International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

A Survey on Trustworthy and Automated Digital Forensics: Hardware-Assisted Integrity, Domain-Specific Languages, and ATT&CK-Mapped Analysis

Sindhu S, Keerthana N., Adithya V., Devana M. D., Merlin M. S.
Department of Computer Science and Engineering (Cybersecurity)
Royal College of Engineering and Technology, Akkikkavu, Thrissur, Kerala, India

***Abstract* - Investigations today stretch across mixed endpoints, connected devices, compiled programs and hardware whose trustworthiness is often unclear, yet whatever is gathered from them must still stand up from the moment of acquisition to the moment of presentation. In practice, forensic workflows are usually stitched together from separate tools, platform-specific procedures and records kept by hand. This survey examines six recent studies (2025–2026) on hardware-security assessment, hardware Trojan detection, cryptographic hardware, third-party device trust, binary-code analysis and resource-aware password-recovery computation, together with 41 works drawn from their bibliographies. The studies are compared by problem, method, reported outcome and limitation, and the discussion is built around four themes: trustworthy hardware and supply-chain assurance, cryptographic support and roots of trust, behavioural and binary analysis, and automation through domain-specific languages (DSLs) with cross-platform execution. A gap recurs across the material. The techniques for individual layers are strong, but none of the studies shows one workflow that joins investigator-authenticated acquisition, interpretation and a tamper-evident evidence history. We therefore derive a six-layer reference architecture: a constrained, investigator-oriented DSL; a compiler that yields reviewable execution plans; lightweight endpoint agents; central correlation with ATT&CK-aligned interpretation; hash-chained provenance; and hardware-assisted investigator authentication. Block diagrams, a trust-boundary analysis, a threat model and an evaluation framework accompany it. The architecture is a design proposal, and no performance results are claimed.**

***Index Terms*—Digital forensics, evidence integrity, hardware root of trust, domain-specific language, compiler-assisted automation, behavioural analysis, binary analysis, MITRE ATT&CK, chain of custody, endpoint forensics.**

I. INTRODUCTION

A. Motivation

Digital evidence no longer sits neatly on a single computer. One case can involve endpoint activity, operating-system artefacts, network events, running programs, attached devices and the hardware beneath all of them, and not every one of these can be taken at face value. Whether an artefact is worth anything as evidence depends on much more than the fact that it could be collected. The investigator has to explain where it came from, how it was obtained, what was done to it afterwards, whether it has changed, and how the observations lead to the conclusion being drawn.

This creates an awkward tension in day-to-day practice. Some forensic tools dig deep into artefacts but are tied to one operating system or one kind of task. Monitoring platforms collect plenty of telemetry yet keep little in the way of forensic provenance. Cryptographic protection can reveal that something was altered later, but it cannot show that the original acquisition was complete or that the person carrying it out was entitled to do so. Machine-learning analysis can flag unusual patterns, though a flag is only useful when it points back to real source events and is read with care. Even the lowest level can be unreliable: bit errors in NAND flash make reading storage media harder [10], and recovering data from mobile devices can call for specialised in-system programming [25].

B. Problem Statement

We treat trustworthy automated forensics as an integration problem. Every trust dependency in a forensic workflow, whether platform hardware, cryptographic primitives, acquisition procedure, analysis method or evidence history, has

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

an active research community of its own. Those communities rarely meet around the question an investigator actually faces: can this evidence be defended, from the moment it was acquired to the moment it is presented? We put four questions to the recent literature.

- **RQ1.** What do recent studies of hardware security, cryptographic hardware and third-party devices tell us about the trust assumptions behind a forensic acquisition platform?
- **RQ2.** What can binary-code analysis and behavioural analysis add to forensic interpretation, and where does their reliability run out?
- **RQ3.** What would a constrained, reviewable, cross-platform workflow language have to offer before forensic procedures could be called reproducible?
- **RQ4.** Which gaps remain between these strands, and how might an integrated architecture close them in a way that can be evaluated honestly?

Our central argument is that trustworthy automation has to bring five concerns together: (i) authorised acquisition, (ii) reproducible execution of workflows, (iii) contextual analysis and correlation, (iv) verifiable evidence provenance, and (v) transparent accountability of the investigator.

C. Contributions

- The survey places six recent studies in one analytical frame that spans hardware trust, cryptographic support, binary and behavioural analysis, and workflow automation, instead of treating them as unrelated security topics.
- It sets out the limits of applying hardware assurance and automated analysis directly to forensic workflows, and it records what each study leaves unreported.
- It identifies five architectural gaps and derives a six-layer reference architecture from them, presented through block diagrams, a map from layers back to the literature, a data-flow description, a trust-boundary analysis and a threat model.
- It proposes an evaluation framework with technical, forensic and security measures and a scenario matrix, keeping design proposals clearly apart from empirical results.

D. Organisation

Section II introduces the background concepts. Section III explains the scope and method. Section IV reviews the six core studies and compares them, and Section V draws out the themes that emerge. Research gaps are summarised in Section VI, after which Section VII presents the integrated reference architecture and Section VIII the evaluation framework. Section IX covers limitations together with legal, ethical and privacy considerations, Section X points to future work, and Section XI concludes.

II. BACKGROUND AND TERMINOLOGY

A. Evidence Integrity, Provenance, and Chain of Custody

Integrity means that an artefact has not changed since some reference point. Provenance is the recorded history of the artefact: its source, what was done to it, and who was responsible. Chain of custody is the documented sequence of handling events that lets a reviewer follow the artefact from acquisition to presentation. A cryptographic digest supports integrity only relative to the moment it was computed. It says nothing about whether the source was genuine or the collection complete. Provenance and custody records therefore need acquisition metadata, the operator’s identity, tool versions, and a trustworthy entry for every handling event.

B. Hardware Trust and Roots of Trust

Software-level assurance sits on hardware that is assumed to behave as specified, and hardware-security research shows that this assumption is an attack surface in its own right. Integrated-circuit supply chains are global, and untrusted parties may influence design or fabrication stages [2], [36]. A root of trust is a component whose behaviour is assumed rather than checked at runtime; it typically offers protected key storage, random-number generation and cryptographic services. Existing designs range from processor-integrated security extensions [23] and secure microcontrollers [19] to dedicated root-of-trust blocks [45] and cryptographic accelerators [4], [9], [46]. They differ greatly in the assurance they give and in what they cost, a distinction that matters once a low-cost token is proposed for investigator authentication (Section VII-D).

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

C. Domain-Specific Languages and Compiler-Assisted Automation

A domain-specific language (DSL) offers a deliberately small vocabulary fitted to one problem area. For forensics, that would let an investigator state what should be collected or analysed, while a compiler works out how to do it on each platform. A compiler usually lowers source text through an intermediate representation (IR), a structured form that can be validated, inspected and transformed before any code or command is produced. Because the compiler and the agents that run its output become part of the evidence path, they add a new trust boundary that has to be controlled (Section V-D).

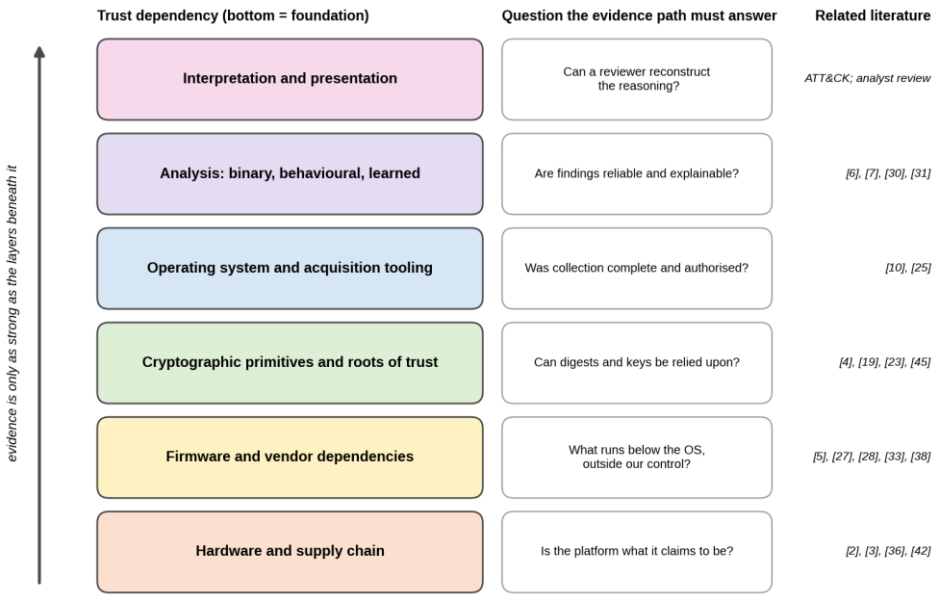
D. Behavioural and Binary Analysis, and ATT&CK

Behavioural analysis looks at what a program or user actually did: processes created, network connections made, files and registry entries changed. Binary analysis looks at the compiled code itself, often without source, using similarity, fingerprinting, and vulnerability or authorship analysis [6]. The MITRE ATT&CK knowledge base supplies a shared vocabulary of adversary tactics and techniques, which lets observations be grouped around behaviours such as execution, persistence, privilege escalation, discovery, lateral movement and command and control. Throughout this survey ATT&CK is only an analytical aid. A mapping to a technique is a hypothesis backed by evidence and is not proof that the technique took place.

E. Trust Dependencies in a Forensic Workflow

Fig. 1 lays out the dependencies that a forensic conclusion inherits. The foundation is the hardware and supply chain of the devices doing the acquisition and analysis. Above that come firmware and vendor services, cryptographic primitives, the operating system and acquisition tooling, the analysis methods, and finally interpretation and presentation. Each layer raises a question the evidence path must be able to answer, and the related literature differs from layer to layer. The figure also explains how this survey is organised: the six core studies speak mostly to the lower and middle layers, while the workflow-level questions near the top are the ones on which the reviewed studies give the least direct evidence.

Fig. 1. Trust dependencies inherited by a forensic conclusion, with the question each layer raises and the related literature.



Two points follow from this. First, a failure in a lower layer cannot be repaired by diligence in a higher one: a carefully documented analysis of data from a compromised acquisition platform is still weak evidence. Second, the dependencies are not all of one kind. Some, such as hardware provenance, are settled before an investigation begins and can only be documented. Others, such as the acquisition procedure and the evidence history, are produced during the investigation and can be recorded and verified as it goes. A trustworthy architecture ought to treat the two differently, documenting the first kind explicitly and recording the second kind automatically.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

III. SURVEY SCOPE AND METHODOLOGY

A. Core Corpus and Supporting Literature

The core corpus is made up of six research papers published in or around 2025–2026. One deals with hardware-security assessment [2], one with machine-learning-based hardware Trojan detection [3], one with a cryptographic accelerator tile [4], and one with third-party hardware in a trustworthy-AI deployment [5]. A systematic review of binary-code analysis [6] is included, as is one study of heuristic optimisation for WPA2 dictionary recovery [1]. The last of these is there only as a limited example of resource-aware forensic computation and is not central to the architecture discussed later. An earlier IoT malware-forensics paper from 2020 was left out so that the focus stays on recent work.

The reference list also holds 41 supporting works, [7]–[47], taken from the bibliographies of the core papers. They supply technical background and earlier work. We did not read each of them in full, and none is treated as an additional core study.

B. Selection Criteria

A study was considered relevant if it addressed at least one trust dependency of a forensic workflow: hardware and supply-chain integrity, cryptographic primitives and roots of trust, analysis of compiled or behavioural artefacts, or resource-aware investigative computation. Peer-reviewed journal articles with identifiable bibliographic records were preferred. The only exclusion on record is the 2020 IoT malware-forensics paper mentioned above, set aside because it is older.

C. Extraction and Synthesis Procedure

For every core paper we extracted the attributes listed in Table I. The extracted details were then sorted into four themes: (i) trustworthy hardware and supply-chain assurance, (ii) cryptographic hardware and evidence integrity, (iii) behavioural and binary analysis, and (iv) automation through structured workflows. Fig. 2 summarises the procedure. The architecture in Section VII comes from comparing these themes and the gaps between them, and it is not lifted from any single paper.

Fig. 2. Survey procedure: from the core corpus and supporting literature to themes, gaps, and the proposed architecture and evaluation framework.

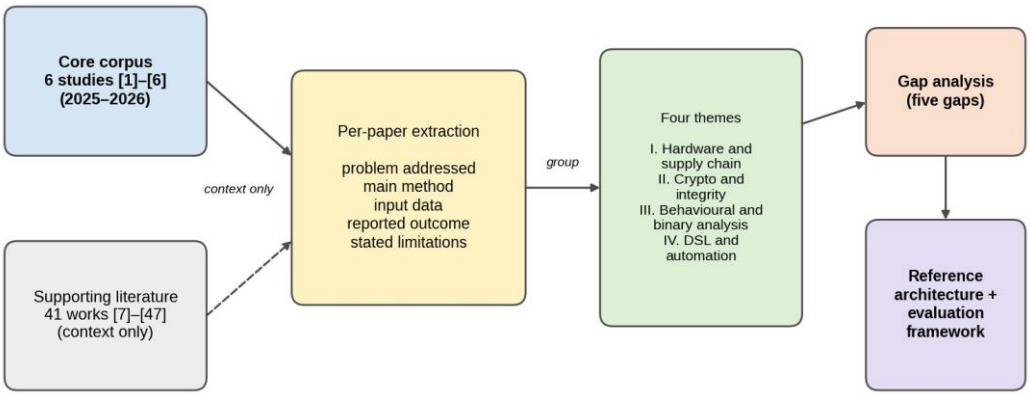


TABLE I. ATTRIBUTES EXTRACTED FROM EACH CORE STUDY

Attribute	What was recorded	Why it matters for the synthesis
Problem addressed	The question or threat the study targets	Shows which trust dependency the study speaks to
Main method	Technique, model or design contribution	Keeps detection methods apart from frameworks, reviews and designs
Input data	Dataset, case study or design artefact used	Sets the limits on how far results can be generalised
Reported outcome	Result as stated by the authors, attributed to them	Prevents inferred or pooled scores

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

Attribute	What was recorded	Why it matters for the synthesis
Stated or observable limitation	Limits the authors acknowledge or that are evident from the design	Feeds the gap analysis
Forensic relevance	Observation made by the survey authors	Kept distinct from reported results

Note: "Forensic relevance" is an observation we made while comparing the studies. It is not a claim made by the original papers.

D. Limitations of the Review

The review is thematic and is not a registered systematic review, so no claim to exhaustive coverage is made. The core corpus is small and bounded, which means the conclusions describe this corpus and not the whole field. We compute no pooled performance score, since the studies differ in task, dataset, threat model and evaluation method, and the numbers quoted belong to their original studies and cannot be compared directly. Throughout, we try to separate what an article reports from what we conclude by setting the articles side by side. Established endpoint forensic platforms fell outside the scope of this corpus, so placing the proposed architecture against them is left for future work.

IV. REVIEW OF CORE STUDIES

Each study is summarised below in a similar pattern: what it set out to do, what it reports, what that could mean for a forensic workflow, and where its limits lie. The statements about meaning for forensics are our own readings and should not be attributed to the original authors.

A. Hardware-Security Attack Assessment

Maragkou et al. [2] put forward an assessment model for real-world hardware-security attacks and stress how complicated the global integrated-circuit supply chain has become. Their contribution is to frame hardware trust as a matter of lifecycle and ecosystem rather than a narrow property of a chip. Related threat-modelling work on hardware supply-chain security [36] and the security analysis of open-source hardware design [39] point the same way.

For forensics, the framing bears directly on the assumptions behind acquisition devices, storage media and authentication components. A forensic platform takes on whatever supply-chain exposure its hardware carries, so its documentation should say which assumptions are made and which are actually verified. The study is an assessment framework and not a detection method; it gives no detection rates and does not evaluate any forensic application.

B. Hardware Trojan Detection in Open Designs

Hayashi and Ruggiero [3] bring natural-language processing and machine learning to hardware Trojan detection in open-source hardware designs. They work with established benchmark sources and synthetic examples, in a dataset of 3,808 instances, and report strong classification results for selected approaches. The paper belongs to a wider line of work that includes graph-neural-network detection [26], [47], topology-aware detection [37], RTL-level subgraph analysis [34], hardware reverse-engineering frameworks [29] and benchmark creation [41].

Screening designs automatically can add to design-time assurance, but it helps a forensic platform only indirectly, because finding suspicious logic in a design is not the same as validating the behaviour of an appliance already deployed. The limits of such approaches are well documented elsewhere. Reinforcement-learning attacks on static detectors [40], for instance, show that high benchmark scores do not guarantee that previously unseen modifications will be caught.

C. Cryptographic Accelerator Tile

Nannipieri et al. [4] describe Crypto-Tile, a suite of hardware accelerators developed within the European Processor Initiative. It covers symmetric and public-key cryptography, digest computation, random-number generation and post-quantum functions, along with system integration and key-management concerns. It shows how cryptographic services can be moved closer to the processor architecture, next to other hardware roots of trust [19], [23], [45] and secure crypto-processors [9], [46].

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

What this suggests for forensics is that the primitives an integrity layer needs (hashing, random numbers, key handling and signatures) can be delivered efficiently in hardware, and that post-quantum functions are being considered already at the design stage. The paper is a hardware-design contribution and does not look at forensic use. Work on hardware Trojans aimed at post-quantum schemes [22] is a useful reminder that cryptographic hardware is an attack surface too.

D. Third-Party Hardware as the Weakest Link

Lorenz et al. [5] use a smart-TV case study to show that a platform's security can be weakened by dependencies outside the primary vendor's control. The domain is consumer devices, but the lesson carries over: a device taking part in an evidence workflow may inherit risk from firmware, vendor services, embedded components and external integrations. Firmware-analysis research [27], [28], [33], [38] offers ways of inspecting such components.

Components that handle evidence should therefore avoid unnecessary external services, document the ones that remain, and keep a controlled offline mode. The study itself is a qualitative case study in a single domain. It offers a trust-boundary analysis and no quantitative results.

E. Binary-Code Analysis

Javed et al. [6] present a systematic review of binary-code analysis that covers binary similarity, fingerprinting, vulnerability analysis, clone detection, authorship attribution and anti-evasion strategies. Investigators often meet compiled artefacts with no source code, so these techniques are directly relevant. Methods cited in the review include neural function recognition [13], tensor and self-attentive embeddings for binary search and similarity [16], [31], simhash-based search [32], function fingerprinting [18], patch-semantic vulnerability verification [30] and symbolic analysis [14]. Surveys of vulnerability mining [44] add further context.

Binary characteristics can suggest that an executable resembles a known family or contains suspicious functionality, and that complements behavioural telemetry showing whether and how it ran. The weak point is evasion. Obfuscation, including control-flow obfuscation with self-modifying code [7], degrades these tools, and learning-based classifiers raise robustness concerns [12], [15], [21]. The review reports on the techniques in the field and does not propose a forensic workflow.

F. Resource-Aware Password Recovery

Chalyi [1] evaluates heuristic ordering strategies for offline WPA2 dictionary recovery, looking at how preprocessing cost, dictionary structure and hardware throughput relate to one another. WPA2 security rests on the IEEE 802.11i amendment [24], and studies of password patterns [43] inform how dictionaries are designed. The topic is peripheral to the architecture discussed here.

Its value for us is methodological. Computationally heavy investigative tasks need resource-aware scheduling, transparent recording of parameters and clear boundaries of authorisation. The study does not establish any general advantage for forensic workloads of other kinds.

G. Comparative Summary

Table II summarises the studies. Metrics, datasets and evaluation protocols differ from paper to paper, so the table is descriptive and is not a ranking. Table III then restates each study in terms of the forensic evidence path, separating what it contributes from what it does not demonstrate.

TABLE II. COMPARISON OF THE SIX CORE STUDIES

Study	Focus	Main contribution	Reported outcome	Main limitation
[1] Chalyi (2026)	WPA2 dictionary recovery	Heuristic ordering framework	Hardware-aware ordering analysis	Peripheral to forensic workflows; not general
[2] Maragkou et al. (2025)	Hardware-security attacks	Assessment model for real-world attacks	Assessment framework (no detection metric)	Not a detection or mitigation method

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

Study	Focus	Main contribution	Reported outcome	Main limitation
[3] Hayashi & Ruggiero (2025)	Hardware Trojans in open designs	NLP and ML classification	Strong results on a 3,808-instance dataset	Benchmark scores may not transfer to unseen Trojans
[4] Nannipieri et al. (2025)	Cryptographic hardware	Crypto-Tile accelerator suite	Hardware design and integration	No forensic evaluation
[5] Lorenz et al. (2025)	Third-party hardware	Smart-TV case study	Qualitative trust-boundary analysis	Single-domain case study
[6] Javed et al. (2025)	Binary-code analysis	Systematic review of techniques	Taxonomy of techniques and anti-evasion	Review only; no workflow evaluation

Note: results are attributed to the cited papers and are not directly comparable.

TABLE III. RELEVANCE OF EACH STUDY TO THE FORENSIC EVIDENCE PATH (ANALYTICAL)

Study	Evidence-path element	Transferable lesson	Not demonstrated
[2]	Platform and supply-chain trust	Trust assumptions behind acquisition devices, storage media and authentication components should be stated openly	Detection or mitigation; any forensic application
[3]	Design-time assurance	Automated screening can support assurance, though only indirectly	Validation of a deployed forensic appliance; detection of unseen Trojans
[4]	Integrity and key services	Hashing, random numbers, key management and post-quantum functions can be supplied close to the processor	Use inside an evidence workflow
[5]	External dependencies	Devices inherit risk from firmware, vendor services and embedded components	Generality beyond one consumer-device case
[6]	Compiled artefacts	Similarity, fingerprinting and anti-evasion techniques address binaries without source	A forensic workflow; robustness to obfuscation
[1]	Investigative computation	Resource-aware scheduling, parameter recording and authorisation boundaries	Any general advantage for other workloads

H. Supporting Literature

The 41 supporting works are grouped by role in Table IV. We used them to understand the setting of the core studies and to qualify their limits. None of them counts as additional core evidence.

TABLE IV. SUPPORTING LITERATURE GROUPED BY ROLE

Role	Works	Use in this survey
Hardware Trojan detection, benchmarks and reverse engineering	[11], [22], [26], [29], [34], [37], [39], [40], [41], [47]	Methods and limits of design-level assurance; adversarial evasion of static detectors; Trojans targeting post-quantum schemes
Supply-chain and security management	[8], [35], [36], [42]	Organisational documentation of trust assumptions; threat modelling; root of trust over untrusted chiplets
Secure hardware and cryptographic processors	[9], [19], [23], [45], [46]	Levels of assurance available from dedicated secure hardware, and the gap to low-cost tokens
Firmware analysis	[27], [28], [33], [38]	Ways of inspecting the firmware dependencies of devices in an evidence workflow
Binary similarity, fingerprinting and vulnerability analysis	[13], [14], [16], [18], [30], [31], [32], [44]	Techniques and limits of analysing compiled artefacts
Obfuscation and robustness of learning-based classifiers	[7], [12], [15], [21]	Reasons why automated binary and malware analysis may be unreliable under evasion
Software hardening and autonomous defence	[17], [20]	Context for automating security tasks; no forensic evaluation
Forensic acquisition	[10], [25]	Examples of low-level acquisition difficulty (NAND bit errors; mobile in-system programming)
Wireless security and passwords	[24], [43]	Background for resource-aware password recovery

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

V. THEMATIC ANALYSIS

A. Trustworthy Hardware and Supply-Chain Assurance

A forensic platform relies on the integrity of the devices that acquire, store and process data. If a component is compromised before an investigation starts, software-level logging alone may not say enough about where the collected data came from or whether it is complete [2]. This does not mean every workstation needs specialised hardware inspection. It means the trust assumptions behind acquisition devices, endpoint agents, storage media and authentication components should be explicit. Procurement provenance, firmware update controls, component identity, secure configuration and auditable maintenance all bear on how reliable the evidence path is, and a command that returns successfully should never be read as proof that the platform underneath was clean. Management-system standards such as ISO/IEC 27001 [35], along with general security-management practice [8], give organisational context for writing such assumptions down.

Automated hardware inspection [3] supports this goal only indirectly, since spotting suspicious logic in a design says little about how a deployed forensic appliance behaves. A practical architecture should pair design-time assurance with runtime identity, measured configuration, update provenance and independent checks on the evidence. Third-party dependencies [5] strengthen the case for cutting unnecessary external services, documenting those that stay, and keeping a controlled offline mode. Trust has to be established at each boundary and not assumed from a product label or a vendor's reputation; proposals for a chiplet-level root of trust over untrusted components [42] illustrate the same principle at system level. Three practical consequences follow for the design of a forensic workflow.

- **Make assumptions explicit.** Record which hardware and firmware the platform relies on, and what is known about where they came from.
- **Prefer runtime identity to labels.** Component and agent identity should be checkable while the system runs and should not be inferred from a vendor name.
- **Limit external dependencies.** Acquisition and evidence handling should not need a vendor cloud service, and an offline mode should exist.

B. Cryptographic Hardware and Evidence Integrity

Hashing is a practical way to detect changes to acquired artefacts. A digest computed at acquisition can be checked again during transfer, storage, analysis or disclosure, and accelerators such as Crypto-Tile [4] show that these primitives can be supplied efficiently. A matching digest, though, only establishes consistency with the data that was hashed. It does not show that the source was genuine, that the acquisition was complete, or that the operator followed a valid procedure. Integrity therefore has to rest on further support: acquisition metadata, trusted timestamps where they exist, operator identity, tool versions and a documented chain of custody.

A useful evidence record ties the artefact to the context of its acquisition: case identifier, device identity, collection scope, acquisition time, operator identity, tool and agent version, digest, and every handling event that follows. In a chained metadata structure each event refers to the digest of the one before it, which makes unauthorised changes easier to see. The chain is only as reliable as the protection of its keys, the process that generates its events and its access controls. An attacker who can rewrite the whole record, or who gets hold of a signing key, defeats a hash chain on its own. Several decisions are left open, including key provisioning, recovery, secure storage, clock trust, access revocation and independent verification.

C. Behavioural and Binary Analysis

Forensic interpretation often means weighing several weak observations together. A process event, a network connection or a file modification may each look ordinary alone, but when they are correlated by host, time, process lineage, user context and related artefacts they can form a meaningful sequence. That is the argument for event correlation and attack reconstruction over reliance on a single alert. Correlated results should keep the original events

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

and their provenance, and derived conclusions should be stored apart from the source evidence so a reviewer can tell observed facts from interpretation.

Binary analysis [6] complements behavioural telemetry. The characteristics of a binary may suggest that an executable resembles a known family or has suspicious functionality, while endpoint events show whether and how it was run. The limitations deserve equal attention. Similarity does not necessarily mean common authorship or malicious intent. Compiler optimisation, function inlining, library reuse and build configuration all affect similarity [30], [31], and anti-analysis behaviour [7] can lower the reliability of automated tools. Learning-based malware classifiers face related robustness concerns [12], [15], [21]. A forensic workflow should therefore keep tool versions and analysis parameters, record confidence and uncertainty, and allow manual validation of conclusions with high impact. Computationally heavy tasks such as dictionary-based recovery [1] call, in addition, for resource-aware scheduling and explicit authorisation boundaries.

D. Domain-Specific Languages and Compiler-Assisted Workflows

In cross-platform investigations the same question needs different commands and artefact locations on each operating system. Asking investigators to remember platform-specific syntax makes inconsistent collection more likely and workflows harder to reproduce. A DSL can give them a higher-level vocabulary for investigative intent, with a compiler translating that intent into platform-specific operations. The direction fits the wider interest in autonomous and automated cyber defence [20], although none of the reviewed core papers evaluates it for forensics.

To be useful forensically, the language has to be constrained. Its commands should express permitted collection and analysis operations and not arbitrary execution, and a workflow definition should declare its scope, target, required privileges, expected output and evidence-handling behaviour. The compiler should check syntax and semantics, reject unsupported or unsafe operations, and produce a traceable execution plan. A readable IR lets reviewers see what the system intends to do before any agent acts. When a platform cannot supply a requested artefact, the system should report an explicit limitation and not quietly substitute another source.

A compiled workflow should also yield an execution manifest listing the workflow, compiler and agent versions, the target platform, parameters, start and completion times, privileges used and outcomes. The manifest should tell apart successful collection, partial collection, permission denial, unsupported artefacts and execution failure. Since the DSL creates a new trust boundary, investigators have to be able to trust the compiler and the agent, which makes compiler testing, signed releases, least-privilege execution, code review and validation against known forensic test cases essential. Automation can improve consistency, but it can just as easily repeat a flawed procedure at scale. The aim is controlled and reviewable automation.

E. ATT&CK-Mapped Interpretation

The MITRE ATT&CK knowledge base gives a common vocabulary for adversary tactics and techniques. Mapping observed events onto it helps organise evidence around behaviours such as execution, persistence, privilege escalation, discovery, lateral movement and command and control. The mapping is an analytical aid and not proof that a technique occurred. A sound implementation keeps the raw telemetry, the rule or model that proposed the mapping, the ATT&CK version used, and the reasoning that links observations to the technique. It should also allow several hypotheses when evidence is incomplete, which matters most where legitimate administrative activity resembles adversarial behaviour.

VI. RESEARCH GAPS

The reviewed literature supplies useful building blocks, but its coverage is uneven. Hardware-security research concentrates on device and supply-chain trust, cryptographic-hardware research provides efficient primitives, binary-analysis research addresses compiled artefacts and anti-evasion, and studies of third-party devices show how trust can be weakened by outside components. Not one of the six core studies describes, on its own, a complete workflow that runs from investigator-authenticated acquisition through cross-platform execution, event correlation and ATT&CK

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

interpretation to chained evidence provenance. Five gaps follow, and Table V relates each to its supporting evidence and to the architectural layer that responds to it.

- **G1 Reproducible workflows.** Platform-specific forensic procedures are still hard to express and reproduce consistently across mixed endpoints.
- **G2 Integrity bound to procedure.** Integrity mechanisms tend to be discussed apart from the acquisition procedure and the identity of the operator.
- **G3 Explainable findings.** Automated detection and binary analysis can produce findings that are hard to interpret unless they are linked to original evidence and explicit reasoning.
- **G4 Hardware trust in the forensic chain.** Hardware trust is usually considered at the level of components or supply chains and not as part of a full forensic chain.
- **G5 Comparable evaluation.** The studies use different datasets and metrics, so reported performance cannot be compared directly.

TABLE V. RESEARCH GAPS, SUPPORTING EVIDENCE, AND ARCHITECTURAL RESPONSE

Gap	Where it shows in the reviewed work	Reasoning	Responding layer
G1	No core study evaluates a DSL or compiler for forensics; automation appears only as general interest in autonomous cyber defence [20]	Reproducibility needs a shared, constrained vocabulary and a reviewable plan	L2, L3, L4
G2	Hashing and cryptographic hardware [4] are treated as primitives; a digest shows consistency only	Integrity means something only when it is bound to operator identity, scope and tool version	L1, L4, L6
G3	Binary analysis [6] and ML detection [3] give findings whose reliability depends on tooling and data [30], [31], [40]	Findings need pointers to source events, parameters and a stated uncertainty	L5, L6
G4	Hardware-trust studies [2], [5] address components and ecosystems, not the evidence path	Trust assumptions of tokens, agents and storage must be explicit and checked	L1, cross-cutting
G5	Different tasks, datasets and metrics across all six studies	A shared plan and metric definitions are needed before comparison is meaningful	Section VIII

VII. AN INTEGRATED REFERENCE ARCHITECTURE

A. Design Principles and Assumptions

The architecture is derived from the gaps in Section VI and from the cautions recorded in Section V. Six principles guide it.

- **Authorise before acting.** Nothing is collected without an authenticated investigator, a bound case and session, and a plan validated against that investigator's authority.
- **Constrain, then review.** Workflows are written in a restricted language and compiled to a human-readable IR that is reviewed before anything runs.
- **Least privilege on endpoints.** Agents carry out only the operations in a signed plan, with the minimum privileges those operations need.
- **Record procedure with evidence.** Each artefact is bound to the manifest of the execution that produced it.
- **Separate facts from interpretation.** Original artefacts, normalised telemetry, derived indicators, mappings and conclusions are stored as distinct but linked layers.
- **State limitations explicitly.** Unsupported artefacts, partial collection and uncertain mappings are reported and never hidden or silently replaced.

Four assumptions also need to be stated openly: that compiler and agent releases can be signed and verified; that investigators can be enrolled with a credential and later revoked; that a time source of known quality is available, or that

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

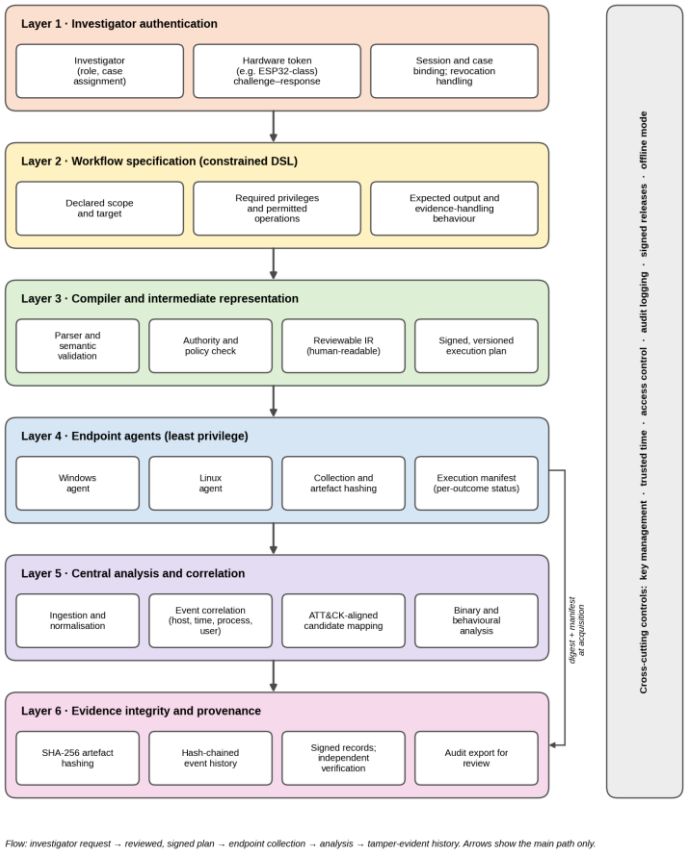
its absence is recorded; and that an independent party can hold a checkpoint of the evidence chain. If any of these fails, the architecture falls back to weaker guarantees, and the manifest and records should show that.

B. Overview

Building on the gaps above, we outline an integrated, hardware-assisted reference architecture for trustworthy automated forensics. It has six cooperating layers: (1) investigator authentication; (2) workflow specification in a constrained DSL; (3) a compiler and intermediate representation; (4) platform-specific endpoint agents; (5) centralised evidence analysis and correlation; and (6) evidence integrity and provenance management. Fig. 3 shows the layers and the main path of a request.

The layers are meant to form a continuous, reviewable path from the investigator's request to the resulting evidence and analytical output. A set of cross-cutting controls supports every layer: key management, trusted time, access control, audit logging, signed releases and an offline mode. A side path carries the artefact digest and execution manifest straight from the endpoint to the provenance layer at the moment of acquisition, so that integrity evidence does not depend on later analysis stages.

Fig. 3. Six-layer reference architecture for trustworthy automated forensics (design proposal).



C. Traceability to the Reviewed Literature

Fig. 4 maps each core study to the layers it informs. The mapping is our own interpretation, and the papers do not state it. Hardware-assessment work [2] and third-party hardware analysis [5] inform the cross-cutting trust assumptions and the design of endpoint components. Crypto-Tile [4] informs the level of assurance expected of the token and the integrity primitives, while binary-code analysis [6] informs the analysis layer. Hardware Trojan detection [3] contributes design-time assurance at the cross-cutting level, and the resource-aware recovery study [1] adds the requirement that heavy tasks be scheduled, parameterised and authorised. The DSL layer (L2) has no direct counterpart among the six core studies, which is itself one of the gaps this survey identifies.

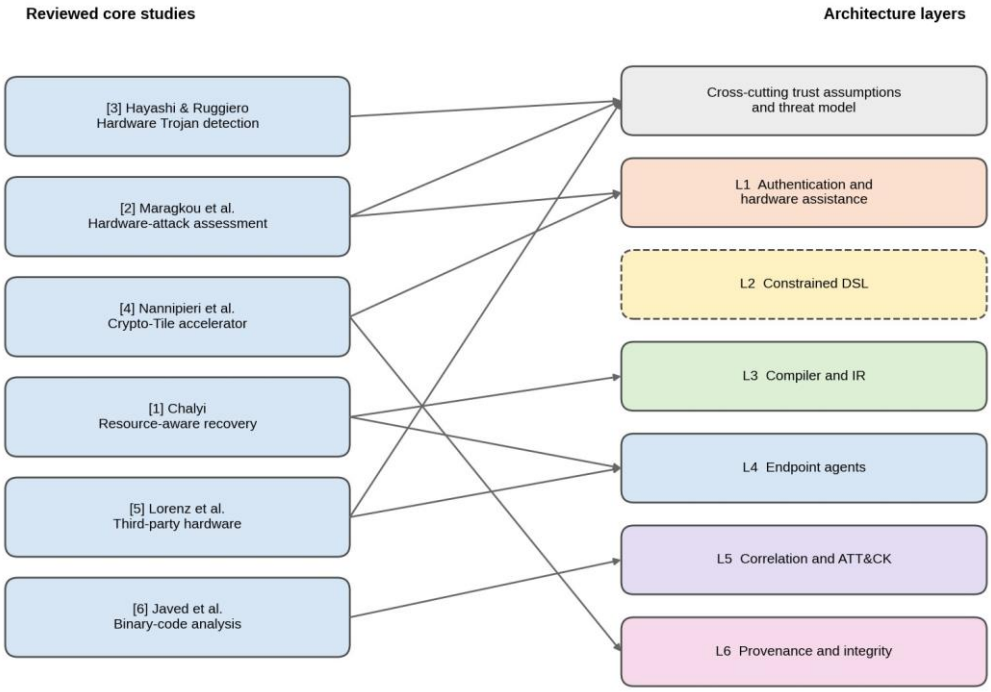
(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

D. Layer 1: Investigator Authentication and Hardware Assistance

A low-cost, microcontroller-based hardware security module, for example an ESP32-class device, could authenticate investigators and identify investigation sessions, adding a physical trust element to the workflow. Such a design must make clear how credentials are stored or derived, how the device talks to the host, how authentication is tied to a case and session, and how loss or revocation is handled. A hardware token should not be treated as sufficient proof of identity by itself. It should work inside a broader authentication and authorisation process. The roots-of-trust literature [4], [19], [23], [45] indicates the level of assurance that dedicated secure hardware can offer, and also the gap between that hardware and a low-cost token.

Fig. 4. Analytical mapping from the six core studies to the architecture layers.



Layer L2 (DSL) has no direct counterpart among the six core studies; [20] gives only general context. Mapping is analytical, not stated by the papers.

Four questions define the design space for this layer and should be answered before any prototype is evaluated. On *credential protection*: can the credential be extracted from a low-cost device, and what protection does the chosen microcontroller offer? On *session binding*: is each request bound to a case and session so that a captured exchange cannot be replayed? On *revocation*: how is a lost or retired token invalidated, and how do records already signed remain verifiable afterwards? On *fallback*: what happens when the token is unavailable during an urgent investigation, and how is that exception recorded?

E. Layers 2 and 3: Constrained DSL, Compiler, and Intermediate Representation

The DSL gives investigators a vocabulary of intent. Its constraint is the whole point: it expresses permitted collection and analysis operations and not arbitrary execution. Table VI lists the design requirements derived from Section V-D, each with a rationale and a candidate mechanism.

TABLE VI. DESIGN REQUIREMENTS FOR THE INVESTIGATOR-ORIENTED DSL

Requirement	Rationale	Candidate mechanism
Constrained vocabulary	Workflows should express permitted collection and analysis, not arbitrary execution	Closed set of named operations; no free-form shell access
Declared scope and target	Reproducibility and avoidance of over-collection	Mandatory case, target and scope fields checked at compile time

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

Requirement	Rationale	Candidate mechanism
Declared privileges	Least privilege and auditability	Each operation annotated with the privilege it needs; compiler rejects requests beyond the investigator's authority
Explicit limitations	No silent substitution of one artefact source for another	Unsupported artefact reported in IR and manifest; behaviour on unsupported artefacts declared by the workflow
Reviewable IR	Reviewers should see intended actions before execution	Readable IR generated deterministically from source and compiler version
Versioned and signed	The compiler is a new trust boundary	Signed compiler releases; plan pins compiler and knowledge-base versions
Resource declarations	Heavy tasks need scheduling and authorisation [1]	Optional block declaring resource limits and parameters, recorded in the manifest

Fig. 5 shows the compilation pipeline. Source text is parsed, checked semantically and checked against the investigator's authority, and any failure returns an explicit error with nothing executed. An accepted workflow becomes a reviewable IR, is lowered to platform back-ends and is packaged as a signed execution plan that agents verify before acting. Listing 1 gives an illustrative workflow and Listing 2 the corresponding IR. Both are design examples and do not describe an implemented language.

Listing 1. Illustrative DSL workflow (design example).

```
workflow "persistence-triage" version "1.0" {
  case "CASE-2026-014";
  target host("WS-017");
  platforms [windows, linux];
  privilege read_only;
  collect process_list, network_connections;
  collect persistence_points;
  collect event_logs since "2026-10-01T00:00Z";
  on_unsupported report;   # never substitute silently
  analyse attack_map(kb_version = "pinned");
  store sha256, chain = true;
}
```

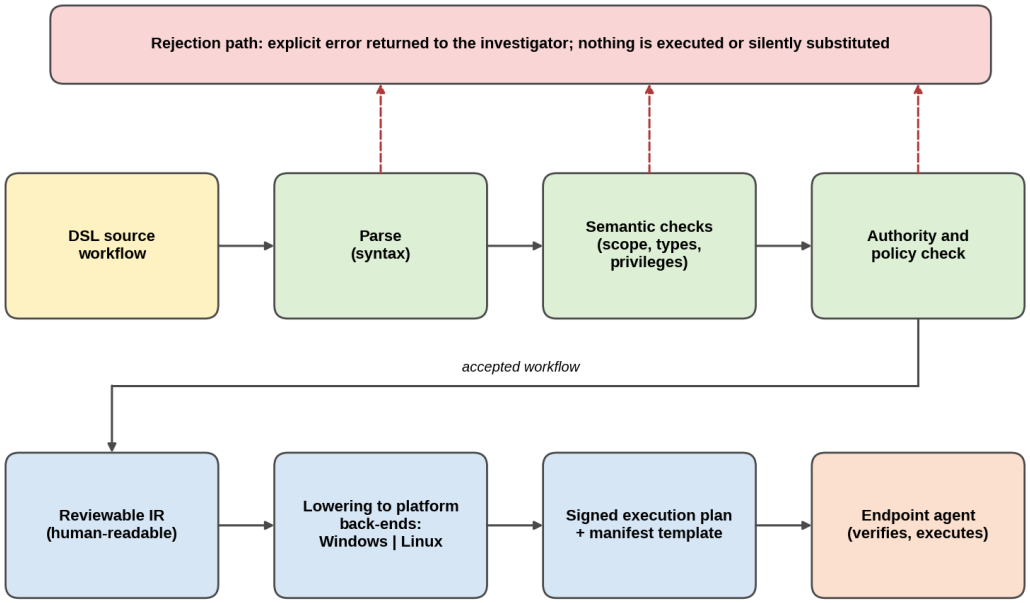
Listing 2. Corresponding reviewable IR (illustrative; symbolic fields in angle brackets).

```
PLAN id=<plan-id> wf=persistence-triage@1.0 compiler=<ver> kb=<ver>
SCOPE case=CASE-2026-014 target=WS-017 privilege=read_only
STEP 1 op=process_list         windows -> <backend op>  linux -> <backend op>
STEP 2 op=network_connections  windows -> <backend op>  linux -> <backend op>
STEP 3 op=persistence_points   windows -> <backend op>  linux -> <backend op>
STEP 4 op=event_logs           windows -> <backend op>  linux -> <backend op>
LIMIT none declared
OUTPUT store=sha256 chain=true manifest=required
SIGN <compiler signature> <investigator countersignature>
```


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

Fig. 5. Compilation pipeline from workflow source to a signed execution plan, with the rejection path.



Investigator reviews the IR before the signed plan is released; agents refuse unsigned or altered plans.

The value of a shared intent is that one question maps to different sources on different systems. Table VII illustrates the kind of lowering involved. The entries are illustrative and not exhaustive, and they also show why the compiler must report differences instead of hiding them: process lineage, connection attribution and timestamp semantics are not identical across platforms.

TABLE VII. ILLUSTRATIVE LOWERING OF INVESTIGATIVE INTENT TO PLATFORM SOURCES

Intent	Windows source (examples)	Linux source (examples)	Difference to record
process_list	Process enumeration through operating-system interfaces; process-creation events where logging is enabled	Enumeration of the process file system; process accounting where enabled	Availability of parent–child lineage differs
network_connections	TCP and UDP connection tables	Kernel socket tables and socket-statistics tools	Process attribution depends on privilege
persistence_points	Scheduled tasks, services and start-up registry entries	Scheduled jobs, service-manager units and shell start-up files	Mechanisms differ; mapped to one persistence intent
event_logs	Native event logs	System journal and log files	Schemas differ; normalisation required
file_metadata	File-system timestamps and attributes of the Windows file system	Timestamps and attributes of the Linux file system in use	File-system semantics differ

Note: entries illustrate the lowering concept. A prototype would define each back-end operation precisely and test it against known cases.

F. Layer 4: Endpoint Agents and the Execution Manifest

Endpoint agents are kept deliberately lightweight. An agent verifies the signature and version of a plan, executes only the listed operations with the declared privileges, hashes every collected artefact at acquisition and produces an execution manifest. The manifest records the workflow, compiler and agent versions, the target platform, parameters, start and completion times, privileges used and outcomes. Table VIII defines the outcome states, which make the difference between a complete and an incomplete acquisition visible and not merely implied.

TABLE VIII. EXECUTION-MANIFEST OUTCOME STATES

Status	Meaning	Detail that must be recorded
Complete	The requested artefact was collected in full	Digest, size and source identifier
Partial	Collection succeeded only in part	What is missing, why, and counts of items or bytes collected

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

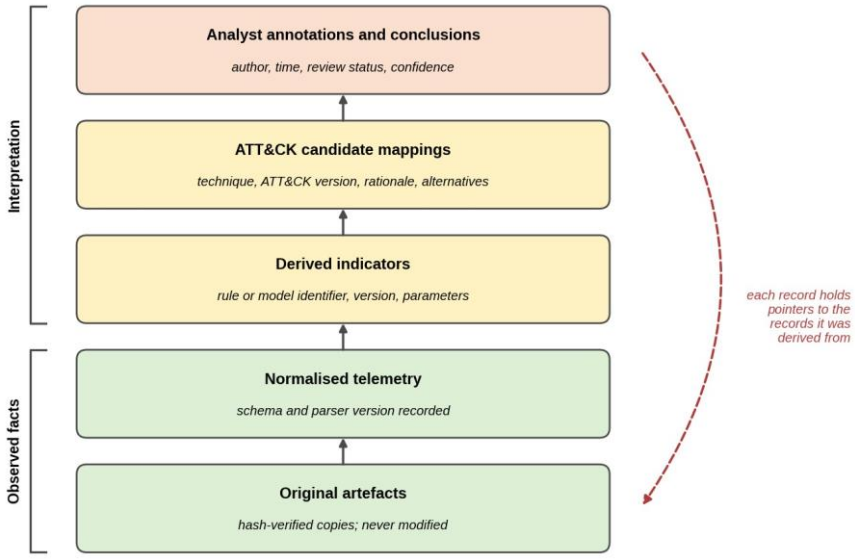
Status	Meaning	Detail that must be recorded
Permission denied	The operating system refused the operation	Privilege used, operation and error code
Unsupported artefact	The artefact does not exist on this platform or version	Platform and version; explicit limitation, no substitute source
Execution failure	The agent or a tool failed or was interrupted	Stage, error, and whether a retry occurred
Plan rejected	The agent refused the plan	Reason, such as signature failure or version mismatch

Two further design points follow from the third-party-dependency lesson [5]. The agent should not depend on external services to perform collection, and an offline mode should let plans and results be exchanged without network access. The endpoint under investigation may itself be altered or hostile, so the agent's results are treated as claims whose consistency is checked and not as ground truth. This is reflected in the trust-boundary analysis of Section VII-J.

G. Layer 5: Central Analysis, Correlation, and ATT&CK-Aligned Interpretation

The central analysis component correlates endpoint events (process, network, event-log, persistence, file and driver-related artefacts) by host, time, process lineage, user context and related artefacts, and it supports ATT&CK-aligned investigation. Binary analysis results [6] can be attached to the executables that events refer to. Its output should keep original artefacts, normalised telemetry, derived indicators, analyst annotations and system-generated conclusions separate, so that an independent reviewer can reconstruct the analytical path. Fig. 6 shows the separation: the lower two layers hold observed facts, the upper three hold interpretation, and each record keeps pointers to the records it was derived from.

Fig. 6. Separation of observed facts from interpretation, with pointers from every derived record back to its sources.



A candidate ATT&CK mapping should carry the elements listed in Table IX. Alternatives matter here. Where legitimate administrative activity looks like adversarial behaviour, the system should keep more than one hypothesis, mark its confidence, and leave high-impact conclusions to human review.

TABLE IX. ELEMENTS OF A CANDIDATE ATT&CK MAPPING RECORD

Element	Content	Purpose
Source events	Identifiers of the normalised events that support the mapping	Lets a reviewer trace a claim to evidence
Technique candidate	ATT&CK tactic and technique identifier	Common vocabulary for reporting
Knowledge-base version	The ATT&CK version used	Reproducibility as the knowledge base changes
Rule or model	Identifier, version and parameters of what produced the mapping	Explainability and auditability
Reasoning	Statement linking the observations to the technique	Distinguishes inference from observation

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

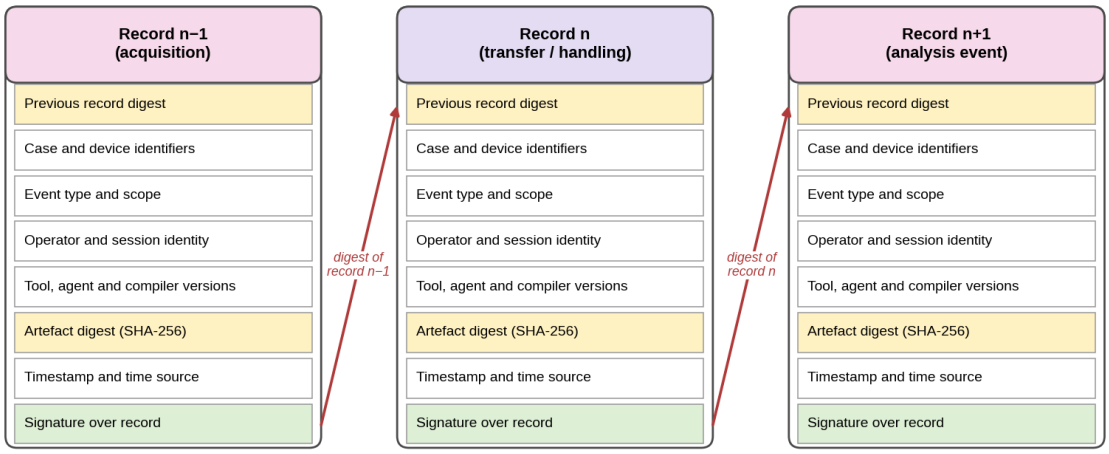
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

Element	Content	Purpose
Alternative hypotheses	Other explanations, including legitimate administration	Avoids over-claiming from ambiguous activity
Confidence and review status	Stated confidence and whether an analyst has reviewed it	Marks which results are provisional

H. Layer 6: Evidence Integrity and Provenance

The integrity layer would use SHA-256 artefact hashing and chained metadata to make changes to evidence detectable, with the signing and verification considerations discussed in Section V-B. Fig. 7 shows a chained record structure. Each record carries the digest of its predecessor, so altering an earlier record invalidates every later link, and each record is signed. Table X lists the fields together with the trust dependency of each, which is the part most often overlooked: a field is only as trustworthy as the process that produced it.

Fig. 7. Hash-chained evidence records with signatures and an independent verification step.



Verification: recompute each record digest from its content, check the link to its predecessor and the signature, and compare the chain head with an independently held checkpoint. A hash chain alone does not stop an attacker who can rewrite the whole chain or who holds the signing key.

TABLE X. FIELDS OF AN EVIDENCE RECORD AND THEIR TRUST DEPENDENCIES

Field	Purpose	Depends on
Case identifier	Binds the record to an investigation	An authorised case registry
Device identity	Identifies the source system	Enrolment or attestation; strength depends on the platform
Collection scope	States what was requested	The signed execution plan
Acquisition time and time source	Ordering and timing of events	A trusted time source; clock discrepancies must be flagged
Operator identity and session	Accountability for the action	The authentication process, including the token and its revocation
Tool, agent and compiler versions	Reproducibility	Signed releases
Artefact digest (SHA-256)	Detects later change to the artefact	The hash primitive; shows consistency only, not authenticity
Previous record digest	Links the record into the chain	Integrity of the chain and its storage
Signature	Authenticity of the record	Protection of the signing key
Handling events	Transfer, access, disclosure and analysis steps	The event-generation process and access controls

A hash chain makes tampering *visible*, but it does not make it *impossible*. An attacker who can rewrite the whole chain, or who holds a signing key, can produce a consistent forgery. The design therefore adds three mitigations:

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

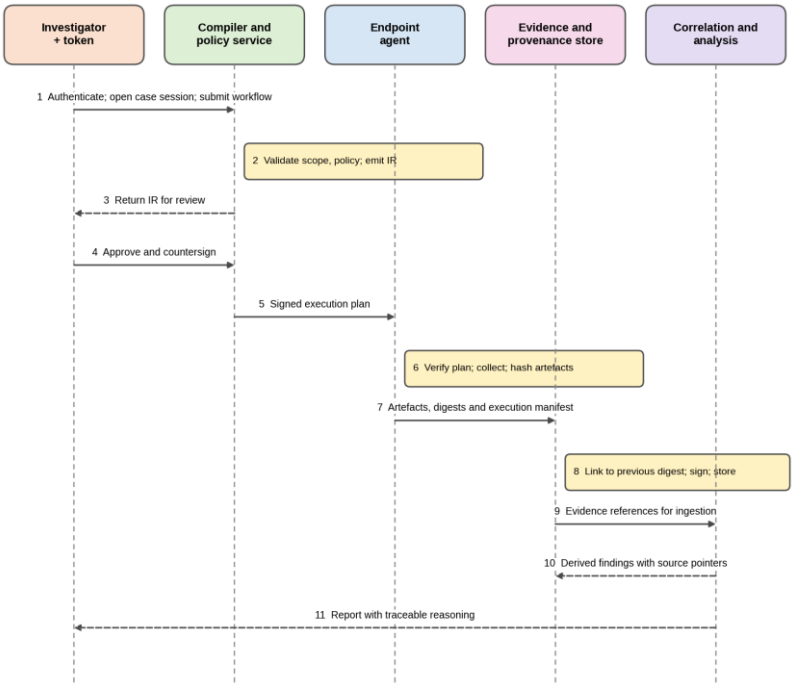
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

signatures on each record, protection of signing keys in secure hardware where practicable [4], [45], and periodic checkpoints of the chain head held by an independent party so that wholesale replacement can be detected. The remaining open decisions (key provisioning, recovery, clock trust and revocation) should be settled in the prototype and tested under the scenarios in Section VIII.

I. End-to-End Workflow

Fig. 8 follows one investigation across the five principal actors. The investigator authenticates with the token, opens a case session and submits a workflow (steps 1–2). The compiler validates it, emits the IR and returns it for review, and the investigator approves and countersigns (steps 3–4). A signed plan goes to the endpoint agent, which verifies it, collects and hashes the artefacts (steps 5–6). Artefacts, digests and the execution manifest then pass to the provenance store, where the record is linked to its predecessor, signed and stored (steps 7–8). Evidence references are handed to the analysis component, whose derived findings are stored separately with source pointers (steps 9–10), and a report with traceable reasoning returns to the investigator (step 11).

Fig. 8. End-to-end sequence from authenticated request to a report with traceable reasoning.



J. Trust Boundaries and Threat Model

Fig. 9 divides the architecture into four zones and five boundaries. The endpoint under investigation (Z3) is drawn with a dashed outline because it may be altered or hostile, and its local artefacts are therefore treated with the most suspicion. Boundary B5 stands for the hardware, firmware and third-party components beneath every zone, which is the supply-chain exposure described in [2] and [5]. Table XI lists twelve threats with the zones and layers they affect, the design mitigation and the residual risk. The residual column is deliberate, because a mitigation reduces a risk and only rarely removes it.

TABLE XI. THREAT MODEL FOR THE REFERENCE ARCHITECTURE

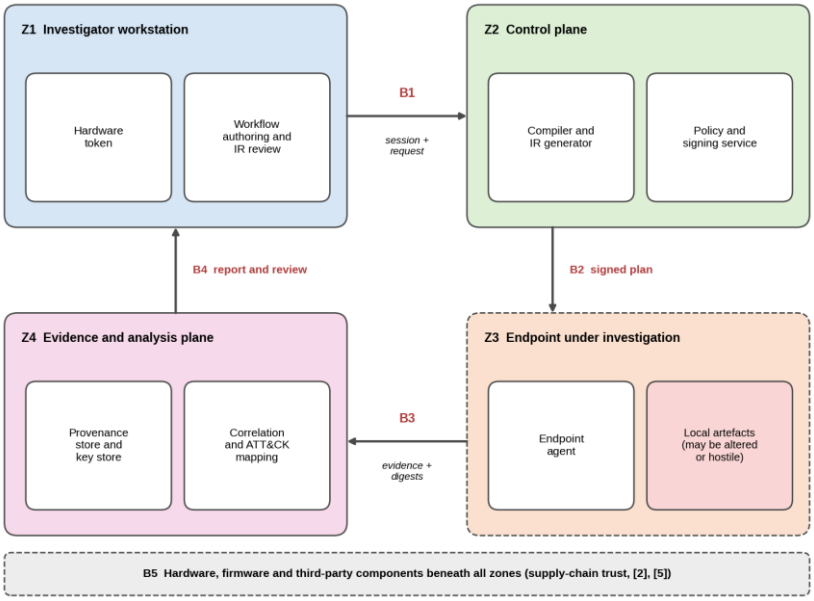
ID	Threat	Where	Design mitigation	Residual risk
T1	Compromised or substituted endpoint agent	Z3, L4	Signed agent releases; agent identity checked; manifest consistency checks	A deeply compromised endpoint can still misreport what it sees
T2	Lost or stolen investigator token	Z1, L1	Session binding; revocation; token only one factor in broader authentication	Low-cost token may resist extraction poorly [4], [19], [23], [45]

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

ID	Threat	Where	Design mitigation	Residual risk
T3	Over-collection through an overly broad workflow	L2, L3	Mandatory scope and target; compiler checks against authority; data minimisation	Authority definitions may themselves be too broad
T4	Compromised compiler or IR generator	Z2, L3	Signed releases; compiler testing; reviewable IR; investigator countersignature	Subtle miscompilation may pass review
T5	Tampering with stored evidence or records	Z4, L6	Hash chain; record signatures; independent checkpoints	Whole-chain rewrite if checkpoints are absent
T6	Signing-key compromise	Z2, Z4	Key protection in secure hardware where practicable; rotation and revocation	Records signed before detection remain suspect
T7	Clock manipulation	Z3, L6	Trusted time source; clock discrepancies flagged	Absent trusted time weakens ordering claims
T8	Replay or duplicate submission	B2, B3	Session binding; nonces; duplicate detection	Depends on correct implementation of freshness checks
T9	Anti-analysis and obfuscated binaries	L5	Record tool versions and parameters; confidence levels; manual validation [6], [7]	Automated similarity may be evaded
T10	Misattribution through ATT&CK mapping	L5	Mappings stored as hypotheses with alternatives and review status	Analysts may over-trust prioritised results
T11	Supply-chain compromise of agents or dependencies	B5, L4	Minimise external services; offline mode; component identity; update provenance [5]	Hardware-level compromise is hard to detect at runtime
T12	Privacy harm from excessive data collection	L2, L4	Scope limits; role-based access; retention policy	Policy enforcement depends on organisational practice

Fig. 9. Trust zones (Z1–Z4) and boundaries (B1–B5) of the reference architecture.



K. Design Decisions and Alternatives

Table XII records the principal design decisions, the alternatives that were considered and the reason for each choice. Several of the choices trade assurance against cost or complexity, so a prototype should be used to test whether each trade-off is acceptable and not simply assume it.

TABLE XII. PRINCIPAL DESIGN DECISIONS AND ALTERNATIVES

Decision	Adopted option	Alternative	Reason and trade-off
Investigator credential	Microcontroller-based token inside a broader authentication process	Dedicated secure element or smart card; software-only credential	Low cost and accessibility against lower assurance than dedicated secure hardware [4], [19], [23], [45]

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

Decision	Adopted option	Alternative	Reason and trade-off
Workflow language	Constrained DSL	General scripting	Constraint enables compile-time checks and review, at the cost of expressiveness
Plan form	Readable IR reviewed before execution	Direct translation to commands	Review adds a step but exposes intent before it is carried out
Agent privilege	Least privilege per signed plan	Resident administrator agent	Smaller attack surface, but some artefacts need elevated rights
Provenance structure	Signed hash chain with external checkpoints	Plain hash list; database audit log only	Greater tamper evidence for more key management
Analysis placement	Central correlation with minimal endpoint normalisation	Full on-endpoint analysis	Cross-host correlation and consistent tooling against more data movement
Interpretation vocabulary	ATT&CK-aligned, version pinned	Ad hoc taxonomy	Shared vocabulary; mapping remains a hypothesis
Connectivity	Controlled offline mode; few external services	Cloud-dependent operation	Reduces third-party exposure [5] at the cost of convenience

L. Maturity

The architecture is a design proposal. No accuracy, latency, scalability, detection-rate or integrity-validation result is claimed. The next phase is a controlled prototype, evaluated as described in Section VIII.

VIII. EVALUATION FRAMEWORK

A. Principles

A meaningful evaluation should measure technical behaviour and forensic quality separately, so that a fast system is never mistaken for a trustworthy one. Fig. 10 outlines the framework. A controlled prototype with a fixed build and test plan is exercised by five test suites, and results are reported together with the hardware, parameters and metric definitions needed to reproduce them. Following good practice in comparative surveys, any future prototype study should fix its test plan before tuning, keep test data apart from development data, and report hardware, parameters and metric definitions. What follows are proposed evaluation dimensions and not reported results.

B. Technical Measures

TABLE XIII. PROPOSED TECHNICAL MEASURES

Measure	Definition	Reporting note
Workflow compilation success	Share of valid workflows that compile without error	Report by workflow class
Compiler rejection correctness	Share of invalid, ambiguous and out-of-scope workflows that are rejected	Report false accepts and false rejects separately
Endpoint execution time	Time to execute a compiled workflow on an endpoint	State endpoint hardware and operating system
Resource consumption	Processor, memory and input/output use at the endpoint during collection	Report peak and average
Event-ingestion throughput	Events processed per unit time by the central component	State hardware and event mix
Correlation latency	Time from ingestion to a correlated result	Report the distribution, not only the mean
Availability	Share of time the services are operational	State the test duration

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

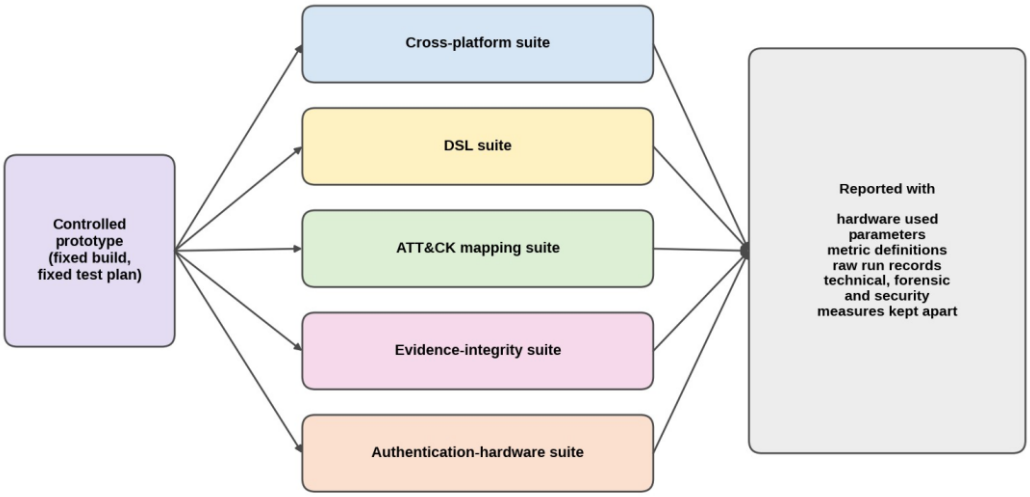
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

C. Forensic Measures

TABLE XIV. PROPOSED FORENSIC MEASURES

Measure	Definition	Reporting note
Acquisition completeness	Share of ground-truth artefacts present in the collected set	Requires a seeded, documented ground truth
Repeatability	Agreement between repeated runs against the same system state	State what varies between runs
Provenance-field completeness	Share of records with every required field populated	List the required fields (Table X)
Hash-verification outcome	Pass or fail of verification, including injected modifications	Report detection of every injected case
Audit-log consistency	Agreement between manifests, chain records and logs	Report inconsistencies found
Traceability ratio	Share of conclusions traceable to original evidence	Count unreferenced conclusions
ATT&CK mapping quality	Precision, recall and analyst agreement	Do not treat every mapping as a binary classification where context is essential

Fig. 10. Evaluation framework: five test suites run against a controlled prototype and reported with full conditions.



Protocol: fix the plan before tuning; keep test data separate from development data; repeat runs.

D. Test Scenarios

The scenarios in Table XV exercise the failure and attack conditions that the threat model anticipates. Each one is defined by an injected condition and an expected observation, so a prototype either shows the behaviour or it does not. Together they also cover the five proposed test dimensions. These are cross-platform execution (equivalent investigative objectives on supported Windows and Linux configurations, recording differences in logs, permissions, file-system semantics and event sources), DSL behaviour (valid, invalid, ambiguous and out-of-scope commands), ATT&CK mapping, evidence integrity, and authentication hardware (reliability, session binding, failure handling, and resistance to replay or device loss).

TABLE XV. TEST SCENARIO MATRIX

ID	Scenario	Injected condition	Expected observation
S1	Modified artefact copy	Alter bytes of a stored artefact after acquisition	Verification fails and identifies the affected record
S2	Metadata alteration	Change an operator or time field in a record	Signature or chain verification fails

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

ID	Scenario	Injected condition	Expected observation
S3	Interrupted transfer	Cut the connection during upload	Manifest records partial collection or failure; no silent success
S4	Duplicate submission	Resubmit the same artefact	Duplicate flagged; chain does not fork
S5	Clock discrepancy	Skew the endpoint clock	Discrepancy flagged; ordering relies on the trusted time source
S6	Key revocation	Revoke a token or signing key	New records under the revoked key are rejected; earlier records remain verifiable under the recorded policy
S7	Out-of-scope workflow	Request an operation beyond the investigator's authority	Compiler rejects before any execution
S8	Unsupported artefact	Request an artefact absent on the platform	Explicit limitation recorded; no substitute source
S9	Token loss or replay	Replay a captured authentication exchange; use a revoked token	Session binding rejects the replay; revocation takes effect
S10	Altered plan	Modify a signed plan in transit	Agent refuses and records "plan rejected"
S11	Benign administration	Run legitimate administrative activity that resembles adversary behaviour	Multiple hypotheses retained; mapping not asserted as fact

E. Experimental Protocol and Threats to Validity

The protocol should (i) fix the test plan and success criteria before tuning; (ii) use separate development and test data, with seeded ground truth for acquisition completeness; (iii) repeat runs so that variation is visible; (iv) record hardware, operating-system versions, parameters and metric definitions; and (v) keep raw run records so that results can be audited. Four threats to validity should be reported. The prototype's test environment may not represent real investigations, seeded ground truth may be easier than real evidence, the authors both build and test the system, and ATT&CK mapping quality depends on analyst judgement. Where a result cannot be generalised, the report should say so.

IX. DISCUSSION

A. Summary of Findings

Returning to the research questions of Section I-B: for RQ1, the hardware and third-party studies imply that the trust assumptions of acquisition platforms should be made explicit and checked at runtime and not taken for granted. For RQ2, binary and behavioural analysis can complement one another, yet both are bounded by obfuscation, build variation and classifier robustness, so findings need preserved parameters, a stated uncertainty and manual validation. For RQ3, a constrained DSL would need declared scope, privileges and expected output, a reviewable IR, explicit limitations and signed releases. For RQ4, the remaining gaps are those in Table V, which the six-layer architecture is meant to address and which only a controlled prototype can test.

B. Limitations of This Survey

The corpus is small and was chosen thematically, so conclusions about the field as a whole would be unsafe. The six studies differ widely in kind: an assessment model, a detection study, a hardware design, a case study, a review and a computational study. They shed light on different parts of the problem and cannot be ranked. The architecture was derived analytically and has not been built, and the traceability map in Fig. 4 is the authors' interpretation. The threat model is not exhaustive, which is why residual risks are stated alongside each mitigation.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

C. Legal, Ethical, and Privacy Considerations

Automation does not remove the need for authority. Collection must be authorised for the case and the target, and for this reason the architecture binds every plan to an authenticated investigator and a declared scope. Automated systems must not collect more data than the authorised purpose of the investigation requires, so scope limits, data minimisation, role-based access and retention policies are design requirements and not optional extras. Whether a given record would be accepted as evidence in a particular jurisdiction depends on local rules and practice that this survey does not address, and the architecture should not be read as legal advice or as a guarantee of admissibility. Human review remains essential for high-impact interpretations, especially where machine-learning outputs or ATT&CK mappings are used to prioritise evidence.

D. Deployment Considerations

A prototype would have to settle several practical matters that the literature leaves open: how agents are installed and updated on endpoints without weakening the system being investigated; how investigators are enrolled and how tokens are issued, replaced and retired; where signing keys are held and how they are recovered; how an independent party receives chain checkpoints; and how the system behaves with limited connectivity. Each should be written down as an explicit assumption, in keeping with the principle that trust is established at every boundary.

X. FUTURE DIRECTIONS

Several lines of work follow from the gaps identified.

- **Controlled prototype.** Build the architecture and evaluate it against Section VIII, publishing the test plan, the scenarios and the raw results.
- **Shared benchmark.** A benchmark of forensic workflows with ground truth for acquisition completeness and provenance would let the field compare approaches fairly.
- **Authentication hardware study.** Compare low-cost tokens with dedicated secure elements on resistance to extraction, session binding, revocation and cost.
- **Compiler assurance.** Develop test suites and, where feasible, formal checks for the compiler and for the lowering from IR to platform operations.
- **Privacy-preserving collection.** Investigate how to minimise personal data and apply role-based access and clear retention policies in automated collection.
- **Secure update and audit export.** Provide signed agent packages, secure update mechanisms and independent audit export so that a third party can verify a case.
- **Broader evidence sources.** Extend the vocabulary to connected and mobile devices, where acquisition is hard [10], [25], and to firmware components [27], [28], [33], [38].
- **Comparison with existing platforms.** Position the architecture against established endpoint forensic platforms, which were outside the scope of this corpus.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

XI. CONCLUSION

Trustworthy digital forensics takes more than a collection tool or a cryptographic hash. It depends on the integrity of the hardware and software involved, on investigative procedures that can be repeated, on interpreting events in their context, and on being able to show how evidence was acquired and handled. The reviewed studies each light up an important part of this problem. Hardware-security assessment exposes supply-chain dependencies, hardware Trojan detection explores automated assurance, cryptographic accelerators provide trusted primitives, third-party device research highlights external trust boundaries, and binary-code analysis tackles compiled and evasive artefacts.

The main finding of this survey is that these strands remain unintegrated. The reference architecture we outline is offered as one response to that gap. It centres on a DSL-driven workflow, compiler-assisted cross-platform execution, endpoint agents, central correlation with ATT&CK-mapped analysis, SHA-256 hashing with chained metadata and hardware-assisted investigator authentication, and it is supported by a trust-boundary analysis, a threat model and a scenario-based evaluation framework. Because it is only a proposal, any claim about its effectiveness or operational readiness has to wait for a controlled prototype, transparent evidence provenance and reproducible cross-platform testing.

ACKNOWLEDGMENT

The authors thank the authors of the studies reviewed in this survey, whose work shaped the thematic analysis.

REFERENCES

[1] O. Chalyi, “Beyond hardware acceleration: A heuristic decision framework for optimizing WPA2 dictionary attacks,” IEEE Access, vol. 14, 2026, doi: 10.1109/ACCESS.2026.3737853.

[2] S. Maragkou, L. Rappel, H. Dettmer, T. Sauter, and A. Jantsch, “The pains of hardware security: An assessment model of real-world hardware security attacks,” IEEE Open J. Ind. Electron. Soc., vol. 6, 2025, doi: 10.1109/OJIES.2025.3561675.

[3] V. T. Hayashi and W. V. Ruggiero, “Hardware Trojan detection in open-source hardware designs using machine learning,” IEEE Access, vol. 13, 2025, doi: 10.1109/ACCESS.2025.3546156.

[4] P. Nannipieri, L. Crocetti, S. Di Matteo, L. Fanucci, and S. Saponara, “Hardware design of an advanced-feature cryptographic tile within the European Processor Initiative,” IEEE Trans. Comput., vol. 74, no. 3, 2025, doi: 10.1109/TC.2023.3278536.

[5] S. Lorenz, S. Stinehour, A. Chennamaneni, A. B. Subhani, and M. Nadim, “A case study of 3rd party hardware: The weakest link in Google’s trustworthy artificial intelligence implementation,” IEEE Access, vol. 13, 2025, doi: 10.1109/ACCESS.2025.3570633.

[6] H. Javed, F. Ali, B. Shah, and D. Kwak, “Binary code analysis for cybersecurity: A systematic review of forensic techniques in vulnerability detection and anti-evasion strategies,” IEEE Access, vol. 13, 2025, doi: 10.1109/ACCESS.2025.3610616.

[7] G. Morse, M. Alqaradaghi, and T. Kozsik, “Control flow obfuscation with irreducible loops and self-modifying code,” Publ. Math. Debrecen, vol. 100, pp. 617–637, Dec. 2022, doi: 10.5486/pmd.2022.suppl.5.

[8] S. Snedaker and B. Cunningham, The Best Damn IT Security Management Book Period, 1st ed. Burlington, MA, USA: Elsevier, 2007.

[9] D.-E.-S. Kundi, A. Khalid, A. Aziz, C. Wang, M. O’Neill, and W. Liu, “Resource-shared crypto-coprocessor of AES enc/dec with SHA-3,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 67, no. 12, pp. 4869–4882, Dec. 2020.

[10] J. P. van Zandwijk and A. Fukami, “NAND flash memory forensic analysis and the growing challenge of bit errors,” IEEE Secur. Privacy, vol. 15, no. 6, pp. 82–87, Nov. 2017.

[11] S. Amir, B. Shakya, X. Xu, Y. Jin, S. Bhunia, M. Tehranipoor, and D. Forte, “Development and evaluation of hardware obfuscation benchmarks,” J. Hardw. Syst. Secur., vol. 2, no. 2, pp. 142–161, Jun. 2018.

[12] J. Ferdous, R. Islam, A. Mahboubi, and M. Z. Islam, “AI-based ransomware detection: A comprehensive review,” IEEE Access, vol. 12, pp. 136666–136695, 2024.

[13] E. C. R. Shin, D. Song, and R. Moazzezi, “Recognizing functions in binaries with neural networks,” in Proc. 24th USENIX Secur. Symp., 2015, pp. 611–626.

[14] L.-A. Daniel, “Symbolic binary-level code analysis for security: Application to the detection of microarchitectural timing attacks in cryptographic code,” Université Côte d’Azur, Nice, France, Tech. Rep., 2021.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

[15] H. Javed, S. El-Sappagh, and T. Abuhmed, "Robustness in deep learning models for medical diagnostics: Security and adversarial challenges towards robust AI applications," Artif. Intell. Rev., vol. 58, no. 1, pp. 1–107, Nov. 2024, doi: 10.1007/s10462-024-11005-9.

[16] J. Yang, C. Fu, X.-Y. Liu, H. Yin, and P. Zhou, "Codee: A tensor embedding scheme for binary code search," IEEE Trans. Softw. Eng., vol. 48, no. 7, pp. 2224–2244, Jul. 2022, doi: 10.1109/TSE.2021.3056139.

[17] C. Groh, S. Proskurin, and A. Zarras, "Free Willy: Prune system calls to enhance software security," in Proc. 38th ACM/SIGAPP Symp. Appl. Comput., Mar. 2023, pp. 1522–1529, doi: 10.1145/3555776.3577593.

[18] S. Alrabaee, "A stratified approach to function fingerprinting in program binaries using diverse features," Expert Syst. Appl., vol. 193, May 2022, Art. no. 116384, doi: 10.1016/j.eswa.2021.116384.

[19] NXP, "Building a secure system using NXP secure MCU LPC54S0xx," Tech. Rep., Mar. 2019. [Online]. Available: https://www.nxp.com/docs/en/application-note/AN12385.pdf

[20] M. Oreyomi and H. Jahankhani, "Challenges and opportunities of autonomous cyber defence (ACyD) against cyber attacks," in Blockchain and Other Emerging Technologies for Digital Business Strategies, 2022, pp. 239–269, doi: 10.1007/978-3-030-98225-6_9.

[21] H. Yakura, S. Shinozaki, R. Nishimura, Y. Oyama, and J. Sakuma, "Malware analysis of imaged binary samples by convolutional neural network with attention mechanism," in Proc. 8th ACM Conf. Data Appl. Secur. Privacy, Mar. 2018, pp. 127–134, doi: 10.1145/3176258.3176335.

[22] P. Ravi, S. Deb, A. Baksi, A. Chattopadhyay, S. Bhasin, and A. Mendelson, "On threat of hardware trojan to post-quantum lattice-based schemes: A key recovery attack on SABER and beyond," in Proc. Int. Conf. Secur., Privacy, Appl. Cryptogr. Eng., Jan. 2022, pp. 81–103.

[23] ARM, "ARM security technology: Building a secure system using TrustZone technology," Tech. Rep., 2005–2009. [Online]. Available: https://developer.arm.com/documentation/PRD29-GENC-009492/c

[24] IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Amendment 6: Medium Access Control (MAC) Security Enhancements, IEEE Std 802.11i, Jul. 2004.

[25] C. M. da Silveira et al., "Methodology for forensics data reconstruction on mobile devices with Android operating system applying in-system programming and combination firmware," Appl. Sci., vol. 10, no. 12, Art. no. 4231, Jun. 2020.

[26] W. Fu, H. Yu, O. Arias, K. Yang, Y. Jin, T. Yavuz, and X. Guo, "Graph neural network based hardware trojan detection at intermediate representative for SoC platforms," in Proc. Great Lakes Symp. VLSI, Jun. 2022, pp. 481–486.

[27] P. Shirani et al., "BINARM: Scalable and efficient detection of vulnerabilities in firmware images of intelligent electronic devices," in Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment, 2018, pp. 114–138, doi: 10.1007/978-3-319-93411-2_6.

[28] M. Beninger, P. Charland, S. H. H. Ding, and B. C. M. Fung, "ERS0: Enhancing military cybersecurity with AI-driven SBOM for firmware vulnerability detection and asset management," in Proc. 16th Int. Conf. Cyber Conflict (CyCon), May 2024, pp. 141–160, doi: 10.23919/CyCon62501.2024.10685598.

[29] M. Fyrbiak et al., "HAL—The missing piece of the puzzle for hardware reverse engineering, trojan detection and insertion," IEEE Trans. Dependable Secure Comput., vol. 16, no. 3, pp. 498–510, May 2019.

[30] S. Yang et al., "Towards practical binary code similarity detection: Vulnerability verification via patch semantic analysis," ACM Trans. Softw. Eng. Methodol., vol. 32, no. 6, pp. 1–29, Nov. 2023, doi: 10.1145/3604608.

[31] L. Massarelli, G. A. D. Luna, F. Petroni, R. Baldoni, and L. Querzoni, "SAFE: Self-attentive function embeddings for binary similarity," in Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment, 2019, pp. 309–329, doi: 10.1007/978-3-030-22038-9_15.

[32] Y. Zhang, A. X. Xu, and Y. Jiang, "Scalable and accurate binary code search method based on simhash and partial trace," in Proc. IEEE 19th Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom), Dec. 2020, pp. 818–826, doi: 10.1109/TrustCom50675.2020.00111.

[33] P. Sun, L. Garcia, G. Salles-Loustau, and S. Zonouz, "Hybrid firmware analysis for known mobile and IoT security vulnerabilities," in Proc. 50th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN), Jun. 2020, pp. 373–384, doi: 10.1109/DSN48063.2020.00053.

[34] F. Demrozi, R. Zucchelli, and G. Pravadelli, "Exploiting sub-graph isomorphism and probabilistic neural networks for the detection of hardware trojans at RTL," in Proc. IEEE Int. High Level Design Validation Test Workshop (HLDVT), Oct. 2017, pp. 67–73.

[35] Information Security, Cybersecurity and Privacy Protection—Information Security Management Systems—Requirements, ISO/IEC 27001, Oct. 2022.

[36] B. Halak, Ed., "CIST: A threat modelling approach for hardware supply chain security," in Hardware Supply Chain Security. Berlin, Germany: Springer, 2021, pp. 3–65, doi: 10.1007/978-3-030-62707-2_1.

[37] R. Hassan, X. Meng, K. Basu, and S. M. P. Dinakarrao, "Circuit topology-aware vaccination-based hardware trojan detection," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 42, no. 9, pp. 2852–2862, Sep. 2023.

[38] G. Hernandez, F. Fowze, D. J. Tian, T. Yavuz, P. Traynor, and K. R. B. Butler, "Toward automated firmware analysis in the IoT era," IEEE Secur. Privacy, vol. 17, no. 5, pp. 38–46, Sep. 2019, doi: 10.1109/MSEC.2019.2926462.

[39] J. Baehr, A. Hepp, M. Brunner, M. Malenko, and G. Sigl, "Open source hardware design and hardware reverse engineering: A security analysis," in Proc. 25th Euromicro Conf. Digit. Syst. Design (DSD), Aug. 2022, pp. 504–512.

[40] V. Gohil, H. Guo, S. Patnaik, and J. Rajendran, "ATTRITION: Attacking static hardware trojan detection techniques using reinforcement learning," in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., Nov. 2022, pp. 1275–1289.

[41] H. Salmani, M. Tehranipoor, S. Sutikno, and F. Wijitrisnanto, "Trusthub trojan benchmark for hardware trojan detection model creation using machine learning," ACM Trans. Embed. Comput. Syst., vol. 22, Mar. 2022, Art. no. 46.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

[42] M. Nabeel, M. Ashraf, S. Patnaik, V. Soteriou, O. Sinanoglu, and J. Knechtel, "2.5D root of trust: Secure system-level integration of untrusted chiplets," IEEE Trans. Comput., vol. 69, no. 11, pp. 1611–1625, Nov. 2020.

[43] W. Han, Z. Li, L. Yuan, and W. Xu, "Regional patterns and vulnerability analysis of Chinese web passwords," IEEE Trans. Inf. Forensics Security, vol. 11, no. 2, pp. 258–272, Feb. 2016, doi: 10.1109/TIFS.2015.2490620.

[44] P. Xu, Z. Mai, Y. Lin, Z. Guo, and V. S. Sheng, "A survey on binary code vulnerability mining technology," J. Inf. Hiding Privacy Protection, vol. 3, no. 4, pp. 165–179, 2021, doi: 10.32604/jihpp.2021.027280.

[45] Rambus, "Introducing the Rambus CryptoManager root of trust (CMRT)," 2018. [Online]. Available: https://www.rambus.com/blogs/introducing-the-rambus-cryptomanager-root-of-trust-cmrt

[46] S. Di Matteo, L. Baldanzi, L. Crocetti, P. Nannipieri, L. Fanucci, and S. Saponara, "Secure elliptic curve crypto-processor for real-time IoT applications," Energies, vol. 14, no. 15, pp. 1–20, Aug. 2021.

[47] R. Yasaei, L. Chen, S.-Y. Yu, and M. A. A. Faruque, "Hardware trojan detection using graph neural networks," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 44, no. 1, pp. 25–38, Jan. 2025.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)