Chapter 11: Formal Verification

Learning Objectives

By the end of this chapter, students will be able to:

1. Explain the difference between testing and formal verification, and describe what each method can guarantee.
2. Build protocol models in the Applied Pi Calculus, using free names, private names, and equational theories.
3. Describe the Dolev-Yao attacker model and list the deduction rules an attacker can use.
4. Define and verify six key security properties: secrecy, authentication, forward secrecy, key freshness, replay protection, and sequence integrity.
5. Develop a bounded model checker that systematically explores states using breadth-first search, applying Dolev-Yao deduction rules at each step. Students should submit the full Python code of the checker with clear documentation and examples showing it working on a simple protocol.
6. Detect protocol downgrade vulnerabilities, including POODLE, DROWN, FREAK, Logjam, ROBOT, and SWEET32, through analysis of TLS configuration data. Students should produce a report that explains the detection criteria, includes sample configuration data, and summarizes findings using the detection tool or code they build.
7. Translate natural language security requirements into precise temporal logic formulas to enable automated verification. Students should provide a short written justification for each translation, clearly showing the original requirement and the resulting temporal logic formula side by side.

Agentic Lens

This chapter highlights why humility matters in formal verification. Agents build models, choose properties, and review counterexamples. Still, every claim must stay within the limits set by the model and its assumptions.

- **Agent role.** Assist with protocol modeling, property selection, and counterexample interpretation.
- **Observations.** Protocol transcripts, configuration data, attacker models, stated requirements, and prior proof results.
- **Tools.** Applied Pi Calculus models, bounded model checking, symbolic verification, downgrade checks, and runtime monitors.
- **State.** Current assumptions, abstractions introduced into the model, proof obligations, and unresolved counterexamples.
- **Verifier.** Proof engines, model sanity checks, assumption reviews, and implementation-level follow-up tests.
- **Guardrails.** A symbolic proof does not provide a comprehensive deployment guarantee. A failed proof may indicate deficiencies in the model as much as in the protocol itself.
- **A key lesson is that confusing what is true** in the model with what is true in the real system can lead to exaggerated security claims, especially when accuracy is crucial. For example, a protocol may be formally verified as secure under modeled assumptions. Still, in deployment, an overlooked implementation detail—such as a firmware update that enables protocol fallback or disablesa required security check—might eexposee a real vulnerability. In one scenario, a device that passed all model-based secrecy and authentication checks was later compromised in the field after a vendor patch changed certificate validation behavior, a factor not accounted for in the original model. This example shows why model-based confidence must always be tested against changes in real systems and adversarial conditions.

Threat Model and Assumptions

This chapter explains the attacker model and shows that the strength of any proof depends on how realistic its assumptions about the attacker are.

- **Threat model.** The primary attacker is symbolic and network-powerful: able to intercept, replay, reorder, compose, and inject messages within the modeled protocol environment.
- **Threat model.** Side channels, hardware faults, unsafe randomness, and implementation bugs matter too, but they are out of scope unless the model introduces them directly.
- **Assumption.** The extracted protocol model is a faithful abstraction of the implementation features that matter for the property being checked.
- **Assumption.** Security claims are conditional on the chosen abstraction, deduction rules, freshness assumptions, and search bounds. A proof result never outruns those premises.
- **Assumption.** An inconclusive bounded search is not evidence of safety. It means the current model and depth did not produce a counterexample.
- **Scope boundary:** Success in symbolic verification does not guarantee safe deployment. To be confident in the results, you must connect model findings to real systems through thorough testing and review.

11.1 Introduction: From Testing to Proving

Chapters 1 through 10 evaluate security empirically by probing ports, fuzzing inputs, consulting CVE databases, conducting penetration testing campaigns, and collecting deception telemetry. These approaches identify vulnerabilities in current deployments but cannot guarantee their absence. A scanner reporting zero vulnerabilities in a TLS implementation indicates only that no vulnerability was detected under the tested configuration at that time. Formal verification provides stronger assurances than conventional testing, but only within the constraints of the selected model and its assumptions. A verified security property encompasses *all p*rotocol executions represented by the model, including those not generated by fuzzers or scanners. For instance, the Needham-Schroeder Public-Key Protocol was considered secure until Gavin Lowe discovered an attack via formal model checking in 1996. Empirical testing did not reveal this flaw because it relied on a specific interleaving of three concurrent protocol sessions. Chapter 11 incorporates formal verification into the analytical workflow. For each device running TLS, MQTT, or CoAP, the engine extracts a formal model from the protocol transcript, defines security properties such as secrecy, authentication, and forward secrecy, and verifies them using either ProVerif (if available) or a built-in bounded model checker. Any violations produce counterexample attack traces that illustrate how an attacker could exploit the protocol. **Evidence label: Illustrative.** Method note: The analyst readout demonstrates the chapter's workflow and proof methodology, not a universal benchmark. Assumptions: The model extractor, bounded depth, and protocol selection are internally consistent with the chapter's toolchain. Boundary: This assessment is model-based and does not constitute a comprehensive claim of protocol correctness. A total of *42 devices were evaluated across TLS, MQTT, and CoAP. Thirty-eight received a GREEN verdict. Three were marked YELLOW due to inconclusive bounded analysis at the selected depth. One device (TLS 1.2 on 192.168.1.47) received a RED verdict for failing forward secrecy, as it uses RSA key exchange without ephemeral Diffie-Hellman. The corresponding attack trace consists of five modeled steps.*

11.2 Formal Methods Primer

11.2.1 Model Checking vs. Theorem Proving

Formal verification comes in two families:

Approach	Technique	Guarantees	Limitations
Model checking	Exhaustive state space exploration	Complete for finite models	State space explosion
Theorem proving	Logical deduction from axioms	Complete for all models	Requires expert guidance

Approach	Technique	Guarantees	Limitations
Symbolic verification	ProVerif-style Horn clause resolution	Unbounded sessions	May produce false positives

Breakwater uses model checking (bounded) as the primary engine and ProVerif (symbolic) when the external tool is installed. The bounded checker explores all reachable states up to a configurable depth, while ProVerif reasons about an unbounded number of protocol sessions.

11.2.2 Why IoT Protocols Need Formal Verification

IoT protocols are especially vulnerable to protocol-level attacks f**or several** reasons. For example, some TLS 1.3 implementations might agree to use weaker cipher suites.

1. **Legacy protocols** (MQTT 3.1.1, CoAP over UDP) were designed without encryption in mind.
2. **Downgrade attacks** exploit version negotiation to force connections to vulnerable protocol versions.
3. **Compositional failures** can arise when secure protocols interact and share state. For example, MQTT over TLS gets confidentiality from TLS, but MQTT's Quality of Service (QoS) features can still allow replay attacks.

```
graph TD
  A[Phase 1-6 Scan Results] --> B[Model Extractor]
  B --> C[Protocol Model]
  C --> D[Property Specification]
  D --> E{ProVerif Available?}
  E -->|Yes| F[ProVerif Engine]
  E -->|No| G[Bounded Model Checker]
  F --> H[Property Verdicts]
  G --> H
  H --> I[Attack Trace Generator]
  I --> J[Conformance Tester]
  J --> K[Downgrade Detector]
  K --> L[Verification Result]
  L --> M[HYDRA Integration]

  style B fill:#e1f5fe
  style C fill:#e8f5e9
  style D fill:#f3e5f5
  style F fill:#fff3e0
  style G fill:#fff3e0
  style I fill:#ffcdd2
  style K fill:#fce4ec
```

The verification pipeline runs for each host and protocol, adjust**ing its checks for** every unique case. It uses a Pi Calculus model built from protocol transcripts or a standard template.

1. **Property Specification** – load standard security properties for the protocol (6 properties for TLS, 5 for MQTT, 4 for CoAP).
2. **Verification** – try ProVerif first (subprocess), fall back to bounded model checker.
3. **Attack Trace Generation** – if a property is violated, extract a step-by-step counterexample.
4. **Conformance Testing** – check RFC compliance (minimum version, cipher preferences, certificate validation).
5. **Downgrade Detection** – check for POODLE, DROWN, FREAK, Logjam, ROBOT, and SWEET32 vulnerabilities.

11.4 The Applied Pi Calculus

11.4.1 Syntax

The Applied Pi Calculus (Abadi & Fournet, 2001) extends the pi calculus with user-defined functions and equational theories. A protocol model in Breakwater consists of:

- **Roles**: protocol participants (e.g., client, server).
- **Free names**: public channel names visible to the attacker (e.g., c for the network channel).
- **Private names**: secrets not initially known to the attacker (e.g., sk_server, premaster_secret).
- **Messages**: ordered sequence of send/receive actions on channels.
- **Equational theory**: rules that define how cryptographic functions reduce (e.g., dec(enc(m, k), k) = m).

11.4.2 CryptoTerm Representation

In Breakwater, the CryptoTerm dataclass represents terms in the equational theory:

```
CryptoTerm(
  type="enc",
  name="client_key_exchange",
  args=[
    CryptoTerm(type="nonce", name="premaster_secret"),
    CryptoTerm(type="pk", name="pk_server" This stands for enc(premaster_secret, pk(server)), which means the client encrypts the premaster secret with the server's public key, as in the RSA key exchange in TLS 1.2.y exchange.
```

11.4.3 TLS 1.3 Model

The ModelExtractor builds a complete TLS 1.3 model with four messages:

1. Client -> Server: (nonce_client, ecdhe_client)
2. Server -> Client: (nonce_server, ecdhe_server)
3. Server -> Client: enc((sign(hash(nonces), sk_server), Finished), session_key)
4. Client -> Server: enc(client_finished_verify, session_key)

Where session_key = hkdf_expand(ecdh(ecdhe_client, ecdhe_server), nonce_client, nonce_server).

```
# From apps/api/app/scanning/formal_verify/model_extractor.py

session_key = CryptoTerm(
  type="func",
  name="hkdf_expand",
  args=[
    CryptoTerm(
      type="func",
      name="ecdh",
      args=[ecdhe_c, ecdhe_s]
    ),
```

```
        nonce_c, nonce_s,
    ],
)
```

The equational theory includes:

```
dec(enc(m, pk(x)), sk(x)) = m     -- Asymmetric decryption
dec(enc(m, k), k) = m             -- Symmetric decryption
verify(sign(m, sk(x)), pk(x)) = m  -- Signature verification
fst(pair(x, y)) = x               -- Pair projection
snd(pair(x, y)) = y
```

11.5 The Dolev-Yao Attacker Model

11.5.1 Capabilities

The Dolev-Yao model (1983) describes a very powerful attacker, assuming perfect cryptography with no bugs or side channels. The attacker:

1. **Controls the network** – can intercept, drop, modify, replay, and inject messages on public channels.
2. **Has initial knowledge** – public keys of all participants, free names, and any data sent on public channels.
3. **Cannot break cryptography** – cannot decrypt without the correct key, cannot forge signatures without the private key, cannot invert hash functions.

11.5.2 Deduction Rules

The attacker's knowledge grows through seven deduction rules, implemented in the bounded checker:

```
# From apps/api/app/scanning/formal_verify/bounded_checker.py

DOLEV_YAO_RULES = [
    {"name": "pair", "arity": 2, "apply": lambda a, b: pair(a, b)},
    {"name": "fst",  "arity": 1, "apply": lambda p: p.args[0] if p.type == "pair"},
    {"name": "snd",  "arity": 1, "apply": lambda p: p.args[1] if p.type == "pair"},
    {"name": "enc",  "arity": 2, "apply": lambda m, k: enc(m, k)},
    {"name": "dec",  "arity": 2, "apply": lambda ct, k: ct. args[0] if key_matches},
    {"name": "hash", "arity": 1, "apply": lambda m: hash(m)},
    {"name": "sign", "arity": 2, "apply": lambda m, sk: sign(m, sk)},
]
```

The dec rule is conditional: the attacker can only decrypt enc(m, k) if they know k. The verify rule works the same way: it only succeeds if the signature was made with the matching private key.

11.5.3 Knowledge Closure

The deduce() method computes the closure of attacker knowledge under the deduction rules, iterating until a fixpoint is reached (no new terms can be derived):

```
def deduce(self, knowledge, rules):
    current = set(knowledge)
    changed = True
```

```
    while changed and iterations < 50:
        changed = False
        new_terms = set()
        for k in list(current):
            if k.startswith("pair:"):
                # fst and snd projections
                new_terms.add(first_component)
                new_terms.add(second_component)
            if k.startswith("enc:") and key_in_knowledge:
                new_terms.add(plaintext)
        additions = new_terms - current
        if additions:
            current |= additions
            changed = True
    return current
```

11.6 Security Properties

11.6.1 Property Taxonomy

The property_spec.py module defines six classes of security properties, each with a formal semantics expressed in temporal logic:

Property	Temporal Logic	Intuition
Secrecy	G(not attacker_knows(term))	The attacker never learns the secret term
Authentication	G(end_A(B) => F_past(begin_B(A)))	If A completes, B was genuinely participating
Forward secrecy	G(compromised(LTK) => G(not attacker_knows(SK)))	Past session keys survive key compromise
Key freshness	G(established(K) => not F_past(established(K)))	No key reuse across sessions
Replay protection	G(received(M) => not F_past(received(M)))	Captured messages cannot be re-injected
Sequence integrity	recv(M1) < recv(M2) < ... < recv(Mn)	Message ordering is preserved

11.6.2 Property Factory Functions

Each property type has a factory function that constructs a SecurityProperty with the appropriate target term, temporal formula, and attacker model:

```
# From apps/api/app/scanning/formal_verify/property_spec.py
def secrecy_property(term, attacker="attacker"):
    return SecurityProperty(
        name=f"secrecy_{term.name}",
        property_type="secrecy",
        target_term=term,
        temporal_formula=f"G(not attacker_knows({term.to_proverif()}))",
        attacker_model="dolev_yao",
    )
```

```
def forward_secrecy_property(session_key):
  return SecurityProperty(
    name=f"forward_secrecy_{session_key.name}",
    property_type="forward_secrecy",
    target_term=session_key,
    temporal_formula=(
      f"G(compromised(long_term_key) => "
      f"G(not attacker_knows({session_key.to_proverif()})))"
    ),
  )
```

11.6.3 Standard Property Sets

Each protocol has a predefined set of properties:

- **TLS** (6 properties): secrecy of session key, secrecy of premaster, client-server authentication, forward secrecy, key freshness, and replay protection of Finished message.
- **MQTT** (5 properties): secrecy of password, secrecy of payload, client-broker authentication, replay protection of PUBLISH, sequence integrity (CONNECT -> CONNACK -> PUBLISH).
- **CoAP** (4 properties): secrecy of payload, client-server authentication, replay protection of message ID, and token freshness.

11.7 Bounded Model Checking

11.7.1 Algorithm

The BounThe BoundedChecker uses breadth-first search to explore states. Each state includes the current step number, what the attacker knows, the protocol step index, and the list of actions taken so far.
apps/api/app/scanning/formal_verify/bounded_checker.py

```
class BoundedChecker:
  def check(self, model, prop):
    initial_knowledge = self._initial_attacker_knowledge(model)
    initial_state = {
      "step": 0,
      "knowledge": initial_knowledge,
      "protocol_step": 0,
      "trace": [],
    }
    queue = deque([initial_state])
    visited = set()
    while queue and steps_explored < self.max_depth:
      state = queue.popleft()
      if self.is_bad_state(state, prop):
        return PropertyVerdict(result="RED", attack_trace=trace)
      for successor in self.expand_state(state, model):
        if successor not in visited:
          queue.append(successor)
    return PropertyVerdict(result="GREEN")
```

11.7.2 State Expansion

At each state, two types of successors are generated:

1. **Protocol message steps: the next message in the protocol is sent. If it goes over a public channel, the attacker learns the payload and any parts that can be broken down. The deduction steps** – the Dolev-Yao rules are applied to expand the attacker's knowledge.

11.7.3 Bad State Detection

The checker evaluates property violations by type:

- **Secrecy**: the target term's hash appears in the attacker's knowledge set.
- **Authentication**: the attacker can construct a Finished message without legitimate party participation (no signature or encryption wrapping).
- **Forward secrecy**: the attacker knows both a long-term key (sk:*) AND the session key.
- **Replay protection**: the target message is in the attacker without nonce protection.
- **Key freshness**: the target key appears more than once in the trace.

11.7.4 Verdicts

The checker produces a traffic-light verdict:

- **GREEN** – GREEN: No violation found within the depth limit. The property holds for all states that were checked. Violation found. The attack trace shows the exact sequence of steps.
- **YELLOW** – inconYELLOW: The result is inconclusive (for example, ProVerif timed out or the depth limit was reached before all states could be checked). A[Initial State] --> B{Property Violated?}
  B -->|Yes| C[RED: Extract Attack Trace]
  B -->|No| D[Expand Successors]
  D --> E[Protocol Message Step]
  D --> F[Attacker Deduction Step]
  E --> G{Depth Limit?}
  F --> G
  G -->|No| B
  G -->|Yes| H[GREEN: No Violation in Bound]

  style C fill:#ffcdd2
  style H fill:#c8e6c9

11.8 ProVerif-Style Symbolic Verification

11.8.1 Integration Architecture

When the proverif binary is on $PATH, the VerificationEngine uses it as the primary verification backend. The engine generates a .pv input file from the model and property, runs ProVerif as a subprocess with a configurable timeout, and parses the output for verdicts.

```
# From apps/api/app/scanning/formal_verify/verification_engine.py
class VerificationEngine:
  def __init__(self):
    self._proverif_path = shutil.which("proverif")

  def verify_property(self, model, prop, timeout=120):
    if self._proverif_path:
      result = self.try_proverif(model, prop, timeout)
      if result is None:
        return result
```

```
        return self.fallback_bounded_check(model, prop, timeout)
```

11.8.2 ProVerif Input Generation

The generate_proverif_input() method translates the Breakwater model into ProVerif's input language:

```
(* Auto-generated by Breakwater formal verification *)
free c: channel.
free sk_server: bitstring [private].

fun enc(bitstring, bitstring): bitstring.
fun dec(bitstring, bitstring): bitstring.
fun pk(bitstring): bitstring.
fun sign(bitstring, bitstring): bitstring.
fun hash(bitstring): bitstring.

reduc forall m: bitstring, k: bitstring; dec(enc(m, k), k) = m.
reduce forall m: bitstring, sk: bitstring;
  verify_sig(sign(m, sk), pk(sk)) = m.

(* Secrecy query *)
free session_key_query: bitstring [private].
query attacker(session_key_query).

(* Authentication query *)
event begin_server(x: bitstring).
event end_client(x: bitstring).
query x: bitstring; event(end_client(x)) ==> event(begin_server(x)).
```

11.8.3 Output Parsing

ProVerif output is parsed for the patterns RESULT not attacker(...) is true (property holds) or RESULT not attacker(...) is false (property violated). Attack traces are extracted from the Derivation section of the output.

11.8.4 Advantages of ProVerif

ProVerif provides two key advantages over bounded model checking:

1. **Unbounded sessions** – ProVerif reasons about an arbitrary number of concurrent protocol sessions rather than a single-session bounded exploration.
2. **Automatic proof** – ProVerif uses Horn clause resolution to prove or disprove properties without manual guidance automatically.

The tradeoff is that ProVerif may produce false positives (reporting a property as violated when no real attack exists) due to over-approximation in its abstraction.

11.9 Model Extraction from Protocol Transcripts

11.9.1 The Extraction Problem

In an ideal deployment, the formal model would be written by hand from the protocol specification. In practice, Breakwater must verify devices on the network without access to their source code or specification. The ModelExtractor solves this by extracting formal models from observed protocol transcripts.

11.9.2 Three Extraction Strategies

The extractor supports three protocol-specific strategies and a generic fallback:

TLS (extract_tls_model): Inspects the TLS record layer to identify the version (1.2 vs 1.3), then generates the appropriate formal model. TLS 1.3 uses ephemeral ECDHE key shares and HKDF key derivation. TLS 1.2 uses RSA key exchange and PRF key derivation.

MQTT (extract_mqtt_model): Models the CONNECT/CONNACK/SUBSCRIBE/SUBACK/PUBLISH sequence with credential terms. MQTT over TLS adds a confidentiality assumption on the channel.

CoAP (extract_coap_model): Models CON/ACK request-response pairs with message IDs and tokens.

11.9.3 Generic Extraction via State Machine Inference

For unknown protocols, the extractor applies L*-style state machine learning:

```
# From apps/api/app/scanning/formal_verify/model_extractor.py
def infer_state_machine(self, messages):
    states = ["S0"]
    transitions = []
    current_state = "S0"
    for msg in messages:
        symbol = f"{msg['sender']}:{msg['type']}"
        key = f"{current_state}{symbol}"
        if key not in seen_patterns:
            next_state = f"S{state_index}"
            seen_patterns[key] = next_state
        transitions.append({"from": current_state, "to": next_state, ...})
        current_state = next_state
    return {"states": states, "transitions": transitions, "initial": "S0"}
```

Cryptographic operations are identified from byte patterns: TLS record headers (0x16 for handshake, 0x17 for application data), ASN.1 SEQUENCE headers (0x30 for certificates), and high-entropy blocks (>7.0 bits per byte, which is often consistent with encrypted or compressed data).

```
graph LR
    A[Protocol Transcript] --> B{Known Protocol?}
    B -->|TLS| C[TLS-Specific Extractor]
    B -->|MQTT| D[MQTT-Specific Extractor]
    B -->|CoAP| E[CoAP-Specific Extractor]
    B -->|Unknown| F[Generic State Machine]
    C --> G[Applied Pi Calculus Model]
    D --> G
    E --> G
    F --> H[Identify Crypto Ops]
    H --> G

    style G fill:#e8f5e9
    style F fill:#fff3e0
```

11.10 Attack Trace Generation

11.10.1 Counterexample Extraction

When the bounded checker finds a property violation (bad state), it extracts the trace – the sequence of actions from the initial state to the violation. Each trace step records:

```
{
  "step_number": 3,
  "actor": "attacker",
  "action": "dolev_yao_deduction",
  "message": "Apply deduction rules",
  "knowledge_gained": ["nonce:premaster_secret", "var:session_key"],
}
```

11.10.2 Example: Forward Secrecy Violation

Consider TLS 1.2 with RSA key exchange (no ephemeral Diffie-Hellman). The attack trace for forward secrecy violation:

1. **Step 1**: Client sends (nonce_client) on public channel. Attacker learns nonce_client.
2. **Step 2**: Server sends (nonce_server, pk_server) on public channel. Attacker learns nonce_server and pk_server.
3. **Step 3**: Client sends enc(premaster_secret, pk_server). Attacker records but cannot decrypt (lacks sk_server).
4. **Step 4**: Long-term key compromise event – attacker obtains sk_server.
5. **Step 5**: Attacker applies dec(enc(premaster_secret, pk_server), sk_server) = premaster_secret. Then computes session_key = prf(premaster, nonce_client, nonce_server).

The forThe forward secrecy property is broken here because if the long-term key is compromised, an attacker can decrypt the premaster secret and then get the session key. TLS 1.3 avoids this problem by using temporary ECDHE key shares. Even if the server's long-term key is later compromised, the temporary Diffie-Hellman secret is already gone.3 Counterexample-Guided Exploit Synthesis

The counterexample_exploiter.py module takes verified attack traces and attempts to synthesize concrete exploits – translating the symbolic attack into network-level actions (packet crafting, timing, injection points). This bridges the gap between formal verification and actionable security testing.

11.11 RFC Conformance Testing

11.11.1 Conformance Architecture

The ConformanceTester runs per-protocol checks against RFC specifications. Unlike formal verification (which reasons about the protocol *design*), conformance testing checks the protocol *implementation* on the scanned device.

TLS (RFC 8446 / RFC 5246) – 7 checks: 1. Minimum TLS version (>=1.2) 2. Cipher preference ordering (AEAD ciphers preferred) 3. Certificate validation (not expired, not self-signed) 4. Extension support (SNI, ALPN) 5. Renegotiation safety (secure renegotiation extension) 6. Compression disabled (CRIME mitigation) 7. Key rotation frequency

MQTT (OASIS Standard) – checks for authentication requirement, QoS level support, and payload encryption.

CoAP (RFC 7252) – checks for DTLS usage, message ID handling, and token randomness.

11.12 Downgrade Attack Detection

11.12.1 Known Downgrade Attacks

The DowngradeDetector checks for six known protocol downgrade vulnerabilities:

Attack	CVE	Type	Severity	Condition
POODLE	CVE-2014-3566	Version downgrade	High	SSLv3 enabled
DROWN	CVE-2016-0800	Cross-protocol	Critical	SSLv2 enabled
FREAK	CVE-2015-0204	Cipher downgrade	High	Export ciphers accepted
Logjam	CVE-2015-4000	Key exchange	High	DH groups < 1024 bits
ROBOT	CVE-2017-13099	Cipher downgrade	High	RSA PKCS#1 v1.5 padding oracle
SWEET32	CVE-2016-2183	Cipher weakness	Medium	64-bit block ciphers (3DES)

11.12.2 Detection from Scan Data

Each downgrade check examines TLS configuration data collected during Phase 2 enrichment:

```
# From apps/api/app/scanning/formal_verify/downgrade_detector.py
KNOWN_DOWNGRADE_ATTACKS = {
  "POODLE": {
    "cve_id": "CVE-2014-3566",
    "description": "Attacker forces fallback to SSLv3 then exploits CBC padding oracle.",
    "severity": "high",
    "mitigation": "Disable SSLv3 entirely; use TLS_FALLBACK_SCSV.",
  },
  "DROWN": {
    "cve_id": "CVE-2016-0800",
    "description": "SSLv2 exposure allows cross-protocol attack on TLS sessions.",
    "severity": "critical",
    "mitigation": "Disable SSLv2 on all servers sharing the RSA key.",
  },
  # ...
}
```

The detector examines supported protocol versions, accepted cipher suites, key exchange parameters, and certificate chain properties. A device that accepts SSLv3 connections is flagged for POODLE. A device with export-grade RSA ciphers enabled is flagged as vulnerable to FREAK.

11.13 Compositional Verification

11.13.1 The Composition Problem

Real-world deployments stack multiple protocols: MQTT over TLS, CoAP over DTLS, HTTP over TLS with client certificates. Each protocol may be individually secure, but their composition can introduce vulnerabilities through shared state, channel confusion, or authentication bypass.

11.13.2 Compositional Verification Strategy

The CompositionalVerifier module analyses protocol stacks by:

1. **Layer identification** – determine which protocols are stacked (e.g., MQTT at application layer, TLS at transport layer).
2. **Interface extraction** – identify shared terms between layers (e.g., TLS session key used by MQTT, the certificate identity used for MQTT authentication).
3. **Cross-layer property checking** – verify that properties at each layer are preserved through the composition. For example, if TLS provides session key secrecy, does MQTT correctly use this key for all payload encryption?
4. **Binding verification** – check that the transport-layer (TLS certificate) authentication is correctly bound to the application-layer (MQTT username/password) authentication.

This is an area of active research. The Breakwater implementation provides a practical first approximation, flagging common composition errors (e.g., MQTT sending plaintext credentials before TLS handshake completion, or CoAP falling back to unencrypted transport when DTLS fails).

Several open problems remain in compositional verification. One challenge is automated reasoning about security properties that span multiple protocol layers, where properties may not be preserved under composition or may require sophisticated cross-layer invariants. Another unresolved question involves the scalability of verification techniques as protocol stacks become more complex, especially in resource-constrained IoT and OT deployments. These open research questions invite doctoral students to develop new tools, compose stronger security guarantees, or design verification workflows capable of handling dynamic, multi-protocol environments.

```
graph TD
  A[MQTT Application Protocol] --> B[Shared Interface]
  C[TLS Transport Protocol] --> B
  B --> D[Compositional Verifier]
  D --> E{Properties Preserved?}
  E -->|Yes| F[GREEN: Composition Safe]
  E -->|No| G[RED: Cross-Layer Vulnerability]

  A --> A1[MQTT Properties]
  C --> C1[TLS Properties]
  A1 --> D
  C1 --> D

  style F fill:#c8e6c9
  style G fill:#ffcdd2
```

11.14 Runtime Monitors and NL-to-Logic Translation

11.14.1 Runtime Module The RuntimeMonitor module creates simple monitoring hooks based on verified security properties. Once formal verification shows that a property should hold, a runtime monitor continuously checks that it actually does in the live system. This helps connect the formal model to the real system. station.

Monitors are built as finite-state machines from temporal-logic rules. For example, the forward secrecy monitor checks if any session uses a non-temporary key exchange and raises an alert if RSA is used instead of ECDHE.Real Language to Temporal Logic

The NLToLogic translator (enabled when BREAKWATER_FORMAL_VERIFY_NL_ENABLED=true) converts natural language security requirements into temporal logic formulae using LLM-assisted parsing:

Input: "The MQTT password must never be visible to an eavesdropper."

Output: G(not attacker_knows(mqtt_password))

Input: "If the client believes it completed the handshake with the server, then the server was genuinely participating."

Output: G(end_client(server) => F_past(begin_server(client)))

This security engineers who are not specialists to define custom security properties without needing to know temporal logic syntax. Technical Expansion

11.B1 Proof artifact review

A proof artifact should be reviewable without running the tool. It should name the property, model, attacker, assumptions, timeout, verdict, and trace if present, the trace. That makes the proof portable across teams.

The review should distinguish proof from configuration. A green result for a model with no downgrade path does not prove the deployed device lacks a downgrade path. A green result for a bounded depth does not prove unbounded safety. A green result under a symbolic abstraction does not prove that the parser handles malformed packets safely. The artifact should say exactly what was modeled.

> **Caution:** A green verification result does not mean the device is secure. It means the model, under its stated assumptions, satisfies the property. The implementation may still diverge from the model in ways the proof cannot see.

For an OT protocol, the trace is often more valuable than the verdict. A five-step trace that shows nonce reuse, fallback, or unauthenticated state transition can teach an engineering team where to patch. A proof report that hides the trace behind a badge does not help the field team fix firmware.

11.B2 Implementation distance

The distance between the model and the firmware is the hard part. A verified abstract handshake can still fail when the device adds retries, fallback modes, vendor extensions, or state reuse. The chapter should train students to measure that distance.

The implementation distance should be written as a delta. The model says one session key per handshake. The firmware caches the state across reconnects. The model says there should be no plaintext credentials after authentication. The device logs one field for support. The model says certificate validation is required. The deployment turns off validation for an expired plant certificate. Each delta is a possible exploit path.

World-class verification does not end with the theorem. It builds a bridge from the theorem to packet capture, configuration export, firmware behavior, and operator exception. The student should be able to say which proof remains valid after an exception and which proof has been invalidated by the field reality.

11. A Advanced Practitioner Fieldbook: What Experts Still Miss

This fieldbook expands Chapter 11 for doctoral students and senior cybersecurity practitioners. It assumes the reader knows the vocabulary. The goal is harder. The reader must reason under pressure, preserve evidence boundaries, and explain why the analysis should be trusted when the environment fights back.

The prose uses short sentences on purpose. Short sentences expose weak claims. They also make the material teachable in a live seminar.

11.A1 The proof that did not cover the outage

A vendor claims the handshake is verified. The outage is due to retry behavior and downgrade fallback. The proof was scoped too narrowly.

Figure 11.8: Counterexample Trace for Authentication Violation

Figure 11.A1: ch11 fig08 counterexample trace.

Figure note: The counterexample trace belongs with the outage story. It shows how a failed property becomes an engineering script, not just a red badge.

For the proof that did not cover the outage, the review starts with the protocol trace and ends only when the claim has a named owner. The analyst should state what the system observed, what transformation changed the record, and what conclusion is still forbidden.

The specific danger in the proof that did not cover the outage is false closure. The attacker assumption may support one interpretation while counterexample traces toward another. The case should remain open until the proof boundary explains why the stronger claim is admissible or why it has been refused.

A senior answer for the proof that did not cover the outage produces three artifacts: a leadership sentence, a reproducible evidence note, and a hostile objection. The three artifacts should agree on the same boundary. If they do not, the analyst has confused persuasion with proof.

11.A2 Applied pi calculus as a discipline

The calculus forces analysts to name channels, terms, secrets, events, and attacker knowledge.

Figure 11.1: Applied Pi-Calculus Syntax

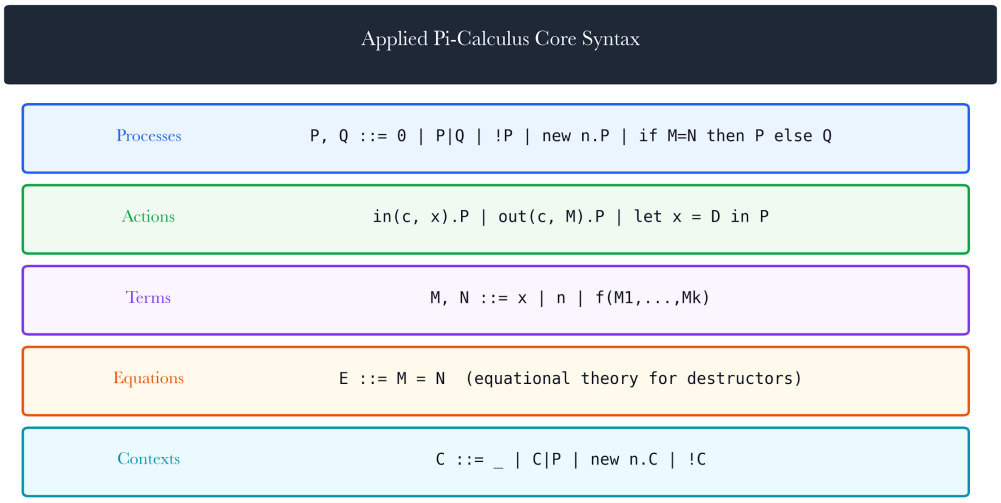

Figure 11.A2: ch11 fig01 applied pi calculus syntax.

Figure note: Applied pi calculus syntax belongs to the modeling discipline. The figure reminds students that notation is a contract with assumptions.

For applied pi calculus as a discipline, the review starts with the protocol trace and ends only when the claim has a named owner. The analyst should state what the system observed, what transformation changed the record, and what conclusion is still forbidden.

The specific danger in applied pi calculus as a discipline is false closure. The attacker assumption may support one interpretation, while a counterexample trace points toward another. The case should remain open until the proof boundary explains why the stronger claim is admissible or why it has been refused.

A senior answer for applied pi calculus as discipline produces three artifacts: a leadership sentence, a reproducible evidence note, and a hostile objection. The three artifacts should agree on the same boundary. If they do not, the analytic has confused persuasion with proof.

11.A3 Dolev-Yao as a useful fiction

The attacker controls the network and symbolic cryptography is perfect. This fiction makes message-flow errors visible.

Figure 11.2: Dolev-Yao Attacker Rules

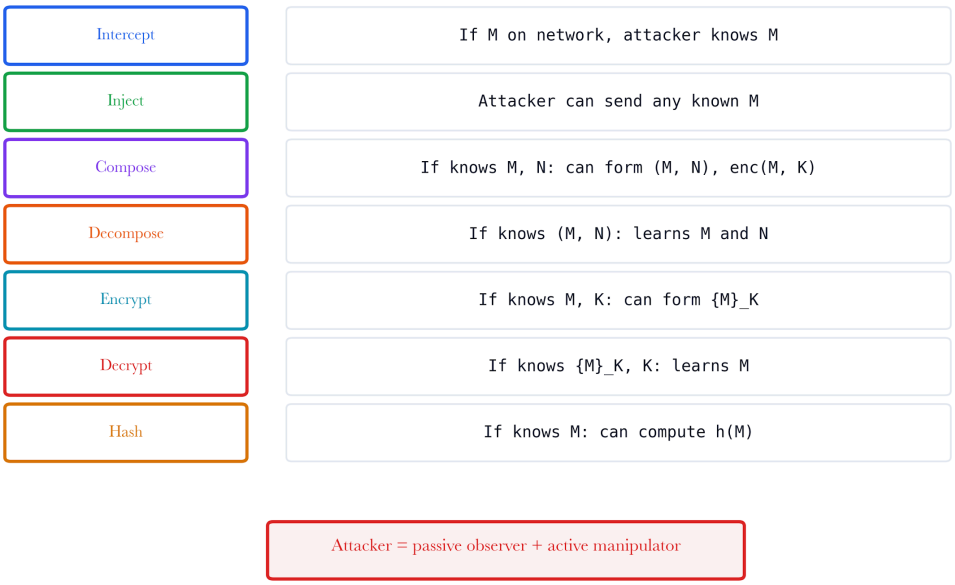

Figure 11.A3: ch11 fig02 dolev yao rules.

Figure note: Dolev-Yao rules belong with useful fiction. The figure states adversary powers before anyone overclaims a proof.

For dolev-yao as a useful fiction, the review starts with the protocol trace and ends only when the claim has a named owner. The analyst should state what the system observed, what transformation changed the record, and what conclusion is still forbidden.

The specific danger in dolev-yao as a useful fiction is the temptation to false closure. Attacker assumptions may support one interpretation, while a counterexample traces toward another. The case should remain open until the proof boundary explains why the stronger claim is admissible or why it has been refused.

A senior answer for Dolev-Yao as a useful fiction produces three artifacts: a leadership sentence, a reproducible evidence note, and a hostile objection. The three artifacts should agree on the same boundary. If they do not, the analyst has confused persuasion with proof.

11.A4 Bounded checking and the cliff of depth

A bounded checker can find a counterexample. It cannot prove absence beyond the bounds.

Figure 11.3: Bounded Model Checker BFS Exploration

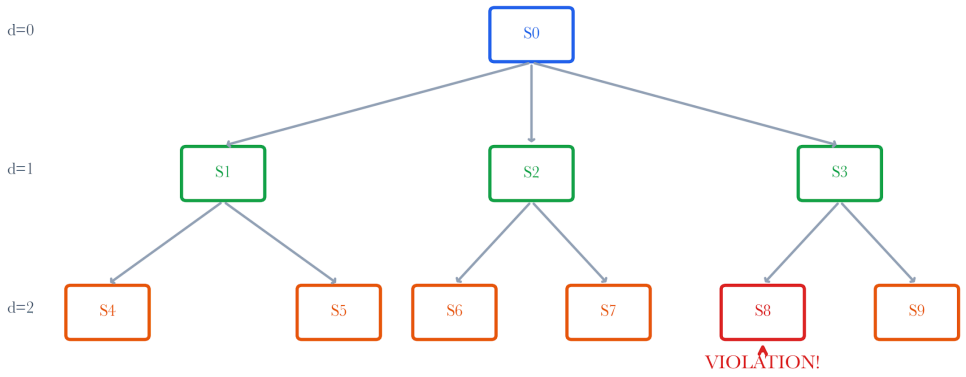

Figure 11.A4: ch11 fig03 bounded model checker bfs.

Figure note: The bounded checker figure belongs with depth. It shows why a green result must travel with its explored boundary.

For bounded checking and the depth cliff, the review starts with the protocol trace and ends only when the claim has a named owner. The analyst should state what the system observed, what transformation changed the record, and what conclusion is still forbidden.

The specific danger in bounded checking and the cliff of depth is false closure. The attacker's assumption may support one interpretation, while the counterexample traces point toward another. The case should remain open until the proof boundary explains why the stronger claim is admissible or why it has been refused.

A senior answer for bounded checking and the depth cliff produces three artifacts: a leadership sentence, a reproducible evidence note, and a hostile objection. The three artifacts should agree on the same boundary. If they do not, the analytic has confused persuasion with proof.

11.A5 Counterexamples as incident scripts

A counterexample trace is a rehearsal. It shows which messages, ordering, and assumptions created the violation.

Figure 11.8: Counterexample Trace for Authentication Violation

Figure 11.A5: ch11 fig08 counterexample trace.

Figure note: A counterexample trace is the best way to teach failure. It lets the reviewer replay the break and ask whether the model matches the device.

For counterexamples as incident scripts, the review starts with the protocol trace and ends only when the claim has a named owner. The analyst should state what the system observed, what transformation changed the record, and what conclusion is still forbidden.

The specific danger in counterexamples as incident scripts is false closure. Attacker assumptions may support one interpretation, while the a counterexample traces toward another. The case should remain open until the proof boundary explains why the stronger claim is admissible or why it has been refused.

A senior answer for counterexamples as incident scripts produces three artifacts: a leadership sentence, a reproducible evidence note, and a hostile objection. The three artifacts should agree on the same boundary. If they do not, the analyst has confused persuasion with proof.

11.A6 Model extraction from messy transcripts

Real transcripts have retries, clock drift, vendor extensions, and missing messages. Extraction is forensic work.

Figure 11.12: TLS 1.3 Handshake Message Flow (Formal)

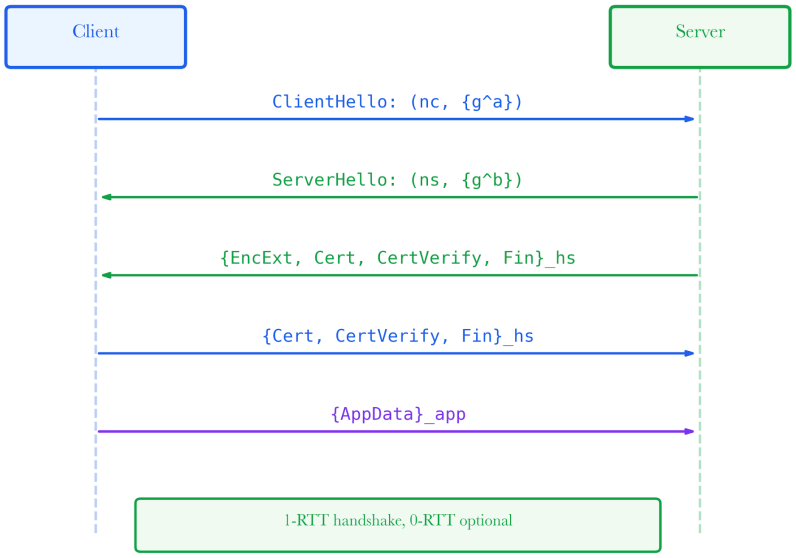

Figure 11.A6: ch11 fig12 handshake message flow.

Figure note: The handshake flow belongs with transcript extraction. It connects packet evidence to the abstract model used by the verifier.

For model extraction from messy transcripts, the review starts with the protocol trace and ends only when the claim has a named owner. The analyst should state what the system observed, what transformation changed the record, and what conclusion is still forbidden.

The specific danger in model extraction from messy transcripts is the risk of false closure. The attacker assumption may support one interpretation, while the counterexample traces point toward another. The case should remain open until the proof boundary explains why the stronger claim is admissible or why it has been refused.

A senior answer for model extraction from messy transcripts produces three artifacts: a leadership sentence, a reproducible evidence note, and a hostile objection. The three artifacts should agree on the same boundary. If they do not, the analytic has confused persuasion with proof.

11.A7 Downgrade attacks as negotiation failures

Downgrade is often a broken negotiation story. The attacker forces both sides to accept lower security.

Figure 11.6: Protocol Downgrade Attack Patterns

Figure 11.A7: ch11 fig06 protocol downgrade attacks.

Figure note: Downgrade attacks are negotiation failures. The figure shows how protocol choice becomes an attack surface.

For downgrade attacks as negotiation failures, the review starts with the protocol trace and ends only when the claim has a named owner. The analyst should state what the system observed, what transformation changed the record, and what conclusion is still forbidden.

The specific danger in downgrade attacks, as negotiation failures is false closure. The attacker's assumption may support one interpretation, while the counterexample points toward another. The case should remain open until the proof boundary explains why the stronger claim is admissible or why it has been refused.

A senior answer for downgrade attacks as negotiation failures produces three artifacts: a leadership sentence, a reproducible evidence note, and a hostile objection. The three artifacts should agree on the same boundary. If they do not, the analyst has confused persuasion with proof.

11.A8 Compositional verification

A device is TLS, firmware update, onboarding, and a cloud API. Local proofs can fail when protocols compose.

Figure 11.15: Compositional Verification Approach

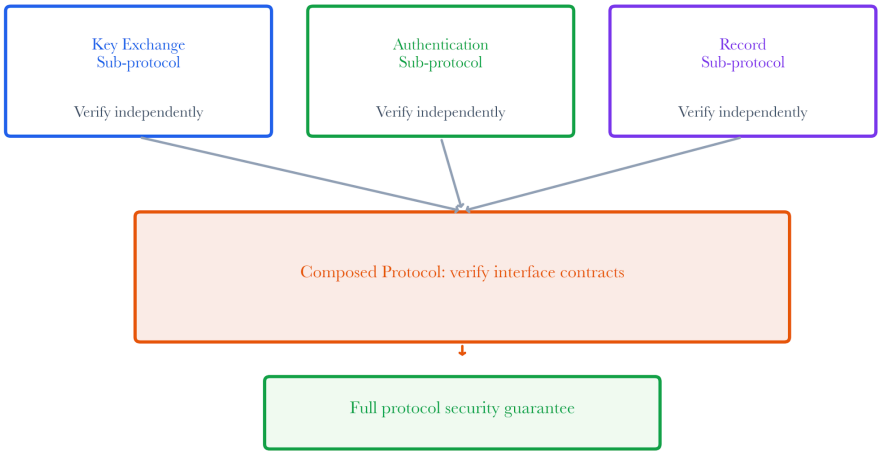

Figure 11.A8: ch11 fig15 compositional verification.

Figure note: Compositional verification belongs with subsystem claims. The figure keeps local proofs from being inflated into whole-system guarantees.

For compositional verification, the review starts with the protocol trace and ends only when the claim has a named owner. The analyst should state what the system observed, what transformation changed the record, and what conclusion is still forbidden.

The specific danger in compositional verification is false closure. The attacker's assumption may support one interpretation, while the counterexample traces toward another. The case should remain open until the proof boundary explains why the stronger claim is admissible or why it has been refused.

A senior answer for compositional verification produces three artifacts: a leadership sentence, a reproducible evidence note, and a hostile objection. The three artifacts should agree on the same boundary. If they do not, the analyst has confused persuasion with proof.

11.A9 NL-to-logic translation with humility

Natural language requirements are ambiguous. Translation to temporal logic needs reviewInor NL-to-logic translatityhumility and , evidence qualitareis not evenly distributed. The airport operations center may provide a precise protocol model and still hide the condition that matters. That is why chapter-level mastery requires evidence boundaries, not vocabulary recall.

The review question is practical: where can an adversary make the system overconfident? The answer usually sits between the attacker's assumption, the counterexample trace, and the implementation delta. That gap is where expert students should work.

The section should end with an action. Quarantine, shadow-test, narrow the claim, collect one more artifact, or reject the conclusion. A vague recommendation is an analytic failure.

For NL-to-logic translation with humility, the review starts with the protocol trace and ends only when the claim has a named owner. The analyst should state what the system observed, what transformation changed the record, and what conclusion is still forbidden.

The specific danger in NL-to-logic translation with humility is the risk of false closure. The attacker's assumption may support one interpretation, while the counterexample trace points toward another. The case should remain open until the proof boundary explains why the stronger claim is admissible or why it has been refused.

A senior answer for NL-to-logic translation with humility produces three artifacts: a leadership sentence, a reproducible evidence note, and a hostile objection. The three artifacts should agree on the same boundary. If they do not, the analyst has confused persuasion with proof.

11.A10 Runtime monitors as proof witnesses

A runtime monitor checks whether the live system stays inside the verified envelope.

Runtime monitors as proof witnesses should change how the analyst briefs risk. In the police evidence facility, leadership does not need a tour of the algorithm. They need to know which protocol model is admissible, which assumption is fragile, and which decision can be taken now.

The system should resist false closure. When the attacker's assumption and implementation delta disagree, the correct state may be unresolved. That is a professional answer, provided the unresolved state has an owner, expiry, and next evidence request.

The highest-grade student will keep model behavior, governance, and mission consequence separate. Merging them into one confidence score destroys the audit trail that makes cyber analytics defensible.

For runtime monitors as proof witnesses, the review starts with the protocol trace and ends only when the claim has a named owner. The analyst should state what the system observed, what transformation changed the record, and what conclusion is still forbidden.

The specific danger in runtime monitors as proof witnesses is the false closure problem. The attacker assumption may support one interpretation, while counterexample traces toward another. The case should remain open until the proof boundary explains why the stronger claim is admissible or why it has been refused.

A senior answer for runtime monitors as proof witnesses produces three artifacts: a leadership sentence, a reproducible evidence note, and a hostile objection. The three artifacts should agree on the same boundary. If they do not, the analyst has confused persuasion with proof.

11.A11 The property matrix

Secrecy, authentication, freshness, replay protection, forward secrecy, and downgrade resistance are different properties.

In the factory cell, the property matrix becomes a claim-control problem inside formal verification. The analyst starts by naming the observed protocol model, the transformation that touched it, and the smallest defensible conclusion. Anything stronger has to wait for evidence.

The failure path is subtle. The dashboard combines the attacker assumption, counterexample trace, and implementation delta into a single status. A senior reviewer should split them apart and preserve the disagreement as a first-class record.

The stop rule matters. The subsection should leave the reader able to say what blocks the claim, what promotes it, and what keeps it in review. That discipline is the difference between analytics and decorative scoring.

For the property matrix, the review starts with the protocol trace and ends only when the claim has a named owner. The analyst should state what the system observed, what transformation changed the record, and what conclusion is still forbidden.

The specific danger in the property matrix is false closure. The attacker assumption may support one interpretation, while the counterexample traces toward another. The case should remain open until the proof boundary explains why the stronger claim is admissible or why it has been refused.

A senior answer for the property matrix produces three artifacts: a leadership sentence, a reproducible evidence note, and a hostile objection. The three artifacts should agree on the same boundary. If they do not, the analytic has confused persuasion with proof.

11.A12 Protocol state machines as memory

A state machine records what the implementation remembers. Many protocol bugs are memory bugs.

Treat protocol state machines as memory, not as an adversarial hearing, and not as a feature description. The municipal water site gives the system partial data, delayed data, and political pressure. The answer must still bind the protocol model to a named decision.

A strong implementation keeps the uncomfortable edge visible. If the attacker assumption says proceed while the implementation delta says wait, the system should not average the conflict into its confidence. It should record the conflict and assign ownership.

The doctoral move is to ask for the counterfactual. What observation would make the original claim false? If the workflow cannot answer, it has built a belief engine rather than a cyber-analytic control.

For protocol state machines as memory, the review starts with the protocol trace and ends only when the claim has a named owner. The analyst should state what the system observed, what transformation changed the record, and what conclusion is still forbidden.

The specific danger in protocol state machines as memory is false closure. Attacker assumptions may support one interpretation, while a counterexample trace points toward another. The case should remain open until the proof boundary explains why the stronger claim is admissible or why it has been refused.

A senior answer for protocol state machines as memory produces three artifacts: a leadership sentence, a reproducible evidence note, and a hostile objection. The three artifacts should agree on the same boundary. If they do not, the analytic has confused persuasion with proof.

11.A13 Formal results for executives

Executives need proof posture. They need what passed, failed, timed out, and fell outside the model.

The operational test for formal results for executives is whether a second expert can replay the reasoning. In a residential safety platform, this means the input record, transformation, exception handling, and decision authority must survive handoff.

The dangerous shortcut is to trust the most convenient signal. The counterexample trace may look stable, while the implementation delta carries the real warning. Attacker assumptions may look mathematically clean, while the mission context changes the cost of error.

A useful report should be modest and sharp. It should say what the analytic saw, what it inferred, what it refused to infer, and which collection step would move the case forward.

For formal results for executives, the review starts with the protocol trace and ends only when the claim has a named owner. The analyst should state what the system observed, what transformation changed the record, and what conclusion is still forbidden.

The specific danger in formal results for executives is the risk of false closure. An attacker's assumption may support one interpretation, while a counterexample trace points toward another. The case should remain open until the proof boundary explains why the stronger claim is admissible or why it has been refused.

A senior answer for formal results for executives produces three artifacts: a leadership sentence, a reproducible evidence note, and a hostile objection. The three artifacts should agree on the same boundary. If they do not, the analyst has confused persuasion with proof.

11.A14 When proof meets firmware

Firmware often implements partial, stale, or vendor-tuned protocol variants.

For when proof meets firmware, evidence quality is not evenly distributed. The regional hospital may provide a precise protocol model and still hide the condition that matters. That is why chapter-level mastery requires evidence-based boundaries, not vocabulary recall.

The review question is practical: where can an adversary make the system overconfident? The answer usually sits between the attacker assumption, the counterexample trace, and the implementation delta. That gap is where expert students should work.

The section should end with an action. Quarantine, shadow-test, narrow the claim, collect one more artifact, or reject the conclusion. A vague recommendation is an analytic failure.

For when proof meets firmware, the review starts with the protocol trace and ends only when the claim has a named owner. The analyst should state what the system observed, what transformation changed the record, and what conclusion is still forbidden.

The specific danger when proof meets firmware is false closure. The attacker's assumption may support one interpretation, while the counterexample points toward another. The case should remain open until the proof boundary explains why the stronger claim is admissible or why it has been refused.

A senior answer to the question of when proof meets firmware produces three artifacts: a leadership sentence, a reproducible evidence note, and a hostile objection. The three artifacts should agree on the same boundary. If they do not, the analyst has confused persuasion with proof.

11.A15 Doctoral Mastery Check

A student has mastered Chapter 11 when they can say exactly what a proof covers and what it leaves outside the model. The answer must tie together property, attacker model, depth, symbolic abstraction, transcript evidence, counterexample trace, and implementation delta into a single reviewable artifact.

Perturb the case with five verification shocks: make one fallback mode undocumented, one nonce reused after reconnect, one certificate exception field-only, one bounded search stop before the exploit depth, and one firmware extension absent from the model. The student should downgrade the verdict with precision.

11.D Seminar War Rooms

This section is intentionally demanding. Each war room asks the reader to work like a senior analyst responsible for verification model decisions in live IoT or OT environments. The task is not to recite the chapter. The task is to make a bounded claim under pressure, then defend the evidence boundary when another expert attacks it.

11.D1 Tls Field Exception

War Room 1 begins with the TLS field exception. The first analyst wants to accept the protocol trace because it fits the expected story. The second analyst refuses to move until the attacker's assumption is tied to an observed artifact, an owner, and a clock. The disagreement is productive. It prevents the team from treating a plausible explanation as verified.

The class should interrogate the case by changing one fact that a real adversary could influence: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The

answer must state which claim survives the change and which claim fails. A doctoral answer does not hide behind caution. It chooses a precise operational state: admit, hold, shadow- test, collect, rollback, or reject.

The final artifact for the TLS field exception is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and proof boundary. It also contains one leadership sentence that is weaker than the analyst wants and stronger than a lawyer would fear. That sentence is the craft. It carries enough truth to guide action without smuggling in facts the system did not prove.

For TLS field exception, change one adversary-controlled fact: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives and which claim fails. A doctoral answer chooses a precise operational state rather than hiding behind generic caution.

The final artifact for the TLS field exception is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and recovery or hold gate. It also contains one leadership sentence that guides action without smuggling in facts that the system did not prove.

11.D2 Mqtt Reconnect

War Room 2 begins with the MQTT reconnect. The first analyst wants to accept the protocol trace because it fits the expected story. The second analyst refuses to move until the attacker's assumption is tied to an observed artifact, an owner, and a clock. The disagreement is productive. It prevents the team from treating a plausible explanation as verified.

The class should challenge the case by changing one fact that a real adversary could influence: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives the change and which claim fails. A doctoral answer does not hide behind caution. It chooses a precise operational state: admit, hold, shadow- test, collect, rollback, or reject.

The final artifact for the MQTT reconnect is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and proof boundary. It also contains one leadership sentence that is weaker than the analyst wants and stronger than a lawyer would fear. That sentence is the craft. It carries enough truth to guide action without smuggling in facts the system did not prove.

For MQTT reconnect, change one adversary-controlled fact: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives and which claim fails. A doctoral answer chooses a precise operational state rather than hiding behind generic caution.

The final artifact for MQTT reconnect is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and recovery or hold gate. It also contains one leadership sentence that guides action without smuggling in facts that the system did not prove.

11.D3 Coap Shallow Search

War Room 3 begins with the CoAP shallow search. The first analyst wants to accept the protocol trace because it fits the expected story. The second analyst refuses to move until the attacker's assumption is tied to an observed artifact, an owner, and a clock. The disagreement is productive. It prevents the team from treating a plausible explanation as verified.

The class should bound the case by changing one fact that a real adversary could influence: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives the change and which claim fails. A doctoral answer does not hide behind caution. It chooses a precise operational state: admit, hold, shadow-test, collect, rollback, or reject.

The final artifact for the CoAP shallow search is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and proof boundary. It also contains one leadership

sentence that is weaker than the analyst wants and stronger than a lawyer would fear. That sentence is the craft. It carries enough truth to guide action without smuggling in facts the system did not prove.

For a CoAP shallow search, change one adversary-controlled fact: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives and which claim fails. A doctoral answer chooses a precise operational state rather than hiding behind generic caution.

The final artifact for the CoAP shallow search is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and recovery or hold gate. It also contains one leadership sentence that guides action without smuggling in facts that the system did not prove.

11.D4 Certificate Bypass

War Room 4 begins with the certificate bypass. The first analyst wants to accept the protocol trace because it fits the expected story. The second analyst refuses to move until the attacker's assumption is tied to an observed artifact, an owner, and a clock. The disagreement is productive. It prevents the team from treating a plausible explanation as verified.

The class should quarantine the case by changing one fact that a real adversary could influence: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives the change and which claim fails. A doctoral answer does not hide behind caution. It chooses a precise operational state: admit, hold, shadow- test, collect, rollback, or reject.

The final artifact for the certificate bypass is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and proof boundary. It also contains one leadership sentence that is weaker than the analyst wants and stronger than a lawyer would fear. That sentence is the craft. It carries enough truth to guide action without smuggling in facts the system did not prove.

For certificate bypass, change one adversary-controlled fact: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives and which claim fails. A doctoral answer chooses a precise operational state rather than hiding behind generic caution.

The final artifact for certificate bypass is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and recovery or hold gate. It also contains one leadership sentence that guides action without smuggling in facts that the system did not prove.

11.D5 Downgrade Appliance

War Room 5 begins with the downgrade appliance. The first analyst wants to accept the protocol trace because it fits the expected story. The second analyst refuses to move until the attacker's assumption is tied to an observed artifact, an owner, and a clock. The disagreement is productive. It prevents the team from treating a plausible explanation as verified.

The class should simulate the case by changing one fact that a real adversary could influence: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives the change and which claim fails. A doctoral answer does not hide behind caution. It chooses a precise operational state: admit, hold, shadow- test, collect, rollback, or reject.

The final artifact for the downgrade appliance is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and proof boundary. It also contains one leadership sentence that is weaker than the analyst wants and stronger than a lawyer would fear. That sentence is the craft. It carries enough truth to guide action without smuggling in facts the system did not prove.

For a downgrade appliance, change one adversary-controlled fact: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim

survives and which claim fails. A doctoral answer chooses a precise operational state rather than hiding behind generic caution.

The final artifact for the downgrade appliance is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and recovery or hold gate. It also contains one leadership sentence that guides action without smuggling in facts that the system did not prove.

11.D6 Badge-Reader Transcript

War Room 6 begins with the badge-reader transcript. The first analyst wants to accept the protocol trace because it fits the expected story. The second analyst refuses to move until the attacker's assumption is tied to an observed artifact, an owner, and a clock. The disagreement is productive. It prevents the team from treating a plausible explanation as verified.

The class should replay the case by changing one fact that a real adversary could influence: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives the change and which claim fails. A doctoral answer does not hide behind caution. It chooses a precise operational state: admit, hold, shadow-test, collect, rollback, or reject.

The final artifact for the badge-reader transcript is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and proof boundary. It also contains one leadership sentence that is weaker than the analyst wants and stronger than a lawyer would fear. That sentence is the craft. It carries enough truth to guide action without smuggling in facts the system did not prove.

For badge-reader transcript, change one adversary-controlled fact: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives and which claim fails. A doctoral answer chooses a precise operational state rather than hiding behind generic caution.

The final artifact for the badge-reader transcript is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and recovery or hold gate. It also contains one leadership sentence that guides action without smuggling in facts that the system did not prove.

11.D7 Vendor Extension

War Room 7 begins with the vendor extension. The first analyst wants to accept the protocol trace because it fits the expected story. The second analyst refuses to move until the attacker's assumption is tied to an observed artifact, an owner, and a clock. The disagreement is productive. It prevents the team from treating a plausible explanation as verified.

The class should audit the case by changing one fact that a real adversary could influence: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives the change and which claim fails. A doctoral answer does not hide behind caution. It chooses a precise operational state: admit, hold, shadow-test, collect, rollback, or reject.

The final artifact for the vendor extension is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and proof boundary. It also contains one leadership sentence that is weaker than the analyst wants and stronger than a lawyer would fear. That sentence is the craft. It carries enough truth to guide action without smuggling in facts the system did not prove.

For vendor extension, change one adversary-controlled fact: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives and which claim fails. A doctoral answer chooses a precise operational state rather than hiding behind generic caution.

The final artifact for vendor extension is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and recovery or hold gate. It also contains one leadership sentence that guides action without smuggling in facts that the system did not prove.

11.D8 Counterexample Ticket

War Room 8 begins with the counterexample ticket. The first analyst wants to accept the protocol trace because it fits the expected story. The second analyst refuses to move until the attacker's assumption is tied to an observed artifact, an owner, and a clock. The disagreement is productive. It prevents the team from treating a plausible explanation as verified.

The class should stage the case by changing one fact that a real adversary could influence: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives the change and which claim fails. A doctoral answer does not hide behind caution. It chooses a precise operational state: admit, hold, shadow-test, collect, rollback, or reject.

The final artifact for the counterexample ticket is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and proof boundary. It also contains one leadership sentence that is weaker than the analyst wants and stronger than a lawyer would fear. That sentence is the craft. It carries enough truth to guide action without smuggling in facts the system did not prove.

For a counterexample ticket, change one adversary-controlled fact: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives and which claim fails. A doctoral answer chooses a precise operational state rather than hiding behind generic caution.

The final artifact for the counterexample ticket is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and recovery or hold gate. It also contains one leadership sentence that guides action without smuggling in facts that the system did not prove.

11.D9 Runtime Witness

War Room 9 begins with the runtime witness. The first analyst wants to accept the protocol trace because it fits the expected story. The second analyst refuses to move until the attacker's assumption is tied to an observed artifact, an owner, and a clock. The disagreement is productive. It prevents the team from treating a plausible explanation as verified.

The class should defer the case by changing one fact that a real adversary could influence: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives the change and which claim fails. A doctoral answer does not hide behind caution. It chooses a precise operational state: admit, hold, shadow-test, collect, rollback, or reject.

The final artifact for the runtime witness is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and proof boundary. It also contains one leadership sentence that is weaker than the analyst wants and stronger than a lawyer would fear. That sentence is the craft. It carries enough truth to guide action without smuggling in facts the system did not prove.

For runtime witness, change one adversary-controlled fact: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives and which claim fails. A doctoral answer chooses a precise operational state rather than hiding behind generic caution.

The final artifact for the runtime witness is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and recovery or hold gate. It also contains one leadership sentence that guides action without smuggling in facts that the system did not prove.

11.D10 Executive Proof Report

War Room 10 begins with the executive proof report. The first analyst wants to accept the protocol trace because it fits the expected story. The second analyst refuses to move until the attacker's assumption is tied to an observed artifact, an owner, and a clock. The disagreement is productive. It prevents the team from treating a plausible explanation as verified.

The class should escalate the case by changing one fact that a real adversary could influence: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives the change and which claim fails. A doctoral answer does not hide behind caution. It chooses a precise operational state: admit, hold, shadow- test, collect, rollback, or reject.

The final artifact for the executive proof report is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and proof boundary. It also contains one leadership sentence that is weaker than the analyst wants and stronger than a lawyer would fear. That sentence is the craft. It carries enough truth to guide action without smuggling in facts the system did not prove.

For the executive proof report, change one adversary-controlled fact: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives and which claim fails. A doctoral answer chooses a precise operational state rather than hiding behind generic caution.

The final artifact for the executive proof report is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and recovery or hold gate. It also contains one leadership sentence that guides action without smuggling in facts that the system did not prove.

11.D11 Nonce Reuse Hearing

War Room 11 begins with the nonce reuse hearing. The first analyst wants to accept the protocol trace because it fits the expected story. The second analyst refuses to move until the attacker's assumption is tied to an observed artifact, an owner, and a clock. The disagreement is productive. It prevents the team from treating a plausible explanation as verified.

The class should rollback the case by changing one fact that a real adversary could influence: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives the change and which claim fails. A doctoral answer does not hide behind caution. It chooses a precise operational state: admit, hold, shadow- test, collect, rollback, or reject.

The final artifact for the nonce reuse hearing is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and proof boundary. It also contains one leadership sentence that is weaker than the analyst wants and stronger than a lawyer would fear. That sentence is the craft. It carries enough truth to guide action without smuggling in facts the system did not prove.

For nonce reuse hearing, change one adversary-controlled fact: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives and which claim fails. A doctoral answer chooses a precise operational state rather than hiding behind generic caution.

The final artifact for the nonce reuse hearing is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and recovery or hold gate. It also contains one leadership sentence that guides action without smuggling in facts that the system did not prove.

11.D12 Implementation-Distance Review

War Room 12 begins with the implementation-distance review. The first analyst wants to accept the protocol trace because it fits the expected story. The second analyst refuses to move until the attacker's assumption is tied to an observed artifact, an owner, and a clock. The disagreement is productive. It prevents the team from treating a plausible explanation as verified.

The class should instrument the case by changing one fact that a real adversary could influence: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives the change and which claim fails. A doctoral answer does not hide behind caution. It chooses a precise operational state: admit, hold, shadow- test, collect, rollback, or reject.

Chapter 11: Formal Verification

The final artifact for the implementation-distance review is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and proof boundary. It also contains one leadership sentence that is weaker than the analyst wants and stronger than a lawyer would fear. That sentence is the craft. It carries enough truth to guide action without smuggling in facts the system did not prove.

For implementation-distance review, change one adversary-controlled fact: timing, identity, provenance, replay, exception handling, sampling, maintenance state, or operator pressure. The answer must state which claim survives and which claim fails. A doctoral answer chooses a precise operational state rather than hiding behind generic caution.

The final artifact for implementation-distance review is a compact evidence packet. It contains the input record, transformation, unresolved conflict, decision authority, and recovery or hold gate. It also contains one leadership sentence that guides action without smuggling in facts that the system did not prove. ## 11. E Adversarial Oral Examination

These oral-exam prompts are designed to make expert students uncomfortable in the right way. Each prompt forces a precise claim about verification review, then changes one operational fact that could occur in a real IoT or OT environment.

11.E1 When Bounded depth stops before reconnect

The examiner gives the student a clean initial narrative, then reveals that bounded depth stops before reconnect. The student must decide whether the original recommendation survives. A passing answer does not retreat into general caution. It names the exact sentence that must be weakened and the exact evidence that would restore it.

The expected response has four parts: current operational state, evidence still admissible, evidence now contaminated or incomplete, and next collection step. The answer should be short enough to brief during an incident and rigorous enough for another senior analyst to reproduce. If the student cannot name the boundary, they have not mastered the chapter.

The expected response for bounded-depth stops before reconnect has four parts: current operational state, evidence still admissible, evidence now contaminated or incomplete, and the next collection step. The answer should be short enough to brief during an incident and rigorous enough for another senior analyst to reproduce. If the student cannot name the boundary for this prompt, the student has not mastered the chapter.

11.E2 When the symbolic model omits parser failure

The examiner gives the student a clean initial narrative, then reveals that the symbolic model omits parser failure. The student must decide whether the original recommendation survives. A passing answer does not retreat into general caution. It names the exact sentence that must be weakened and the exact evidence that would restore it.

The expected response has four parts: current operational state, evidence still admissible, evidence now contaminated or incomplete, and next collection step. The answer should be short enough to brief during an incident and rigorous enough for another senior analyst to reproduce. If the student cannot name the boundary, they have not mastered the chapter.

The expected response for the symbolic model, omitting parser failure, has four parts: current operational state, evidence still admissible, evidence now contaminated or incomplete, and next collection step. The answer should be short enough to brief during an incident and rigorous enough for another senior analyst to reproduce. If the student cannot name the boundary for this prompt, the student has not mastered the chapter.

11.E3 When Certificate exception invalidates the proof

The examiner gives the student a clean initial narrative, then reveals that the certificate exception invalidates the proof. The student must decide whether the original recommendation survives. A passing answer does not retreat into general caution. It names the exact sentence that must be weakened and the exact evidence that would restore it.

The expected response has four parts: current operational state, evidence still admissible, evidence now contaminated or incomplete, and next collection step. The answer should be short enough to brief during an incident and rigorous enough for another senior analyst to reproduce. If the student cannot name the boundary, they have not mastered the chapter.

The expected response for a certificate exception invalidating proof has four parts: current operational state, evidence still admissible, evidence now contaminated or incomplete, and the next collection step. The answer should be short enough to brief during an incident and rigorous enough for another senior analyst to reproduce. If the student cannot name the boundary for this prompt, the student has not mastered the chapter.

11.E4 When the downgrade branch is hidden in the firmware

The examiner gives the student a clean initial narrative, then reveals that the downgrade branch is hidden in the firmware. The student must decide whether the original recommendation survives. A passing answer does not retreat into general caution. It names the exact sentence that must be weakened and the exact evidence that would restore it.

The expected response has four parts: current operational state, evidence still admissible, evidence now contaminated or incomplete, and next collection step. The answer should be short enough to brief during an incident and rigorous enough for another senior analyst to reproduce. If the student cannot name the boundary, they have not mastered the chapter.

The expected response for the downgrade branch hidden in firmware has four parts: current operational state, evidence still admissible, evidence now contaminated or incomplete, and next collection step. The answer should be short enough to brief during an incident and rigorous enough for another senior analyst to reproduce. If the student cannot name the boundary for this prompt, the student has not mastered the chapter.

11.E5 When Trace contradicts the executive summary

The examiner gives the student a clean initial narrative, then reveals that the trace contradicts the executive summary. The student must decide whether the original recommendation survives. A passing answer does not retreat into general caution. It names the exact sentence that must be weakened and the exact evidence that would restore it.

The expected response has four parts: current operational state, evidence still admissible, evidence now contaminated or incomplete, and next collection step. The answer should be short enough to brief during an incident and rigorous enough for another senior analyst to reproduce. If the student cannot name the boundary, they have not mastered the chapter.

The expected response for trace contradicts the executive summary, which has four parts: current operational state, evidence still admissible, evidence now contaminated or incomplete, and next collection step. The answer should be short enough to brief during an incident and rigorous enough for another senior analyst to reproduce. If the student cannot name the boundary for this prompt, the student has not mastered the chapter.

11.E6 When the Runtime monitor sees a state not modeled

The examiner gives the student a clean initial narrative, then reveals that the runtime monitor sees a state not modeled. The student must decide whether the original recommendation survives. A passing answer does not

retreat into general caution. It names the exact sentence that must be weakened and the exact evidence that would restore it.

The expected response has four parts: current operational state, evidence still admissible, evidence now contaminated or incomplete, and next collection step. The answer should be short enough to brief during an incident and rigorous enough for another senior analyst to reproduce. If the student cannot name the boundary, they have not mastered the chapter.

The expected response for the the runtime monitor'state not modeled' has four parts: current operational state, evidence still admissible, evidence now contaminated or incomplete, and next collection step. The answer should be short enough to brief during an incident and rigorous enough for another senior analyst to reproduce. If the student cannot name the boundary for this prompt, the student has not mastered the chapter.

11.E7 When Vendor extension changes nonce lifecycle

The examiner gives the student a clean initial narrative, then reveals that vendor extension changes the nonce lifecycle. The student must decide whether the original recommendation survives. A passing answer does not retreat into general caution. It names the exact sentence that must be weakened and the exact evidence that would restore it.

The expected response has four parts: current operational state, evidence still admissible, evidence now contaminated or incomplete, and next collection step. The answer should be short enough to brief during an incident and rigorous enough for another senior analyst to reproduce. If the student cannot name the boundary, they have not mastered the chapter.

The expected response for vendor extension changes to the nonce lifecycle has four parts: current operational state, evidence still admissible, evidence now contaminated or incomplete, and next collection step. The answer should be short enough to brief during an incident and rigorous enough for another senior analyst to reproduce. If the student cannot name the boundary for this prompt, the student has not mastered the chapter.

11.E8 When the Proof report lacks attacker powers

The examiner gives the student a clean initial narrative, then reveals that the proof report lacks attacker powers. The student must decide whether the original recommendation survives. A passing answer does not retreat into general caution. It names the exact sentence that must be weakened and the exact evidence that would restore it.

The expected response has four parts: current operational state, evidence still admissible, evidence now contaminated or incomplete, and next collection step. The answer should be short enough to brief during an incident and rigorous enough for another senior analyst to reproduce. If the student cannot name the boundary, they have not mastered the chapter.

The expected response for a proof report lacking attacker powers has four parts: current operational state, evidence still admissible, evidence now contaminated or incomplete, and next collection step. The answer should be short enough to brief during an incident and rigorous enough for another senior analyst to reproduce. If the student cannot name the boundary for this prompt, the student has not mastered the chapter.

> **Key Takeaway:** Doctoral Mastery Check. Return a bounded-model-check trace, a ProVerif-style correspondence proof, and a counterexample-response protocol that states which assumption broke and how to strengthen it.

For **when the Proof report lacks attacker powers**, report four outcomes: current operational state, evidence still admissible, evidence now contaminated or incomplete, and the next collection action. Bind each outcome to a single owner and a single failure condition. If the chain cannot produce a bounded state, escalate rather than soften the decision.