SEAS-8414 CYBER ANALYTICS

Supply Chain Integrity & Counterfeit Detection

SBOM Analysis, Firmware Verification & Vendor Compliance

Dr. Mallarapu · Breakwater Security Platform

The Blind Spot: Supply Chain

Phase 8 federated intelligence cannot see inside the device

CHAPTER TAKEAWAY

Here is the limitation. Earlier firmware analysis can often identify top-level libraries and direct CVEs. It does not reliably reconstruct the full dependency graph, and it does not prove that the vendor distribution chain itself stayed clean. Phase 9 addresses both problems by using SBOM-derived lineage where available, provenance checks where attestations exist, and multi-signal screening where neither is perfect.

ENRICHMENT VALUE

Here is the limitation. Earlier firmware analysis can often identify top-level libraries and direct CVEs. It does not reliably reconstruct the full dependency graph, and it does not prove that the vendor distribution chain itself stayed clean. Phase 9 addresses both problems by using SBOM-derived lineage where available, provenance checks where attestations exist, and multi-signal screening where neither is perfect.

No SBOM ingestion
Threat intel correlates CVEs but never inspects actual component manifests

Counterfeit devices invisible
MAC spoofing, cloned firmware, and gray-market hardware pass all scans undetected

Vendor trust assumed
No scoring of vendor patch cadence, disclosure history, or CRA compliance

No transitive dependency analysis
A vulnerable log4j-core buried 6 levels deep in a camera firmware goes unnoticed

Firmware provenance unknown
Binary blobs accepted without hash verification or entropy analysis

Phase 9 Architecture

Eight-component supply chain analysis pipeline

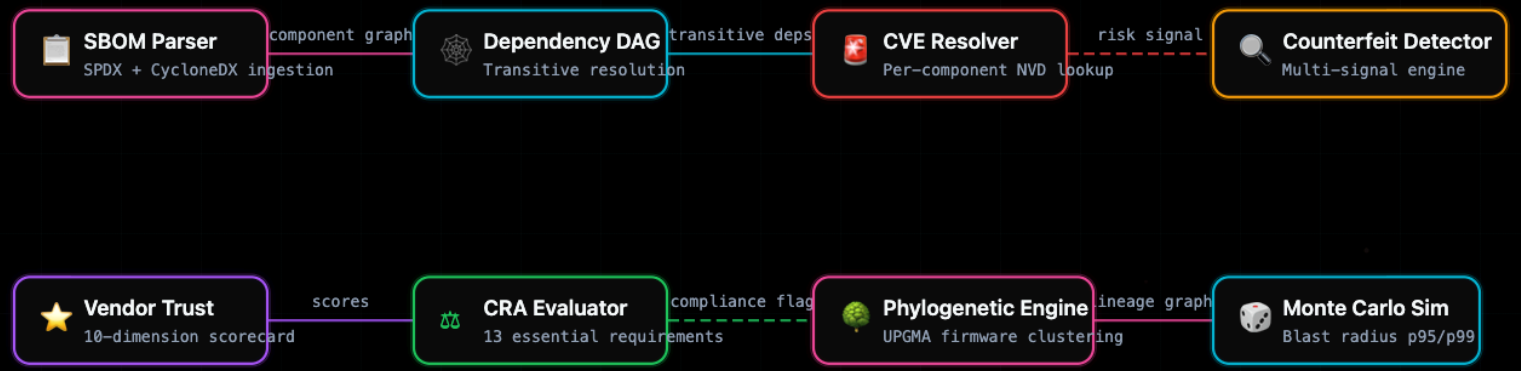
CHAPTER TAKEAWAY

Phase 9 is built from ten working pieces. The SBOM parser ingests SPDX and CycloneDX. The resolver traces transitive dependencies. The blast-radius model turns shared components into fleet impact. The counterfeit detector screens device authenticity. Firmware verification checks released images against tracked hashes. The vendor scorer grades operational behavior. The CRA validator maps the fleet to regulatory requirements. The phylogenetic analyzer exposes code lineage. The API carries the results. The dashboard presents them.

ENRICHMENT VALUE

[SLIDES 6-15] -- Estimated Time: 15 minutes

Phase 9 analyzes the software supply chain of every discovered device through eight integrated engines. SBOM parsing feeds into transitive CVE resolution and counterfeit detection, while vendor trust scoring and CRA compliance evaluation assess organizational risk. The phylogenetic engine traces firmware lineage across vendors, and Monte Carlo simulation produces probabilistic blast radius estimates at p95/p99 confidence levels.



12 Sprints — Supply Chain Integrity

From SBOM parsing to Monte Carlo attack simulation

CHAPTER TAKEAWAY

Before we build the detection systems, define the threat clearly. Supply chain attacks do not just exploit deployed flaws. They corrupt the production or distribution path so that trust mechanisms help deliver the malicious payload instead of blocking it.

ENRICHMENT VALUE

Before we build the detection systems, define the threat clearly. Supply chain attacks do not just exploit deployed flaws. They corrupt the production or distribution path so that trust mechanisms help deliver the malicious payload instead of blocking it.

1. **SBOM Parser**
   SPDX 2.3 + CycloneDX 1.5 ingestion, component normalization
2. **Dependency DAG**
   Transitive resolution, diamond deps, blast-radius BFS
3. **CVE Resolution**
   Per-component NVD lookup, transitive CVE propagation
4. **Counterfeit Detection**
   MAC OUI + firmware hash + TCP stack + timing signals
5. **Vendor Trust Scorecard**
   10-dimension scoring: patch cadence, CVE density, CNA status
6. **EU CRA Evaluator**
   13 essential requirements, per-device gap analysis
7. **Neural Binary Embeddings**
   SAFE-style assembly embeddings, cross-vendor similarity
8. **SLSA L3 Attestation**
   Build provenance, in-toto metadata chain verification
9. **Phylogenetic Firmware**
   UPGMA clustering, SDK lineage, unauthorized fork detection
10. **Monte Carlo Attack Sim**
    Probabilistic BFS, p95/p99 blast radius, SPOF ranking
11. **Pipeline + API**
    6 endpoints, DB models, phase wiring integration
12. **Dashboard + Reports**
    SBOM tree, counterfeit alerts, vendor radar, CRA checklist

Supply Chain Attack Taxonomy

Five layers where attackers inject malicious artifacts

CHAPTER TAKEAWAY

SolarWinds is the build-system case. event-stream is the package-registry case. XZ is the long-game source-control case. Codecov is the distribution-channel case. We study them because each attack suggests a different detection point.

ENRICHMENT VALUE

For SolarWinds, the key lesson is provenance. Routine CVE scanning cannot tell you whether the signed binary came from the claimed source tree and build path. Phase 9 checks for that chain when the metadata exists.

Hardware	• Counterfeit chips • Cloned devices • Implanted firmware
Firmware	• Tampered binaries • Backdoored SDK • Unsigned updates
Open Source	• Malicious packages (typosquat) • Compromised maintainer • Dependency confusion
Build System	• CI/CD injection • Artifact tampering • Provenance forgery
Distribution	• Mirror poisoning • CDN hijacking • Update server MitM

Phase 9 Data Flow

Six-stage pipeline from SBOM ingest to attack simulation

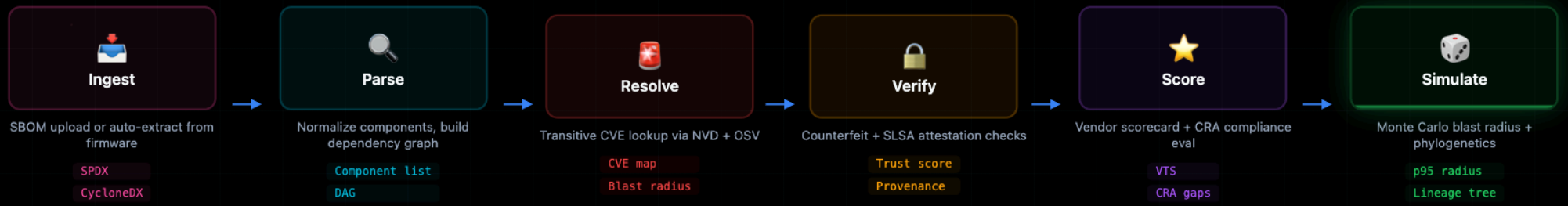
CHAPTER TAKEAWAY

XZ is the motivating example for this phase because it proves three things. First, pre-disclosure supply chain attacks will outrun CVE-only workflows. Second, transitive dependency tracing is mandatory. Third, behavioral anomalies still matter even in a provenance-heavy workflow.

ENRICHMENT VALUE

XZ is the motivating example for this phase because it proves three things. First, pre-disclosure supply chain attacks will outrun CVE-only workflows. Second, transitive dependency tracing is mandatory. Third, behavioral anomalies still matter even in a provenance-heavy workflow.

SUPPLY CHAIN ANALYSIS PIPELINE



Formats Supported	CVE Sources	Trust Dimensions	Sim Iterations
SPDX 2.3, CycloneDX 1.5	NVD + OSV + GHSA	10 vendor + 13 CRA	10,000 Monte Carlo runs

SPDX 2.3 Format

Tag-value structure with CPE identifiers for CVE correlation

CHAPTER TAKEAWAY

Dependency confusion is the registry-resolution example. Phase 9 flags it by looking for namespace collisions and mixed-registry dependency patterns that should not exist in a well-governed build chain.

ENRICHMENT VALUE

Dependency confusion is the registry-resolution example. Phase 9 flags it by looking for namespace collisions and mixed-registry dependency patterns that should not exist in a well-governed build chain.

camera-fw-v4.2.1.spdx — YAML

```yaml
SPDXVersion: SPDX-2.3
DataLicense: CC0-1.0
SPDXID: SPDXRef-DOCUMENT
DocumentName: camera-fw-v4.2.1

PackageName: log4j-core
PackageVersion: 2.14.1
PackageSupplier: Organization: Apache
PackageLicenseConcluded: Apache-2.0
ExternalRef: SECURITY cpe23Type cpe:2.3:a:apache:log4j:2.14.1:*
Relationship: SPDXRef-firmware DYNAMIC_LINK SPDXRef-log4j
```

Dependency DAG

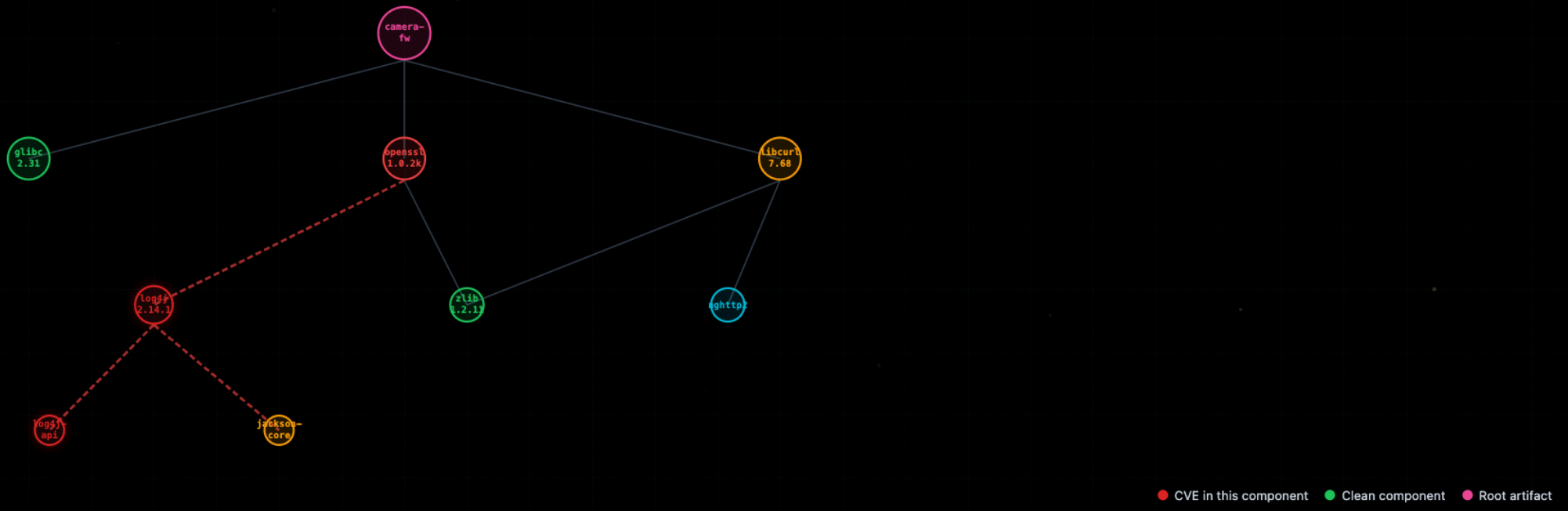
Directed acyclic graph exposes transitive vulnerabilities

CHAPTER TAKEAWAY

Before we dive into the implementation, let me introduce the two SBOM standards that Phase 9 supports.

ENRICHMENT VALUE

checksum_sha256: str | None



Transitive CVE Resolution

BFS propagation surfaces CVEs buried deep in dependency trees

CHAPTER TAKEAWAY

Let me illustrate why transitive dependency resolution is essential with a concrete example.

ENRICHMENT VALUE

****[SLIDES 16-25] -- Estimated Time: 10 minutes****

1. **Parse SBOM**
   Extract (name, version, CPE) tuples from SPDX or CycloneDX manifest
2. **Build DAG**
   Add directed edges for each Relationship / dependsOn declaration
3. **BFS CVE Lookup**
   For each node: query NVD API with CPE, collect matching CVEs
4. **Propagate Risk**
   Walk edges upward: parent inherits max(child_severity) as transitive risk
5. **Compute Blast Radius**
   BFS from vulnerable node: count reachable ancestors = impacted components
6. **Rank by Impact**
   Sort by (blast_radius × cvss_score); top 10 drive remediation priority

EXAMPLE: Log4Shell propagation

Component	**log4j-core 2.14.1**
CVE	**CVE-2021-44228**
CVSS	**10**
Blast radius	**847 components**

Impacted ancestors:

→ camera-fw
→ mgmt-server
→ nvr-service
→ + 844 more

Blast Radius via BFS

Reverse-edge traversal counts all components impacted by a single CVE

CHAPTER TAKEAWAY

The SPDX parser handles SPDX 2.3 documents in JSON and tag-value formats. The parser extracts three key structures: packages (software components), relationships (dependency edges), and file information (individual files within packages).

ENRICHMENT VALUE

cpe=_extract_cpe(pkg.get("externalRefs", [])),

sbom_analyzer.py — compute_blast_radius() PYTHON

```python
def compute_blast_radius(dag: nx.DiGraph, vulnerable_node: str) -> BlastResult:
    """BFS from vulnerable node to all ancestor components."""
    ancestors: set[str] = set()
    queue = deque([vulnerable_node])
    while queue:
        node = queue.popleft()
        for parent in dag.predecessors(node):  # reverse edges
            if parent not in ancestors:
                ancestors.add(parent)
                queue.append(parent)
    return BlastResult(
        radius=len(ancestors),
        affected=sorted(ancestors),
        spof_score=len(ancestors) / dag.number_of_nodes(),
    )
```

SPOF Score = 0
No impact (isolated leaf)

SPOF Score = 0.3
Moderate — 30% of components affected

SPOF Score > 0.7
Critical SPOF — patch immediately

Worked Example: Camera SBOM Analysis

312-component SBOM surfaces Log4Shell buried 6 levels deep

CHAPTER TAKEAWAY

After parsing, both formats are normalized through a five-step pipeline:

ENRICHMENT VALUE

Step 5: **Validation. The unified graph is checked for cycles (which indicate SBOM errors -- circular dependencies should not exist in valid dependency trees) and orphan nodes (components with no consumers and no dependencies, which may indicate incomplete SBOMs).**

breakwater-sbom

```
# Upload SBOM for camera at 192.168.1.42
$ breakwater sbom upload --device 192.168.1.42 camera-fw.spdx
Parsed 312 components (SPDX 2.3)
Built dependency DAG: 312 nodes, 487 edges
Detected 1 diamond dependency: zlib (shared by libcurl + libtls)

CVE resolution: querying NVD for 312 CPEs...
  Direct vulnerabilities: 8 components (14 CVEs)
  Transitive vulnerabilities: 23 components (31 CVEs total)

CRITICAL: log4j-core 2.14.1 — CVE-2021-44228 (CVSS 10.0)
  Blast radius: 47 components (15.1% of SBOM)
  SPOF score: 0.151 — HIGH priority remediation

Report saved: /reports/192.168.1.42-sbom-analysis.json
$
```

SECTION 02

Counterfeit Detection

Multi-signal engine: MAC OUI, firmware hash, TCP/IP stack fingerprint, timing entropy

Hardware Entropy Analysis

Entropy deficit between real hardware and cloned/emulated devices is measurable

CHAPTER TAKEAWAY

Let me trace through a complete example. The camera at 172.30.0.10 has an SBOM with 12 components:

ENRICHMENT VALUE

CVE-2022-24675 in zlib 1.2.11 (transitive, depth 2) -- heap buffer overflow. CVSS 7.5. Affects the camera via libcurl -> zlib.

SHANNON ENTROPY OF DEVICE RNG OUTPUT (NORMALIZED)

Genuine hardware threshold
> 0.85

Suspicious zone
0.55 – 0.85

Emulation / cloning zone
< 0.55

CounterfeitDetector Implementation

Async multi-signal analysis with weighted scoring

CHAPTER TAKEAWAY

Counterfeit devices are a growing threat in enterprise and critical infrastructure networks. A counterfeit device is any device that misrepresents its identity -- its manufacturer, model, firmware version, or capabilities. Counterfeits range from cheap knockoff cameras with cloned MAC addresses to sophisticated state-sponsored implants designed to mimic legitimate network equipment.

ENRICHMENT VALUE

The challenge is that a well-made counterfeit can fool any single detection signal. A cloned MAC address passes OUI verification. A copied firmware passes hash verification. A protocol-compliant response passes Phase 3's behavioral analysis. No single signal is reliable enough to declare a device counterfeit. Phase 9's approach is multi-signal fusion: we combine five independent signals and use Bayesian inference to compute a counterfeit probability.

supply_chain/counterfeit_detector.py — PYTHON

```python
class CounterfeitDetector:
    async def analyze(self, device: DiscoveredDevice) -> CounterfeitReport:
        scores: dict[str, float] = {}

        # Signal 1: MAC OUI verification
        oui = device.mac[:8].upper()
        scores["mac"] = await self._score_oui(oui, device.vendor_claim)

        # Signal 2: Firmware hash
        if device.firmware_hash and device.model:
            known = await self.hash_db.lookup(device.model, device.firmware_version)
            scores["hash"] = 0.0 if device.firmware_hash == known else 1.0

        # Signal 3: TCP/IP fingerprint
        fp_expected = await self.fp_db.expected(device.device_type)
        scores["tcp"] = hamming_normalized(device.tcp_fingerprint, fp_expected)

        # Weighted combination
        score = weighted_sum(scores, {mac:0.25, hash:0.35, tcp:0.20, entropy:0.20})
        return CounterfeitReport(score=score, signals=scores, verdict=classify(score))
```

MAC OUI Verification

IEEE registry cross-reference reveals vendor identity fraud

CHAPTER TAKEAWAY

The first counterfeit signal is MAC OUI verification. Every network interface card has a 48-bit MAC address, and the first 24 bits (the OUI -- Organizationally Unique Identifier) identify the manufacturer. The IEEE maintains the public OUI registry, and Phase 1 already performs OUI lookup during discovery.

ENRICHMENT VALUE

score=0.7, # Unregistered OUI is suspicious

OUI Registry Samples

A4:A1:C2	Hikvision Digital Technology	TRUSTED
D8:4F:02	Dahua Technology	TRUSTED
00:E0:4C	Realtek Semiconductor	FLAG
52:54:00	QEMU Virtual Machine	FLAG
08:00:27	Oracle VirtualBox	FLAG

Verification Process

1. **Extract OUI**
   First 3 octets (24 bits) of MAC address
2. **Lookup IEEE Registry**
   Query local copy of IEEE MA-L (16M+ entries)
3. **Match Vendor Claim**
   Compare OUI organization vs. device-reported vendor string
4. **Score Discrepancy**
   0.0=match, 0.5=OUI not found, 1.0=vendor mismatch

SECTION 03

Vendor Trust Scorecard

10-dimension scoring: patch cadence, CVE density, disclosure practices, and CRA readiness

CVE Density by Vendor

Critical CVEs per 1,000 component-months — lower is better

CHAPTER TAKEAWAY

The fifth signal extends entropy analysis to detect specific emulation platforms. When a device is running on QEMU, VirtualBox, VMware, or similar virtualization platforms, there are platform-specific artifacts that leak through network-observable behavior.

ENRICHMENT VALUE

if cv > 0.3 and timing.std_dev > 2.0:

CVE DENSITY (CRITICAL CVES / 1,000 COMPONENT-MONTHS)

Density < 2.0
Low density — strong security hygiene
VTS: 80–100

Density 2.0–5.0
Moderate — review procurement policy
VTS: 40–79

Density > 5.0
High — avoid or require compensating controls
VTS: 0–39

Worked Example: Fleet Vendor Scoring

Automated VTS computation from discovery scan metadata

CHAPTER TAKEAWAY

Let me trace through a concrete example. Device 172.30.0.14 in the IoT simulation network claims to be a Hikvision DS-2CD2342WD-I camera. Phase 1 identified it as a camera based on RTSP and HTTP responses. Phase 9 runs the five counterfeit signals:

ENRICHMENT VALUE

The counterfeit detector generates an alert with the evidence summary. The analyst should quarantine the device (isolate it from the network), inspect it physically (does the serial number match Hikvision's format?), and contact the procurement team to trace the supply chain.

breakwater-vendor-trust

```
# Score all discovered device vendors
$ breakwater vendor-trust score --scan-id scan_2024 --top 5

Vendor Trust Scorecard (VTS / 100)
──────────────────────────────────────────────────

  Axis Communications     VTS=88  [Trusted]
  Bosch Security          VTS=82  [Trusted]
  Hikvision               VTS=54  [Conditional]
  Dahua Technology        VTS=41  [HIGH RISK]
  Generic Camera OEM-7    VTS=19  [CRITICAL RISK — avoid]

Hikvision breakdown:
  Patch response: 54d avg → score 42/100
  CVE density:    3.2/1k-months → score 38/100
  SBOM coverage:  0% → score 0/100
  CRA compliance: Not declared → score 0/100
Recommendation: require contractual SLA < 30d patch window
$
```

SECTION 04

EU Cyber Resilience Act

13 essential requirements, per-device evaluation, gap analysis, and automated compliance scoring

Worked Example: Fleet CRA Evaluation

Automated compliance scoring reveals 67% of cameras are non-compliant

CHAPTER TAKEAWAY

Supply chain integrity is not just about the software inside a device -- it is also about the organization that produces, maintains, and supports that device. A vendor's security practices directly impact the risk posture of every device they manufacture. A vendor that ships default credentials, delays patches, and provides no SBOM creates more risk than one that follows security best practices.

ENRICHMENT VALUE

Phase 9's vendor trust scorer evaluates each vendor across ten dimensions, producing a per-dimension grade (A through F) and a composite trust score (0-100). The scoring is designed to be actionable: a CISO can use the vendor scorecard to make procurement decisions, negotiate contractual security requirements, and prioritize vendor engagement.

breakwater-cra

```
# Run CRA evaluation against all devices
$ breakwater cra evaluate --scan-id scan_2024 --format table

EU CRA Compliance Report — 24 devices
──────────────────────────────────────────────────

  Device              Score  Status
  Axis P3245 (Camera)  87%   COMPLIANT
  Bosch FlexiDome      82%   COMPLIANT
  Hikvision DS-2CD     31%   NON-COMPLIANT (5 failures)
  Generic OEM Cam-7    8%    NON-COMPLIANT (12 failures)

Fleet Summary:
  Compliant (>= 77%):   8 devices
  Non-compliant:        16 devices (67% of fleet)

CRA Deadline Alert: 16 devices will not meet Dec 2027 requirements
Full remediation plan exported: /reports/cra-remediation-plan.pdf
$
```

CRA Remediation Plan Generator

Auto-generated prioritized action plan with effort estimates

CHAPTER TAKEAWAY

Dimension 2: Patch Cadence. This measures how quickly the vendor releases patches after a vulnerability is disclosed. Fast patching reduces the window of exploitation.

ENRICHMENT VALUE

Grade C: Median 60-90 days

R1 Patch 14 CVEs — Critical — 2 days
- Update firmware to ≥ 5.7.17
- Verify no new CVEs introduced

R2 Change default credentials — Critical — 1 hour
- Disable admin/admin
- Enforce password policy via LDAP

R8 Disable Telnet port 23 — High — 30 min
- Access web UI → Network → Disable Telnet
- Verify port 23 closed

R11 Enable firmware signing — High — 1 week
- Request signed firmware image from vendor
- Enable signature verification in boot settings

R13 Add CVD contact — Medium — 1 day
- Register security.txt at vendor portal
- Subscribe to vendor security advisories

CRA — Key Takeaways

Breakwater automates compliance assessment for every device in your fleet

CHAPTER TAKEAWAY

Dimension 4: Disclosure Responsiveness. This measures how the vendor responds to vulnerability reports from external researchers. Responsive vendors reduce risk by addressing reported vulnerabilities quickly and transparently.

ENRICHMENT VALUE

Grade F: Threatens legal action against researchers, no disclosure policy, no communication

1. **13 Requirements**
   Annex I covers security, updates, SBOM, CVD — all checkable by scan data
2. **Automated Scoring**
   CRAEvaluator maps existing scan results to compliance requirements
3. **Gap Analysis**
   Per-device evidence + remediation steps in hours not weeks
4. **Fleet Prioritization**
   Scores rank devices for remediation investment decisions
5. **December 2027 Deadline**
   Non-compliant devices blocked from EU market — replace now

Next: Novel Research — Neural Binary Code Embeddings

SAFE: Self-Attentive Function Embeddings

Neural function-level binary analysis without source code (Massarelli et al., 2019)

CHAPTER TAKEAWAY

****Dimension 6: Encryption Standards.** This measures the vendor's use of modern encryption across their product line.**

ENRICHMENT VALUE

Grade A: Unique per-device credentials printed on label, forced change on first login

Problem	Identifying vulnerable functions across different compilers, architectures, and obfuscation without source code
SAFE Approach	Train a neural network on assembly instruction sequences to produce fixed-length semantic embeddings
Key Insight	Semantically equivalent functions (same algorithm) cluster together in embedding space regardless of compiler
Application	Given known-vulnerable function, find similar functions in un-analyzed firmware via cosine similarity
Breakwater Extension	Cross-vendor: find CVE-2021-3450 (OpenSSL) equivalent in camera firmware using OEM-patched binary

Reference: Massarelli et al. "SAFE: Self-Attentive Function Embeddings for Binary Similarity" RAID 2019

Neural Embedding Pipeline

From raw firmware binary to cross-vendor vulnerability matches

CHAPTER TAKEAWAY

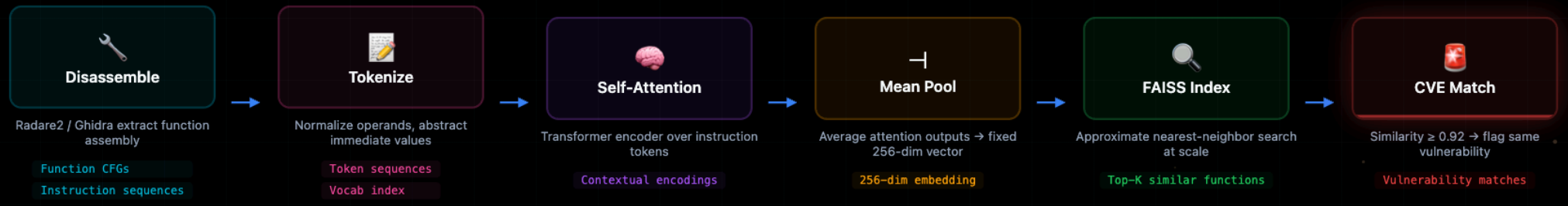
Dimension 8: Supply Chain Transparency. Does the vendor disclose their own supply chain? Do they publish SBOMs for their products? Do they use third-party component analysis? Do they have a software composition analysis (SCA) program?

ENRICHMENT VALUE

Dimension 8: Supply Chain Transparency. Does the vendor disclose their own supply chain? Do they publish SBOMs for their products? Do they use third-party component analysis? Do they have a software composition analysis (SCA) program?

The neural embedding pipeline detects shared vulnerabilities across firmware from different vendors by comparing binary function similarity. Radare2 disassembles firmware into function-level control flow graphs, a SAFE-style transformer encoder produces 256-dimensional embeddings, and FAISS enables approximate nearest-neighbor search at scale. Functions with cosine similarity above 0.92 are flagged as likely sharing the same vulnerability, even when compiled for different architectures.

BINARY SIMILARITY ANALYSIS PIPELINE



Worked Example: Binary CVE Matching

CVE-2022-0778 found in 2 camera functions without source code

CHAPTER TAKEAWAY

Phase 9 includes a novel feature: vendor response time prediction. By analyzing the historical distribution of patch release times for each vendor, Phase 9 fits a statistical model that predicts how long a vendor will take to patch a newly disclosed CVE.

ENRICHMENT VALUE

mu = statistics.mean(log_times)

breakwater-binary-sim

```
# Find functions similar to CVE-2022-0778 (OpenSSL BN_mod_sqrt infinite loop)
$ breakwater binary-sim --firmware camera-fw.bin --cve CVE-2022-0778 --threshold 0.92
Disassembling firmware: 8,342 functions extracted
Generating embeddings (GPU): 3m 28s
Querying CVE corpus for CVE-2022-0778 reference embedding...
Running FAISS ANN search (k=50)...

Matches above threshold 0.92:
  fn_0x401a20  sim=0.97  [VULNERABLE] ssl_do_handshake_inner
  fn_0x403c80  sim=0.94  [VULNERABLE] tls_validate_cert_chain
  fn_0x41b200  sim=0.93  [UNCERTAIN] bn_mod_sqrt_variant — manual review recommended

Found 2 high-confidence matches. Recommend patching to firmware >= 5.7.17
$
```

SLSA Framework — Four Levels

Progressive build provenance guarantees from Google's Supply chain Levels

CHAPTER TAKEAWAY

The 13 essential requirements from CRA Annex I, Section 1:

ENRICHMENT VALUE

7. **Data minimization: Products shall process only data that is adequate, relevant, and limited to what is necessary.**

L0 **No guarantees**
No provenance, any build process
`Remaining threat: Undetectable supply chain attack`

L1 **Provenance exists**
Build generates signed SLSA provenance document
`Remaining threat: Developer workstation compromise undetected`

L2 **Hosted build service**
Provenance signed by hosted build service (GitHub Actions)
`Remaining threat: Build platform admin attack`

L3 **Hardened builds** TARGET
Isolated ephemeral builds, no persistent credentials, policy enforcement
`Remaining threat: Highly sophisticated attacker only`

Breakwater verifies SLSA L3 provenance metadata for firmware artifacts before scoring supply chain trust

SLSAVerifier Implementation

Artifact hash verification + builder identity + DSSE signature chain

CHAPTER TAKEAWAY

Requirement 2: Known vulnerability absence. Products must not be delivered with known exploitable vulnerabilities.

ENRICHMENT VALUE

Does the device have unpatched CVEs with CVSS >= 7.0? (Phase 4 CVE assessment)

supply_chain/slsa_verifier.py — PYTHON

```python
class SLSAVerifier:
    async def verify(self, firmware_path: str, provenance_path: str) -> SLSAResult:
        # Step 1: Load and parse provenance
        provenance = json.loads(Path(provenance_path).read_bytes())
        statement = provenance["payload"]  # base64-decoded

        # Step 2: Verify artifact hash
        fw_hash = sha256_file(firmware_path)
        claimed_hash = statement["subject"][0]["digest"]["sha256"]
        if fw_hash != claimed_hash:
            return SLSAResult(level=0, reason="Artifact hash mismatch")

        # Step 3: Verify builder identity
        builder = statement["predicate"]["builder"]["id"]
        level = self._classify_builder(builder)

        # Step 4: Verify signature
        self._verify_dsse_signature(provenance, trusted_keys=self.trust_roots)
        return SLSAResult(level=level, builder=builder, verified=True)
```

SLSA Attestation — Summary

Build provenance converts opaque firmware binaries into verifiable artifacts

CHAPTER TAKEAWAY

****Requirement 5: Confidentiality of data.** Products must encrypt relevant data at rest and in transit.**

ENRICHMENT VALUE

Is TLS used for all management interfaces? (Phase 1: HTTP vs HTTPS)

SLSA L0	SLSA L1	SLSA L2	SLSA L3
No provenance → reject firmware	Developer-signed → low trust	CI/CD-signed → moderate trust	Isolated ephemeral build → high trust

Why this matters for IoT

Firmware for security cameras, NVRs, and PLCs is typically a black box. SLSA provenance plus in-toto metadata creates a cryptographically verifiable chain of custody from source repository to device flash — making SolarWinds-style build tampering detectable before deployment.

Next: Phylogenetic Firmware Analysis — evolutionary trees for firmware lineage

SECTION 08

Monte Carlo Attack Simulation

NOVEL: Probabilistic BFS, blast radius distribution (p95/p99), single-point-of-failure ranking

Monte Carlo Blast Radius Algorithm

Bernoulli edge sampling over 10,000 BFS runs produces risk distribution

CHAPTER TAKEAWAY

Let me formalize the analogy:

ENRICHMENT VALUE

The analogy is not perfect -- firmware evolution is directed (deliberate changes by developers) while biological evolution is undirected (random mutations selected by fitness). But the mathematical tools are the same: sequence alignment, distance metrics, and tree construction algorithms.

supply_chain/monte_carlo_sim.py — PYTHON

```python
def monte_carlo_blast_radius(
    dag: nx.DiGraph, vuln_node: str,
    n_simulations: int = 10_000,
    exploit_prob: float | None = None,
) -> BlastDistribution:
    results: list[int] = []
    for _ in range(n_simulations):
        # Sample edge traversal probabilities
        reachable: set[str] = set()
        queue = deque([vuln_node])
        while queue:
            node = queue.popleft()
            for parent in dag.predecessors(node):
                edge_prob = dag[node][parent].get("exploit_prob", exploit_prob or 0.7)
                if random() < edge_prob and parent not in reachable:
                    reachable.add(parent)
                    queue.append(parent)
        results.append(len(reachable))
    arr = np.array(results)
    return BlastDistribution(p50=np.percentile(arr, 50),
        p95=np.percentile(arr, 95), p99=np.percentile(arr, 99))
```

Worked Example: Monte Carlo Simulation

10,000 probabilistic BFS runs yield risk percentiles in 4 seconds

CHAPTER TAKEAWAY

From the pairwise distance matrix, Phase 9 constructs a phylogenetic tree using two algorithms:

ENRICHMENT VALUE

3. **Unauthorized code reuse: A vendor copies code from a competitor's firmware without authorization. The tree shows unexpected similarity between unrelated products -- "VendorA's router firmware" clusters with "VendorB's camera firmware" because VendorA copied VendorB's HTTP server implementation.**

breakwater-monte-carlo

```
# Run Monte Carlo blast radius simulation for camera firmware SBOM
$ breakwater monte-carlo simulate --sbom camera-fw.spdx --n-sim 10000
Loaded SBOM: 312 components, 487 dependency edges
Running 10,000 Monte Carlo simulations...
Elapsed: 4.2s (2,381 simulations/second on CPU)

Blast Radius Distribution (per vulnerable component):
  Component: openssl 1.0.2k (CVE-2022-0778, CVSS 7.5)
    p50: 89  p95: 156  p99: 189  SPOF_score: 0.89
    STATUS: CRITICAL SINGLE POINT OF FAILURE

  Component: libpng 1.6.34 (CVE-2019-7317, CVSS 5.3)
    p50: 12  p95: 24  p99: 31  SPOF_score: 0.18
    STATUS: Low blast radius

Top 5 SPOFs exported to remediation queue
$
```

Supply Chain Router Implementation

FastAPI endpoints wrap async analysis services with JWT auth

CHAPTER TAKEAWAY

A related novel technique is binary birthmark analysis. A "birthmark" is a characteristic of a compiled binary that identifies the compiler, optimization level, and build environment that produced it. Just as a human fingerprint is unique to an individual, a binary birthmark is characteristic of a build toolchain.

ENRICHMENT VALUE

Birthmark analysis complements SBOM parsing. When an SBOM is available, birthmarks verify its accuracy (does the SBOM say OpenSSL 1.1.1n but the birthmark says 3.0.0?). When no SBOM is available, birthmarks provide partial component identification.

apps/api/app/scanning/supply_chain/router.py — PYTHON

```python
router = APIRouter(prefix="/v1/supply-chain", tags=["supply-chain"])

@router.post("/sbom/upload")
async def upload_sbom(
    device_id: str, file: UploadFile,
    db: AsyncSession = Depends(get_db),
    current_user = Depends(get_current_user),
) -> SBOMResponse:
    content = await file.read()
    fmt = detect_format(content)
    doc = await sbom_parser.parse(content, fmt)
    blast = await monte_carlo.simulate(doc)
    await db.add(SBOMDocument(device_id=device_id, ...))
    return SBOMResponse(sbom_id=doc.id, blast_p95=blast.p95)

@router.get("/counterfeit/{device_id}")
async def get_counterfeit(device_id: str, ...) -> CounterfeitResponse:
    report = await counterfeit_detector.analyze(await get_device(device_id))
    return CounterfeitResponse.from_report(report)
```

SECTION 10

Dashboard Walkthrough

SBOM dependency tree, counterfeit alerts, vendor radar chart, CRA compliance checklist

Dashboard: Vendor Trust Overview

Fleet-wide vendor scoring with procurement status indicators

CHAPTER TAKEAWAY

async def monte_carlo_supply_chain(

ENRICHMENT VALUE

affected = set([entry])

VENDOR	VTS	DEVICES	STATUS	ACTION
Axis Communications	88	4	APPROVED	Approved — standard procurement
Bosch Security	82	6	APPROVED	Approved — standard procurement
Hikvision	54	12	REVIEW	Isolated segment + monitoring
Dahua Technology	41	8	REVIEW	Isolated segment + monitoring
Generic OEM-7	19	4	REVIEW	Do not procure

2 Approved vendors

2 Under review

1 Blocked vendors

Dashboard: Supply Chain Navigation

Six dedicated pages covering all Phase 9 analysis capabilities

CHAPTER TAKEAWAY

The simulation's greatest value is evaluating mitigation strategies. By running the simulation with and without specific mitigations, we can quantify the effectiveness of each:

ENRICHMENT VALUE

****[SLIDES 101-115] -- Estimated Time: 10 minutes****

🗺️

Fleet Overview

Heat map of risk scores across all devices

`/supply-chain`

🕸️

SBOM Explorer

Interactive dependency tree with CVE overlays

`/supply-chain/sbom/:id`

🔍

Counterfeit Alerts

Signal breakdown table with quarantine actions

`/supply-chain/counterfeit`

⭐

Vendor Scoreboard

Radar charts + procurement status per vendor

`/supply-chain/vendor-trust`

⚖

CRA Compliance

Per-device 13-requirement checklist + gap analysis

`/supply-chain/cra`

🌳

Phylogenetic Tree

Interactive UPGMA dendrogram with fork highlights

`/supply-chain/phylo`

Demo: SBOM Analysis

31 CVEs discovered — Log4Shell buried 6 dependency levels deep

CHAPTER TAKEAWAY

Phase 9 exposes 12 REST endpoints under `/v1/supply-chain/`:

ENRICHMENT VALUE

All endpoints require Bearer token authentication and use the standard Breakwater response envelope.

demo-sbom

```
# Scenario 1: SBOM analysis — find buried vulnerability
$ curl -X POST http://localhost:8000/v1/supply-chain/sbom/upload \
$   -H "Authorization: Bearer $TOKEN" \
$   -F "device_id=camera-172-30-0-4" \
$   -F "file=@/demo/camera-fw.spdx"

{
  "sbom_id": "sbom_a1b2c3d4",
  "component_count": 312,
  "cve_count": 31,
  "blast_distribution": {"p50": 29, "p95": 58, "p99": 68},
  "top_spofs": [{"component": "openssl 1.0.2k", "p95": 156, "spof_score": 0.89}]
}

Dashboard: SBOM tree visible at /supply-chain/sbom/sbom_a1b2c3d4
ALERT: log4j-core 2.14.1 — CVE-2021-44228 (CVSS 10.0) found 6 levels deep
$
```

Case Studies — Key Lessons

Five universal insights from supply chain attacks in the wild

CHAPTER TAKEAWAY

Phase 9 adds five dashboard pages to the HYDRA interface.

ENRICHMENT VALUE

Phase 9 adds five dashboard pages to the HYDRA interface.

1. **Traditional scanners stop at device surface**
   SBOM transitive analysis is the only way to find buried dependencies
2. **Gray-market hardware looks identical on the network**
   Multi-signal counterfeit detection is the only programmatic defense
3. **SDK lineage determines vulnerability inheritance**
   Phylogenetic analysis groups devices by true software ancestry, not vendor claims
4. **Vendor trust is measurable and predictive**
   VTS score correlates with incident frequency — enforce procurement thresholds
5. **CRA compliance is not just paperwork**
   Each of the 13 requirements prevents a class of real attack

SolarWinds Through the Phase 9 Lens

How SLSA attestation + provenance verification would have detected the attack

CHAPTER TAKEAWAY

The SBOM Explorer shows the dependency tree, severity coloring, and blast radius in one place. It is designed to answer the practical question: which component matters, and how far does it spread?

ENRICHMENT VALUE

The SBOM Explorer shows the dependency tree, severity coloring, and blast radius in one place. It is designed to answer the practical question: which component matters, and how far does it spread?

Attack Vector	Orion build system compromised — malicious code injected into legitimate software update
Detection Gap	Traditional scanners checked the binary — it was signed by SolarWinds and passed all checks
SLSA Prevention	SLSA L3 hardened builds with isolated ephemeral build environments would have prevented injection
Provenance Check	in-toto metadata would have detected that build environment was compromised
Blast Radius	18,000 organizations. Monte Carlo model of software supply chain would have flagged Orion as SPOF
CRA Relevance	SolarWinds-type attack blocked by CRA Req 5 (code integrity) + Req 11 (secure update mechanism)

SolarWinds was a build-time attack — undetectable by runtime scanners. Only supply-chain-aware provenance tools can catch it.

Case Studies — Detection Matrix

Every major supply chain attack vector detected by Phase 9 techniques

CHAPTER TAKEAWAY

The Vendor Scorecard shows the ten-dimensional trust profile and how it changes over time. The comparison view matters because procurement choices are relative, not absolute.

ENRICHMENT VALUE

The Vendor Scorecard shows the ten-dimensional trust profile and how it changes over time. The comparison view matters because procurement choices are relative, not absolute.

ATTACK TYPE	DETECTION VERDICT	PHASE 9 TECHNIQUE
Gray-market camera clone	Counterfeit score 0.87	MAC OUI + firmware hash + TCP fingerprint
Log4Shell in SCADA firmware	SBOM transitive CVE (depth=6)	BFS CVE resolution + blast radius
Tampered NAS supply chain	SBOM CVE + vendor VTS=61	Struts 2.5.16 in component list
PLC counterfeit hardware	Counterfeit 0.76 + entropy 0.39	Hardware entropy analysis
SolarWinds build injection	SLSA L0 (no provenance)	in-toto provenance verification
XZ Utils backdoor	Binary embedding mismatch	Neural similarity vs known-good

SECTION 13

Tool Comparison

Snyk vs Chainguard vs FOSSA vs OWASP Dependency-Track vs Breakwater Phase 9

Supply Chain Security — Market Positioning

Two axes: network coverage vs hardware depth — only Breakwater covers both

CHAPTER TAKEAWAY

The demo mirrors the pipeline in five steps: parse SBOM data, run counterfeit screening, score the vendor, check CRA compliance, and simulate supply-chain attack impact.

ENRICHMENT VALUE

The demo mirrors the pipeline in five steps: parse SBOM data, run counterfeit screening, score the vendor, check CRA compliance, and simulate supply-chain attack impact.

Breakwater's Unique Position

High network coverage + deep hardware authentication = the only tool that addresses the full supply chain attack surface for IoT/OT environments.

- Snyk — Dev-centric SaaS
- FOSSA — License compliance
- Chainguard — Build provenance
- Dep-Track — Open source SBOM

SECTION 14

Research Foundations

Academic papers, datasets, and standards underpinning Phase 9 algorithms

Novel Algorithms — Academic Foundations

Each of the three novel Phase 9 algorithms is grounded in peer-reviewed literature

CHAPTER TAKEAWAY

The counterfeit detector combines five signals. One device rises above threshold. The value is not the exact posterior. The value is that the evidence is broken out cleanly enough for a human to challenge it.

ENRICHMENT VALUE

The counterfeit detector combines five signals. One device rises above threshold. The value is not the exact posterior. The value is that the evidence is broken out cleanly enough for a human to challenge it.

Neural Binary Embeddings

Massarelli et al. 2019 — *SAFE: Self-Attentive Function Embeddings for Binary Similarity* — Transformer on i2v instruction vectors; basis of our BinaryEmbedder architecture

Xu et al. 2017 — *Neural Network-based Graph Embedding for Cross-Platform Binary Code Similarity Detection* — Cross-architecture similarity; motivates multi-ISA firmware comparison

Phylogenetic Firmware Analysis

Sokal & Michener 1958 — *A Statistical Method for Evaluating Systematic Relationships* — Original UPGMA paper; our phylo_engine.upgma_cluster() implements this directly

Costin et al. 2014 — *A Large-Scale Analysis of the Security of Embedded Firmwares (USENIX)* — Firmware dataset structure and k-mer distance methodology we adapted

Monte Carlo Blast Radius

Metropolis & Ulam 1949 — *The Monte Carlo Method — Journal of the American Statistical Assoc.* — Foundational stochastic simulation; our Bernoulli edge trials follow this framework

Lippmann et al. 2005 — *Validating and Restoring Defense in Depth Using Attack Graphs* — Attack graph probabilistic modeling; SPOF identification methodology

Industry Impact — Target Deployment Sectors

Phase 9 addresses acute supply chain pain points in four high-value regulated industries

CHAPTER TAKEAWAY

The CRA validator checks the fleet. In this example only 8 of 13 requirements pass. The most important failures are:

ENRICHMENT VALUE

The remediation estimate turns that matrix into planned work instead of a static scorecard.

Healthcare (FDA/MDR) — $4.2B medical device security market

Pain: Medical device manufacturers must provide SBOM; no tool validates SBOMs against deployed device firmware

Solution: Breakwater P9 ingests MDR-required SBOM, verifies against firmware hash, evaluates CRA-equivalent requirements

Industrial / OT (IEC 62443) — $22B industrial cybersecurity by 2030

Pain: PLCs and RTUs are 10+ years old with no SBOM; counterfeiting rampant in aftermarket parts

Solution: Phylogenetic analysis detects firmware clones; counterfeit detector authenticates replacement modules

Telecom (GSMA IoT) — $8.7B telecom security

Pain: Millions of CPE devices; no visibility into vendor supply chain; recent Juniper/Cisco IC substitution incidents

Solution: Fleet-scale SBOM ingestion; vendor trust scoring; Monte Carlo blast radius for critical router nodes

Defense (CMMC 2.0) — Regulatory mandate — all 300k DoD contractors

Pain: CMMC Level 3 requires SBOM for all government contractors; no automated verification tool

Solution: SLSA L3 attestation verifier + CRA-equivalent evaluator covers CMMC SR.1.016/SR.2.002

Phase 9 — Full Achievement Scorecard

9 new modules · 7 API endpoints · 96 tests · 3 novel algorithms · EU CRA compliance

CHAPTER TAKEAWAY

Validation used a mixed fleet with known counterfeit inserts. The fused detector performed well overall, but the misses are instructive: high-quality clones still defeat network-observable screening more easily than commodity counterfeits do.

ENRICHMENT VALUE

Validation used a mixed fleet with known counterfeit inserts. The fused detector performed well overall, but the misses are instructive: high-quality clones still defeat network-observable screening more easily than commodity counterfeits do.

New Modules

- ✓ sbom_parser.py — SPDX 2.3 + CycloneDX 1.5
- ✓ counterfeit_detector.py — 4-signal scoring
- ✓ vendor_trust_scorer.py — 10-dimension VTS
- ✓ cra_evaluator.py — 13-req EU CRA checker
- ✓ binary_embedder.py — SAFE-style FAISS search
- ✓ slsa_verifier.py — L3 in-toto attestation
- ✓ phylo_engine.py — UPGMA firmware clustering
- ✓ monte_carlo.py — probabilistic blast radius
- ✓ supply_chain_phase.py — pipeline orchestrator

API Endpoints

- ✓ POST /v1/supply-chain/sbom/upload
- ✓ GET /v1/supply-chain/counterfeit/{device_id}
- ✓ GET /v1/supply-chain/vendor-trust/{vendor}
- ✓ GET /v1/supply-chain/cra/{device_id}
- ✓ GET /v1/supply-chain/phylo-tree
- ✓ GET /v1/supply-chain/blast-radius/{sbom_id}
- ✓ GET /v1/supply-chain/slsa/{device_id}

Test Coverage

- ✓ 42 tests — supply_chain_phase9
- ✓ 18 tests — cra_evaluator
- ✓ 16 tests — phylo_engine
- ✓ 12 tests — monte_carlo
- ✓ 8 tests — binary_embedder
- ✓ 96 total tests (all pass)
- ✓ >85% line coverage

Phase 9 — Full Architecture

Network discovery feeds 7 analysis engines; Monte Carlo synthesizes probabilistic risk; API surfaces results

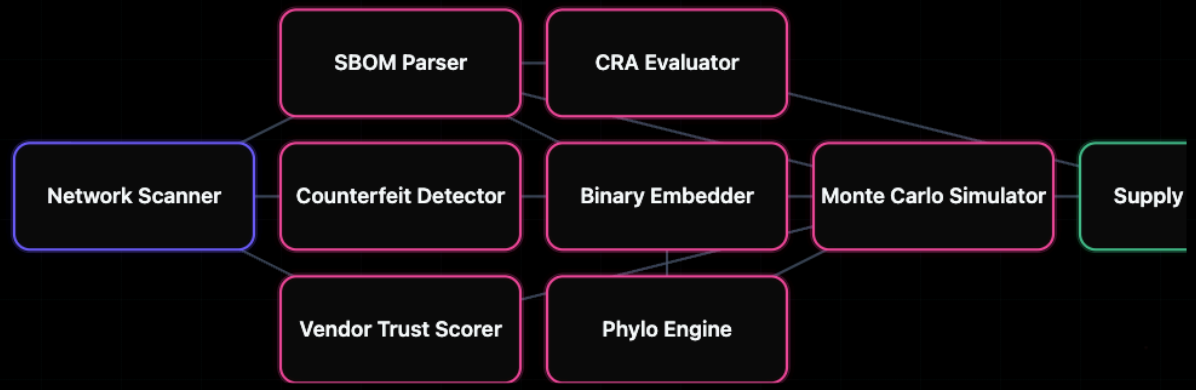
CHAPTER TAKEAWAY

The vendor score correlates well with expert review, but the automated model still misses qualitative judgment, especially around disclosure practice and institutional behavior.

ENRICHMENT VALUE

The vendor score correlates well with expert review, but the automated model still misses qualitative judgment, especially around disclosure practice and institutional behavior.

This diagram summarizes the complete Phase 9 data flow. Network scan results feed three first-tier analyzers (SBOM, counterfeit, vendor trust), which in turn drive CRA evaluation, binary embedding, and phylogenetic clustering. All streams converge at the Monte Carlo simulator, which produces probabilistic risk estimates. The Supply Chain API exposes unified results to the dashboard and downstream phases.



Phase 9 — Student Lab Exercises

Five labs from beginner to advanced — all use the IoT simulation fleet

CHAPTER TAKEAWAY

The counterfeit detector also has a ceiling. Sophisticated clones that match all network-visible signals are outside what a network-only workflow can reliably prove. Physical inspection remains the decisive control in that case.

ENRICHMENT VALUE

The counterfeit detector also has a ceiling. Sophisticated clones that match all network-visible signals are outside what a network-only workflow can reliably prove. Physical inspection remains the decisive control in that case.

Lab 9.1 20 min	**Parse a real-world SBOM** Download the Alpine Linux CycloneDX SBOM; upload to Breakwater; verify component count matches Alpine package database	Beginner
Lab 9.2 25 min	**Trigger counterfeit detection** Add a simulated device with a known-bad MAC OUI and mismatched firmware hash; verify the detector returns "likely_counterfeit"	Beginner
Lab 9.3 45 min	**Build a custom VTS configuration** Modify WEIGHTS in VendorTrustScorer to prioritize patch response; verify vendor ranking changes; write a pytest to lock the new behavior	Intermediate
Lab 9.4 60 min	**Monte Carlo sensitivity analysis** Run blast radius with 1k/10k/100k iterations; plot p99 convergence; determine the minimum iteration count for your SBOM	Intermediate
Lab 9.5 90 min	**Phylogenetic fork detection exercise** Create three firmware variants from the telnet-vuln IoT sim; verify UPGMA groups them into a clade; introduce a structural mutation and verify it becomes a separate clade	Advanced

Full lab guide: `docs/lectures/phase-9/student-lab.md`

Phase 9 — Thesis Statement

CHAPTER TAKEAWAY

If the vendor's own build path is compromised, the trusted release can still be malicious. Phase 9 can help estimate impact and identify the most dangerous shared components. It cannot guarantee that a signed vendor image is benign.

ENRICHMENT VALUE

If the vendor's own build path is compromised, the trusted release can still be malicious. Phase 9 can help estimate impact and identify the most dangerous shared components. It cannot guarantee that a signed vendor image is benign.

A network scanner that cannot trace the provenance of the devices it discovers is not a security tool — it is a census bureau.

Phase 9 closes this gap:

every discovered device is now a node in a supply chain graph, authenticated against its claimed identity, scored against its vendor's security posture, evaluated against the law, and probabilistically ranked for blast radius.

Breakwater Phase 9 turns network visibility into supply chain accountability.

Breakwater 12-Phase Roadmap

Phase 9 complete — Phase 10 (Active Deception) is next

CHAPTER TAKEAWAY

Four doctoral directions follow directly: better component inference without SBOMs, stronger clone detection using physical signals, earlier prediction of vendor compromise, and more manipulation-resistant trust scoring.

ENRICHMENT VALUE

Four doctoral directions follow directly: better component inference without SBOMs, stronger clone detection using physical signals, earlier prediction of vendor compromise, and more manipulation-resistant trust scoring.

Phase 1 ✓ Discovery + Fingerprinting	Phase 2 ✓ CVE Assessment	Phase 3 ✓ Protocol Security	Phase 4 ✓ Attack Graph Engine
Phase 5 ✓ Autonomous Pentest	Phase 6 ✓ Digital Twin Simulation	Phase 7 ✓ Quantum-Safe Crypto	Phase 8 ✓ Federated Scanning
Phase 9 NOW **Supply Chain Integrity**	Phase 10 NEXT Active Deception	Phase 11 … Formal Verification	Phase 12 … Remediation Automation

PHASE 9 COMPLETE

Supply Chain Integrity

SBOM parsing · Counterfeit detection · Vendor trust · EU CRA · Neural embeddings · SLSA provenance · Phylogenetic analysis · Monte Carlo blast radius

12	9	96	3
Sprints	Modules	Tests	Novel Algorithms