

Cybersecurity, flight software and operational resilience

Protect the digital functions of an isolated base: trust architecture, authentication, updates, critical software, logs, segmentation and recovery after compromise.

Before you begin — Prerequisites: modules 00–34 as relevant. Every important symbol is defined at first use.

Mastery objectives

- connect principles to architecture or operational decisions
- repeat simple calculations and verify units and assumptions
- identify degraded modes, interfaces and uncertainty
- produce a verifiable procedure or plan

1. On Mars a cyber incident can become a physical failure

Software controls power, air, water, communications, robots and inventories. Unauthorized actions or software defects can therefore create physical consequences. Cybersecurity must be integrated with functional safety: protecting information is not enough if a falsified command can stop a pump.

2. Reduce attack surface through architecture

Critical systems should not all share one flat network. Segmenting habitat, workshop, science, visitor, robotics and administration networks limits propagation. Gateways should allow only required flows and log sensitive exchanges.

3. Identity, authentication and least privilege

A user account should not have more authority than its role requires. Critical operations may require stronger authentication, two-person approval or separated roles. The system must also remain usable when the Earth link is unavailable.

4. Updates and chain of trust

A useful software patch becomes a risk if it is corrupted or incompatible. Packages should be signed, verified, tested against a reference environment and deployed gradually. Known system images and previous versions must permit rollback.

5. Logging and detection without overwhelming the crew

Logs should record critical commands, configuration changes, failed authentication and communication anomalies. Too many alerts make the system unusable, so detection must prioritize events while retaining enough evidence to reconstruct an incident.

6. Backup, safe mode and restoration

A base needs clean offline recovery configurations. Backups must be separated from the systems they protect, tested and documented. Some critical controllers can maintain a minimal local safe function—pressure, temperature or circulation—even when higher-level networks are unavailable.

7. Building trust boundaries between life-critical and convenience functions

Not every network on a Mars base has the same criticality. A recreation terminal, a science server and the controller of an oxygen loop should not share identical access paths or privileges. Segmentation limits how far a software defect or compromise can propagate, but it must remain compatible with emergency operations. Isolating a network cannot prevent an authorized crew from controlling life-critical equipment locally. The architecture therefore needs trust zones, controlled gateways, essential services, maintenance paths and offline operating modes. Cybersecurity becomes a property of functional architecture rather than a layer added after deployment.

8. Updating software without turning a patch into a new failure

A software update can remove a vulnerability while introducing a compatibility problem. Mars crews cannot rely on rapid hardware replacement or immediate vendor intervention. Patches should therefore be authenticated, tested on representative configurations, deployed in stages and paired with rollback capability. Critical systems may preserve a known-good software image separately from the active version. Logs should make it possible to prove which version was running during an incident. Patch management therefore becomes configuration management and safety engineering as much as cybersecurity.

9. Incident response: preserve the mission before hunting for attribution

When suspicious behavior appears, the operational priority is to preserve life-critical functions and stop propagation. The crew must be able to isolate a machine, enter a degraded mode, preserve logs, verify command integrity and rebuild service from a trusted baseline. Detailed forensic analysis can follow. On a distant system, incident response should be rehearsed in simulation: who authorizes isolation, which functions may be disconnected, what must remain available and how is a clean recovery confirmed? The procedure must still work when the Earth link is unavailable.

10. Worked example: patching workload

Assume 120 software-controlled devices. Manual preparation and verification takes 25 min per device, or 3,000 min = 50 h. If automation removes 80% of the repetitive work and the operator spends 5 min per device, labor falls to 10 h. Automation must still produce evidence and logs rather than hiding the activity.

Deeper engineering: patch a life-critical system without creating a common-cause failure

A software patch can remove a vulnerability and create an operational hazard at the same time. On Mars the update strategy must therefore separate life-critical functions, authenticate packages and preserve a route back to the last known-good configuration. NASA software requirements emphasise planning, security and configuration control because changing code also changes the physical system that code commands. [NASA NPR 7150.2D](#)

Worked example. A habitat has 40 identical controllers. Updating all 40 at once potentially exposes 100% of that family to a common regression. A four-wave strategy of ten controllers does not change the intrinsic probability that the package contains a defect, but it limits the initial exposure radius to $10 \div 40 = 25\%$. If the first wave operates for a defined observation period, the next can begin. A genuinely life-critical function should go further still: independent redundant versions, a representative test bench and a verified rollback path before deployment.

The 25% figure is not a universal recommendation. It illustrates the difference between individual risk and common-cause risk. A perfectly authenticated update can still be dangerous if it installs the same defect everywhere. Cyber resilience for a settlement therefore depends on segmentation, diversity and restoration as much as it depends on encryption or passwords. [NASA — Spacecraft Software Engineering](#)

11. Progressive exercise

Draw a network containing ECLSS, power generation, laboratory, robots and personal devices. Mark which communications are necessary and which should be blocked by default.

Reasoned correction

A defensible architecture separates life-critical operational networks from science and personal-device networks. ECLSS and power distribution exchange only the states required for power management and do not accept inbound sessions from personal devices. The laboratory receives time and approved telemetry and can deposit results through an intermediate zone, but it does not directly command life-support controllers. Robots use an authenticated gateway with a minimal command set. Personal devices reach published services, never control buses. Every exception is documented, logged and constrained by protocol, direction and identity: “deny by default, open only for a demonstrated need” becomes the design rule.

Mini-project

Write an incident-response plan for compromise of a maintenance server: detection, isolation, continuity of vital functions, evidence collection, restoration, validation and return to service.

Cyber resilience on Mars means preventing software faults from becoming physical emergencies

A Mars settlement is a cyber-physical system. Software does not merely display information: it opens valves, controls pumps, allocates power, commands rovers, interprets radiation monitors and may manage life-support loops. A malicious command, corrupted update or accidental configuration error can therefore produce the same operational consequence as a broken cable or failed valve. Cybersecurity must be designed as part of system safety, not added later as an office-computing feature.

The most important architectural move is separation. A crew entertainment network, a scientific workstation and a life-support controller do not deserve the same trust boundary or the same privileges. Critical functions should have limited interfaces, authenticated commands, explicit allowed states and a recoverable safe mode. The goal is not to create an impossible promise of zero intrusion. It is to prevent one compromised component from gaining enough authority to create a colony-wide failure.

Updates are a special hazard because a patch can remove one vulnerability while introducing another defect. Earth can review software, but communication delay and local conditions mean the settlement still needs staged deployment, signed packages, rollback capability, configuration records and a known-good recovery image. A patch is not complete when it installs successfully; it is complete when the new configuration has passed declared functional checks and the team can still return to a safe previous state.

Cyber response also competes for crew attention. Logs can contain millions of events while only a few matter operationally. Detection therefore needs prioritization: which event threatens safety, which event changes trust in a system, which can wait for Earth analysis, and which requires immediate local isolation? A mature response plan preserves the mission first, evidence second and attribution third. Finding the attacker is less urgent than keeping oxygen, thermal control and electrical distribution inside safe limits.

Four ideas that connect cybersecurity, flight software and operational safety

Least privilege

Each account, service and device receives only the authority required for its current function. A camera should not possess credentials that can reconfigure an oxygen controller simply because both share a network. Least privilege reduces the consequences of a compromised component.

Trust boundary

A trust boundary is a controlled interface across which identity, data or commands move between zones with different assurance levels. Crossing the boundary should require explicit checks rather than assuming that everything inside the habitat is trustworthy.

Chain of trust

A chain of trust links boot firmware, operating software, configuration and updates through verifiable signatures or measurements. Its purpose is to make unauthorized modification detectable before a component is allowed to control critical hardware.

Safe mode

Safe mode is a deliberately limited configuration that preserves essential functions while disabling or constraining uncertain functions. It is valuable only if crews can enter, verify and exit it using procedures that do not depend on the same failed software path.

Calculation laboratory

Formula 1 — patch exposure budget

Canonical quantitative mini-lessons

This block is the module quantitative reference; exercise values are pedagogical and must be replaced by mission data before any real decision.

Patch deployment workload

$$H_{\text{patch}} = n_{\text{assets}} \times t_{\text{patch}}$$

1 — Concrete question

What does “ $H_{\text{patch}} = n_{\text{assets}} \times t_{\text{patch}}$ ” compute in “Patch deployment workload”?

2 — Intuition without symbols

Operational patch cost depends on asset count and safe time required for each.

3 — Quantities

H_{patch} : total patch workload [h]; n_{assets} : assets to patch [asset]; t_{patch} : average time per asset [h/actif]

4 — Formula

$$H_{\text{patch}} = n_{\text{assets}} \times t_{\text{patch}}$$

5 — Read aloud

Read “ $H_{\text{patch}} = n_{\text{assets}} \times t_{\text{patch}}$ ” by naming every operation, subscript and grouping explicitly.

6 — Symbols and meaning

H_{patch} : total patch workload [h]; n_{assets} : assets to patch [asset]; t_{patch} : average time per asset [h/actif]

7 — Pronunciation

Say subscripts, powers, parentheses and Greek letters explicitly so the relation can be reconstructed without seeing the page.

8 — Units

H_{patch} [h]; n_{assets} [asset]; t_{patch} [h/actif]

9 — Convention

Keep the same frame, time basis, system boundary and sign convention throughout the substitution.

10 — Why this operation

Multiplication converts a rate, fraction or specific quantity into a total.

11 — Assumptions

Factors must belong to the same scenario and their units must combine correctly.

12 — Independent check

Dividing the result by a non-zero factor should recover the product of the others.

13 — Numerical case

With $n_{\text{assets}} = 20$ asset, $t_{\text{patch}} = 0.5$ h/actif: $H_{\text{patch}} = 20 \times 0.5 = 10$ h.

14 — Why the calculation works

Multiplication converts a rate, fraction or specific quantity into a total.

15 — Verification

Dividing the result by a non-zero factor should recover the product of the others.

16 — Mental estimate

Estimating each factor separately quickly predicts product scale.

17 — Interpretation

Plan enough time for validation and rollback rather than treating a patch as a file copy.

18 — What the result does not prove

The calculation proves neither that inputs are exact nor that the model remains valid outside the declared scenario; both require separate evidence.

19 — Sensitivity

A relative change in one factor transfers directly to the product when the others stay fixed.

20 — Guided and autonomous exercises

Guided exercise. Recalculate this scenario: With $n_{\text{assets}} = 50$ asset, $t_{\text{patch}} = 0.2$ h/actif: $H_{\text{patch}} = 50 \times 0.2$?

Detailed guided correction — open after trying

With $n_{\text{assets}} = 50$ asset, $t_{\text{patch}} = 0.2$ h/actif: $H_{\text{patch}} = 50 \times 0.2 = 10$ h. The decision must then be checked against the module margins and assumptions.

Autonomous exercise. Recalculate this scenario: With $n_{\text{assets}} = 12$ asset, $t_{\text{patch}} = 1.5$ h/actif: $H_{\text{patch}} = 12 \times 1.5$?

Autonomous correction — open after trying

With $n_{\text{assets}} = 12$ asset, $t_{\text{patch}} = 1.5$ h/actif: $H_{\text{patch}} = 12 \times 1.5 = 18$ h. The decision must then be checked against the module margins and assumptions.

21 — Mission decision

Plan enough time for validation and rollback rather than treating a patch as a file copy.

Exposure window

$t_{\text{exposure}} = t_{\text{detect}} + t_{\text{validate}} + t_{\text{deploy}}$

1 — Concrete question

What does " $t_{\text{exposure}} = t_{\text{detect}} + t_{\text{validate}} + t_{\text{deploy}}$ " compute in "Exposure window"?

2 — Intuition without symbols

Exposure persists through detection, validation and deployment; these delays add.

3 — Quantities

t_{exposure} : exposure duration [h]; t_{detect} : detection time [h]; t_{validate} : validation time [h]; t_{deploy} : deployment time [h]

4 — Formula

$t_{\text{exposure}} = t_{\text{detect}} + t_{\text{validate}} + t_{\text{deploy}}$

5 — Read aloud

Read " $t_{\text{exposure}} = t_{\text{detect}} + t_{\text{validate}} + t_{\text{deploy}}$ " by naming every operation, subscript and grouping explicitly.

6 — Symbols and meaning

t_{exposure} : exposure duration [h]; t_{detect} : detection time [h]; t_{validate} : validation time [h]; t_{deploy} : deployment time [h]

7 — Pronunciation

Say subscripts, powers, parentheses and Greek letters explicitly so the relation can be reconstructed without seeing the page.

8 — Units

t_{exposure} [h]; t_{detect} [h]; t_{validate} [h]; t_{deploy} [h]

9 — Convention

Keep the same frame, time basis, system boundary and sign convention throughout the substitution.

10 — Why this operation

Addition combines delays or contributions that accumulate in the same operational chain.

11 — Assumptions

Contributions must use a common unit and the same system boundary.

12 — Independent check

Removing one term from the total should recover the sum of the others.

13 — Numerical case

With $t_{\text{detect}} = 2$ h, $t_{\text{validate}} = 4$ h, $t_{\text{deploy}} = 3$ h: $t_{\text{exposure}} = 2 + 4 + 3 = 9$ h.

14 — Why the calculation works

Addition combines delays or contributions that accumulate in the same operational chain.

15 — Verification

Removing one term from the total should recover the sum of the others.

16 — Mental estimate

Adding dominant terms first gives a robust order of magnitude.

17 — Interpretation

Reduce the term dominating the window without removing necessary safety checks.

18 — What the result does not prove

The calculation proves neither that inputs are exact nor that the model remains valid outside the declared scenario; both require separate evidence.

19 — Sensitivity

The total changes linearly with each term when the others stay fixed.

20 — Guided and autonomous exercises

Guided exercise. Recalculate this scenario: With $t_{\text{detect}} = 0.5$ h, $t_{\text{validate}} = 2$ h, $t_{\text{deploy}} = 1$ h: $t_{\text{exposure}} = 0.5 + 2 + 1$?

Detailed guided correction — open after trying

With $t_{\text{detect}} = 0.5$ h, $t_{\text{validate}} = 2$ h, $t_{\text{deploy}} = 1$ h: $t_{\text{exposure}} = 0.5 + 2 + 1 = 3.5$ h. The decision must then be checked against the module margins and assumptions.

Autonomous exercise. Recalculate this scenario: With $t_{\text{detect}} = 8$ h, $t_{\text{validate}} = 6$ h, $t_{\text{deploy}} = 4$ h: $t_{\text{exposure}} = 8 + 6 + 4$?

Autonomous correction — open after trying

With $t_{\text{detect}} = 8$ h, $t_{\text{validate}} = 6$ h, $t_{\text{deploy}} = 4$ h: $t_{\text{exposure}} = 8 + 6 + 4 = 18$ h. The decision must then be checked against the module margins and assumptions.

21 — Mission decision

Reduce the term dominating the window without removing necessary safety checks.

Restoration time

$t_{\text{restore}} = t_{\text{isolate}} + t_{\text{restore_data}} + t_{\text{verify}}$

1 — Concrete question

What does “ $t_{\text{restore}} = t_{\text{isolate}} + t_{\text{restore_data}} + t_{\text{verify}}$ ” compute in “Restoration time”?

2 — Intuition without symbols

Returning to service requires isolating the fault, restoring a known state and verifying it is actually sound.

3 — Quantities

t_{restore} : total restoration time [h]; t_{isolate} : isolation time [h]; $t_{\text{restore_data}}$: data restoration time [h];
 t_{verify} : verification time [h]

4 — Formula

$t_{\text{restore}} = t_{\text{isolate}} + t_{\text{restore_data}} + t_{\text{verify}}$

5 — Read aloud

Read “ $t_{\text{restore}} = t_{\text{isolate}} + t_{\text{restore_data}} + t_{\text{verify}}$ ” by naming every operation, subscript and grouping explicitly.

6 — Symbols and meaning

t_{restore} : total restoration time [h]; t_{isolate} : isolation time [h]; $t_{\text{restore_data}}$: data restoration time [h];
 t_{verify} : verification time [h]

7 — Pronunciation

Say subscripts, powers, parentheses and Greek letters explicitly so the relation can be reconstructed without seeing the page.

8 — Units

t_{restore} [h]; t_{isolate} [h]; $t_{\text{restore_data}}$ [h]; t_{verify} [h]

9 — Convention

Keep the same frame, time basis, system boundary and sign convention throughout the substitution.

10 — Why this operation

Addition combines delays or contributions that accumulate in the same operational chain.

11 — Assumptions

Contributions must use a common unit and the same system boundary.

12 — Independent check

Removing one term from the total should recover the sum of the others.

13 — Numerical case

With $t_{\text{isolate}} = 1$ h, $t_{\text{restore_data}} = 3$ h, $t_{\text{verify}} = 2$ h: $t_{\text{restore}} = 1 + 3 + 2 = 6$ h.

14 — Why the calculation works

Addition combines delays or contributions that accumulate in the same operational chain.

15 — Verification

Removing one term from the total should recover the sum of the others.

16 — Mental estimate

Adding dominant terms first gives a robust order of magnitude.

17 — Interpretation

Compare this time with safe-mode endurance before accepting the recovery architecture.

18 — What the result does not prove

The calculation proves neither that inputs are exact nor that the model remains valid outside the declared scenario; both require separate evidence.

19 — Sensitivity

The total changes linearly with each term when the others stay fixed.

20 — Guided and autonomous exercises

Guided exercise. Recalculate this scenario: With $t_{\text{isolate}} = 0.5$ h, $t_{\text{restore_data}} = 1$ h, $t_{\text{verify}} = 1$ h:
 $t_{\text{restore}} = 0.5 + 1 + 1$?

Detailed guided correction — open after trying

With $t_{\text{isolate}} = 0.5$ h, $t_{\text{restore_data}} = 1$ h, $t_{\text{verify}} = 1$ h: $t_{\text{restore}} = 0.5 + 1 + 1 = 2.5$ h. The decision must then be checked against the module margins and assumptions.

Autonomous exercise. Recalculate this scenario: With $t_{\text{isolate}} = 2$ h, $t_{\text{restore_data}} = 6$ h, $t_{\text{verify}} = 3$ h:
 $t_{\text{restore}} = 2 + 6 + 3$?

Autonomous correction — open after trying

With $t_{\text{isolate}} = 2$ h, $t_{\text{restore_data}} = 6$ h, $t_{\text{verify}} = 3$ h: $t_{\text{restore}} = 2 + 6 + 3 = 11$ h. The decision must then be checked against the module margins and assumptions.

21 — Mission decision

Verified backup coverage

$$C_{\text{backup}} = n_{\text{verified}} / n_{\text{critical}}$$

1 — Concrete question

What does “ $C_{\text{backup}} = n_{\text{verified}} / n_{\text{critical}}$ ” compute in “Verified backup coverage”?

2 — Intuition without symbols

A backup is a capability only when restoration has been verified for critical assets.

3 — Quantities

C_{backup} : verified coverage [sans dimension]; n_{verified} : assets with verified restore [asset]; n_{critical} : critical assets [asset]

4 — Formula

$$C_{\text{backup}} = n_{\text{verified}} / n_{\text{critical}}$$

5 — Read aloud

Read “ $C_{\text{backup}} = n_{\text{verified}} / n_{\text{critical}}$ ” by naming every operation, subscript and grouping explicitly.

6 — Symbols and meaning

C_{backup} : verified coverage [sans dimension]; n_{verified} : assets with verified restore [asset]; n_{critical} : critical assets [asset]

7 — Pronunciation

Say subscripts, powers, parentheses and Greek letters explicitly so the relation can be reconstructed without seeing the page.

8 — Units

C_{backup} [sans dimension]; n_{verified} [asset]; n_{critical} [asset]

9 — Convention

Keep the same frame, time basis, system boundary and sign convention throughout the substitution.

10 — Why this operation

Division relates a quantity to a coherent capacity, duration or reference.

11 — Assumptions

Numerator and denominator must describe the same scenario and the denominator must remain positive.

12 — Independent check

Multiplying the result by the denominator should reconstruct the numerator.

13 — Numerical case

With $n_{\text{verified}} = 18$ asset, $n_{\text{critical}} = 20$ asset: $C_{\text{backup}} = 18 / 20 = 0.9$.

14 — Why the calculation works

Division relates a quantity to a coherent capacity, duration or reference.

15 — Verification

Multiplying the result by the denominator should reconstruct the numerator.

16 — Mental estimate

Comparing numerator and denominator scales predicts the result.

17 — Interpretation

Treat incomplete coverage as resilience debt rather than a routine IT task.

18 — What the result does not prove

The calculation proves neither that inputs are exact nor that the model remains valid outside the declared scenario; both require separate evidence.

19 — Sensitivity

The result rises with the numerator and falls as the denominator rises.

20 — Guided and autonomous exercises

Guided exercise. Recalculate this scenario: With $n_{\text{verified}} = 30$ asset, $n_{\text{critical}} = 30$ asset: $C_{\text{backup}} = 30 / 30$?

[Detailed guided correction — open after trying](#)

With $n_{\text{verified}} = 30$ asset, $n_{\text{critical}} = 30$ asset: $C_{\text{backup}} = 30 / 30 = 1$. The decision must then be checked against the module margins and assumptions.

Autonomous exercise. Recalculate this scenario: With $n_{\text{verified}} = 9$ asset, $n_{\text{critical}} = 12$ asset: $C_{\text{backup}} = 9 / 12$?

Autonomous correction — open after trying

With $n_{\text{verified}} = 9$ asset, $n_{\text{critical}} = 12$ asset: $C_{\text{backup}} = 9 / 12 = 0.75$. The decision must then be checked against the module margins and assumptions.

21 — Mission decision

Treat incomplete coverage as resilience debt rather than a routine IT task.

Alert precision

$P_{\text{alert}} = n_{\text{true}} / n_{\text{alerts}}$

1 — Concrete question

What does " $P_{\text{alert}} = n_{\text{true}} / n_{\text{alerts}}$ " compute in "Alert precision"?

2 — Intuition without symbols

Precision measures the share of alerts that truly deserve crew attention.

3 — Quantities

P_{alert} : alert precision [sans dimension]; n_{true} : true relevant alerts [alert]; n_{alerts} : total alerts [alert]

4 — Formula

$P_{\text{alert}} = n_{\text{true}} / n_{\text{alerts}}$

5 — Read aloud

Read " $P_{\text{alert}} = n_{\text{true}} / n_{\text{alerts}}$ " by naming every operation, subscript and grouping explicitly.

6 — Symbols and meaning

P_{alert} : alert precision [sans dimension]; n_{true} : true relevant alerts [alert]; n_{alerts} : total alerts [alert]

7 — Pronunciation

Say subscripts, powers, parentheses and Greek letters explicitly so the relation can be reconstructed without seeing the page.

8 — Units

P_{alert} [sans dimension]; n_{true} [alert]; n_{alerts} [alert]

9 — Convention

Keep the same frame, time basis, system boundary and sign convention throughout the substitution.

10 — Why this operation

Division relates a quantity to a coherent capacity, duration or reference.

11 — Assumptions

Numerator and denominator must describe the same scenario and the denominator must remain positive.

12 — Independent check

Multiplying the result by the denominator should reconstruct the numerator.

13 — Numerical case

With $n_{\text{true}} = 80$ alert, $n_{\text{alerts}} = 100$ alert: $P_{\text{alert}} = 80 / 100 = 0.8$.

14 — Why the calculation works

Division relates a quantity to a coherent capacity, duration or reference.

15 — Verification

Multiplying the result by the denominator should reconstruct the numerator.

16 — Mental estimate

Comparing numerator and denominator scales predicts the result.

17 — Interpretation

Low precision creates fatigue and should trigger detector tuning before adding more alarms.

18 — What the result does not prove

The calculation proves neither that inputs are exact nor that the model remains valid outside the declared scenario; both require separate evidence.

19 — Sensitivity

The result rises with the numerator and falls as the denominator rises.

20 — Guided and autonomous exercises

Guided exercise. Recalculate this scenario: With $n_{\text{true}} = 45$ alert, $n_{\text{alerts}} = 60$ alert: $P_{\text{alert}} = 45 / 60$?

Detailed guided correction — open after trying

With $n_{\text{true}} = 45$ alert, $n_{\text{alerts}} = 60$ alert: $P_{\text{alert}} = 45 / 60 = 0.75$. The decision must then be checked against the module margins and assumptions.

Autonomous exercise. Recalculate this scenario: With $n_{\text{true}} = 10$ alert, $n_{\text{alerts}} = 50$ alert: $P_{\text{alert}} = 10 / 50$?

Autonomous correction — open after trying

With $n_{\text{true}} = 10$ alert, $n_{\text{alerts}} = 50$ alert: $P_{\text{alert}} = 10 / 50 = 0.2$. The decision must then be checked against the module margins and assumptions.

21 — Mission decision

Low precision creates fatigue and should trigger detector tuning before adding more alarms.

Operational risk index

$R_{\text{oper}} = p_{\text{compromise}} \times I_{\text{impact}}$

1 — Concrete question

What does " $R_{\text{oper}} = p_{\text{compromise}} \times I_{\text{impact}}$ " compute in "Operational risk index"?

2 — Intuition without symbols

A simple index combines probability and impact to rank scenarios without claiming exact prediction of the future.

3 — Quantities

R_{oper} : risk index [point]; $p_{\text{compromise}}$: scenario probability [sans dimension]; I_{impact} : normalized impact [point]

4 — Formula

$R_{\text{oper}} = p_{\text{compromise}} \times I_{\text{impact}}$

5 — Read aloud

Read " $R_{\text{oper}} = p_{\text{compromise}} \times I_{\text{impact}}$ " by naming every operation, subscript and grouping explicitly.

6 — Symbols and meaning

R_{oper} : risk index [point]; $p_{\text{compromise}}$: scenario probability [sans dimension]; I_{impact} : normalized impact [point]

7 — Pronunciation

Say subscripts, powers, parentheses and Greek letters explicitly so the relation can be reconstructed without seeing the page.

8 — Units

R_{oper} [point]; $p_{\text{compromise}}$ [sans dimension]; I_{impact} [point]

9 — Convention

Keep the same frame, time basis, system boundary and sign convention throughout the substitution.

10 — Why this operation

Multiplication converts a rate, fraction or specific quantity into a total.

11 — Assumptions

Factors must belong to the same scenario and their units must combine correctly.

12 — Independent check

Dividing the result by a non-zero factor should recover the product of the others.

13 — Numerical case

With $p_{\text{compromise}} = 0.1$ sans dimension, $I_{\text{impact}} = 8$ point: $R_{\text{oper}} = 0.1 \times 8 = 0.8$ point.

14 — Why the calculation works

Multiplication converts a rate, fraction or specific quantity into a total.

15 — Verification

Dividing the result by a non-zero factor should recover the product of the others.

16 — Mental estimate

Estimating each factor separately quickly predicts product scale.

17 — Interpretation

Use the index to prioritize mitigations, then review probability and impact assumptions separately.

18 — What the result does not prove

The calculation proves neither that inputs are exact nor that the model remains valid outside the declared scenario; both require separate evidence.

19 — Sensitivity

A relative change in one factor transfers directly to the product when the others stay fixed.

20 — Guided and autonomous exercises

Guided exercise. Recalculate this scenario: With $p_{\text{compromise}} = 0.02$ sans dimension, $I_{\text{impact}} = 10$ point: $R_{\text{oper}} = 0.02 \times 10$?

Detailed guided correction — open after trying

With $p_{\text{compromise}} = 0.02$ sans dimension, $I_{\text{impact}} = 10$ point: $R_{\text{oper}} = 0.02 \times 10 = 0.2$ point. The decision must then be checked against the module margins and assumptions.

Autonomous exercise. Recalculate this scenario: With $p_{\text{compromise}} = 0.3$ sans dimension, $I_{\text{impact}} = 4$ point: $R_{\text{oper}} = 0.3 \times 4$?

Autonomous correction — open after trying

With $p_{\text{compromise}} = 0.3$ sans dimension, $I_{\text{impact}} = 4$ point: $R_{\text{oper}} = 0.3 \times 4 = 1.2$ point. The decision must then be checked against the module margins and assumptions.

21 — Mission decision

Use the index to prioritize mitigations, then review probability and impact assumptions separately.

Starting question

How much aggregate exposure time is created when several vulnerable systems remain unpatched for a known interval?

Read aloud

Read: “exposure equals the number of affected systems times the time each remains open.”

Symbols, pronunciation and meaning

E is aggregate system-hours of exposure; N is the number of affected systems; t_{open} is the time before mitigation or patch completion.

Units

If time is measured in hours, E is system-hours. The quantity is a planning indicator, not a probability of attack.

Origin and status of values

N comes from the verified asset/configuration inventory. The open interval comes from detection time, review, staging and deployment records.

Why this operation

Multiplication is appropriate because every affected system contributes its own vulnerable time to the operational burden.

Substitution and calculation

If 12 controllers require a patch and remain exposed for 18 h, $E = 12 \times 18 = 216$ system-hours.

Calculator entry

Enter 12×18 . Keep the unit “system-hours” attached to the result so it is not misread as elapsed mission time.

Mental estimate

Ten systems for about twenty hours would be roughly 200 system-hours, so 216 is the expected order of magnitude.

Independent check

Divide 216 by 12 and recover 18 h per controller. The reverse operation checks the arithmetic.

Physical or operational interpretation

The indicator helps compare mitigation strategies. Isolating half the systems immediately can reduce aggregate exposure even before every patch is installed.

Plain-English translation

In plain language, the settlement carries the equivalent of 216 controller-hours of known vulnerability during the stated interval.

Variation / sensitivity

If isolation reduces the affected population from 12 to 4 while patch time remains 18 h, exposure falls to 72 system-hours.

Limit / assumption

Exposure time alone says nothing about exploitability, attacker access or consequence. Risk assessment must also consider likelihood, privilege and physical impact.

Formula 2 — recovery time objective against survival margin

Starting question

Does the planned restoration time leave enough margin before a critical resource reaches its safety limit?

Read aloud

Read: "margin equals survival time minus restore time."

Symbols, pronunciation and meaning

T_{survival} is the available safe operating duration after isolation; T_{restore} is the tested time to restore the required function; M is remaining time margin.

Units

All three quantities use the same time unit, such as minutes or hours.

Origin and status of values

Survival time comes from the physical system buffer or emergency reserve. Restore time should come from realistic recovery drills, not a theoretical file-copy duration.

Why this operation

Subtraction compares the resource deadline with the actual recovery task. A positive result means time remains; a negative result means the recovery plan is too slow.

Substitution and calculation

If a habitat can remain safe for 5 h in degraded mode and tested restoration requires 2.5 h, $M = 5 - 2.5 = 2.5$ h.

Calculator entry

Enter $5 - 2.5$. Record the assumptions behind both numbers in the incident plan.

Mental estimate

The two durations are equal halves, so a 2.5 h margin is immediately plausible.

Independent check

Add restore time and margin: $2.5 + 2.5 = 5$ h, recovering the survival window.

Physical or operational interpretation

A cyber recovery procedure that takes longer than the physical buffer is not operationally viable even if it is technically correct.

Plain-English translation

The team has about two and a half hours of timing reserve after the tested restoration task.

Variation / sensitivity

If restoration slows to 4 h, margin falls to 1 h. If it exceeds 5 h, another degraded or manual control path is required.

Limit / assumption

This simple margin ignores uncertainty and simultaneous crew tasks. Real planning should include conservatism, detection delay and the possibility that recovery attempts fail.

Mission reasoning: defend the settlement without making it impossible to operate

Segment by consequence, not by convenience

Network zoning should follow what a compromise could physically change. Life support, power switching, propulsion test equipment and medical systems deserve stricter boundaries than low-consequence services. A flat network is operationally convenient until one credential or device becomes the bridge to every critical controller.

Authenticate both people and machines

Human login is only one part of identity. Services, update packages, maintenance laptops and automated agents also need authenticated identities. The procedure should state what happens when identity cannot be proven: reject the command, move to a limited function, or require a local physical authorization path.

Stage updates and preserve rollback

The safest update is not always the fastest global deployment. A settlement can test on a noncritical twin or one redundant channel, observe declared health indicators, then expand the rollout. A signed update that breaks timing or sensor compatibility is still a dangerous update, so authenticity and functional qualification are separate gates.

Log events that support a decision

Logging every byte is useless if operators cannot identify changes that affect mission state. Critical logs should answer who or what changed configuration, when, from where, with which software identity, and whether the command succeeded. Time synchronization and protected audit records are essential for reconstructing incidents.

Recover from known-good foundations

A backup is only useful if it can be restored and if the restored system is trusted. Recovery drills should verify firmware, software version, configuration, credentials, network rules and critical data. Restoring a compromised configuration from a recent backup can faithfully recreate the incident.

Preserve operations before attribution

During an incident, local teams isolate affected paths, establish safe configuration and preserve enough evidence for later analysis. Chasing attribution while critical functions remain exposed wastes the limited resource that matters most: crew attention and survival margin.

Cyber-resilience exercises — decide what to isolate, trust and restore

Exercise A — Compromised maintenance laptop

A maintenance laptop used on a comfort network shows malware indicators, but it was also connected yesterday to a life-support service port. What is the first operational question?

Reveal the reasoned solution

Determine whether the laptop could have crossed a trust boundary and whether any critical configuration or credentials may now be untrusted. Isolate the laptop and affected interface, preserve logs, verify the controller state from an independent path, and avoid assuming that the absence of visible malfunction proves integrity.

Exercise B — Unsigned emergency patch

Earth sends a patch file through an improvised channel after a communications outage. The file may be legitimate but its normal signature cannot be verified. Should it be installed on the active oxygen controller?

Reveal the reasoned solution

Not by default. The system should remain on the last trusted configuration or move to an approved degraded/manual mode while authenticity is independently established. Urgency does not remove the need to know what code will control a vital function.

Exercise C — Too many alerts

A detector reports 8,000 anomalies after a network change. How should the crew triage?

Reveal the reasoned solution

Prioritize events that cross critical trust boundaries, alter privileges or configuration, affect safety functions, or coincide with physical anomalies. Bulk low-consequence events can be packaged for Earth analysis. The local response should reduce uncertainty around mission consequence first.

Exercise D — Rollback test

A new controller build passes installation but a valve timing test fails. What makes rollback acceptable?

Reveal the reasoned solution

The previous image and configuration must be known-good, compatible with the current hardware/data state, and recoverable through a tested path. Rollback should also preserve the anomaly evidence and record the configuration transition rather than silently erasing it.

Exercise E — Shared administrator account

Several crew members use one administrator password because it is easier during emergencies. Name two problems.

Reveal the reasoned solution

Account sharing destroys attribution and expands the consequence of one credential compromise. Better emergency access uses individually authenticated privileged elevation or a controlled break-glass mechanism with logging and post-use review.

Exercise F — Safe-mode timing

A cyber incident leaves 6 h of thermal buffer. Isolation and restoration drills take 4 h. What timing concern remains?

Reveal the reasoned solution

The nominal margin is only 2 h before adding detection delay, crew workload and possible failed recovery attempts. The plan should consider a longer physical buffer, a faster independent control path, or a tested contingency that does not rely on completing the full restoration within one attempt.

Interactive beginner glossary

These terms connect digital security to the physical and operational consequences that matter in a remote settlement.

- cybersecurity — Protection of digital systems, communications and information against unauthorized access, manipulation, disruption or loss.
- integrity — Property that data, software or configuration remain accurate and are not changed without authorized control.
- authentication — Process of verifying the claimed identity of a user, device, service or software package.
- authorization — Decision that an authenticated identity is allowed to perform a specific action or access a resource.
- least privilege — Principle of granting only the minimum permissions required for the intended task.
- trust boundary — Controlled interface between systems or zones that operate with different assumptions about identity and assurance.
- attack surface — Set of interfaces, services, accounts and pathways through which a system could potentially be influenced or compromised.
- segmentation — Division of networks or functions into controlled zones so compromise does not automatically propagate everywhere.

- air gap — Physical or logical separation intended to prevent direct network communication between systems; it is not proof that data can never cross.
- credential — Information or cryptographic material used to prove identity or obtain authorized access.
- multi-factor authentication — Authentication requiring evidence from more than one independent factor category.
- digital signature — Cryptographic mechanism used to verify who signed data and whether the signed content has been altered.
- hash — Fixed-length cryptographic digest used to detect changes in digital data when an appropriate algorithm is used.
- secure boot — Boot process that verifies approved software components before allowing them to execute.
- chain of trust — Linked sequence of verification steps from a trusted starting point through later software or configuration.
- patch — Software change intended to correct a defect, vulnerability or operational problem.
- rollback — Return from a new software or configuration state to a previously controlled state.
- configuration baseline — Approved record of the software, parameters, interfaces and versions that define a known system state.
- vulnerability — Weakness that could be exploited or accidentally triggered to violate intended security or operation.
- exploit — Method or code that uses a vulnerability to cause unintended behavior or gain unauthorized capability.
- incident response — Structured actions used to detect, contain, recover from and learn from a cybersecurity event.
- containment — Actions that limit propagation or consequence while preserving essential operation.
- forensic evidence — Records or data preserved so an incident can later be reconstructed and analysed.
- logging — Creation of time-stamped records of relevant system events, decisions, commands or errors.
- safe mode — Restricted system configuration designed to preserve essential functions while uncertain or failed functions are limited.
- backup — Protected copy of data, software or configuration kept for recovery after loss or corruption.
- restore — Process of returning required data or system function from a controlled backup or known-good image.
- recovery time objective — Target maximum duration within which a required function should be restored after disruption.
- break-glass access — Controlled emergency mechanism that grants exceptional privilege when normal access paths are unavailable.
- zero trust — Security approach that does not grant broad implicit trust solely because a user or device is inside a particular network location.

Operational depth: build cybersecurity around degraded modes and configuration evidence

Keep an authoritative asset inventory

A patch programme cannot protect devices that nobody knows exist. The inventory should link hardware identity, software build, safety function, network zone, responsible owner and update state. Portable maintenance tools and spare controllers matter because they can reintroduce old vulnerabilities after repair.

Use independent physical safing where justified

Some vital functions deserve a local manual or hardwired safing path that does not depend on the same network and software stack being investigated. Independence is expensive, so it should be reserved for consequences where loss of digital trust could threaten life before restoration is possible.

Treat removable media as a controlled transfer

An isolated system still receives software, logs or scientific data through physical media. The transfer process needs scanning, provenance, write protection where appropriate and clear separation between inbound and outbound media. An “air gap” without transfer discipline creates false confidence.

Drill compromise and software failure together

Operators may not immediately know whether strange behaviour is malicious, accidental or caused by hardware. Initial procedures should stabilize the function under uncertainty. A response that only works after identifying an attacker is too slow for a safety-critical colony.

Protect time and configuration records

Distributed systems need trustworthy clocks to reconstruct command order and correlate cyber events with physical telemetry. Configuration records should survive the incident so investigators can distinguish a corrupted parameter from a hardware failure or operator action.

Design recovery as an engineering test

Recovery images, spare controllers and credentials should be exercised periodically. The test measures real restore time and confirms that dependencies such as certificates, network rules and calibration data are present. An untested backup is an assumption, not a capability.

Operational review checklist

- Separate vital control networks from lower-consequence services.
- Give users, devices and services only the privileges they need.
- Verify signatures and configuration before deploying critical updates.
- Preserve a tested rollback or known-good recovery path.
- Make critical logs time-synchronized and resistant to silent alteration.
- Define local cyber stop rules and safe modes before incidents occur.
- Control removable media and maintenance interfaces as trust-boundary crossings.
- Measure real restoration time against physical survival buffers.
- Preserve evidence after safing the mission, not before it.
- Rehearse incidents in which the cause is still unknown.

Primary sources and bridges

- [NASA Software Engineering Handbook — Cybersecurity](#)
- [NASA NPR 7150.2D §3.11 — SWE-154, SWE-156, SWE-157, SWE-159 \(software cybersecurity\)](#)
- [NASA JSC — Spacecraft Software Engineering](#)

FR ↔ EN parity matrix

Patch deployment workload	Charge de déploiement des correctifs
Exposure window	Fenêtre d'exposition
Restoration time	Temps de restauration
Verified backup coverage	Couverture de sauvegarde vérifiée
Alert precision	Précision des alertes
Operational risk index	Indice de risque opérationnel