SISEN: Attack and Disruption Guide

Security-Informed Safety in Wireless Networks

Purpose of this Guide

This guide explains the attack, observation, and disruption activities used within the SISEN learning environment. It will help you understand what each activity represents, which trust assumption or security property it challenges, what evidence you should collect, and how the resulting security failure may contribute to a safety hazard.

The commands provide bounded ways to conduct each activity within SISEN. You must not use them against systems outside the designated lab environment.

You should use this guide with the relevant Medical IoT, Smart Building, or 6LoWPAN scenario guide and hazard analysis worksheet.

Authorised Lab Use Only

You must perform all security testing, observation, and disruption activities only within the SISEN environment and against the designated lab infrastructure.
Do not target external devices, wireless networks, access points, brokers, services, or systems.
Only use the commands, scripts, interfaces, and targets specified in this guide.

1 How to Conduct an Activity

You will need at least two terminals for most SISEN activities.

- **Scenario terminal:** Use this terminal to start the relevant SISEN scenario. Keep it open so that you can observe system messages and status information.
- **Activity terminal:** Use this terminal to run the observation, attack, or disruption command.

You may need a third terminal to observe MQTT messages, inspect logs, capture network traffic, or monitor the dashboard while the activity is running.

Before starting an activity:

1. Start the relevant SISEN scenario.
2. Confirm that the dashboard and telemetry are updating normally.
3. Open any additional terminal needed for traffic capture, MQTT observation, or log monitoring.
4. Run only one bounded activity at a time.
5. Observe the technical effect and the visible response of the scenario.
6. Stop the activity and confirm that normal operation resumes.

The commands in this guide assume that you are working from the root folder of the SISEN repository. Attack commands can be run from a fresh terminal. The runner uses the repository virtual-environment interpreter when it is available, so you do not need to activate `.venv` manually.

Helper-based telemetry attacks are bounded demonstrations. Where an injected value must remain visible, the helper refreshes it for approximately ten seconds and then exits. Normal scenario telemetry continues while the helper is running and should restore the dashboard

afterwards. Stop continuing observation or capture tools using `Ctrl+C`. Where an activity requires a different stopping or recovery procedure, the relevant section will explain it.

Manual wireless and infrastructure activities are documented in `docs/manual-attacks.md`. The separate hardware access-point activities are documented under `ap/hw-ap/docs/` and are not launched through `launch_sisen.py`.

2 Understanding and Listing the Activity Categories

SISEN groups its activities according to the security property, trust assumption, communication path, or safety condition being examined.

Confidentiality Use these activities to examine the unauthorised observation of telemetry, identifiers, protocol metadata, or operational state.

```
python3 attacks/run_attack.py --list --category confidentiality
```

Authenticity Use these activities to examine whether the system accepts data from a source that is not the legitimate device, sensor, publisher, gateway, or wireless infrastructure.

```
python3 attacks/run_attack.py --list --category authenticity
```

Integrity Use these activities to examine how the system handles altered, extreme, malformed, manipulated, or otherwise untrustworthy data.

```
python3 attacks/run_attack.py --list --category integrity
```

Replay and Message Freshness Use these activities to examine whether the system accepts stale or previously valid telemetry as though it represented the current system state.

```
python3 attacks/run_attack.py --list --category replay
```

Availability Use these activities to examine missing, delayed, suppressed, noisy, dropped, or otherwise disrupted telemetry or communication services.

```
python3 attacks/run_attack.py --list --category availability
```

Protocol and Communication Path Use these activities to examine or disrupt the communication path between devices, wireless interfaces, gateways, relays, brokers, and applications.

```
python3 attacks/run_attack.py --list --category protocol
```

Scenario-Focused Safety Cases Use these activities to examine realistic scenario-specific conditions in which a security failure may contribute to an unsafe state.

```
python3 attacks/run_attack.py --list --category safety-case
```

These properties are connected. You may first observe a topic structure, then impersonate a publisher, inject false values, or create a misleading safety-relevant display.

Listing All Activities

To display the complete list of activities supported by the helper, run:

```
python3 attacks/run_attack.py --list
```

Listing Activities by Scenario

You can also filter the available activities by SISEN scenario.

```
python3 attacks/run_attack.py --list --scenario building
python3 attacks/run_attack.py --list --scenario medical
python3 attacks/run_attack.py --list --scenario 6lowpan
```

These properties are connected. You may first observe a topic structure, then impersonate a publisher, inject false values, and finally create a misleading safety-relevant display.

3 Traffic Observation and Capture

Traffic observation provides the baseline evidence you need to understand how data moves through each SISEN scenario. By examining normal communication first, you can compare it with later attacks or disruptions and identify whether an observed effect begins at the wireless, protocol, or application layer.

Wireless Traffic Capture

Relevant scenarios: Smart Building, Medical IoT, and 6LoWPAN.

What it is: Wireless traffic capture records frames or packets exchanged across the wireless communication path so that you can inspect them during or after the activity. In the Smart Building and Medical IoT scenarios, this includes traffic carried through the wireless access point and device namespaces. In the 6LoWPAN scenario, you can capture traffic from the IEEE 802.15.4 and 6LoWPAN interfaces.

What it demonstrates: This activity shows which traffic is visible at different points in the wireless path. It also provides a record of normal communication that you can compare with traffic collected during a later attack or disruption.

Expected observable effect: The capture should record the normal wireless traffic generated by the scenario without changing its behaviour. The dashboard and telemetry should continue to update normally while the capture is running.

Smart Building

```
sudo tcpdump -i wlan0 -n -vv -s 0 -Z "$USER" -w captures/smart-building-ap-wlan0.
pcap
```

Medical IoT

```
sudo tcpdump -i wlan0 -n -vv -s 0 -Z "$USER" -w captures/medical-ap-wlan0.pcap
```

6LoWPAN: IEEE 802.15.4 Capture

```
sudo ip netns exec node1 tcpdump -i wpan1 -n -vv -s 0 -Z "$USER" -w captures/6lowpan
-node1-wpan.pcap
```

6LoWPAN: 6LoWPAN Capture

```
sudo ip netns exec node1 tcpdump -i lowpan0 -n -vv -s 0 -Z "$USER" -w captures/6
lowpan-node1-lowpan.pcap
```

Safety relevance

Wireless traffic capture can help you determine whether the loss or delay of safety-relevant telemetry begins within the wireless communication path. Gaps in packet delivery or changes in the normal traffic pattern may help explain why current telemetry does not reach the dashboard.

MQTT Topic Observation

Relevant scenarios: Smart Building, Medical IoT, and 6LoWPAN.

What it is: MQTT topic observation allows you to subscribe to the telemetry published by a SISEN scenario and view each topic together with its message payload.

What it demonstrates: This activity shows what information is exposed through the MQTT broker, including topic names, device or sensor identifiers, telemetry values, and message frequency. It also helps you understand how each scenario organises its telemetry before you examine later spoofing, replay, or disruption activities.

Expected observable effect: You should see the scenario telemetry appearing in the terminal as it is published. The activity should not change the telemetry, dashboard, or behaviour of the scenario.

Commands:

Smart Building

```
mosquitto_sub -h localhost -v -t 'building/#'
```

Medical IoT

```
mosquitto_sub -h localhost -v -t 'patient/#'
```

6LoWPAN

```
mosquitto_sub -h localhost -v -t 'industrial/6lowpan/#'
```

Capture MQTT Traffic

```
sudo tcpdump -i any -n -vv -s 0 -Z "$USER" -w captures/mqtt-traffic.pcap port 1883
```

Safety relevance

Observing the MQTT topics helps you identify which messages represent safety-relevant conditions and how frequently they are expected to arrive. This provides a baseline for recognising missing, delayed, stale, or manipulated telemetry during later activities.

4 Confidentiality Activities

Confidentiality activities allow you to examine information that may be observed without altering the operation of the system. This may include wireless frames, network traffic, identifiers, protocol metadata, telemetry values, and operational patterns.

Access Point Traffic Observation

Relevant scenario: Any scenario using a wireless access point.

What it is: Access point traffic observation allows you to examine network traffic exchanged through an authorised lab access point. The traffic visible in the capture depends on the access-point configuration, the protocols in use, and whether link-layer or application-layer protection is enabled.

What it demonstrates: This activity demonstrates what network and protocol information is exposed when traffic passes through a wireless access point. Depending on the scenario, you may observe source and destination addresses, TCP connections, MQTT messages, topic names, telemetry publications, IPv6 control traffic, and other protocol exchanges generated by controlled clients.

Expected observable effect: You should be able to identify the authorised lab access point, generate controlled scenario traffic, and observe the resulting packets without disrupting the access point or connected clients.

Check Wireless Interfaces

```
iw dev
ip -brief addr
```

Example: The following command captures traffic from an Open AP configured on `wlan0`.

Capture Open AP Traffic

```
sudo tcpdump -i wlan0 -n -vv -s 0 -Z "$USER" -w captures/open-ap-traffic.pcap
```

Safety relevance

Access point traffic observation can reveal safety-relevant telemetry, device identifiers, protocol behaviour, or operational communication. The information visible depends on the access-point configuration and the link-layer or application-layer protection in use. An observer may use exposed information to understand the system structure and prepare later spoofing, replay, or disruption activities.

Note: Observation of WEP- or WPA2-protected traffic does not by itself expose the protected application data because the wireless payload is encrypted. Some metadata, such as the SSID, BSSID, channel, client addresses, frame types, and connection activity, may still remain visible. Attacks against WEP and WPA2 protection are discussed separately in Section **??**.

Hidden Access Point Observation

Relevant scenario: Any scenario using the Hidden AP option.

What it is: Hidden access point observation examines the raw IEEE 802.11 management traffic generated by an access point that does not advertise its SSID normally. Although the network name may be absent from beacon frames, authorised client probe and association behaviour may reveal it.

What it demonstrates: This activity demonstrates that hiding an SSID is not an effective security control. The access point remains detectable through its wireless activity, and its SSID may be disclosed when an authorised client searches for or connects to the network.

Expected observable effect: Before an authorised client connects, the SSID may be absent from beacon frames. When the client probes for or associates with the network, probe or association frames may reveal the SSID. The observation should not disrupt the access point or client.

Create and Enable Monitor Interface

```
sudo iw dev wlan0 interface add mon-sisen type monitor
sudo ip link set mon-sisen up
```

Confirm Monitor Interface

```
iw dev
```

Capture Hidden AP Traffic

```
sudo tcpdump -i mon-sisen -n -vv -s 0 -Z "$USER" -w captures/hidden-ap-observation.
pcap
```

Remove Monitor Interface

```
sudo iw dev mon-sisen del
```

Useful Wireshark filters:

Hidden AP Display Filters

```
wlan.ssid
wlan.fc.type_subtype == 0x04
wlan.fc.type_subtype == 0x05
```

Safety relevance

Hidden access point observation may reveal the existence and identity of a wireless network that operators assume is concealed. The disclosed SSID, BSSID, channel, client behaviour, and association activity may help an observer map the system and prepare rogue access point, impersonation, or subsequent traffic-observation activities.

Note: In the SISEN environment, the host system manages the access point and can therefore display its configured SSID directly using `iw dev`, even when the network is configured as hidden. In a real external observation scenario, an attacker would not have access to this local configuration and would normally need to use monitor mode to observe IEEE 802.11 management traffic and wait for authorised client activity that may reveal the hidden SSID.

Medical IoT Gateway Log Observation

Relevant scenario: Medical IoT.

What it is: Gateway log observation allows you to examine how the Medical IoT gateway receives, processes, and forwards simulated wearable telemetry.

What it demonstrates: This activity demonstrates what operational information may be exposed through application logs, including telemetry values, device identifiers, processing events, and gateway behaviour.

Expected observable effect: You should be able to observe simulated wearable telemetry being processed by the BLE-to-Wi-Fi gateway without disrupting the scenario.

Observe BLE-to-Wi-Fi Gateway Activity

```
tail -f /tmp/sisen-ble-wifi-gateway.log
```

Safety relevance

Gateway logs may reveal medical measurements, device identifiers, processing behaviour, and operational state. Exposure of this information may help an observer understand the telemetry path and prepare later spoofing, replay, or disruption activities.

Note: The current Medical IoT scenario simulates the BLE communication path and does not generate observable over-the-air BLE traffic. A dedicated BLE traffic-capture activity will be developed in a later version of the platform.

5 Identity, Authenticity, and Spoofing Attacks

Spoofing attacks introduce data or identities that appear to originate from a legitimate device, sensor, client, or infrastructure component. These activities examine whether the system verifies the source of telemetry and whether operational decisions rely on identifiers that can be imitated.

MQTT Telemetry Spoofing

Relevant scenarios: Smart Building and Medical IoT.

What it is: MQTT telemetry spoofing publishes plausible but unauthorised values to legitimate scenario topics. The messages use the same topic structure and expected value format as normal telemetry, allowing them to appear as though they originated from an authorised sensor or wearable.

What it demonstrates: This activity demonstrates the consequences of weak publisher authentication and source validation. A subscriber may accept a correctly formatted message because it appears on an expected topic, even though the message was published by an unauthorised source.

Expected observable effect: You should observe the spoofed values appearing on the relevant MQTT topics and, where the scenario consumes those topics, on the monitoring dashboard. Normal telemetry may subsequently overwrite the spoofed values.

Smart Building example: Run the Smart Building scenario before using the following commands.

Observe Smart Building Telemetry

```
mosquitto_sub -h localhost -t 'building/#' -v
```

Inject Spoofed Smart Building Telemetry

```
python3 attacks/run_attack.py --category authenticity --scenario building --attack
spoofed
```

Medical IoT example: Run the Medical IoT scenario before using the following commands.

Observe Medical IoT Telemetry

```
mosquitto_sub -h localhost -t 'patient/#' -v
```

Inject Spoofed Medical IoT Telemetry

```
python3 attacks/run_attack.py --category authenticity --scenario medical --attack
spoofed
```

Safety relevance

Spoofed telemetry may cause a monitoring or decision-making system to accept false information as authentic. Plausible values may conceal an unsafe condition, create false reassurance, or cause operators and automated controls to act on data that did not originate from the expected sensor or wearable.

6LoWPAN Telemetry Spoofing

Relevant scenario: 6LoWPAN.

What it is: 6LoWPAN telemetry spoofing injects a fabricated JSON telemetry message into the validated UDP and IPv6 communication path used by the constrained network.

What it demonstrates: This activity demonstrates that possession of a valid network path does not establish the authenticity of the transmitting sensor. If the gateway validates only the message structure and destination path, a fabricated reading may be processed as legitimate telemetry.

Expected observable effect: You should observe the spoofed telemetry passing through the 6LoWPAN path and reaching the gateway or MQTT relay. The injected message is marked as spoofed within the demonstration payload, allowing it to be distinguished during analysis.

Inject Spoofed 6LoWPAN Telemetry

```
python3 attacks/run_attack.py --category protocol --scenario 6lowpan --attack
spoofed
```

Safety relevance

A spoofed constrained-device reading may cause the gateway or monitoring system to report an incorrect physical or operational condition. Where sensor identity is not cryptographically verified, an attacker may imitate a trusted device and introduce false telemetry into a safety-relevant information path.

MAC Identity Spoofing

Relevant scenario: Smart Building using MAC-based access filtering.

What it is: MAC identity spoofing changes a wireless interface address so that it imitates an address permitted by the access point. The activity tests whether possession of an allowed MAC address is treated as sufficient evidence of device identity.

What it demonstrates: This activity demonstrates that a MAC address is an observable and changeable identifier rather than a strong authentication credential. An access-control decision based only on an allowed MAC address may therefore be bypassed by another controlled client using the same identity.

Expected observable effect: The controlled client should initially be rejected when using an unapproved MAC address. After adopting an address included in the laboratory allow-list, the same client may be accepted by the access point. Results depend on the selected AP configuration and whether another interface is already using the chosen address.

Manual activity: Start the Smart Building scenario using the MAC-filtered AP configuration, identify an authorised laboratory MAC address, and change only the controlled test interface to that address. Restore the original address when the activity is complete.

Safety relevance

Reliance on MAC filtering may allow an unauthorised device to imitate a trusted sensor or client and gain access to the wireless network. Once connected, the device may observe traffic, publish false telemetry, or interfere with safety-relevant communication while appearing to use an approved identity.

Note: This is a manual infrastructure activity. The repository identifies the activity as `mac-filter-bypass`, but the exact interface and authorised MAC address depend on the running Smart Building scenario and must be confirmed before any command is issued.

6 Spoofed Telemetry and Source Impersonation

Spoofing activities examine whether the system can verify the origin of received telemetry. They demonstrate how an unauthorised source may imitate a legitimate sensor, device, gateway, or publisher and cause downstream components to trust false data.

6.1 Spoofed Telemetry

What it is:

Spoofed telemetry is data published by an unauthorised source while claiming to represent a legitimate sensor, wearable, node, or publisher. The message may use the expected topic and format, causing downstream components to accept it without verifying its origin.

What it demonstrates:

This activity demonstrates an authenticity failure. It shows that correct formatting is not evidence that data came from the correct source. It also exposes the risk created when brokers, gateways, controllers, or dashboards trust a topic name or source address without cryptographic authentication or access control.

Safety relevance:

A monitoring system may make decisions based on a false representation of the physical or operational environment. Spoofed medical telemetry may create an incorrect view of a patient's condition. Spoofed building data may alter the apparent state of temperature, air quality, or occupancy. Spoofed 6LoWPAN telemetry may make a constrained sensor appear healthy, unsafe, or unavailable when the opposite is true.

Example SISEN commands:

```
python3 attacks/run_attack.py --category authenticity --scenario building --attack spoofed
python3 attacks/run_attack.py --category authenticity --scenario medical --attack spoofed
python3 attacks/run_attack.py --category protocol --scenario 6lowpan --attack spoofed
```

Expected observable effect:

The dashboard or subscriber should display telemetry that appears to come from a legitimate source. Logs or packet captures should show the injected message and allow comparison with the normal publisher.

Stopping and recovery:

Allow the bounded helper activity to finish or stop it using `Ctrl+C`. Confirm that subsequent readings originate from the normal publisher and that the dashboard returns to its baseline state.

6.2 Rogue 6LoWPAN Sensor

What it is:

A rogue sensor is an unauthorised constrained node or process that presents itself as a valid telemetry source within the 6LoWPAN scenario.

What it demonstrates:

This activity demonstrates the consequences of weak node identity and insufficient validation at the border router, relay, or application layer. A network address or correctly structured payload may be accepted even when the source is not trusted.

Safety relevance:

The monitoring application may treat rogue telemetry as evidence of a real environmental or operational condition. This can create false alarms, conceal a genuine hazard, or undermine confidence in all telemetry from the constrained network.

Example SISEN command:

```
python3 attacks/run_attack.py --category protocol --scenario 6lowpan --attack spoofed
```

Expected observable effect:

An additional or substituted sensor reading should become visible in the telemetry path. Evidence should be collected from the dashboard, relay output, logs, or packet capture.

Stopping and recovery:

Allow the helper to complete or stop it using `Ctrl+C`. Confirm that the rogue readings cease and that the legitimate sensor remains visible.

7 Extreme and Implausible Values

These activities examine telemetry that is correctly formatted but semantically untrustworthy. They test whether the system validates plausible operating ranges and whether abnormal values are treated as genuine physical conditions without sufficient verification.

7.1 Extreme Telemetry Values

What it is:

An extreme-value activity sends telemetry that is correctly structured but far outside the expected operating range. The payload may be syntactically valid while representing an implausible, unsafe, or physically impossible condition.

What it demonstrates:

This activity demonstrates an integrity failure and the limitations of systems that validate message format but not meaning. It tests whether the gateway, controller, or dashboard performs range checking, plausibility checking, correlation, or rate-of-change validation.

Safety relevance:

Extreme values may trigger unnecessary alarms or automated actions, or they may be accepted as genuine evidence of a dangerous condition. The important question is not merely whether the dashboard turns red, but whether the system distinguishes a genuine hazard from corrupted telemetry and moves to a

safe response.

Example SISEN commands:

```
python3 attacks/run_attack.py --category integrity --scenario building --attack extreme
python3 attacks/run_attack.py --category integrity --scenario medical --attack extreme
python3 attacks/run_attack.py --category protocol --scenario 6lowpan --attack extreme
```

Expected observable effect:

The dashboard should display one or more abnormal values. Depending on the scenario implementation, alerts or controller-state changes may also be visible.

Stopping and recovery:

Allow the bounded helper to complete or stop it using `Ctrl+C`. Verify that normal telemetry replaces the injected value and that any alert or controller state returns to baseline.

7.2 Scenario-Specific Extreme Conditions

What it is:

Scenario-specific extreme conditions package an integrity failure as a realistic safety case. The activity uses values selected to create a meaningful interpretation within the Medical IoT or Smart Building scenario.

What it demonstrates:

This activity demonstrates that the same technical weakness can have different consequences depending on the system context. A value that is merely unusual in one environment may indicate a critical condition in another.

Safety relevance:

Critical medical values may lead to unnecessary intervention or conceal genuine deterioration if false stable values are injected. Extreme building values may trigger inappropriate environmental responses or cause occupants and operators to misinterpret the actual condition.

Example SISEN commands:

```
python3 attacks/run_attack.py --category safety-case --scenario medical --attack critical-
    vitals
python3 attacks/run_attack.py --category safety-case --scenario building --attack gas-leak-
    hidden
```

Expected observable effect:

The scenario dashboard should present a clearly abnormal safety-relevant state. Students should record both the technical evidence and the resulting operational interpretation.

Stopping and recovery:

Allow the helper to complete or stop it using `Ctrl+C`. Confirm that live values return to their normal range and that any persistent alert is cleared appropriately.

8 Malformed and Invalid Data

Malformed-data activities examine how the system handles messages that do not conform to the expected structure, type, encoding, or field requirements. They focus on validation, parser robustness, error handling, and continued processing of legitimate telemetry.

9 Malformed Medical Telemetry

What it is:

Malformed telemetry does not conform to the structure, data type, field set, or encoding expected by the receiving component. Examples include missing fields, incorrect types, invalid JSON, or values placed in the wrong field.

What it demonstrates:

This activity demonstrates an integrity and robustness failure. It tests whether the gateway, broker consumer, dashboard, or parser rejects invalid input safely, logs the problem, and continues processing legitimate telemetry.

Safety relevance:

A malformed message may crash a consumer, corrupt displayed state, interrupt processing, or silently remove a patient from effective monitoring. A safe system should reject invalid data without treating it as a valid measurement and without losing the availability of unrelated telemetry.

Example SISEN command:

```
python3 attacks/run_attack.py --category integrity --scenario medical --attack malformed
```

Expected observable effect:

The malformed message may be rejected, logged, ignored, or displayed incorrectly, depending on the implemented validation. Students should record the actual behaviour rather than assume the expected defence exists.

Stopping and recovery:

Allow the helper to complete or stop it using `Ctrl+C`. Confirm that the dashboard and gateway continue processing subsequent legitimate readings.

10 Replay and Stale Telemetry

Replay activities examine whether previously valid telemetry can be accepted again as though it represented the current system state. They introduce the security property of message freshness and show why valid origin and correct formatting do not guarantee that data is current.

10.1 Replayed Telemetry

What it is:

A replay attack resends a previously valid message so that old information is presented as if it were new. The message may be authentic in origin and correct in format, but it no longer represents the current physical or operational state.

What it demonstrates:

This activity demonstrates a message freshness failure. It shows why authenticity and integrity alone are insufficient when a system must know whether information is current. Effective defences may require timestamps, sequence numbers, nonces, replay protection, or explicit age checks.

Safety relevance:

Stale information can create false reassurance. A stable medical reading may be replayed while a patient deteriorates. A normal building value may hide a developing environmental hazard. An old 6LoWPAN reading may make a failed or disconnected sensor appear active.

Example SISEN commands:

```
python3 attacks/run_attack.py --category replay --scenario building --attack replay
python3 attacks/run_attack.py --category replay --scenario medical --attack replay
python3 attacks/run_attack.py --category protocol --scenario 6lowpan --attack replay
```

Expected observable effect:

A previously observed value should reappear in the telemetry path. Students should compare timestamps, message order, repeated values, and dashboard update behaviour.

Stopping and recovery:

Allow the helper to complete or stop it using `Ctrl+C`. Confirm that the display resumes using newly generated telemetry rather than the replayed value.

10.2 Stale Medical Vitals

What it is:

The stale-vitals case applies replay and Message Freshness failure specifically to patient monitoring. It represents an old set of observations continuing to appear current.

What it demonstrates:

This activity demonstrates the distinction between receiving data and receiving current data. A dashboard that continues to update visually may still be unsafe if it does not verify the age or sequence of the underlying readings.

Safety relevance:

Stale stable values may delay recognition of patient deterioration. The safety issue is therefore not simply missing information, but the presentation of misleading confidence that monitoring remains trustworthy.

Example SISEN command:

```
python3 attacks/run_attack.py --category replay --scenario medical --attack replay
```

Expected observable effect:

The dashboard should continue presenting old vital-sign information. Students should identify the evidence that distinguishes stale data from genuine live readings.

Stopping and recovery:

Allow the helper to complete or stop it using `Ctrl+C`. Confirm that current timestamps and live physiological variation resume.

11 Missing Telemetry and Sensor Blackout

These activities examine complete or partial loss of telemetry delivery. They test whether the system recognises missing updates, distinguishes unavailable data from normal values, and provides an explicit indication that monitoring confidence has been lost.

11.1 Missing 6LoWPAN Telemetry

What it is:

Missing telemetry occurs when expected measurements are not delivered to the gateway, relay, broker, or dashboard. The sensor may still exist, but its current state is no longer visible to the monitoring application.

What it demonstrates:

This activity demonstrates an availability failure. It tests whether the system detects absent updates, marks data as stale or unavailable, and distinguishes loss of communication from a normal or safe measured condition.

Safety relevance:

No data must not be interpreted as normal data. If environmental, industrial, or operational telemetry disappears without an explicit warning, an unsafe condition may remain undetected.

Example SISEN command:

```
python3 attacks/run_attack.py --category protocol --scenario 6lowpan --attack missing
```

Expected observable effect:

The expected 6LoWPAN telemetry should stop or fail to reach the next stage of the communication path. The dashboard may freeze, show an error, or continue displaying the last value.

Stopping and recovery:

Allow the helper to complete or stop it using `Ctrl+C`. Confirm that new telemetry arrives and that stale or unavailable indicators clear correctly.

11.2 Smart Building Sensor Blackout

What it is:

A sensor blackout interrupts telemetry from multiple Smart Building sensors for a bounded period. It represents a wider availability failure than the loss of a single sensor stream.

What it demonstrates:

This activity demonstrates how dependence on a shared communication path or common service can create a broad loss of operational visibility. It also tests whether the system degrades explicitly or silently.

Safety relevance:

A building management system may lose awareness of temperature, humidity, air quality, and occupancy at the same time. Without clear failure indication or fallback procedures, operators may assume the building remains within safe conditions.

Example SISEN command:

```
sudo python3 attacks/run_attack.py --category availability --scenario building --attack sensor
    -blackout --duration 10
```

Expected observable effect:

Multiple sensor feeds should stop updating for the specified duration. Students should observe whether the dashboard freezes values, marks them unavailable, or raises an alert.

Stopping and recovery:

The activity should end automatically after the specified duration. Use `Ctrl+C` if early termination is required. Confirm that all sensor streams resume and that the dashboard does not retain stale values as current.

12 Selective Client or Sensor Disruption

Selective disruption affects one telemetry source while the remainder of the scenario continues operating. It demonstrates how a system can appear broadly healthy while a localised and potentially safety-relevant loss remains undetected.

12.1 Smart Building Client Drop

What it is:

A client-drop activity disconnects or suppresses one selected Smart Building telemetry source while leaving other sources operating normally.

What it demonstrates:

This activity demonstrates selective availability loss. It shows that a system may appear generally healthy while one safety-relevant measurement is absent. It also tests whether the dashboard identifies exactly which stream has failed.

Safety relevance:

Selective loss can be more difficult to recognise than a complete outage. Missing occupancy, temperature, humidity, or air-quality data may conceal a localised hazard while the rest of the dashboard continues to appear normal.

Example SISEN commands:

```
sudo python3 attacks/run_attack.py --category availability --scenario building --attack client
    -drop --target room-101
```

Expected observable effect:

Only the selected telemetry stream should stop or become unavailable. Other Smart Building streams should continue updating.

Stopping and recovery:

Allow the bounded activity to complete or stop it using `Ctrl+C`. Confirm that the selected stream resumes and that the dashboard distinguishes recovered data from previously cached values.

13 Telemetry Noise and Delivery Stress

These activities examine conditions in which communication remains available but the telemetry stream becomes unstable, excessive, repetitive, or difficult to interpret. They test whether the system can preserve usable information under delivery stress.

What it is:

Telemetry noise introduces repeated, unstable, or distracting messages into the data path. Unlike a single false reading, noise creates a sequence of abnormal events that may compete with legitimate telemetry or overload the interpretation process.

What it demonstrates:

This activity demonstrates an availability and data-quality problem. The service may remain technically reachable while useful information becomes harder to identify, process, or trust. It can also reveal whether the system applies rate limiting, filtering, deduplication, or source validation.

Safety relevance:

A noisy monitoring system may produce alarm fatigue, obscure genuine deterioration, delay operator response, or cause automated components to oscillate between states. Availability therefore includes the ability to obtain usable and timely information, not merely the continued existence of a network connection.

Example SISEN commands:

```
python3 attacks/run_attack.py --category availability --scenario building --attack noise --
    count 5
python3 attacks/run_attack.py --category availability --scenario medical --attack noise --
```

```
    count 5
```

Expected observable effect:

The subscriber, logs, or dashboard should show a burst or sequence of abnormal messages. Students should determine whether legitimate telemetry remains visible and whether the system enters an unstable state.

Stopping and recovery:

The helper should stop after the specified count. Use `Ctrl+C` if required. Confirm that the normal message rate and dashboard behaviour return.

14 Scenario-Specific Safety Cases

14.1 False Smart Building Occupancy

What it is:

False occupancy telemetry reports that a zone is occupied when it is empty, or empty when it is occupied.

What it demonstrates:

This activity demonstrates how a small integrity failure can influence several dependent functions. Occupancy data may affect lighting, ventilation, access-control interpretation, or operator awareness.

Safety relevance:

Incorrect occupancy state may contribute to poor ventilation, inadequate lighting, inappropriate access decisions, or an inaccurate understanding of who may be present during an incident or evacuation.

Example SISEN command:

```
python3 attacks/run_attack.py --category safety-case --scenario building --attack blocked-exit
    -hidden
```

Expected observable effect:

The dashboard should show an incorrect occupancy state. Any dependent controller or status change implemented by the scenario should also be recorded.

Stopping and recovery:

Allow the helper to complete or stop it using `Ctrl+C`. Confirm that legitimate occupancy updates resume and that dependent system states return to baseline.

15 6LoWPAN Protocol-Path Activities

The 6LoWPAN scenario carries telemetry through several stages, including the constrained node, IEEE 802.15.4 link, 6LoWPAN adaptation, UDP forwarding, border-router processing, relay, and application layer. Students should use evidence from these stages to identify where spoofing, replay, loss, or manipulation entered the communication path.

15.1 Tracing an Abnormal 6LoWPAN Reading

What it is:

The 6LoWPAN protocol-path activities exercise telemetry as it moves from a constrained node through the low-power communication path and towards the application or MQTT relay. The available modes represent spoofed, extreme, missing, or replayed telemetry.

General 6LoWPAN helpers select one eligible industrial asset at the start of the run and keep that target fixed for the full bounded refresh or replay window. With four active nodes, the eligible assets are Boiler Room (`node-01`), Process Line (`node-02`), Cold Storage (`node-03`), and Loading Bay (`node-04`). Target selection uses a shuffled per-attack rotation, and the injected payload remains specific to the selected asset.

What it demonstrates:

These activities demonstrate that the same application-visible failure can originate at different points in the end-to-end path. Students should distinguish the application symptom from the communication-layer evidence and identify where trust is lost.

Safety relevance:

A dashboard may show false, stale, or missing environmental state regardless of whether the initiating problem occurred at the sensor, low-power link, border router, UDP forwarding stage, relay, or application layer. Safety reasoning therefore requires evidence across the whole path.

Example SISEN commands:

```
python3 attacks/run_attack.py --category protocol --scenario 6lowpan --attack spoofed
python3 attacks/run_attack.py --category protocol --scenario 6lowpan --attack extreme
python3 attacks/run_attack.py --category protocol --scenario 6lowpan --attack missing
python3 attacks/run_attack.py --category protocol --scenario 6lowpan --attack replay
```

Expected observable effect:

The application should show the selected abnormal telemetry condition. Students should correlate that effect with UDP, 6LoWPAN, relay, log, or MQTT evidence available in the running scenario.

Stopping and recovery:

Allow the helper to complete or stop it using `Ctrl+C`. Confirm that the legitimate constrained-node telemetry path resumes.

16 Medical IoT Gateway-Path Activities

16.1 Gateway-Path Spoofing, Manipulation, Replay, and Critical Vitals

What it is:

These activities alter patient telemetry as it is represented through the implemented Medical IoT gateway and MQTT path. They model the consequences of weak trust between the wearable source, gateway, broker, and dashboard.

What it demonstrates:

The activities demonstrate source impersonation, integrity failure, Message Freshness failure, and unsafe interpretation of critical values. They do not by themselves prove that a real BLE protocol attack has occurred.

Safety relevance:

The dashboard may show false, stale, or missing patient data. Such failures can create unnecessary intervention, delayed intervention, or loss of confidence in monitoring. The safety argument must therefore consider both data correctness and explicit failure indication.

Example SISEN commands:

```
python3 attacks/run_attack.py --category authenticity --scenario medical --attack spoofed
python3 attacks/run_attack.py --category integrity --scenario medical --attack extreme
python3 attacks/run_attack.py --category replay --scenario medical --attack replay
python3 attacks/run_attack.py --category safety-case --scenario medical --attack critical-
```

```
vitals
```

Expected observable effect:

The patient dashboard should display the selected false, abnormal, or stale telemetry condition. Logs and MQTT evidence should be used to identify the source of the displayed state.

Stopping and recovery:

Allow the helper to complete or stop it using `Ctrl+C`. Confirm that current readings from the legitimate medical telemetry generator resume.

BLE Scope Limitation

The current Medical IoT activities operate through the implemented gateway and MQTT-visible telemetry path. They must not be described as true BLE spoofing, BLE jamming, or BLE relay attacks unless the runtime produces and exposes the corresponding BLE-layer behaviour.

17 Evidence and Security-to-Safety Analysis

For each activity, record:

- the normal baseline before the activity;
- the exact command used;
- the affected scenario component, interface, topic, and communication path;
- the security property affected;
- the abnormal technical effect;
- the dashboard, log, terminal, or packet-capture evidence, including a precise filename, timestamp, interface, topic, packet number, or screenshot reference;
- whether the system explicitly detected the failure;
- the resulting operational or physical-system interpretation;
- the potential safety hazard and consequence;
- any assumptions required to connect the observed event to the safety consequence;
- the stopping and recovery behaviour, together with evidence that normal operation resumed;
- one or more proposed mitigations;
- whether each control is proposed, implemented, tested, or only partially effective; and
- the residual risk after mitigation.

The final explanation should distinguish clearly between direct observation, inference, and assumption, and between the initiating security event, the immediate system effect, the hazardous system condition, and the possible safety consequence.