

Immutable Execution Provenance for high-risk mobile communications: model-checked containment, rollback-resistant update and scenario-based security evaluation

Jonathan K. Bily

Independent Researcher, Cella, 44370 Teruel, Spain

Corresponding author: Jonathan K. Bily, Cella, 44370 Teruel, Spain.

Tel.: +34 674 98 52 17. E-mail: jonathan@cloudwashlaundry.com.

ORCID: 0009-0004-3526-7271

Article type: Research paper

Word count (main text, excluding title page, abstract, references, tables and figure legends): 5,540

Running title: Immutable Execution Provenance

Key words: mobile security; mercenary spyware; hardware security; execution provenance; secure software update; formal verification; TLA+; supply-chain security; security risk modelling

Abstract

Mercenary spyware compromises the phones of journalists, activists and officials through zero-click exploit chains, while traffic analysis reveals whom they communicate with. This article presents Immutable Execution Provenance (IEP), an endpoint architecture for high-risk mobile communication that roots code integrity in physical media rather than in privileged software. All executable content is fetched from authenticated write-once storage. Writable memory can never become executable, whatever the state of the operating system. An independent security controller outside the application processor's reach gates every egress path, captures evidence, zeroises state and governs replacement of the immutable image. The article makes three contributions. First, it defines the IEP trust boundary and its residual attack surface: code reuse, data-oriented computation and authorised interpreters. Second, it specifies the containment controller and a dual-authorised, rollback-resistant image-replacement protocol in TLA+ and model-checks both exhaustively within stated bounds. The IEP designs satisfy all nine controller properties and all six update properties, including when either signing authority is compromised. Model checking also yields six design requirements, each backed by a counterexample against a plausible alternative: a synchronous hardware hold line; gating of every egress path; no processor-writable resume command; fail-closed exhaustion of the evidence store; burning the anti-rollback fuse before a new image executes; and applying that rule to cold boot as well as to updates. Third, a reproducible Monte Carlo scenario model with unique-chain exploit-burn accounting estimates remote surveillance duty of 0.416% of a three-year horizon for a fortified write-once handset, 0.186% with IEP and 0.057% with IEP plus capability-based isolation (0.032–0.095% across 5th–95th-percentile efficacy priors). Absolute values scale with the assumed re-attack rate, but the strongest variant stays at 13–14% of the write-once duty across all attacker sweeps. Supply-chain exposure ranges from 0.0073% to 5.659% across the interdiction and tamper-detection grid, so at high levels of endpoint hardening the integrity of device logistics can become the binding constraint. The scenario outputs are model estimates rather than measured breach rates, and they define falsifiable hypotheses for a hardware prototype.

1 Introduction

Targeted spyware is a demanding test of the assumption that signed software, verified boot, end-to-end encryption and conventional exploit mitigations are enough for a high-risk mobile communicator. Forensic work continues to document zero-click exploitation of current smartphones, including a September 2026 Citizen Lab report of Pegasus infection of a Serbian pro-democracy student activist through iMessage [1]. Apple characterises mercenary exploit chains as unusually sophisticated and expensive, and has responded with Lockdown Mode and hardware-assisted memory-safety mechanisms such as Memory Integrity Enforcement (MIE) [2, 3]. GrapheneOS similarly combines memory tagging, hardened allocation, sandboxing, verified boot and exploit mitigations [4]. These measures substantially raise the cost of exploitation, but they retain a general-purpose execution environment whose security ultimately depends on a large, mutable software stack.

At the network layer, anonymity systems such as Vuvuzela, Loopix and Karaoke show that communication metadata can be protected against powerful observers by mixing, continuous noise or cover traffic [5–7]. That protection is orthogonal to endpoint compromise: an attacker who controls an endpoint sees content before encryption and after decryption.

This article asks what security properties follow if instruction fetch is physically restricted to authenticated immutable regions, writable memory cannot become executable under compromised privileged software, and containment, egress and image-replacement decisions are rooted outside the mutable operating system. We call the composed property *Immutable Execution Provenance* (IEP). Its central design choice is to move the root of executable integrity from page-table policy over writable memory, however well protected, to the physical instruction path and to physically immutable media, and to give the response to any violation to a separate controller that the application processor (AP) cannot program.

The article makes five contributions.

1. **Architecture and trust boundary.** It defines IEP for a deliberately constrained communicator: whole-AP instruction-source restriction, immutable control-flow authorisation metadata, an independent controller that holds egress, evidence and anti-rollback state, minimal plaintext retention, route reduction, and termination into an established metadata-resistant relay layer.
2. **Residual-attack analysis.** It identifies what remains possible when writable memory is permanently non-executable (code reuse, data-oriented programming and interpreter “weird machines”) and states the architectural measures each residual requires.
3. **Formal specification and model checking.** It specifies the containment controller and a dual-authorized, rollback-resistant image-replacement protocol in TLA+ and checks them exhaustively within stated bounds. Model checking derives six concrete design requirements, each justified by a counterexample against a plausible weaker design (Section 5).
4. **Reproducible scenario model.** It quantifies remote surveillance duty with unique-chain exploit-burn accounting and sensitivity analyses for IEP efficacy, attacker intensity, physical patch lag and supply-chain parameters. Quantities that are true by construction (fixed-cadence timing suppression and a $1/N$ uniform-guess reference) are reported as design assumptions and analytic references.
5. **Deployment implications.** It draws out consequences for organisations that equip high-risk users: the conditions under which device logistics, rather than exploit resistance, become the

binding constraint, and how transparency-logged dual authorisation constrains compelled or targeted malicious updates.

2 Background and related work

2.1 Hardware-anchored code integrity

Apple’s Kernel Integrity Protection (KIP) uses a hardware-protected physical memory region whose write protection is locked after early boot, and constrains the memory management unit so that privileged code cannot be mapped from outside that region [8]. The Page Protection Layer (PPL) prevents user-space code from being modified after code-signature verification. On A15/M2 and later systems the Secure Page Table Monitor (SPTM), which runs at a higher privilege level than the kernel, protects user and kernel page tables and replaces the PPL, while the Trusted Execution Monitor (TXM) enforces code-signing and execution policy [8]. RISC-V Physical Memory Protection provides architecturally enforced permissions on physical regions, and the ratified Smepmp extension adds sticky Machine-Mode Lockdown, under which locked rules cannot ordinarily be modified until reset [9].

Authenticating instructions at the moment they are fetched is also established. Execute-only memory (XOM) and AEGIS protected code held in untrusted external memory with encryption and integrity verification [10, 11]. SOFIA decrypts each instruction at fetch with a key bound to the control flow and checks precomputed message authentication codes, resetting the processor on tampering [12]. Commercial microcontrollers now provide authenticated execute-in-place (AXiP): code in external flash carries a message authentication code for every 32-byte block, which is verified before the instructions execute, and a failure raises a hard fault [13]. Execution directly from mask ROM is older still.

These mechanisms show that hardware can prevent compromised privileged software from redefining executable content after boot, and that instructions can be authenticated as they are fetched from storage. IEP uses both as building blocks and differs in scope and in where the guarantee is rooted. Executable content originates on physically immutable media rather than on signed pages in writable DRAM governed by a privileged monitor. The fetch restriction is enforced on the physical instruction path rather than through page-table policy. Violation handling is delegated to an independent controller that closes egress, captures forensic state and controls which immutable image may run next.

2.2 Control-flow and memory authority

Control-flow integrity (CFI) constrains indirect control transfers to an allowed graph [14]. Arm Pointer Authentication and Branch Target Identification (PAC/BTI) and Intel Control-flow Enforcement Technology (CET) provide hardware support against important return-, jump- and call-oriented programming patterns [15, 16]. CHERI attaches bounds, permissions and integrity metadata to capabilities, substantially reducing ambient memory authority [17]. Arm Memory Tagging Extension and related mechanisms address further memory-corruption classes [18]. The stronger IEP variant composes these controls with fetch provenance, and the seL4 verified microkernel is an optional isolation substrate [19, 20].

2.3 Residual computation inside authorised code

Preventing instruction injection does not prevent attacker-controlled computation if authorised code already contains an expressive interpreter or data-manipulation substrate. Hu *et al.* showed that data-oriented programming can be Turing-complete without control-flow hijacking [21]. Project Zero’s analysis of FORCEDENTRY documents an image-decoder vulnerability used to build a “weird machine” capable of general computation before any conventional native payload ran [22]. Every native instruction fetched in such an attack comes from authenticated code, so fetch provenance alone does not stop it. IEP therefore pairs fetch provenance with parser minimisation, privilege separation, reduced memory authority and a strict egress policy, and excludes rich interpreters and just-in-time compilation from the target profile.

2.4 Secure update, rollback protection and transparency

The Update Framework (TUF) separates signing roles and thresholds so that a software-update system survives the compromise of individual keys [23]. Android Verified Boot stores a rollback index in tamper-evident storage and, in A/B systems, advances it only after the new slot has booted successfully, which preserves the ability to fall back to the previous slot [24]. Certificate Transparency introduced public, append-only logs whose inclusion proofs make misissuance visible [25]. CHAINIAC applies collective signing and verified builds to software-update transparency [26], and Sigstore makes transparency-logged signing practical at scale [27]. IEP adapts these ideas to a setting in which the executable image is a physical object: replacement is a hardware event, the anti-rollback state lives in one-time-programmable (OTP) fuses held by the independent controller, and admission requires both a build signature and a transparency-log inclusion proof.

2.5 Metadata-resistant communication

IEP is an endpoint architecture. The intended endpoint terminates into an established mix or cover-traffic design whose own threat model has been evaluated independently [5–7]. Dummy traffic can fail under repeated observation, and traffic-analysis attacks can be powerful [28–30], which is why the network layer is delegated rather than re-invented here.

2.6 Formal methods for security-critical control logic

TLA+ is a specification language for concurrent and reactive systems, and its explicit-state model checker TLC exhaustively explores the reachable states of a finite instance [31, 32]. Industrial practice has shown that model checking small, precise specifications of control protocols finds subtle design errors before implementation [33]. IEP uses TLA+ for the two parts of the architecture whose correctness is a matter of ordering and race conditions rather than cryptography or circuit behaviour: the containment controller and the image-replacement protocol.

2.7 Positioning

IEP takes authenticated execution from immutable storage as a building block and contributes the system built around it. Relative to the work above, it is distinctive in five respects. First, scope: hardware-locked instruction provenance covers the *complete* executable surface of the application processor of a

communicator, rather than a boot stage, a kernel region or the firmware of a microcontroller. Second, response: a single independent controller holds egress on every path, evidence capture, zeroisation and anti-rollback state, so that a compromised AP can neither report on nor remediate itself. Third, lifecycle: the executable image is a physical object, and its replacement is governed by a dual-authorised, rollback-resistant protocol. Fourth, verification: the controller and the replacement protocol are specified formally, and the ordering requirements they must satisfy are derived from counterexamples. Fifth, quantification: a reproducible model shows how these choices change remote surveillance duty and when supply-chain integrity becomes the limiting factor.

3 Threat model, trust boundaries and metrics

3.1 Adversary model

We consider a state or state-contracted adversary with five relevant capabilities. A_N observes a large fraction of the network and performs long-term traffic correlation. A_E holds scarce zero-click or one-click exploit chains against exposed parsers and may re-attack after reboot. A_S can interdict hardware or compromise a supplier with some probability. A_P can physically seize a device or coerce its user. A_U targets the image-replacement process: it can present any genuine medium the vendor has ever released (replay), present forged media of any version number, interrupt power at any step, and hold the private key of one of the two admission authorities. A compromised correspondent is in scope as a residual, because conversation confidentiality depends on both endpoints.

Explicit non-goals are indefinite physical possession of an unlocked device, coercion, simultaneous compromise of both admission authorities, compromise of all manufacturing trust anchors, malicious foundries outside the assumed attestation chain, and an already-compromised correspondent. They are stated as exclusions rather than hidden as small probabilities in the model.

3.2 Trust boundary

The AP, kernel, parsers and ordinary writable memory are treated as compromisable. The trusted computing base contains: (i) a minimal immutable boot root; (ii) a hardware instruction-fetch provenance mechanism; (iii) immutable control-flow authorisation metadata in the stronger variant; (iv) capability enforcement or comparable hardware isolation where used; (v) an independent security controller, including its OTP anti-rollback fuse and write-once evidence store, that the AP cannot reprogram; and (vi) the roots needed to authenticate the immutable image. The anonymity network is a separate dependency whose guarantees are not re-proved here.

3.3 Metrics

The evaluation uses four kinds of result. *Formal results* are TLC verdicts on stated properties of bounded models (Section 5). *Remote surveillance duty* is a Monte Carlo scenario output: the fraction of the modelled horizon during which remote exploitation controls the endpoint. *Supply-chain exposure* is computed analytically from explicit interdiction and tamper-detection assumptions. *Derived information exposure* is deterministic consequence accounting that applies the assumed egress-blocking, session-rotation, reboot and archive policies to a compromise-duty value. For the network regressions, a fixed-

cadence profile has zero partner-specific timing signal by construction, and $1/N$ is an analytic uniform-guess reference.

4 Architecture

Fig. 1 shows the partition.

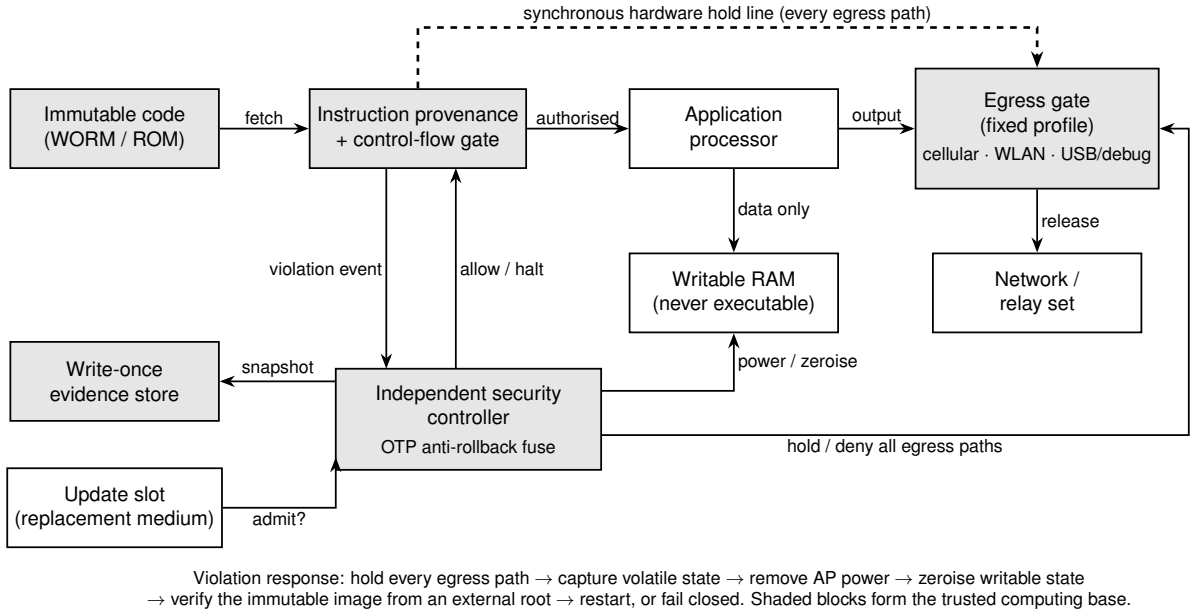


Figure 1: IEP architecture. Writable memory is usable as data but can never become executable. The provenance gate drives a synchronous hardware hold line into every egress path, and the independent security controller performs containment, holds the OTP anti-rollback fuse and admits replacement media. Shaded blocks form the trusted computing base.

4.1 Immutable instruction provenance

The executable system image resides on physically non-rewritable, write-once, read-many (WORM) media, or an equivalently immutable authenticated instruction store, plus a minimal boot ROM. A hardware fetch policy rejects instruction fetches from writable regions and cannot be relaxed by the AP after the trusted boot transition. Writable RAM remains usable as data. The property is stronger than ordinary $W \oplus X$, which a compromised kernel controlling page tables could otherwise relax. Applied to the complete executable surface of a constrained communicator and coupled to independent response and egress control, it removes persistence in executable storage and native-code injection as attacker options.

4.2 Control-flow and capability authorisation

Instruction-source restrictions do not stop code reuse. The stronger IEP variant therefore adds control-flow authorisation and capability-style memory authority, drawing on CFI, PAC/BTI, CET and CHERI [14–17]. A verified microkernel such as seL4, for which machine-checked functional-correctness and security proofs exist, including AArch64 security proofs reported in 2026, can provide the isolation substrate [19, 20].

4.3 Independent containment and egress

A security controller outside AP software control receives provenance events and controls network release on every egress path: cellular, WLAN and wired or debug interfaces. Normal traffic leaves only according to a prescribed profile and only to an approved relay set. On a violation the provenance gate asserts a hardware hold line into the egress gate in the same step, and the controller then captures volatile state to write-once evidence storage, removes AP power, zeroises writable state, verifies the immutable image from an external root and restarts (Fig. 2). If verification fails, or if the evidence store is exhausted, the device fails closed with egress denied until physical service. The controller exposes no AP-writable command that can clear a latched violation or reopen egress. Section 5 shows that each of these choices is necessary.

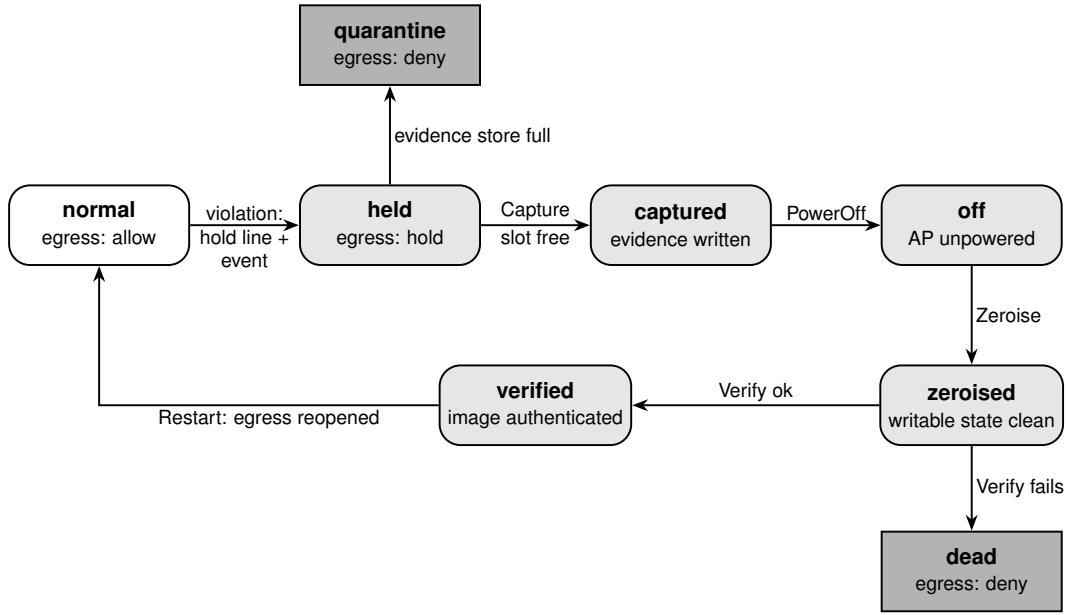


Figure 2: Containment-controller state machine as specified in TLA+. Egress is open only in *normal*; the only transition that reopens it is *Restart* from a verified image over zeroised writable state. Dark states are terminal and fail closed.

The scenario model gives failed exploit attempts no forensic-burn credit. Burn accounting starts only after a successful deployment creates the resident state from which the forensic sequence is defined. Detection of failed attempts, if a prototype demonstrates it, would require a separately measured probability.

4.4 Minimal data value at compromise

The device keeps no bulk plaintext archive. Stored content is ciphertext under per-item keys; only material currently open is released for use, and working keys are destroyed when the item closes. Independent egress policy further limits what a transient compromise can transmit. These policies affect consequences rather than exploit probability, which is why the model reports their effect as derived information exposure.

4.5 Immutable-image replacement

An immutable image cannot receive an in-place patch; remediation installs an authenticated replacement medium. The controller admits a medium only if it carries a build-authority signature, an inclusion proof from an independent append-only transparency log, and a version number strictly greater than the OTP fuse value. It holds every egress path while checking, raises the fuse to the new version *before* the new medium becomes executable, and applies the same raise-before-execute rule when the device cold-boots from any medium. A cold boot accepts a medium whose version is at least the fuse value, so the current image can always restart. Section 5 verifies this protocol against replay, forgery with one compromised authority and power interruption at every step.

Physical replacement takes longer than an over-the-air patch. The central scenario uses a 30-day patch lag for the baseline smartphone and a 90-day physical replacement lag for WORM and IEP configurations. Every exploit chain remains usable for its silent lifetime plus the device’s patch or replacement lag, so the longer lag applies to every vendor-patched vulnerability. Across 30-, 90- and 180-day lags, remote-only duty is 0.411, 0.416 and 0.423% for fortified WORM and 0.057, 0.057 and 0.059% for IEP plus capability isolation. The effect is small because, under the central acquisition assumptions, the attacker almost always holds a usable chain; per-attempt exploit success, not chain availability, limits exposure. An adversary with fewer chains would make the result more sensitive to replacement latency.

4.6 Network layer

The endpoint terminates into an established mix or cover-traffic design such as Loopix, Vuvuzela or a comparable system [5–7]. The included network code is a regression harness. It checks whether simple dummy-traffic choices obviously leak under repeated statistical disclosure and whether variable-rate radio or account activity remains correlatable [28–30].

5 Formal specification and model checking

Summary. The two parts of IEP whose correctness depends on the order of events are specified in TLA+: the containment controller (module `IEPController`) and the image-replacement protocol (module `IEPUpdate`). TLC explores every reachable state of bounded instances and checks safety invariants, action properties and, under weak fairness of controller actions, liveness. The IEP designs satisfy all properties in every configuration checked, including when either admission authority is compromised. Each of six weaker alternatives that a designer might reasonably choose violates a property, and TLC produces a counterexample of 3 to 26 states. These counterexamples become design requirements R1–R6. The models abstract away timing, circuits and cryptography, so they verify the protocol design rather than an implementation. The specifications, configurations, logs and a one-command runner are in the supplementary artifact.

5.1 Method

Both modules are written in TLA+ [31] and checked with TLC [32] (tla2tools build 2026.10.06, revision 94d0c50) using breadth-first exhaustive search. Deadlock checking is disabled because the terminal fail-closed states are intended. Liveness properties are checked under weak fairness of the controller’s

actions; adversary actions carry no fairness, so the adversary may act or remain idle at any point. Each property is listed in Table 1, and every configuration with its result is in Table 2. The specifications, configurations and runner scripts were drafted with the assistance of Anthropic Claude (Claude Opus 5.5, model identifier `claude-opus-5-5`) and reviewed by the author against the architecture of Section 4; every verdict, state count and counterexample reported here was produced by TLC and is reproduced by the supplementary artifact.

Table 1: Properties checked by TLC. Each module also checks a type invariant. Kinds: I, state invariant; A, action property; L, liveness under weak fairness of controller actions.

ID	Property	Kind	Meaning
<i>Containment controller (IEPController)</i>			
C1	NoReleaseAfterViolation	I	No message leaves the device on any path after a provenance violation in the same epoch
C2	EgressClosedWhileLatched	I	Every egress path is closed while a violation is latched
C3	EvidenceBeforeErase	I	Writable state is powered down or erased only after evidence has been captured
C4	AllowImpliesVerified	I	Any open egress path implies a verified immutable image
C5	TerminalIsClosed	I	Failed verification and evidence exhaustion are terminal, with all egress denied and the AP unpowered
C6	ReopenOnlyByRestart	A	The only step that reopens a closed egress path is a restart from a verified image over clean writable state
C7	CleanPowerOn	A	The AP is never powered on over dirty writable state or an unverified image
C8	ViolationLatched	L	Every violation is eventually latched by the controller
C9	Recovery	L	Every latched violation ends in a clean restart or a fail-closed terminal state
<i>Image replacement (IEPUpdate)</i>			
U1	NoRollback	I	The running image is always the highest version that has ever executed
U2	AuthenticExecution	I	Only vendor-built images execute
U3	EgressClosedDuringUpdate	I	Egress is closed whenever no admitted image is running
U4	Recoverable	I	The fuse never exceeds the newest genuine release, so a bootable genuine medium always exists
U5	FuseMonotone	A	The anti-rollback fuse never decreases
U6	Restarts	L	After any power interruption the device returns to a running image

Table 2: Model-checking results. Distinct states are those reachable in the configuration (for counterexamples, those explored before the violation was found). Each counterexample motivates the design requirement listed.

Configuration	States	Result	Trace	Requirement
<i>Containment controller; three egress paths</i>				
IEP, 3 evidence slots	32	C1–C9 hold	–	–
IEP, 8 evidence slots	77	C1–C9 hold	–	–
Event-only notification, no hold line	4	C1 violated	3	R1
Hold line on primary radio only	4	C1 violated	3	R2
AP-writable resume command	4	C6 violated	3	R3
Erase without capture when store is full	33	C3 violated	26	R4
<i>Image replacement; dual authorisation, burn-first, cold boot burns</i>				
No key compromised; 3 versions, 2 faults	318	U1–U6 hold	–	–
Build key compromised; 3 versions, 2 faults	318	U1–U6 hold	–	–
Log key compromised; 3 versions, 2 faults	318	U1–U6 hold	–	–
No key compromised; 5 versions, 4 faults	2,235	U1–U6 hold	–	–
Build key compromised; 5 versions, 4 faults	2,235	U1–U6 hold	–	–
Log key compromised; 5 versions, 4 faults	2,235	U1–U6 hold	–	–
<i>Weaker replacement designs; 3 versions, 2 faults</i>				
Trial-then-commit fuse update	86	U1 violated	6	R5
Cold boot does not raise fuse	131	U1 violated	6	R6
Build signature only; build key compromised	23	U2 violated	3	threshold
Dual authorisation; both keys compromised	23	U2 violated	3	threshold

5.2 Containment controller

The AP is adversarial. It may write attacker-controlled data to writable memory, trigger a provenance violation (an instruction fetch outside the authenticated immutable regions, which the gate refuses), and request release of a profile-conforming message on any of three egress paths: cellular, WLAN and USB/debug. The controller progresses through the phases of Fig. 2. Image verification is non-deterministic, so the model covers a damaged or substituted medium. The write-once evidence store holds a bounded number of incident records (3 in the base configuration, 8 in the large one), and the adversary may trigger violations repeatedly across restarts. A ghost variable records whether any message has left the device after a violation in the same epoch. Four design parameters select the variants: whether the gate drives a synchronous hold line (SyncHold), whether that line covers every egress path or only the primary radio (HoldAll), whether the controller exposes an AP-writable resume command (ApResume), and whether it erases without capture when the evidence store is full (EraseWhenFull).

5.3 Image-replacement protocol

The vendor releases successive versions. The adversary may present any released genuine medium or a forged medium of any version, may hold the build key or the log key (each forged medium then carries the corresponding authorisation), and may cut power at any step up to a bound. Admission, fuse burning, activation and cold boot follow the protocol of Section 4.5. Three parameters select the variants: whether admission requires both authorities (DualAuth), whether the fuse is burned before the new medium executes or only after a successful trial boot (BurnFirst), and whether cold boot raises the fuse (BootBurns). A ghost variable records the highest version that has ever executed; the no-rollback invariant requires every running image to equal it. The base configuration has 3 versions and 2 power

faults; the large configuration has 5 versions and 4 power faults.

5.4 Results and derived design requirements

In the IEP configurations TLC finds no violation. The controller model has 32 distinct reachable states with 3 evidence slots and 77 with 8; the replacement model has 318 distinct states in the base configuration and 2,235 in the large one, identical whether no key, the build key or the log key is compromised (Table 2). The weaker designs fail as follows.

R1. Synchronous hold line. If the gate only posts an event for the controller to process, there is a window between violation and processing in which the AP can still release a message. TLC finds this in 3 states (violate, then send). The provenance gate must close egress in hardware in the same step that it refuses the fetch.

R2. Every egress path is held. If the hold line covers only the primary radio, the AP releases a message over USB/debug or WLAN before the controller reacts (3 states). Debug and wired interfaces are egress paths.

R3. No processor-writable resume. A resume command writable by the AP, a common driver-to-monitor interface, lets a compromised AP reopen egress after a violation (3 states, violating the property that only a verified restart reopens egress). The controller must expose no such path.

R4. Fail closed when evidence storage is exhausted. If the controller erases writable state without capture once the write-once evidence store is full, a fourth incident with three slots erases unrecorded state (26 states). An adversary can therefore deliberately exhaust the store before the incident that matters. When the store is full the controller must quarantine the device.

R5. Raise the fuse before the new image executes. Trial-then-commit, in which the fuse advances only after a successful boot of the new image, permits rollback: release, power cut, cold boot into a trial of the newer image, power cut before commit, then boot of the older medium (6 states). Android Verified Boot defers the rollback-index update by design to preserve fallback [24]; with physically removable media and an adversary who can interrupt power, the trial window becomes a rollback window.

R6. Cold boot is an activation path. With burn-first updates but a cold boot that does not raise the fuse, the adversary bypasses the update protocol entirely: cold boot a newer genuine medium, cut power, and cold boot the older one (6 states). Every path that makes a medium executable must apply the same rule.

Two further configurations establish the authorisation threshold. With a single build signature and the build key compromised, a forged image executes (3 states); with dual authorisation and both keys compromised, the same occurs. Authentic execution therefore holds exactly when at least one of the two independent authorities is uncompromised, which is the intended 2-of-2 guarantee.

5.5 Scope of the formal results

The models assume that the provenance gate never lets a refused fetch commit, that each modelled step is atomic, that the OTP fuse is monotone in hardware, and that signatures and log inclusion proofs are unforgeable without the corresponding key, which is abstracted as authority compromise. The transparency log appears only as an independent admission authority: its append-only consistency, resistance to split views and independent monitoring are assumed rather than checked. They contain no timing, so they say nothing about the latency of the hold line or about physical side channels. Results hold for the stated bounds: up to 8 evidence slots and, for replacement, 5 versions and 4 power faults. Small exhaustive instances of this kind are effective at exposing ordering errors, as the counterexamples above illustrate, but the results are not an unbounded proof. Liveness holds under controller fairness; an adversary who triggers violations repeatedly degrades availability without breaking any safety property, and quarantine on evidence exhaustion turns that into a denial of service by design.

6 Scenario model

The scenario artifact is a single Python program using NumPy and Matplotlib with fixed seeds, CSV outputs and figures. Its purpose is to expose assumptions and sensitivity, not to estimate any vendor's breach probability. The code was debugged and consistency-checked with the assistance of OpenAI ChatGPT ([CHATGPT MODEL AND VERSION]) and edited and re-run with the assistance of Anthropic Claude (Claude Opus 5.5); every value reported in Sections 6–7 is produced by the code in the supplementary artifact, and the author checked each against the generated CSV files.

6.1 Remote route closure

The scenario assigns illustrative weights to messaging zero-click, SMS or one-click, carrier injection, browser drive-by, cell-site-simulator and baseband routes. Data-only service, no browser or mail client, no automatic rendering, admission control, pinned encrypted transport, legacy-radio refusal and radio separation multiplicatively reduce those assumed routes. The resulting remote-delivery capacity is 2.519% of the starting remote weight, after which a legacy parser and memory-safety factor of 0.25 is applied. Fig. 3 displays these assumed residual route weights; they are not a forensic frequency estimate.

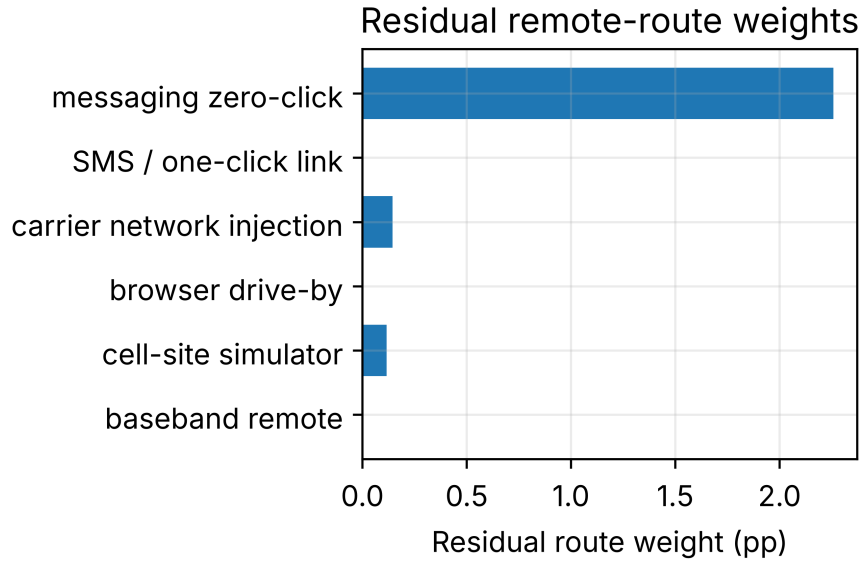


Figure 3: Assumed residual remote-route weights after route-closure controls, in percentage points of the starting remote-delivery weight (they sum to 2.519). The inputs are scenario parameters, not measured prevalence.

6.2 Unique-chain exploit-burn accounting

In the central scenario the attacker acquires four chains a year, each with a 270-day mean silent lifetime, and attempts reinfection when access is absent. The patch lag is 30 days for the baseline smartphone and 90 days centrally for immutable-media configurations. The fortified configurations use a forensic chain-burn probability equal to the product of six explicit post-deployment stages: resident implant in AP DRAM (0.75), anti-forensic survival (1.00), snapshot coverage (0.85), malicious recognition (0.55), exploit recoverability (0.35) and patch shipment (0.80), giving 0.0982. Table 3 lists every central parameter.

Table 3: Central scenario parameters. All are assumptions; Section 7.3 sweeps the attacker parameters.

Parameter	Central value
Horizon; trials	1,095 days; 2,000 per configuration; fixed seeds
Attacker chains	3 initial; 4 acquired per year; inventory cap 8
Silent chain lifetime	Exponential, mean 270 days
Re-attack while access absent	0.80 per day
Exploit success per attempt	Baseline 0.85; fortified 0.00535 ($0.85 \times 2.519\% \times 0.25$)
IEP residual factor (median)	0.458 basic; 0.140 capability
Chain burn after success	0.0982, at most once per chain
Patch/replacement lag	30 days baseline; 90 days immutable media
Reboot/re-image	Mean interval 30 days baseline; every simulated day (1-day time step) fortified
Persistence	Baseline only; detection 1/900 per day
Shipments	Provisioning on day 0; 0.5 per year baseline; 2 per year fortified
Interdiction / tamper detection	0.15 / 0 baseline; 0.02 / 0.97 fortified
Device replacement	Mean 730 days; clears implants; re-exposed to interdiction
Egress block; session rotation; archive	0, none, 1,825 days baseline; 0.99, 5 min, none fortified

The burn probability is applied only after successful exploit deployment. Failed attempts neither create

a resident implant nor burn a chain. Each chain carries a unique burn state: once detected and recovered it is counted as burned once. It may remain operational until the patch or replacement lag expires, but a later reinfection with the same chain cannot generate another burn event. Burn rates are therefore coupled to successful deployments and unique chain inventory rather than to raw attack attempts.

6.3 IEP residual-factor uncertainty

IEP is not represented as a single fixed block rate. Exploit utility is partitioned into injected-code, code-reuse, memory-corruption or data, logic or confused-deputy, and peripheral or microarchitectural primitives. The exploit mixture is sampled from a Dirichlet distribution and defence efficacy from broad uniform ranges. For IEP-basic, the 5th, 50th and 95th-percentile residual factors are 0.335, 0.458 and 0.592; for IEP plus capability isolation they are 0.077, 0.140 and 0.234 (Fig. 4).

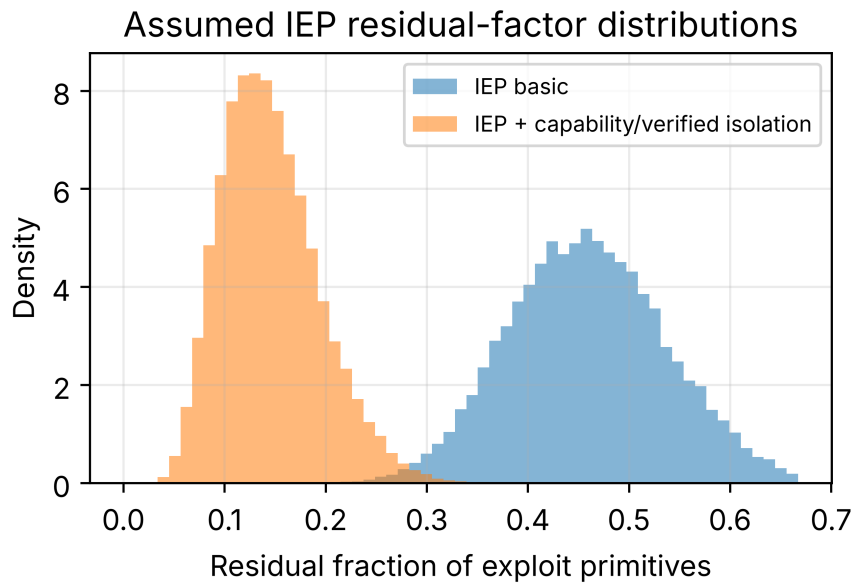


Figure 4: Assumed residual-exploit-factor distributions for IEP-basic and IEP plus capability/verified isolation. These are sensitivity priors, not empirical efficacy measurements.

6.4 Supply-chain process

Supply compromise is modelled separately from remote exploitation. For interdiction probability p_i and tamper-detection probability p_d , the per-shipment probability of undetected interdiction is $p_i(1 - p_d)$. The process assumes a provisioning shipment on day 0, two further shipments a year, and device replacement with a mean interval of 730 days. An undetected interdiction implants the device until the next replacement, which clears the implant but is itself exposed to interdiction. Expected implanted-state duty over the three-year horizon is evaluated analytically for $p_i \in \{0.005, 0.01, 0.02, 0.05, 0.10\}$ and $p_d \in \{0.80, 0.90, 0.97, 0.99, 0.995\}$, which avoids drawing conclusions from rare-event Monte Carlo noise.

7 Scenario results

7.1 Remote-only endpoint exposure

Table 4 reports remote-only runs with supply interdiction disabled, so that the effect of endpoint architecture is not obscured by a separate risk process. Fortified WORM with route closure reduces modelled remote duty to 0.416%. At the median residual factor, IEP-basic gives 0.186% and IEP plus capability isolation gives 0.057%, with 53.1% of trials never remotely compromised over three years. All three rows use the same pseudorandom stream. Burn rates fall as exploit success falls.

Table 4: Remote-only scenario results. Supply-chain compromise is disabled. Duty is the mean $\pm$ one Monte Carlo standard error over 2,000 trials; “Never” is the share of trials with no remote compromise; Burn/yr counts unique exploit chains, each burned at most once. Values are model outputs, not measured breach rates.

Configuration	Duty	Never	Burn/yr
Fortified WORM (90-day lag)	$0.416 \pm 0.004\%$	1.2%	0.148
IEP-basic, median	$0.186 \pm 0.003\%$	14.2%	0.069
IEP + capability, median	$0.057 \pm 0.002\%$	53.1%	0.020

Fig. 5 shows joint scenario exposure and derived consequence accounting. Because rare supply-chain events create noisy joint means, they are not used to support the remote-effect result; the supply component is analysed analytically in Section 7.4.

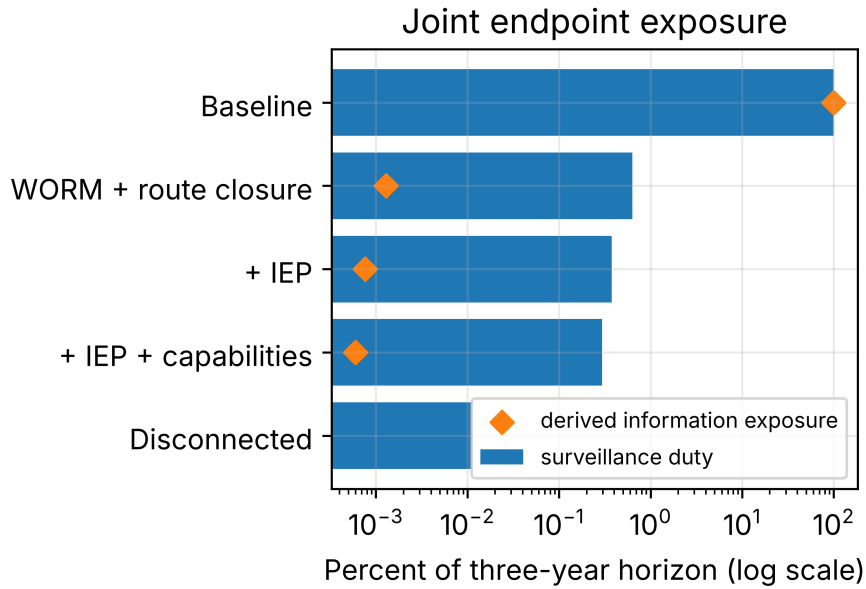


Figure 5: Joint scenario exposure and deterministic derived information exposure. The latter is algebraic consequence accounting, not an independent simulation result. The disconnected reference has zero derived information exposure by construction, so no marker is plotted for it.

7.2 Sensitivity to IEP residual factor

The remote-only endpoint process is run separately at the 5th, 50th and 95th-percentile residual factors (Table 5). These are sensitivity points, not quantiles of an output distribution. For the strongest variant, mean remote duty spans 0.032–0.095%.

Table 5: Mean remote-only endpoint duty evaluated at the 5th, 50th and 95th-percentile IEP residual factors. These are sensitivity points, not quantiles of an output-duty distribution.

Variant	Mean duty at residual-factor percentile		
	5th	50th	95th
IEP-basic	0.135%	0.186%	0.244%
IEP + capability	0.032%	0.057%	0.095%

7.3 Sensitivity to attacker intensity

Table 6 varies the attacker parameters one at a time around the central case, with supply interdiction disabled. Remote duty is close to proportional to the assumed re-attack rate: from 0.051% at 0.1 attempts per day to 0.516% at 1.0 per day for fortified WORM, and from 0.007% to 0.072% for IEP plus capability isolation. The re-attack rate is the most influential attacker assumption; chain acquisition (1–12 per year) and initial inventory (0–8 chains) move duty by at most about 22%. The ratio of IEP-plus-capability to fortified-WORM duty stays between 0.13 and 0.14 in every row, so the relative architectural effect is robust to attacker intensity even though absolute duties are not.

Table 6: Mean remote-only duty under one-at-a-time attacker sweeps (2,000 trials each; Monte Carlo standard errors ≤ 0.005 percentage points). *Central value; the central acquisition rate is 4 chains per year and the central initial inventory is 3 chains.

Parameter	Value	WORM	IEP + cap.	Ratio
Re-attack per day	0.1	0.051%	0.007%	0.13
	0.4	0.214%	0.030%	0.14
	0.8*	0.416%	0.057%	0.14
	1.0	0.516%	0.072%	0.14
Chains per year	1	0.324%	0.047%	0.14
	12	0.427%	0.058%	0.14
Initial chains	0	0.376%	0.053%	0.14
	8	0.420%	0.058%	0.14

7.4 When supply-chain exposure dominates

Under the central fortified assumptions, $p_i = 0.02$ and $p_d = 0.97$, the probability of undetected interdiction per shipment is 0.0006 and the analytic expected supply-chain duty is 0.175%. That exceeds the strongest variant’s median remote-only duty of 0.057%, so under these parameters supply-chain exposure is the larger component. Across the full grid (Fig. 6) expected supply-chain duty ranges from 0.0073% ($p_i = 0.005$, $p_d = 0.995$) to 5.659% ($p_i = 0.10$, $p_d = 0.80$). Aggressive endpoint hardening can therefore make supply integrity the binding constraint, and whether it does depends directly on the interdiction and detection environment.

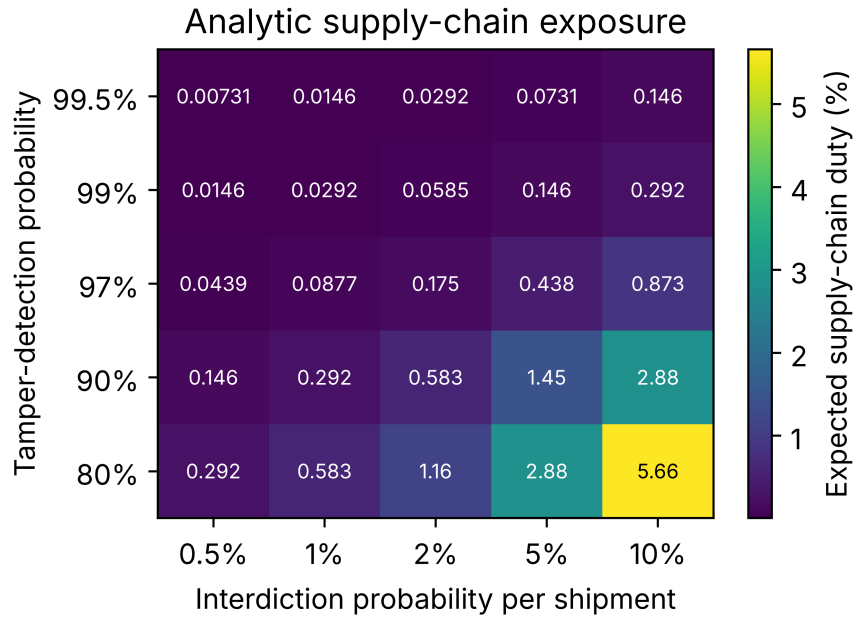


Figure 6: Analytic supply-chain exposure over the interdiction and tamper-detection grid. Cell values are expected supply-chain duty (% of the three-year horizon).

7.5 Network regressions

The network harness retains symmetric cover-traffic regressions only (Fig. 7). Bystander-targeted decoys are excluded on ethical and methodological grounds. The fixed-cadence profile's zero partner-specific timing signal is a design assumption, and $1/50,000$ is an analytic uniform-guess reference. The variable-rate timing experiment and statistical-disclosure regressions are sanity checks on the endpoint's traffic profile; formal anonymity is inherited from the chosen relay layer.

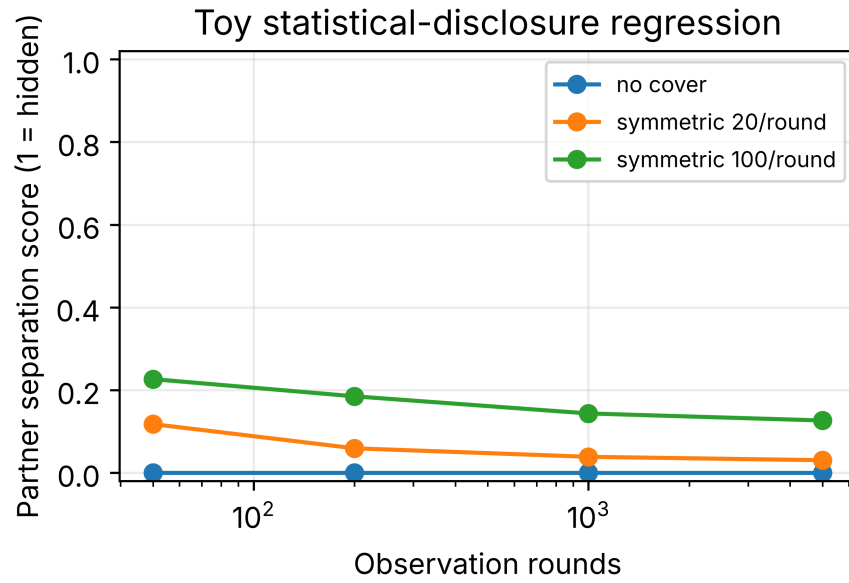


Figure 7: Toy statistical-disclosure regression for no cover traffic and symmetric cover traffic, used as a design sanity check.

8 Security properties and residual risks

Table 7 maps each mechanism to its principal effect and its most important residual. Each row closes some attack paths and leaves others.

Table 7: Mechanism-to-residual mapping. No row is a complete security guarantee.

Mechanism	Primary effect	Important residual
Write-once system media	Blocks ordinary persistence and post-deployment code rewriting	Known vulnerable image persists until authenticated physical replacement
IEP fetch provenance	Blocks native instruction fetch from writable memory	Code reuse; data-oriented programming; authorised interpreters and weird machines
CFI/PAC/BTI/CET + capabilities	Restricts control-flow reuse and memory authority	Authorised-but-vulnerable logic; confused deputies; imperfect policies
Independent controller and egress gate (R1–R4)	No release after a violation on any path; only a verified restart reopens egress; evidence before erasure	Controller implementation bugs; hold-line latency; physical and covert channels; compromised correspondent
Dual-authorised, burn-first replacement (R5–R6)	No rollback; no forged image unless both authorities are compromised	Compromise of both authorities; availability cost of quarantine and replacement logistics
Minimal plaintext retention	Reduces the value of a brief compromise	Currently open material; coerced disclosure
Mix/cover layer	Reduces network metadata linkability under its own model	Endpoint compromise; limits of the relay layer’s network model
Supply-chain controls	Reduce undetected pre-delivery implants	Malicious or colluding trust anchors; uncertain real interdiction rates

The architecture also creates an operational tension. Faster patching needs a trusted channel for introducing newly authenticated code, while stronger immutability deliberately makes that channel harder to use. A deployable system must therefore quantify update latency, replacement logistics, rollback resistance, manufacturing trust and availability during remediation alongside exploit resistance.

9 Implications for deployment

Four results matter to organisations that equip journalists, human-rights defenders or officials with high-risk devices.

First, logistics can matter more than exploit resistance. Once the endpoint is hardened to IEP plus capability isolation, the central supply-chain assumptions already produce more exposure than remote exploitation (0.175% against 0.057%). Investment in tamper-evident shipping, chain-of-custody procedures and device attestation then buys more than further endpoint hardening, and the grid in Fig. 6 shows how quickly the balance shifts with detection rates.

Second, the lag of physical replacement is a smaller cost than intuition suggests. Under the central attacker, moving from 30 to 180 days changes the strongest variant’s duty from 0.057% to 0.059%, because exposure is limited by per-attempt exploit success rather than by chain availability. The result reverses for an adversary with few chains, which is where replacement cadence should be revisited.

Third, dual authorisation with transparency logging constrains compelled or targeted malicious updates. A single coerced or stolen signing key cannot place a new image on a device. Provided the log is append-only, consistent and independently monitored, every admitted image is also publicly visible, so a version built for one user would have to appear in a public log; resistance to split-view attacks then depends on log gossip or witness cosigning, which lie outside the formal model. This is a governance property as much as a technical one: it lets an equipping organisation, or an independent monitor, audit which images its devices are permitted to run.

Fourth, the target profile is narrow by design. No browser, no rich media rendering and no third-party applications are acceptable for a small population whose risk justifies them, not for the general public. IEP is offered as an option for that population, with the costs stated.

10 Limitations and falsification criteria

No IEP processor has been fabricated, no FPGA fetch gate is reported and no mobile operating system has been ported. The formal results cover protocol designs in bounded abstract models, not implementations, and the scenario model states only what follows from its parameters. Whether physically rooted provenance yields measurably greater assurance than page-table monitors such as SPTM/TXM is an empirical question for a prototype.

A prototype would falsify the proposed benefit if: (i) authorised interpreters, data-oriented programming or confused-deputy attacks retain substantially more utility than the sensitivity model assumes; (ii) fetch provenance can be bypassed through DMA, speculative or microarchitectural behaviour, peripheral processors or debug state; (iii) the hold line cannot be made fast enough relative to the egress path to realise requirement R1, or the controller introduces a bypass or covert channel; (iv) physical update latency creates more exploitable time than immutability saves; (v) fixed-profile egress is unusable at the required bandwidth; or (vi) realistic supply-chain controls fail to approach the assumed tamper-detection range.

The endpoint model also omits malicious compilers, foundry-level implants beyond the trust model, compromised secure elements, power and electromagnetic side channels, and social or coercive attacks. The network regression omits Internet routing dynamics and is far weaker than the analyses of mature anonymity systems. These omissions mark work required before a deployable security claim.

11 Implementation and evaluation roadmap

The next step is a prototype. A credible evaluation would include: (1) an FPGA implementation of the physical fetch-provenance policy and of the synchronous hold line, with measured hold latency against the fastest egress path; (2) attempts to execute from writable RAM under a fully compromised privileged software layer; (3) measured latency, energy, area and memory overhead; (4) integration with hardware-enforced memory authority or compartmentalisation; (5) fault-injection and debug-interface testing of the provenance gate, egress gate and controller; (6) adversarial tests using code reuse, data-oriented programming, interpreter weird machines, DMA and peripheral compromise; (7) a measured replacement process, including fuse burning, transparency-log verification and recovery from interrupted updates; and (8) manufacturing and attestation experiments sufficient to replace the supply-chain priors with evidence.

On the formal side, the TLA+ models can be refined towards the hardware: checking the controller's register-transfer-level description against the specified state machine, and extending the replacement model with explicit transparency-log consistency proofs.

12 Conclusion

Immutable Execution Provenance composes physically immutable executable storage, hardware-locked instruction provenance, control-flow and capability authorisation, and an independent controller for egress,

evidence, zeroisation and image replacement into an endpoint architecture for a constrained high-risk communicator. Formal specification and model checking show that the containment controller and the dual-authorized, burn-first replacement protocol satisfy their safety and liveness properties within the checked bounds, including against compromise of either admission authority. The same analysis derives six design requirements, R1–R6, each demonstrated by a counterexample. Under the central remote-only scenario, fortified WORM yields 0.416% surveillance duty, IEP-basic 0.186% and IEP plus capability isolation 0.057%, with the strongest variant spanning 0.032–0.095% across efficacy priors and remaining at 13–14% of the WORM duty across all attacker sweeps. Supply-chain exposure ranges from 0.0073% to 5.659% across the interdiction and detection grid and exceeds remote exposure under the central assumptions. These values are scenario outputs rather than empirical breach rates. The next step is hardware implementation and adversarial evaluation, with particular attention to interpreter and data-only attacks, the latency of the hold line, and the logistics of replacement and supply.

Ethical considerations

This work is defensive. It uses synthetic traffic, simulated compromise events and abstract formal models; it does not develop or deploy exploits. Bystander-targeted false-correlation traffic is excluded because deliberately manufacturing evidence that an uninvolved person communicates with a target creates ethical risk and is not needed to evaluate the endpoint architecture. No human subjects, victim devices, live networks or spyware samples were used.

Acknowledgements

Generative AI was used as follows. Artificial Inventor V63) was used for drafting and editorial restructuring of the manuscript text; for debugging and consistency checking of the scenario model (Section 6); for consistency checking of the scenario tables and figures (Section 7); and for preparation of submission materials, for drafting the TLA+ specifications, model-checking configurations and runner scripts (Section 5); for editing and re-running the scenario model (Sections 6–7); for drawing Figs. 1 and 2; and for formatting and preparing the submission materials. These tools were used to accelerate drafting, coding and formatting by a single author. To validate their outputs, the author reviewed every specification against the architecture description, re-ran all TLC configurations and the scenario model from the supplementary artifact, checked every reported number against the generated output files, and checked every reference against its source. The author takes full responsibility for all content.

Funding

This work received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

Conflict of interest

The author is the inventor and applicant on a pending patent application covering aspects of the IEP architecture described in this article and may therefore have a financial interest in its commercialisation.

Data availability

The supplementary artifact contains the TLA+ specifications, TLC configurations, model-checking logs and summary (`formal_results.csv`), the Python scenario model with its requirements file, explicit assumptions, generated CSV outputs and figures, and one-command scripts that regenerate all results. The scenario model uses fixed pseudorandom seeds; the TLC runs are deterministic. No private dataset is required.

Author contributions

Jonathan K. Bily: Conceptualization, Methodology, Software, Formal analysis, Investigation, Visualization, Writing – original draft, Writing – review & editing.

References

- [1] The Citizen Lab. Pegasus spyware infection of Serbian pro-democracy student activist. University of Toronto, 2 September 2026.
<https://citizenlab.ca/research/pegasus-spyware-infection-of-serbian-activist/> (2 October 2026, date last accessed).
- [2] Apple. Lockdown Mode security for Apple devices. *Apple Platform Security*.
<https://support.apple.com/guide/security/lockdown-mode-security-sec2437264f0/web> (2 October 2026, date last accessed).
- [3] Apple Security Engineering and Architecture. Memory Integrity Enforcement: a complete vision for memory safety in Apple devices. September 2025.
<https://security.apple.com/blog/memory-integrity-enforcement/> (2 October 2026, date last accessed).
- [4] GrapheneOS Project. Features overview. <https://grapheneos.org/features> (2 October 2026, date last accessed).
- [5] van den Hooff J, Lazar D, Zaharia M *et al.* Vuvuzela: scalable private messaging resistant to traffic analysis. In: *Proceedings of the 25th Symposium on Operating Systems Principles*. New York, NY: ACM, 2015, 137–52.
- [6] Piotrowska AM, Hayes J, Elahi T *et al.* The Loopix anonymity system. In: *Proceedings of the 26th USENIX Security Symposium*. Berkeley, CA: USENIX Association, 2017, 1199–216.
- [7] Lazar D, Gilad Y, Zeldovich N. Karaoke: distributed private messaging immune to passive traffic analysis. In: *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation*. Berkeley, CA: USENIX Association, 2018, 711–25.

- [8] Apple. Operating system integrity. *Apple Platform Security*.
<https://support.apple.com/guide/security/operating-system-integrity-sec8b776536b/web> (2 October 2026, date last accessed).
- [9] RISC-V International. Smepmp extension for PMP enhancements for memory access and execution prevention in Machine mode, version 1.0. *RISC-V Ratified Specifications Library*.
<https://docs.riscv.org/reference/isa/priv/smeppmp.html> (2 October 2026, date last accessed).
- [10] Lie D, Thekkath C, Mitchell M *et al*. Architectural support for copy and tamper resistant software. In: *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*. New York, NY: ACM, 2000, 168–77.
- [11] Suh GE, Clarke D, Gassend B *et al*. AEGIS: architecture for tamper-evident and tamper-resistant processing. In: *Proceedings of the 17th Annual International Conference on Supercomputing*. New York, NY: ACM, 2003, 160–71.
- [12] de Clercq R, De Keulenaer R, Coppens B *et al*. SOFIA: software and control flow integrity architecture. In: *Proceedings of the 2016 Design, Automation & Test in Europe Conference*. Dresden: EDAA, 2016, 1172–7.
- [13] Silicon Labs. Authenticated eXecute in Place (Series 3 AXiP). *IoT Security* documentation. <https://docs.silabs.com/iot-security/latest/iot-security-series-3-axip/authenticated-execute-in-place> (6 October 2026, date last accessed).
- [14] Abadi M, Budiu M, Erlingsson Ú *et al*. Control-flow integrity. In: *Proceedings of the 12th ACM Conference on Computer and Communications Security*. New York, NY: ACM, 2005, 340–53.
- [15] Arm. Providing protection for complex software: pointer authentication, branch target identification and memory tagging. Arm Developer.
<https://developer.arm.com/-/media/Arm%20Developer%20Community/PDF/Learn%20the%20Architecture/Providing%20protection%20for%20complex%20software.pdf> (2 October 2026, date last accessed).
- [16] Intel. A technical look at Intel Control-flow Enforcement Technology. Intel Developer, 13 June 2020.
<https://www.intel.com/content/www/us/en/developer/articles/technical/technical-look-control-flow-enforcement-technology.html> (2 October 2026, date last accessed).
- [17] Watson RNM, Woodruff J, Neumann PG *et al*. CHERI: a hybrid capability-system architecture for scalable software compartmentalization. In: *Proceedings of the 2015 IEEE Symposium on Security and Privacy*. Los Alamitos, CA: IEEE Computer Society, 2015, 20–37.
- [18] Serebryany K. ARM Memory Tagging Extension and how it improves C/C++ memory safety. *;login:* 2019;**44**(2).
- [19] Klein G, Elphinstone K, Heiser G *et al*. seL4: formal verification of an OS kernel. In: *Proceedings of the 22nd ACM Symposium on Operating Systems Principles*. New York, NY: ACM, 2009, 207–20.
- [20] seL4 Foundation. seL4 security proofs now complete on AArch64. 24 August 2026.
<https://sel4.systems/news/2026.html> (2 October 2026, date last accessed).

- [21] Hu H, Shinde S, Adrian S *et al.* Data-oriented programming: on the expressiveness of non-control data attacks. In: *Proceedings of the 2016 IEEE Symposium on Security and Privacy*. Los Alamitos, CA: IEEE Computer Society, 2016, 969–86.
- [22] Beer I, Groß S. A deep dive into an NSO zero-click iMessage exploit: remote code execution. Google Project Zero, 15 December 2021.
<https://projectzero.google/2021/12/a-deep-dive-into-nso-zero-click.html> (2 October 2026, date last accessed).
- [23] Samuel J, Mathewson N, Cappos J *et al.* Survivable key compromise in software update systems. In: *Proceedings of the 17th ACM Conference on Computer and Communications Security*. New York, NY: ACM, 2010, 61–72.
- [24] Android Open Source Project. Android Verified Boot 2.0: updating stored rollback indexes.
<https://android.googlesource.com/platform/external/avb/+/-/master/README.md> (6 October 2026, date last accessed).
- [25] Laurie B, Langley A, Kasper E. Certificate Transparency. RFC 6962. Internet Engineering Task Force, 2013.
- [26] Nikitin K, Kokoris-Kogias E, Jovanovic P *et al.* CHAINIAC: proactive software-update transparency via collectively signed skipchains and verified builds. In: *Proceedings of the 26th USENIX Security Symposium*. Berkeley, CA: USENIX Association, 2017, 1271–87.
- [27] Newman Z, Meyers JS, Torres-Arias S. Sigstore: software signing for everybody. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY: ACM, 2022, 2353–67.
- [28] Oya S, Troncoso C, Pérez-González F. Do dummies pay off? Limits of dummy traffic protection in anonymous communications. In: *Privacy Enhancing Technologies (PETS 2014)*. Lecture Notes in Computer Science, Vol. 8555. Cham: Springer, 2014, 204–23.
- [29] Nasr M, Bahramali A, Houmansadr A. DeepCorr: strong flow correlation attacks on Tor using deep learning. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY: ACM, 2018, 1962–76.
- [30] Houmansadr A, Brubaker C, Shmatikov V. The parrot is dead: observing unobservable network communications. In: *Proceedings of the 2013 IEEE Symposium on Security and Privacy*. Los Alamitos, CA: IEEE Computer Society, 2013, 65–79.
- [31] Lamport L. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Boston, MA: Addison-Wesley, 2002.
- [32] Yu Y, Manolios P, Lamport L. Model checking TLA+ specifications. In: *Correct Hardware Design and Verification Methods (CHARME 1999)*. Lecture Notes in Computer Science, Vol. 1703. Berlin: Springer, 1999, 54–66.
- [33] Newcombe C, Rath T, Zhang F *et al.* How Amazon Web Services uses formal methods. *Commun ACM* 2015;**58**:66–73.