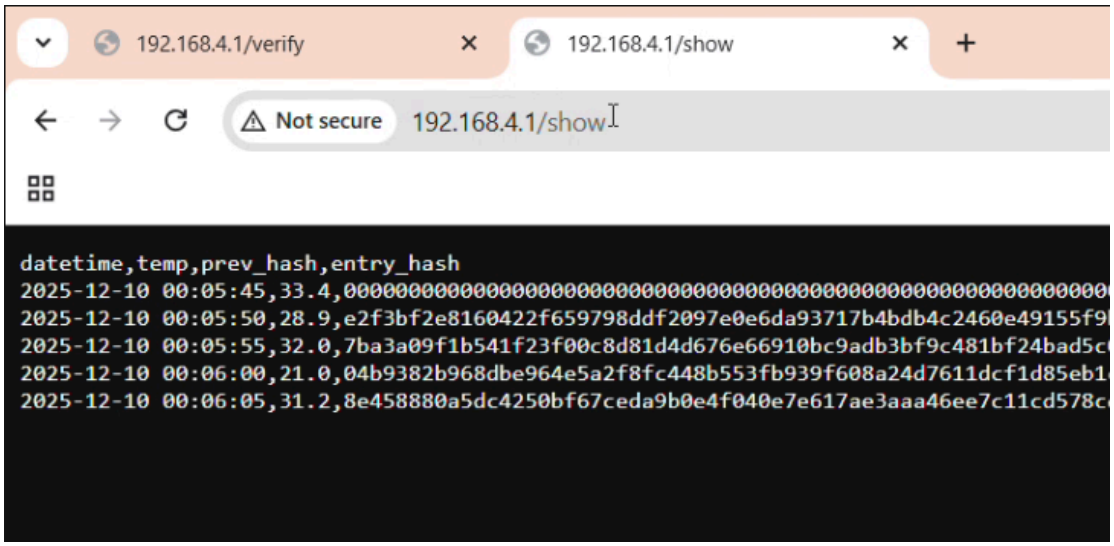


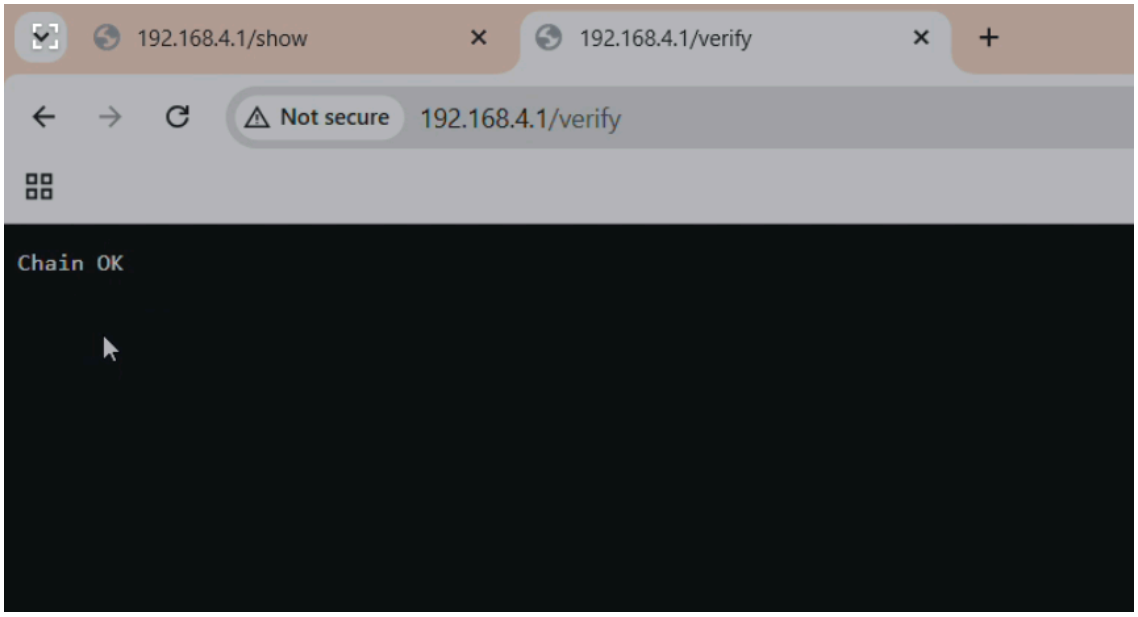


Tampering detected

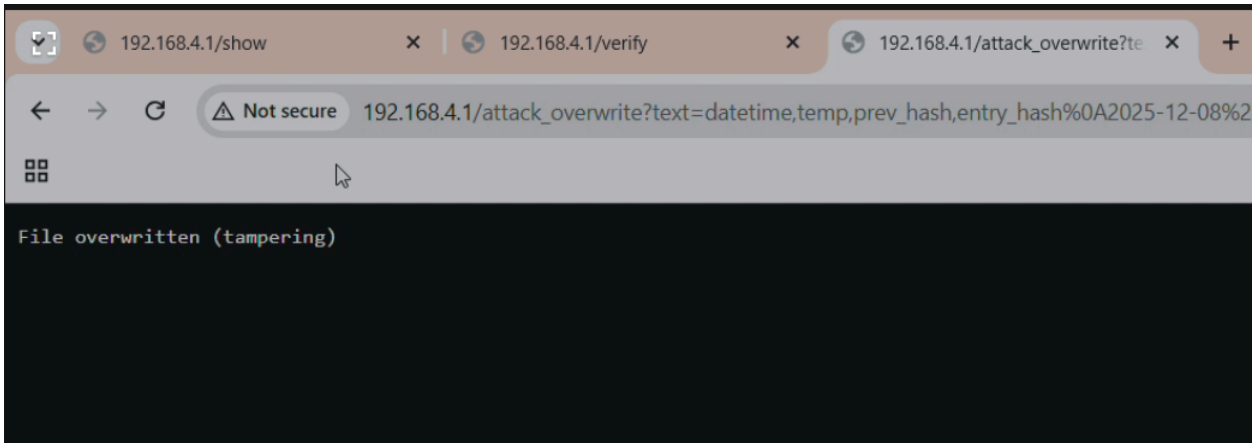


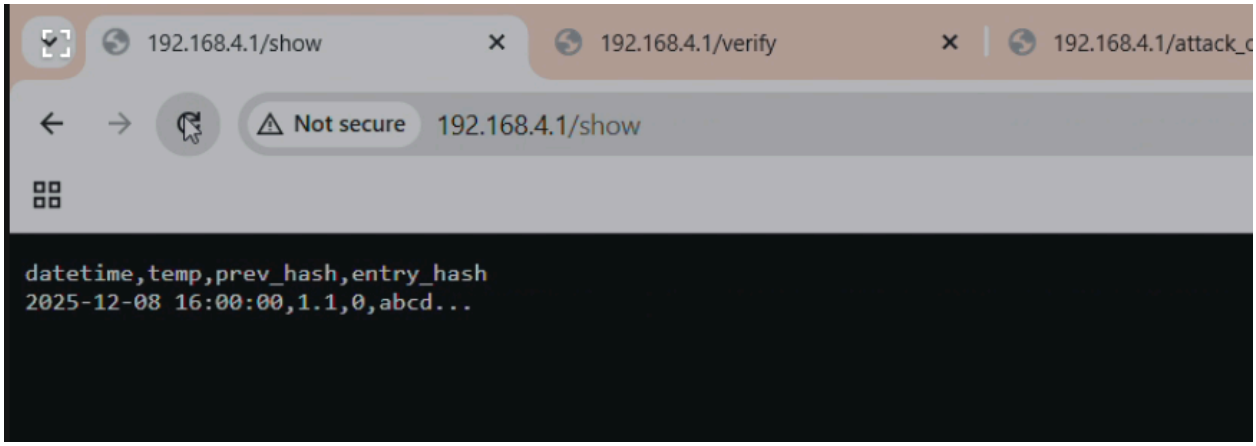
View/Verify and simulate tampering using http commands

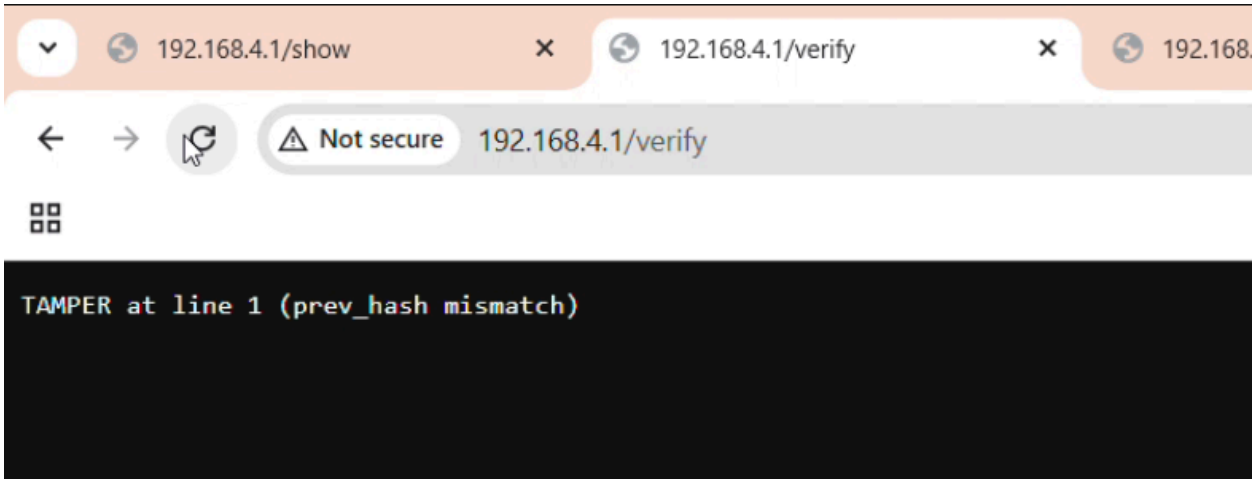




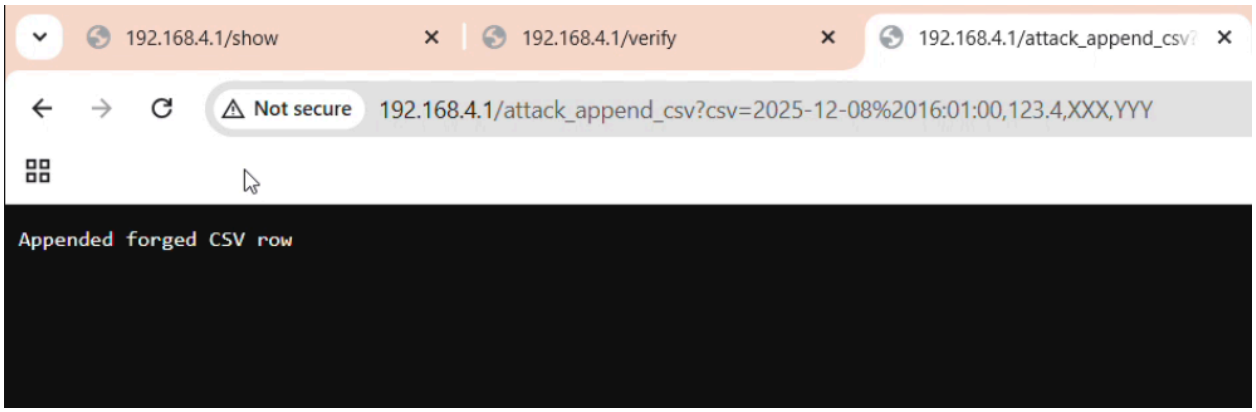
Tampering : Overwrite file

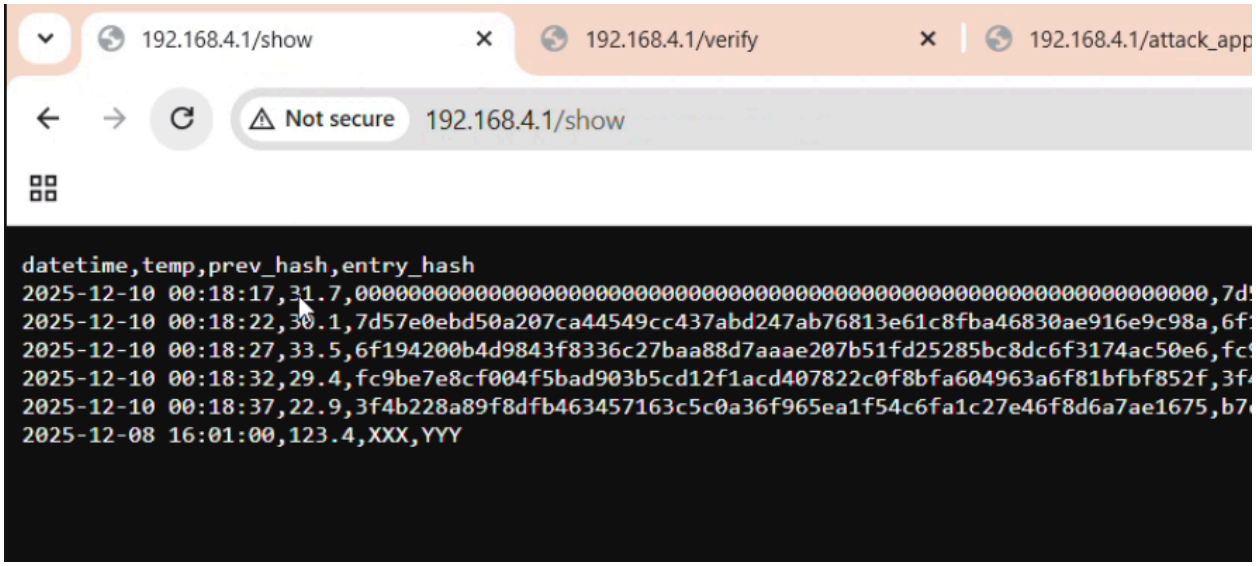


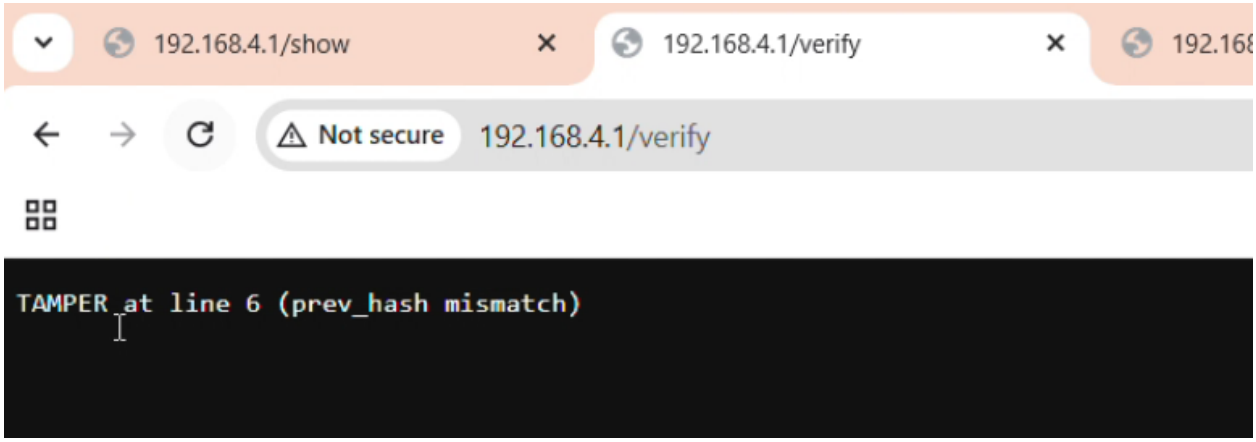




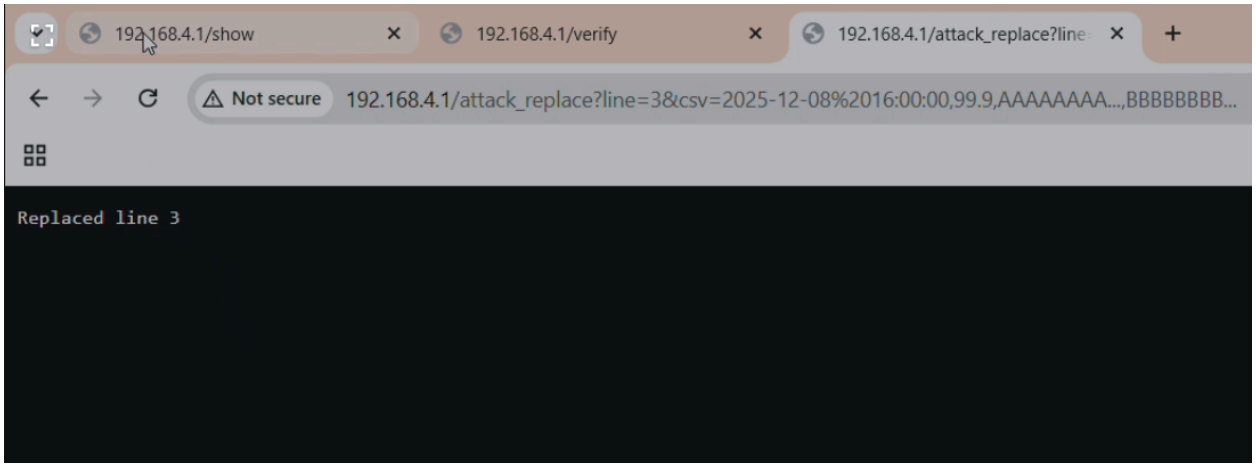
tampering : Alter data

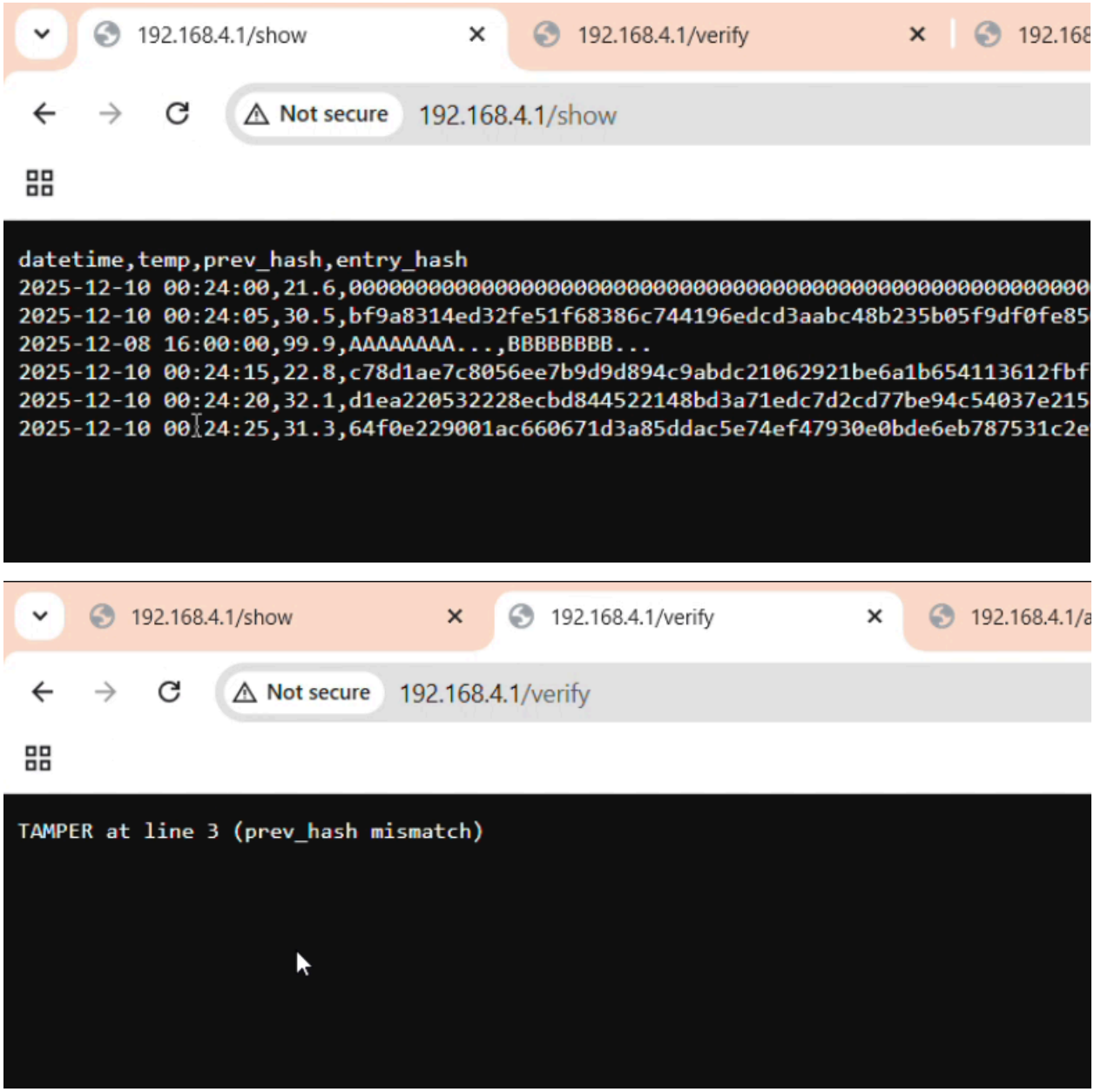






Tampering : Adding duplicate line

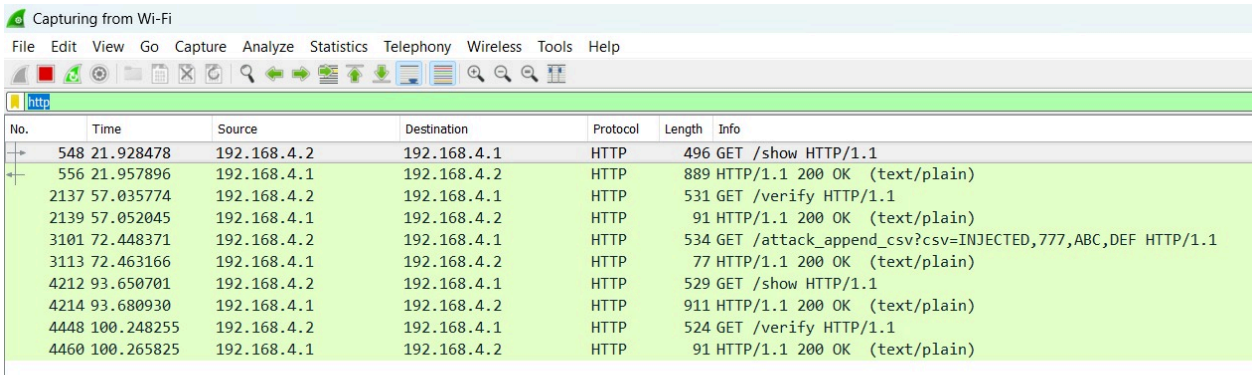




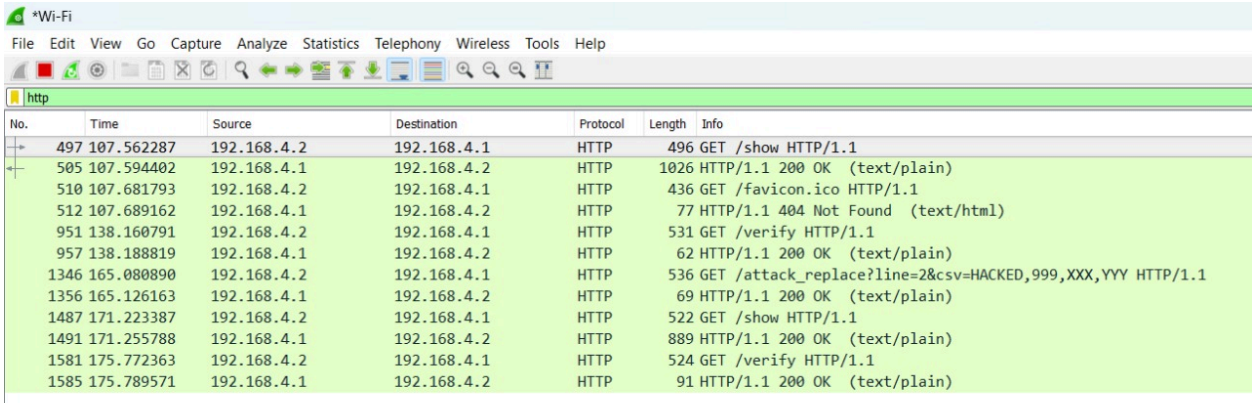
With the addition of cryptographic hash-chaining and NVS-stored final hashes, every recorded entry now becomes part of a verifiable integrity chain. Any modification—whether through manual file edits, injected values, or SD card replacement—breaks this chain and is immediately flagged during verification. Unlike Phase-2, the system can now reliably distinguish authentic data from tampered data, ensuring that no alteration goes undetected.

The final hash stored in NVS is independent of the SD card, acting as a root of trust. Even if the SD card is cloned or replaced, the mismatch is immediately detected.

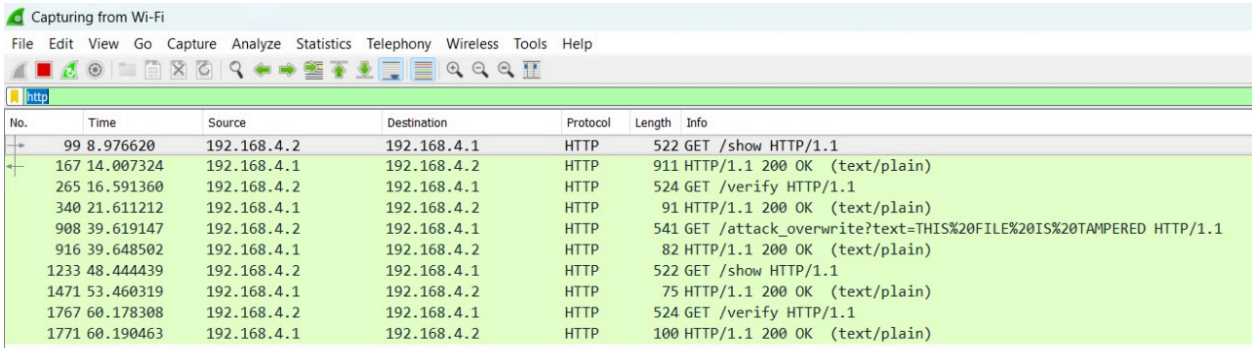
WIRESHARK HTTP TRAFFIC CAPTURE



No.	Time	Source	Destination	Protocol	Length	Info
548	21.928478	192.168.4.2	192.168.4.1	HTTP	496	GET /show HTTP/1.1
556	21.957896	192.168.4.1	192.168.4.2	HTTP	889	HTTP/1.1 200 OK (text/plain)
2137	57.035774	192.168.4.2	192.168.4.1	HTTP	531	GET /verify HTTP/1.1
2139	57.052045	192.168.4.1	192.168.4.2	HTTP	91	HTTP/1.1 200 OK (text/plain)
3101	72.448371	192.168.4.2	192.168.4.1	HTTP	534	GET /attack_append_csv?csv=INJECTED,777,ABC,DEF HTTP/1.1
3113	72.463166	192.168.4.1	192.168.4.2	HTTP	77	HTTP/1.1 200 OK (text/plain)
4212	93.650701	192.168.4.2	192.168.4.1	HTTP	529	GET /show HTTP/1.1
4214	93.680930	192.168.4.1	192.168.4.2	HTTP	911	HTTP/1.1 200 OK (text/plain)
4448	100.248255	192.168.4.2	192.168.4.1	HTTP	524	GET /verify HTTP/1.1
4460	100.265825	192.168.4.1	192.168.4.2	HTTP	91	HTTP/1.1 200 OK (text/plain)



No.	Time	Source	Destination	Protocol	Length	Info
497	107.562287	192.168.4.2	192.168.4.1	HTTP	496	GET /show HTTP/1.1
505	107.594402	192.168.4.1	192.168.4.2	HTTP	1026	HTTP/1.1 200 OK (text/plain)
510	107.681793	192.168.4.2	192.168.4.1	HTTP	436	GET /favicon.ico HTTP/1.1
512	107.689162	192.168.4.1	192.168.4.2	HTTP	77	HTTP/1.1 404 Not Found (text/html)
951	138.160791	192.168.4.2	192.168.4.1	HTTP	531	GET /verify HTTP/1.1
957	138.188819	192.168.4.1	192.168.4.2	HTTP	62	HTTP/1.1 200 OK (text/plain)
1346	165.080890	192.168.4.2	192.168.4.1	HTTP	536	GET /attack_replace?line=2&csv=HACKED,999,XXX,YYY HTTP/1.1
1356	165.126163	192.168.4.1	192.168.4.2	HTTP	69	HTTP/1.1 200 OK (text/plain)
1487	171.223387	192.168.4.2	192.168.4.1	HTTP	522	GET /show HTTP/1.1
1491	171.255788	192.168.4.1	192.168.4.2	HTTP	889	HTTP/1.1 200 OK (text/plain)
1581	175.772363	192.168.4.2	192.168.4.1	HTTP	524	GET /verify HTTP/1.1
1585	175.789571	192.168.4.1	192.168.4.2	HTTP	91	HTTP/1.1 200 OK (text/plain)



No.	Time	Source	Destination	Protocol	Length	Info
99	8.976620	192.168.4.2	192.168.4.1	HTTP	522	GET /show HTTP/1.1
167	14.007324	192.168.4.1	192.168.4.2	HTTP	911	HTTP/1.1 200 OK (text/plain)
265	16.591360	192.168.4.2	192.168.4.1	HTTP	524	GET /verify HTTP/1.1
340	21.611212	192.168.4.1	192.168.4.2	HTTP	91	HTTP/1.1 200 OK (text/plain)
908	39.619147	192.168.4.2	192.168.4.1	HTTP	541	GET /attack_overwrite?text=THIS%20FILE%20IS%20TAMPERED HTTP/1.1
916	39.648502	192.168.4.1	192.168.4.2	HTTP	82	HTTP/1.1 200 OK (text/plain)
1233	48.444439	192.168.4.2	192.168.4.1	HTTP	522	GET /show HTTP/1.1
1471	53.460319	192.168.4.1	192.168.4.2	HTTP	75	HTTP/1.1 200 OK (text/plain)
1767	60.178308	192.168.4.2	192.168.4.1	HTTP	524	GET /verify HTTP/1.1
1771	60.190463	192.168.4.1	192.168.4.2	HTTP	100	HTTP/1.1 200 OK (text/plain)

To demonstrate how tampering can occur over a network, the ESP32 exposes HTTP endpoints that allow both normal viewing and simulated attacks. Using Wireshark, the raw HTTP traffic—including edit, append, or overwrite requests—can be captured and analyzed to show exactly how unauthorized modifications propagate. This provides clear visual proof of network-level tampering attempts and how the forensic protection detects them.

This shows that even if tampering occurs at the HTTP layer, the integrity chain detects it after file modification.

DEMO VIDEO LINK

https://youtu.be/pL5St93XZ0g?si=cstGCZb7T-Y8bX20

CYBER SECURITY LAWS IN INDIA

India's cyber-crime and digital evidence laws directly address such behaviors. Tampering with logs or records—irrespective of the method used—is a punishable offense.

IT Act, 2000 & IT (Amendment) Act, 2008

- **Section 65 – Tampering with Computer Source Documents**
  - Covers alteration, destruction, or concealment of digital records, logs, configurations, or program data.
    Action: Up to **3 years imprisonment** + **₹2 lakh fine**.
- **Section 66 – Computer-Related Offenses**
  - Covers unauthorized data modification, falsification, deletion, or manipulation.
    Includes altering CSV logs, injecting false entries, or modifying SD card contents.
    Action: Up to **3 years imprisonment** + **₹5 lakh fine**.
- **Section 66C & 66D – Identity Theft and Cheating by Personation**
  - Covers impersonating legitimate systems or users, e.g., sending fake HTTP requests or spoofing devices.
    Action: Up to **3 years imprisonment** + **₹1 lakh fine** (each).
- **Section 72 – Breach of Confidentiality & Integrity**
  - Covers unauthorized access, misuse, or alteration of stored data.
    Action: Up to **2 years imprisonment** + **₹1 lakh fine**

Indian Evidence Act – Section 65B (Digital Evidence Integrity)

- For digital records to be admitted as legal evidence, their integrity must be provable. Any detectable tampering—even subtle value changes—invalidates the

evidence.

- **Action:** Evidence is rejected in court unless supported by an integrity mechanism and a 65B certificate.

CONCLUSION

Phase-3 successfully transformed the SD-card data logger from a simple recording device into a tamper-evident forensic system. By integrating SHA-256 hash-chaining and secure NVS-stored reference hashes, every logged entry now becomes cryptographically linked to the previous one—making undetected modification, deletion, or insertion practically impossible.

Both manual SD-card edits and simulated network-based attacks were performed to validate the system. In every case, the verification process accurately identified the exact line of tampering or detected cloned/swapped SD cards. The Wireshark traffic capture further demonstrated how unauthorized changes appear at the protocol level and how the chain-verification mechanism exposes them.

Overall, Phase-3 ensures that recorded temperature data maintains legal-grade integrity, aligns with digital evidence standards (such as Section 65B of the Indian Evidence Act), and provides a robust, practical forensic protection layer on low-cost embedded hardware.