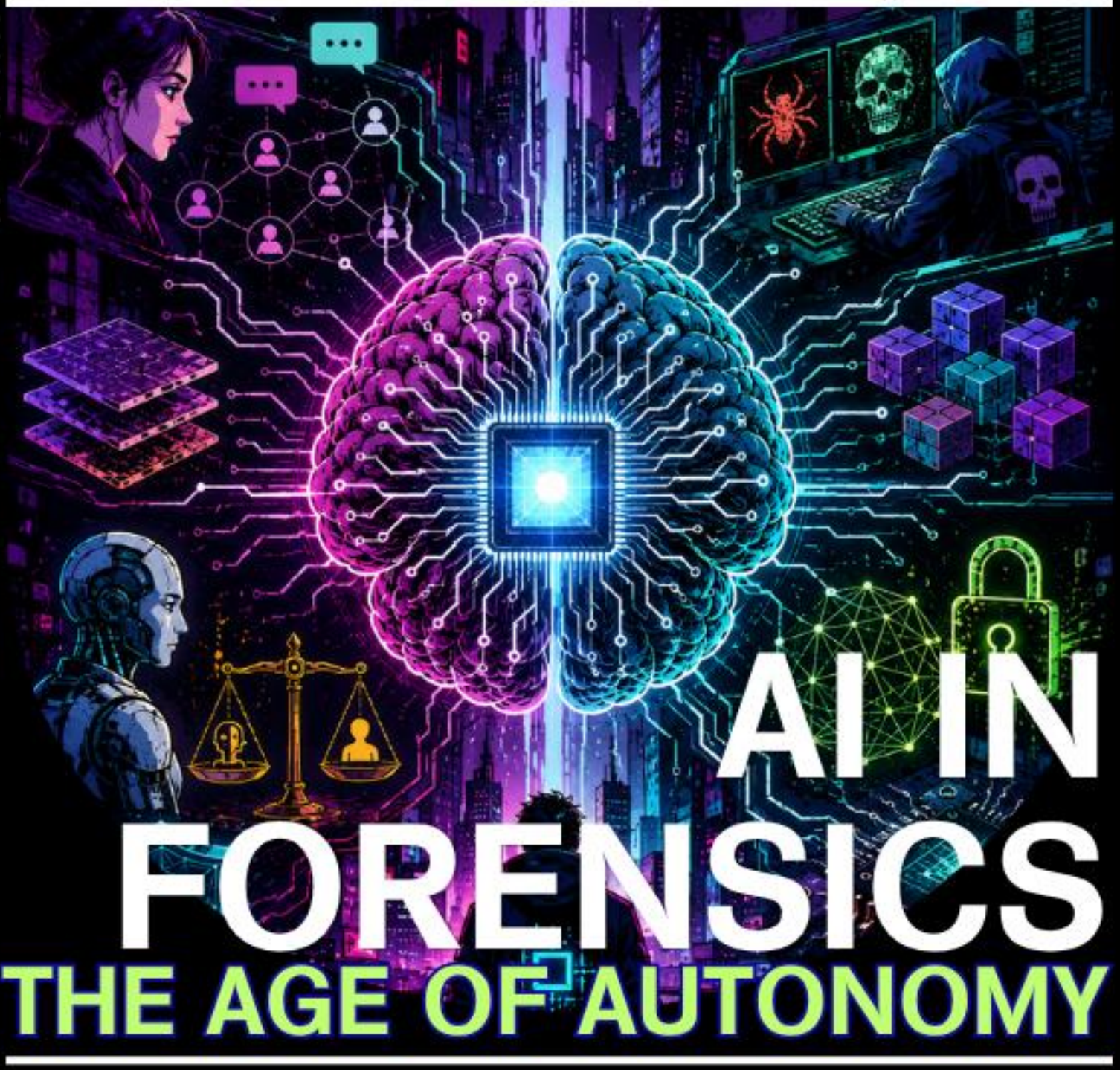
eForensics
VOL. 13
NO. 04
ISSN 2300-6986



AI IN FORENSICS
THE AGE OF AUTONOMY

Editor's Word

Dear Readers,

We live in an era where digital evidence surrounds us completely, and artificial intelligence is fundamentally redefining not only the modern threat landscape but the very fabric of how we conduct investigations. In this latest issue of eForensics, we bring you a highly curated collection of articles that cut straight through the ongoing AI hype to deliver practical insights from the front lines of digital forensics and incident response (DFIR).

The increasing use of artificial intelligence in forensic analysis is a worrying reality that could even create what Sammons, based on Saferstein's work, described in his classic book as "the CSI effect"—which creates the illusion that certain forensic science practices can solve any case and therefore develops unreasonable expectations that lead to erroneous verdicts. As we experience the challenges of AI in forensic investigations, the core issue shifts to the integrity of forensic evidence itself. The key ethical factor centers on how much evidence is a product of AI procedures rather than human analysis. Every verdict based only on AI is a null verdict.

Artificial intelligence has emerged as an indispensable force multiplier for lean security operations centers and overwhelmed forensic labs wrestling with massive data triage. However, this shift introduces complex operational and ethical dilemmas that until recently belonged strictly to the realm of science fiction. Can we maintain full procedural transparency when autonomous algorithms drive critical forensic decisions? Where exactly should we draw the line for AI agent permissions within an active SOC? Our contributors address these pressing questions head-on.

Inside this comprehensive issue, you will discover:

- **macOS Tahoe Binary Analysis via LLMs: Israel Torres walks through an exact, step-by-step methodology to coax ChatGPT into analyzing untrusted binaries and generating production-ready YARA rules, skillfully bypassing vendor-imposed platform guardrails along the way.**
- **Behavioral Identification of Chat-Based Abuse: Andreas Antonsen from STNDRDS AB shares a unique, multi-layered linguistic framework capable of identifying patterns of coercion, control, and psychological manipulation where traditional keyword-based filtering completely fails.**

- **AI Assistants as Targets and Weapons: Igor Korkin, Yuriy Tumanov, and Oksana Dokuchaeva dissect the rapidly expanding attack surfaces of Large Language Models, detailing the mechanics of context poisoning and corporate data exfiltration via active user browser sessions.**
- **The Boundaries of SOC Autonomy: Mayur Agnihotri staves off systemic risks by proposing an operational framework where an AI agent's level of permission is governed strictly by the reversibility of its actions, rather than subjective risk assessments.**
- **Post-Quantum Cryptography & Drone Warfare: Explore the field deployment of quantum-resistant ML-KEM-512 algorithms on resource-constrained ESP32 IoT microcontrollers, alongside a fascinating investigation into how Adversarial Machine Learning (AML) introduces a "forensic fog" during drone swarm crash analysis.**

We close this issue with an engaging, deep-dive debate into the ongoing reality of AI-Malware. Will on-the-fly, LLM-generated polymorphic code truly evade next-generation EDR solutions, or do these machine-generated threats leave behind highly distinct, structural footprints that make them inherently detectable to a trained eye?

Whether you are a seasoned DFIR practitioner, an academic researcher, or a cybersecurity enthusiast, this issue provides the strategic and technical insights necessary to successfully navigate the complex digital battlefield of 2026. AI is becoming a native member of the investigative team—not to replace human intuition, but to allow us to move faster, analyze deeper, and ultimately stay ahead of the curve.

Wishing you an inspiring and insightful read,

Ewa & Paulo & the eForensics Team

Table of Contents

Vet, Trust, and Use the Foundation for Bringing AI into Investigations by *Amber Schroader*

These days, you can't even perform a simple web search without an AI chiming in. This shift has triggered a massive push for investigators to pack their toolkits with AI-powered gadgets from automated triage and image matching to cleaning up audio and extracting data in real-time. AI is often sold as a "magic wand" for digital forensics, and many examiners are skeptical. There is a real fear that these tools are either too good to be true or worse, designed to replace humans entirely. Those concerns are valid. We need to treat AI with the same scientific scrutiny we apply to any other forensic tool, even though it is not a forensic tool in the traditional sense.

First, draw a line between a traditional forensic tool and an AI tool designed for forensics. Forensic tools process data with a specific, repeatable process; functions such as hashing are built into that workflow. AI, on the other hand, is designed to process data in a way that optimizes its ability to recognize patterns and relationships among data points. It does not understand all the files a forensic tool does, nor can it perform the standard verifications a forensic tool can. In this context, AI is closest to behaving like a human examiner, which is where the "job-loss" worry originates. The real risk of AI is not the technology itself replacing us, but the lack of adoption of the technology. AI is a powerful multiplier: it speeds up workflows, uncovers hidden artifacts, and lets small labs punch far above their weight class. The danger of no AI adoption leaves examiners and labs functioning slower, with data only increasing the risk of backlog is high.

The goal for all examiners is to use a "forensic-grade" tool for every piece of work. With AI, that classification is harder because there is no guarantee of repeatable results. So, how can we take a technology like AI and use it for forensics? The answer is not simple, but it remains structured and scientific. It starts with the vetting process.

Why Structured Vetting Matters

Forensic decisions carry legal consequences, and when you rely on an AI tool to prioritize leads, transcribe data, compare data sets, or find the missing piece in a pattern, you must be able to show that the tool's outputs are reliable, reproducible to a reasonable degree, and understood.

AI's answers can vary slightly each time, even with identical input and prompts, since a tool may return different wording. An example from the Paraben Zandra AI tool is that you can give the exact same data, and the exact same prompts and 100% of the time it will give you different words in the reply. That doesn't mean the returned context is different, but the words to form that reply will be different. The AI is similar to a human that might explain to someone one way and the next time have slightly different words to describe the same situation the next time. They are both correct, but this small variance is part of why typical forensic testing and vetting or validation will not work. This does not mean the underlying meaning changes, but the variability means typical forensic testing and validation approaches need adaptation. Instead of vetting the exact output, we should start by examining the AI architecture.

Each manufacturer should know the models and training that went into the AI. You do not simply create an LLM and call it an AI; the foundations of all AI systems come from existing models that are later customized. Therefore, you need to understand what model is being used, what version you are running, and what customizations were made for the data you are processing. This understanding is the key to vetting. If we compare AI to a human examiner, we want a human who understands the difference between data from a Windows Registry and data from an iOS App. Examiners are trained to recognize these distinctions; the AI should be trained similarly. In practice; as part of evaluating the architecture, you ask the vendor, "How does your AI understand my DFIR data?" .

How to Test This Understanding

Test the AI's comprehension with a specific use case and a set of data. Rather than "just using AI," start with a well-defined problem.

1. **Pick Your Use Case** – What question are you trying to answer? Common investigative uses of AI include:
    - *Prioritization*: Ranking thousands of images by "last accessed" dates so a human knows what to examine first. (This requires metadata, not the image pixels.)
    - *Enhancement*: Cleaning up grainy video or muffled audio to make it usable.
    - *Extraction*: Running OCR, redacting PII, or converting speech to text.
2. **Set Your Metrics** – For every task, define how you will grade the AI's work up front:
    - *Accuracy*: If prioritizing files, how many are correctly listed?
    - *Error Rates*: For transcriptions, what is the Word Error Rate (WER)?
    - *The "Flunk" Rate*: What are your limits for false positives or false negatives?
    - *Efficiency*: How much time does this save an analyst per case?

3. **Know the "Why" and the "How"** – Establish acceptable operating points. For example, at what point does an automated flag require a mandatory human doublecheck? The answer is: every time.

4. **Vendor Transparency** – When selecting a vendor, transparency is not a nice-to-have; it is a requirement. Treat these tools with the same scrutiny you would give any other forensic software. For each model, obtain and archive:

    - *Model Details*: Exact model and version (ask for a model card if available).
    - *Training Background*: Source of data, its age, and how it was cleaned.
    - *Weak Spots*: Known limitations and likely failure points.
    - *Legal Fine Print*: Licensing terms, whether output can be used as evidence, and any reporting requirements.
    - *Privacy*: How the data you input is used.

Insist on vendors who openly share benchmarks and documentation. Verify that the tool includes logging and a set testing plan. Confirm that it has been trained to understand the data you will process. If a vendor is evasive about how their technology works, treat that as a red flag and walk away.

To truly trust an AI tool, test it against the kind of data you encounter in the field. Do not rely solely on vendor demos or polished marketing.

1. **Build a Real-World Test Set** – Assemble a test set that mirrors your actual caseload. It should contain:

    - Real-world variety: samples from different devices, file formats, and languages.
    - Tough material: low-quality audio, heavily compressed video, and files that have been edited or manipulated.
    - "Anti-forensic" files: intentionally degraded, noisy, or down-sampled data to see if the AI trips up.
    - Diverse data: a wide range of demographics and languages to check for hidden bias.

Document every file name, hash, and how you obtained the data. Treat this collection as a piece of evidence. Keep it version-controlled and protected as part of your lab's vetting process. If you lack time to create your own set, use standard community test sets, but ensure you can output materials in formats the AI can understand.

2. **Run a Systematic Validation** – Your testing must be repeatable and auditable to a point. Because AI responses are more human-like, there will

be variance in wording; validate the intent rather than exact phrasing. At a minimum:

- Establish a Baseline: Run the tool and calculate core statistics (right vs. wrong).
- Stress Test It: Add noise or alter inputs to see when and how the AI begins to fail.
- Compare Results: Pit the AI against other tools or a human expert to gauge performance.
- Check for Drift: Whenever the software updates, rerun your tests to ensure the AI's behavior has not degraded or changed unexpectedly.

3. **Keep a Paper Trail** – Create a validation checklist that documents your dataset, testing methods, and any limitations discovered. Save prompts, settings, and version numbers to ensure reproducibility. If the tool offers a deterministic mode (same answer every time for the same file), enable it. If not, capture the random seeds the AI uses so you can reproduce findings later.

Understanding AI Context, Logging, and Terms of Use

AI models can embed biases that affect investigative outcomes. As part of validation or vetting, you must understand the context in which the AI was trained and how it interprets data common to your investigations. Consider:

- Demographic performance gaps (speech recognition across accents, face-recognition disparities, etc.).
- Language and dialect coverage for text and speech models.
- Labeling and sampling biases in training data that may skew detection.

If systematic gaps appear, adjust operating thresholds, restrict tool use to appropriate contexts, and require human review. Document bias-testing results and retest when new versions are released.

For tools used in casework, retain detailed logs that support reproducibility not exact verbatim responses, but sufficient context to audit the process. If logs are unavailable, do not use the tool. Those logs serve as the audit or "case notes" the AI generates as part of your investigation team; they should be as accessible and transparent as the notes you would expect from a fellow investigator.

Additionally, scrutinize the license: understand the terms of use and privacy terms. AI tools often lack a standard EULA, so you must clarify how data and tool usage are permitted, whether the vendor may learn from your data, and what sharing restrictions

exist. Review both the terms of use and the company's privacy policy to ensure there are no loopholes that allow your data to be used for model training without consent.

Human in the Loop Is Always Required

AI should augment, not replace, examiner judgment. Establish governance that defines where AI outputs can be used and where human intervention is mandatory:

- *Triage and prioritization*: acceptable to automate, provided a human review the top results.
- *Final conclusions*: require human analyst sign-off and inclusion of the validation before submission to prosecutors or court.

Define roles and specify where AI will be employed and where it will be prohibited, all within the context of the vetting/validation process already completed. If an AI performs poorly in a given area, do not use it there. Retest AI frequently because it evolves faster than any other technology in the past two decades.

Maintain a prompt library as part of your baseline to keep the human-in-the-loop consistent; a set prompt simplifies human review and helps ensure every AI-produced output follows a consistent format.

Practical Checklist for Vetting and Operationalizing an AI Tool

- Define use case and success metrics.
- Require vendor documentation (model card, training summary, versioning).
- Build and document a representative test dataset.
- Run baseline performance, sensitivity, and bias tests.
- Log all inputs/outputs, parameters, and hashes.
- Require human sign-off for case conclusions; record decision governance.
- Confirm the terms of service and data protection.
- Schedule periodic revalidation and monitoring.

Adopting AI in investigations can materially improve timeliness and scope of work, but only if you embrace a disciplined vetting, validation, and lifecycle approach. Aim for tools that are transparent, testable, and accompanied by strong logging and appropriate terms of use. Treat vendor claims skeptically, insists on reproducible validation in your environment, keep humans in the loop, document everything, and refresh validations regularly. Adopting AI with a forensic mindset will ensure that everything remains admissible and that technology serves as a force multiplier rather than a liability.

References

https://www.swgde.org/documents/current-documents/

https://www.nist.gov/itl/ai-risk-management-framework

https://www.nist.gov/itl/csd/secure-systems-and-applications/national-software-reference-library-nsrl

Standards and References Supporting Forensic-Grade AI Validation

1. **Foundational Forensic Principles and Tool Validation:** Established digital forensic standards emphasize repeatability, validation, and evident integrity as prerequisites for admissibility. NIST SP 800-101 Rev. 1 provides guidance specific to mobile device forensics, illustrating core principles of forensic soundness and tool reliability within a defined domain (NIST, 2014). Broader validation expectations are articulated in the Scientific Working Group on Digital Evidence (SWGDE) guidelines, particularly SWGDE IT-3 (Validation of Forensic Software), which defines requirements for repeatable, testable, and documented tool behavior, and SWGDE IT-17 (Digital Evidence Authentication), which addresses integrity and verification practices. Together, these frameworks establish the baseline expectation of "forensic-grade" tooling.

2. **AI Risk, Bias, and Validation Methodologies:** The NIST AI Risk Management Framework (AI RMF 1.0) provides a structured approach to identifying and mitigating risks in AI systems, including bias, lack of transparency, and insufficient human oversight (NIST, 2023). Supporting resources such as the AI RMF Playbook further operationalize testing, evaluation, and continuous monitoring practices. Empirical evidence of AI bias is well documented in the Gender Shades study (Buolamwini & Gebru, 2018), which demonstrated significant demographic disparities in facial recognition performance. These works collectively support the need for systematic bias testing, demographic performance analysis, and documented model limitations in forensic AI applications.

3. **Model Transparency and Documentation Practices:** Emerging best practices for AI transparency are captured in Model Cards for Model Reporting (Mitchell et al., 2019) and Datasheets for Datasets (Gebru et al., 2018). These frameworks propose standardized documentation of model behavior, intended use, training data composition, and known limitations. While not formal standards, they are widely adopted and align directly with requirements for explainability, reproducibility, and defensibility in forensic contexts. NIST's AI evaluation and testing resources further reinforce the importance of baseline testing, stress testing, and drift monitoring in maintaining model reliability over time.

4. **Human Oversight, Auditability, and Legal Admissibility:** ISO/IEC 27037:2012 establishes international guidelines for the identification, collection, acquisition, and preservation of digital evidence, emphasizing audit trails, reproducibility, and process transparency (ISO, 2012). These requirements support the necessity of human oversight and verifiable logging in AI-assisted forensic workflows. In the

United States, admissibility is governed by Federal Rule of Evidence 702 and the Daubert standard, which require that expert methods be reliable, testable, and generally accepted. These legal frameworks underscore the necessity of documented validation and human verification when introducing AI-derived evidence in court.

5. **Test Data and Real-World Validation Resources:** Robust validation requires diverse, representative datasets. The NIST National Software Reference Library (NSRL) provides curated reference data for tool testing and verification. The Computer Forensic Reference Data Sets (CFReDS) project offers scenario-based datasets designed to evaluate forensic tool performance under controlled conditions. Additionally, the Digital Forensics Research Workshop (DFRWS) challenge datasets supply realistic and adversarial test corpora, including anti-forensic and low-quality data scenarios. These resources enable reproducible, version-controlled testing aligned with real-world investigative conditions.

About the Author

Amber Schroader

CEO & Founder, Paraben Corporation

With over 30 years of innovation in digital forensics, Ms. Schroader has developed cutting-edge recovery software for everything from smartphones to cloud storage. She is the architect of the "Forensics of Everything" (FoE) framework, a holistic approach to evidence that has set global standards for seizure and processing protocols. Beyond her technical contributions, she is an influential educator and founder of industry certifications, dedicated to shaping the future of the field through her teaching, writing, and speaking.

Detecting Coercive Control Patterns in Forensic Chat Analysis: A Behavioural Approach by *Andreas Antonsen — STNDRDS AB*

Swedish examples are shown alongside English translations for clarity; detection runs in the original language.

Coercive control leaves few clean traces. Unlike physical assault, the evidence accumulates across hundreds of communication acts that look ordinary in isolation. A message saying *"I'm the only one who really understands you"* reads as genuine affection. The same line repeated across weeks, combined with systematic isolation from friends, memory-questioning of shared events, and gradual assumption of financial control, is something else entirely. The signal is in the pattern, not the phrase.

Most forensic chat-analysis tools handle this poorly. Keyword approaches that work for drug or weapon terminology miss the point because coercive-control utterances are semantically ordinary. Recent deep-learning approaches targeting intimate-partner-violence detection on social-media text show some traction [1] but generally operate as black-box classifiers that flag risk without producing the inspectable, category-level trace a forensic pipeline requires. Large-language-model approaches similarly produce plausible concept matches but often cannot explain *why* a specific message was flagged — a liability when the flagged chat reaches prosecution or cross-examination. This article walks through a pattern-based alternative used in a Swedish forensic pipeline, defines five operational categories, and is explicit about what the approach still gets wrong.

Why keyword detection fails here

The canonical utterances of coercive control are not specialized vocabulary. *"No one else will ever love you like I do." "You're remembering it wrong — I never said that." "Your family is toxic — we don't need them." "You don't know how to handle money, let me take care of the accounts."* Each phrase is semantically ordinary. Each appears in benign contexts: reassurance between spouses, genuine memory disagreements, family conflict, financial-management arrangements between trusting partners.

A keyword list that fires on these produces false positives at a rate that degrades investigator workflows. A keyword list that avoids them misses the cases entirely. The

signal is not in any individual utterance; it is in *which categories of utterance co-occur*, *how often*, and *across what relational context*.

It is worth distinguishing between two superficially similar mechanisms. A lookup table — the canonical keyword approach — answers a single question: is this phrase known? Pattern-plus-context detection answers a structurally different one: is this phrase used in a stance and relational frame consistent with the underlying behavioral category? The first reduces detection to membership testing. The second performs an explicit translation from the semantic category to a set of surface conditions on the text, each of which is independently inspectable. The distinction matters because the failure modes of the two are different in kind, not just in degree.

The implication for detection is straightforward. The unit of analysis must be the category of controlling behaviour rather than the specific word, and the evidentiary weight must come from the combination and accumulation of categories rather than individual lines.

Five operational categories

Across the coercive-control literature and practitioner frameworks — most influentially Evan Stark's work on coercive control as a distinct pattern of intimate abuse [2], along with earlier structured frameworks from the Duluth model's Power and Control Wheel [3] and subsequent measurement research by Hamberger, Larsen & Lehrner [4] — five behavioral categories recur with enough consistency to be operationalized as detection targets. We use these five:

- **Gaslighting** — denial of the victim's reality, memory-questioning, framing the victim as unstable or mistaken
- **Isolation** — cutting off the victim from friends, family, and support networks
- **Love-bombing** — excessive flattery, rapid-escalation attachment, soulmate framings
- **Blame-shifting** — making the victim responsible for the perpetrator's actions
- **Economic control** — restricting financial autonomy, controlling accounts or cards

This set is deliberately shorter than exhaustive clinical taxonomies. We exclude categories that are difficult to distinguish reliably from ordinary conflict in text alone — such as silent-treatment and threat-of-abandonment patterns, which leave few linguistic traces and rely heavily on extra-textual context. The goal is operational: each category we include must be detectable with reasonable precision in chat data.

Each category has characteristic linguistic markers. A representative (non-exhaustive) sample, drawn from the detector implementation covered in this article:

Gaslighting markers (Swedish / English)

- *"du inbillar dig" / "you're imagining things"*
- *"det där hände aldrig" / "that never happened"*
- *"du är paranoid" / "you're paranoid"*

- *"det är bara i ditt huvud" / "it's all in your head"*
- *"jag sa aldrig det" / "I never said that"*

Isolation markers

- *"du behöver inte dina vänner" / "you don't need your friends"*
- *"bara jag förstår dig" / "only I understand you"*
- *"din familj hatar dig" / "your family hates you"*
- *"jag är allt du behöver" / "I'm all you need"*

Love-bombing markers

- *"du är mitt allt" / "you are my everything"*
- *"vi är själsfränder" / "we are soulmates"*
- *"jag har aldrig känt så här förut" / "I've never felt this way before"*
- *"jag klarar inte leva utan dig" / "I can't live without you"*

Blame-shifting markers

- *"det är ditt fel" / "it's your fault"*
- *"du tvingar mig" / "you're forcing me"*
- *"titta vad du har gjort" / "look what you've done"*
- *"du förtjänar det" / "you deserved it"*

Economic control markers

- *"du behöver inte egna pengar" / "you don't need your own money"*
- *"jag sköter ekonomin" / "I handle the finances"*
- *"ge mig ditt kort" / "give me your card"*
- *"du kan inte hantera pengar själv" / "you can't handle money on your own"*

The current implementation covers Swedish and English patterns. Extending to additional languages is straightforward in architecture but labour-intensive in practice: affective registers and control-language idioms vary considerably across cultures and rarely translate by lookup table.

Pattern-plus-context detection

A pattern list alone is not sufficient because many of these patterns fire spuriously in ordinary communication. A therapist might text a client *"you're not imagining it — that was a real slight."* A parent frustrated with a misbehaving child might write *"look what you've done."* A romantic partner saying *"I can't live without you"* may be hyperbolic devotion, not love-bombing.

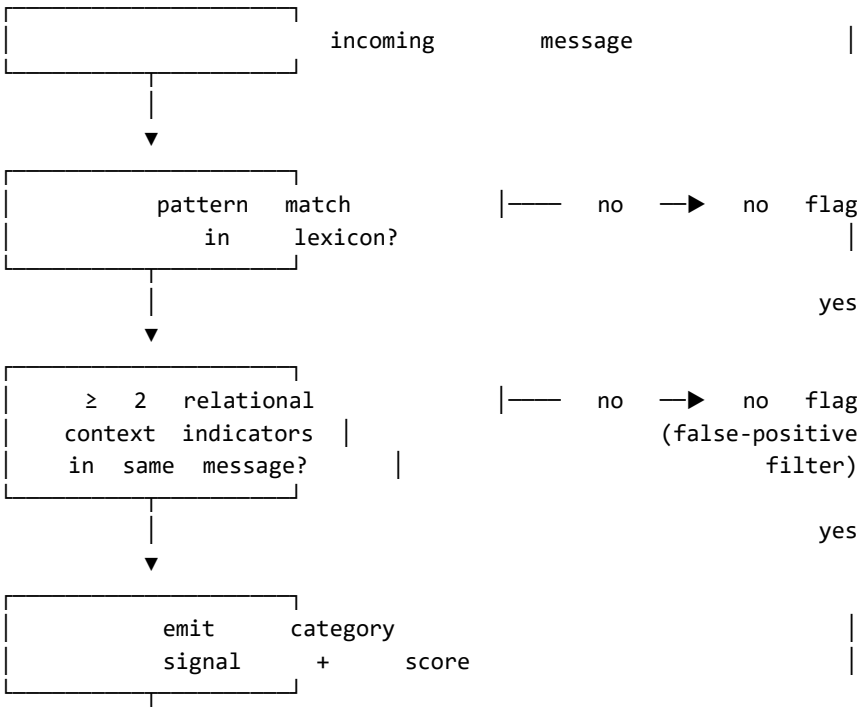
To reduce these false positives without hand-crafting exhaustive exceptions, the detection layer requires a second condition alongside the pattern match: **a relationship-context indicator in the same message**. This is a set of pronouns and relational framing words — Swedish *du, dig, ditt, jag, mig, vi, oss, förstår, älskar, behöver, måste, lyssna*;

English *you, your, I, me, we, us, understand, love, need, must, listen* — that together signal a direct interpersonal communication rather than descriptive or third-party speech.

The reliance on pronouns and other small function words as discriminators of stance and relational register draws on an established body of computational-linguistic work showing that these high-frequency, low-content tokens encode substantial information about speaker intent and discourse type [5]. To make the operational point concrete: Pennebaker showed that depressed writers use first-person singular pronouns substantially more frequently than non-depressed writers. The same empirical principle underpins the relational-context gate here — not the words being said, but the pronoun and stance frequencies around those words, are what distinguish an intimate-communicative register from a descriptive or third-party one.

The detection fires when *both* a behavioural pattern *and* at least two relational-context indicators are present in the same message (window size ±0 — the co-occurrence must be within one message boundary). The compound requirement has a measurable effect on precision: messages describing coercive control from the outside (case notes, news articles, clinical material) typically lack the first-second-person pronoun concentration of direct interpersonal communication, and tend to fail the context gate even when they contain the pattern vocabulary.

This is not a panacea. The context gate is a heuristic, and it can be fooled. But it is a transparent heuristic — an investigator reviewing a finding can see exactly which pattern matched, which context words were present, and why the message passed the gate. A detection that is questioned can be inspected, which is the property that matters for downstream use.



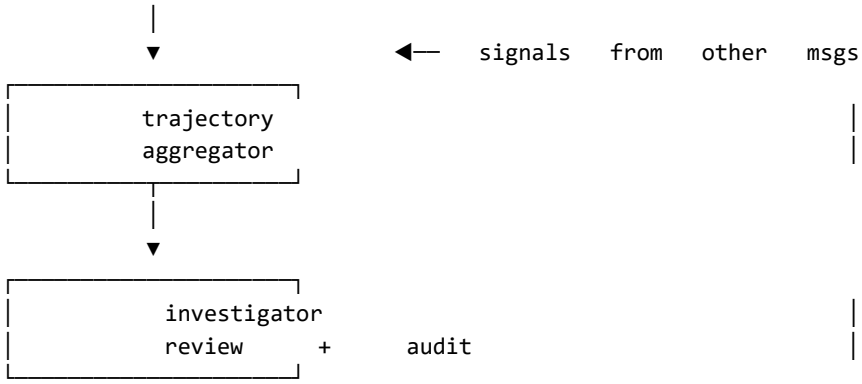


Figure 1. Pattern-plus-context decision flow. A message raises a category signal only when a behavioural pattern co-occurs with at least two relational-context indicators within the same message boundary. Individual signals are then accumulated across the conversation timeline before reaching investigator review.

From semantic category to detectable signal: the translation problem

The pattern-plus-context approach described above sits inside a deeper methodological question that this kind of forensic NLP is forced to confront: what does it mean to detect a semantic category in text at all?

Coercive-control categories are not defined by phrases. They are defined by what phrases do. Gaslighting is not the words "you're imagining things"; it is communication that systematically denies the addressee's reality with the cumulative effect of destabilising them. Love-bombing is not the word "soulmate"; it is rapid attachment-escalation deployed to manufacture dependence. The categories live at the level of intent and effect, not surface form. This is also why both the canonical keyword approach and an end-to-end LLM-only classifier struggle: the first treats the surface as identical to the meaning, and the second treats the meaning as something to be guessed at by a model whose guesses cannot be inspected.

A working forensic detector has to perform an explicit translation: from a semantic category, defined behaviourally and clinically, to a set of surface features in text that correlate with that category in ways that can be audited, reproduced, and challenged. The translation does not need to be perfect; it needs to be legible. Every step from category to flag must be traceable to evidence a reviewer can examine.

The architecture used in the implementation behind this article performs that translation in four layers, each contributing a different kind of surface evidence to the same semantic category.

Layer 1 — pattern lexicon. A curated set of canonical phrases historically associated with each category in clinical and survivor literature ("du inbillar dig" / "you're imagining things", "jag är allt du behöver" / "I'm all you need"). This layer is cheap, fast, and brittle. It catches the canonical formulations and misses everything else. A controlling partner who writes

"are you really sure that's what happened?" exhibits gaslighting without using any canonical phrase, and the lexicon will not see it.

Layer 2 — relational-stance gating via function words. The lexicon match alone fires too often in non-coercive contexts, so a second condition restricts firing to messages that exhibit the pronoun and stance patterns of direct intimate communication. The empirical basis is well established: Pennebaker showed that depressed writers use first-person singular pronouns substantially more frequently than non-depressed writers. The same kind of distinctive frequency holds for stance-taking — an interlocutor positioning themselves as the addressee's epistemic authority ("you don't really feel that", "you're misremembering") uses the second-person + verb-of-mental-state combination at frequencies that descriptive third-party speech does not match. The context gate operationalises this: not what the message says, but how it positions speaker and addressee relative to one another.

Layer 3 — embedding similarity for semantic-field membership. Even with the context gate, novel phrasings outside the canonical lexicon escape the first two layers. A message like "I never said that, you must be remembering it differently" lives in the same semantic field as canonical gaslighting but matches no pattern. To capture this without abandoning auditability, the implementation uses a controlled embedding-similarity rescue: each category has a set of human-curated exemplars; an unmatched message can still be flagged if its embedding sits sufficiently close to the category centroid. The rescue is restricted, scored explicitly, and produces a trace identifying which exemplars the message resembled — so the audit chain is preserved even when the trigger is not a canonical phrase. This layer trades some inspectability for breadth, which is a trade-off the design makes deliberately and visibly.

Layer 4 — co-occurrence and trajectory accumulation. Single-message classifications can be wrong in either direction. The categories themselves are accumulative: the evidentiary signature of coercive control is in which categories co-occur, in what order, and with what density across the conversation timeline. The aggregation layer is where individually-contestable findings combine into a trajectory. A single love-bombing flag is not evidence; love-bombing followed by isolation followed by gaslighting followed by economic control across weeks is.

These four layers are best understood as a chain of progressively more expensive but more semantically informed translations. Each layer fails in a characteristic way on its own. The pattern lexicon fails on novel phrasing. The context gate fails when controlling utterances appear in descriptive or third-party voice. The embedding layer fails when the message vocabulary is too sparse for similarity to be meaningful. The trajectory layer fails when a relationship simply has poor communication dynamics without crossing into control. In combination they constrain one another. A message that survives all four —

pattern match, relational stance, embedding proximity, and trajectory co-occurrence — is a finding the system can defend on the record.

It is worth being explicit about the limit this approach cannot cross. The detector does not understand gaslighting. It detects surface features that correlate with gaslighting in ways that can be audited. The semantic gap between what a category is and what the system finds is real and does not close. What the layered translation produces is not category recognition but category-shaped evidence: inspectable, reproducible, contestable findings that surface where human review is needed. The investigator and the prosecutor render the judgment; the system surfaces what to render judgment about. This is a narrower claim than "AI detects coercive control" — and it is the honest one.

An illustrative timeline

Consider the following compressed, fictitious message sequence from an intimate-partner-violence investigation. Individual messages appear ordinary; the accumulation is recognizable.

```
Week 1 (EN):  "babe you're literally my soulmate, I've never felt
              this way about anyone"
              [love_bombing]

Week 3 (EN):  "you don't actually need to see Sarah this weekend,
              she's been so fake with you lately"
              [isolation]

Week 5 (SV):  "du inbillar dig, jag sa aldrig så där på festen"
              ("you're imagining things, I never said that at
              the party")
              [gaslighting]

Week 6 (SV):  "om du inte hade provocerat mig hade det inte hänt"
              ("if you hadn't provoked me it wouldn't have
              happened")
              [blame_shifting]

Week 8 (EN):  "let me just handle the bank stuff, you don't need
              to worry about any of that"
              [economic_control]

Week 10 (SV): "din familj har alltid behandlat dig illa. Jag är
              den enda som bryr mig om dig på riktigt"
              ("your family has always treated you badly. I'm
              the only one who truly cares about you")
              [isolation]
```

Each message on its own is contestable. The first could be sincere hyperbole. The second could be a friend warning. The third could be honest memory disagreement. And so on.

Read as a trajectory — love-bombing early, isolation escalating, gaslighting and blame-shifting appearing mid-timeline, economic control entering by week 8, reinforcement of isolation by week 10 — it is a recognizable coercive-control pattern. The detection

system's role is to surface this trajectory: not to declare the relationship abusive on its own authority, but to give the investigator an ordered structure through which the chat evidence can be reviewed.

A structured output for the Week 5 message might look like this:

```
Signal:                                                    coercive_control
Category:                                                       gaslighting
Pattern matched:   "du   inbillar   dig"          #   "you're   imagining   things"
Language:                                                                sv
Context            indicators:            ["du",            "jag",            "sa"]
Confidence:                                                            0.78
Message window: Week 5
```

The investigator and, downstream, the prosecutor can see exactly which pattern fired, in what language, with which context indicators co-occurred. The `Confidence` value is derived deterministically from the pattern-match score plus the density of relational-context indicators in the same message window; it is not an LLM-produced probability and will reproduce exactly on re-run. If the detection is disputed, the basis for it is inspectable.

Why not LLM-only detection?

It is worth being precise about the objection to LLM-only detection, because the field has moved. A modern instruction-tuned large-language model can be asked to classify a message, return a category label, extract the relevant span, and articulate reasoning in natural language. The "black box" framing common a few years ago is no longer accurate at that surface level.

The objection that remains is narrower but, in forensic contexts, decisive: **reproducibility and audit transparency**. An LLM can explain why it flagged a message; it cannot reliably produce the same answer twice under identical inputs, and its priors — the training-data mixture that shapes what counts as "coercive" to the model — are not inspectable to an investigator or a defence counsel examining a specific flag after the fact. Sampling temperature, model version, prompt template, and provider-side silent updates all introduce variability that forensic evidence chains are not designed to absorb.

A concrete example: the utterance *"I can't live without you."* A classifier exposed during training to love-bombing literature may flag this as coercive based on accumulated prior. In ordinary hyperbolic affection between romantic partners, it is not a signal. A pattern-plus-context detector does not attempt to classify single messages in isolation. An utterance like this raises no individual flag; it gains evidentiary weight only when the same speaker's messages co-occur with other categories — isolation, gaslighting, and economic control across a trajectory. The investigator gets a direct, re-runnable answer to *"why was this message flagged?"* that does not depend on introspecting an opaque model's training data.

This is not an argument against LLM use in forensic workflows. A hybrid pipeline — pattern-plus-context as the audit-grade primary detector, an LLM as a secondary semantic scout for phrasings outside the lexicon or for investigator-facing summarization — is probably the practical optimum in production. The constraint that matters is not *which technique performs the detection*, but *whether the decision that reaches the investigator is inspectable and reproducible*. In a pattern-plus-context detector this property is structural. In an LLM-primary detector it must be engineered in, with prompt templates, model versions, and sampling settings logged into the evidence chain alongside the flagged content itself.

Legal framing matters

Pattern detection does not exist in a vacuum. The evidentiary value of coercive-control detection depends on whether the relevant jurisdiction recognises coercive control as a distinct offence. Several European jurisdictions now do: the Serious Crime Act 2015 s.76 in England and Wales created a standalone offence of "controlling or coercive behaviour in an intimate or family relationship," followed by Scotland's Domestic Abuse (Scotland) Act 2018 which broadened the framing further. In Sweden, the **Swedish Penal Code (Brottsbalken) Chapter 4, Section 4a — "Gross violation of integrity"** (*grov fridskränkning / grov kvinnofridskränkning*) covers repeated integrity-violating acts in domestic and intimate-partner contexts, and by its structure (repeated acts rather than isolated incidents) aligns more closely with accumulated trajectory evidence than with single-message flags. The scale of the phenomenon in Sweden is well-documented in national criminological surveys [6], which consistently show psychological and controlling forms of partner violence as underreported relative to physical forms — the exact category of behaviour that a pattern-plus-context detector targets.

These legal framings matter for how detection outputs are used downstream. In jurisdictions with a standalone coercive-control offence, the trajectory evidence that pattern detection surfaces can directly support a charging decision. In jurisdictions without such framing, the same evidence functions more as corroborating context for physical-assault or harassment charges. Forensic teams deploying this kind of detection should understand the local legal framework — detection rules are only as useful as the prosecutorial doctrine they feed into.

Beyond the domestic-abuse statutes, the EU AI Act (Regulation 2024/1689) imposes a transverse obligation on any AI system used by law-enforcement authorities for evidence evaluation: Annex III classifies such systems as **high-risk**, triggering explicit requirements for transparency, traceability, technical documentation, and human oversight [7]. Pattern-plus-context detection — by construction inspectable and deterministic — meets these obligations structurally, not by post-hoc explanation. This is not a marketing distinction; it is a design constraint that makes the approach fit for purpose in forensic contexts where the EU AI Act now applies.

Cross-cultural considerations

Coercive-control language varies across cultures. Affective and control registers — tender-consoling phrasing, deference framings, possessive language — are rarely direct translations across languages, and translation-first approaches tend to flatten exactly the cues that matter. A love-bombing utterance in the speaker's native language carries a different evidentiary weight than the same utterance in the victim's second language.

The detection architecture described here runs in the original language using native-language patterns, and reserves translation for display after a signal has already been raised. The detection and the translation live in different layers; the detection is never produced by translation. Lexicons beyond Swedish and English would need to be built with native-speaker input and, where possible, grounded in institutional sources documenting the local registers — not auto-translated from English seeds, which tends to import anglophone framings that do not land in the target language.

Where this approach fails

Honesty about limitations is more useful than marketing. Before the specific failure modes: **empirical validation on a held-out labelled IPV corpus is ongoing**. This article describes the architecture and its reasoning; it does not yet report measured precision and recall. Published numbers will follow once the evaluation infrastructure is in place. What follows are known qualitative failure modes of the approach.

Therapy and mental-health support contexts. Messages between therapists and clients, between peer supporters, or in online mental-health communities share a substantial vocabulary overlap with coercive-control patterns. *"You're not imagining it"*, *"your family hasn't supported you"*, *"I understand you in a way others don't"* — all can appear in legitimate support contexts. Dampener vocabularies reduce this but do not eliminate it, and a system used in intimate-partner-violence triage should not be used to screen mental-health messaging without adjusted thresholds and manual review.

Roleplay and fiction communities. Roleplay scenarios, fan-fiction communities, and some creative-writing spaces intentionally adopt intimate, possessive, or control-laden language between consenting adults. Detection systems cannot, from text alone, reliably distinguish consensual roleplay from coercion. Investigators working in contexts where this confusion is likely should expect high false-positive rates and calibrate their triage accordingly.

Ordinary relationship conflict. Healthy partnerships contain memory disagreements, blame discussions, isolation complaints ("your family is really frustrating"), and possessive expressions. The approach will flag some ordinary conflict messages. The answer is not to suppress them — it is to let investigator review distinguish pattern from noise, which is the operational assumption the system was designed around.

Non-textual coercion. Controlling behaviour enacted primarily through voice tone, physical presence, or visual imagery leaves fewer textual traces. Forensic pipelines that include speech-to-text can bring voice content into textual detection, but visual and physical coercion remain outside the system's reach by construction.

Operational considerations

Two practical questions tend to come up from teams evaluating this kind of detection for deployment: what does it cost to run, and what does it cost to maintain.

Runtime. Pattern matching using compiled regular expressions is effectively linear in message length. On commodity investigator hardware, per-message evaluation against the full multi-language pattern library runs in single-digit milliseconds for typical chat payloads; a full mobile-device extraction of several hundred thousand messages completes in minutes rather than hours. The context-gate check adds a bounded function-word scan that is cheap relative to the pattern match itself, and trajectory aggregation is a single pass over the resulting signal stream. The pipeline is not the bottleneck in a forensic workstation; OCR of message-screenshot exports and ingestion of large mobile extractions typically dominate end-to-end wall-clock time.

A separate question is confidence reliability — whether a flagged message can be relied on as a traceable, defensible finding. The four-layer architecture described in the section "From semantic category to detectable signal" is a structural step toward higher-confidence, traceable findings: no single layer is reliable in isolation, but a flag that survives all four layers (pattern, context, embedding proximity, trajectory presence) is documented to that depth in the audit record. Higher-confidence findings are those backed by multiple converging layers; lower-confidence findings are those resting on fewer, and the system surfaces this distinction in the score breakdown for each finding.

Lexicon maintenance. Patterns are not static. Coercive-control language drifts with vocabulary fashion — new slang, new platform conventions, emoji-mediated euphemisms, cross-cultural borrowings — and regional registers diverge on a shorter timescale than the underlying behavioural categories. Maintaining a production lexicon therefore requires a slow but steady curation cadence: periodic review of false-positive and false-negative samples from real cases, native-speaker validation when extending to additional languages, and a rollback mechanism for pattern changes that turn out to over-fire. Teams deploying this approach should plan for this as an ongoing operational commitment, not a one-time setup. A useful heuristic, from experience, is one lexicon-maintenance review per language per quarter, with out-of-cycle updates triggered by any false-negative or high-severity false-positive observed during casework.

Language extension. Adding a new language is architecturally trivial — drop in a new lexicon file — but practically labour-intensive. Native-speaker-validated pattern lists for affective and control registers rarely exist off-the-shelf and need to be constructed from

scratch or adapted from clinical-linguistic resources where those exist in the target language. Auto-translation from English is counterproductive for this specific task: affective idioms do not translate cleanly, and translated lexicons tend to import anglophone framings that do not match how coercive control is actually expressed in the target culture. A new language is realistically a few weeks of domain work plus a native-speaker review cycle, not an afternoon of `DeepL`-ing.

Reference implementation

A minimal version of the lexicon-matching engine described in this article is published under the Apache 2.0 license:

`github.com/stndrdsab/seva-lexicon-engine` *(to appear alongside this article)*

The repository contains the matching engine and a sample of demonstration lexicons, intended as a starting point for forensic developers and researchers to prototype pattern-plus-context detection on their own data. The full operational coercive-control lexicons and the context-gate implementation used in the deployed pipeline are not part of the reference release; the goal of the release is to make the approach reproducible and peer-reviewable, not to ship a turnkey product.

Closing

Coercive control is difficult to detect because the evidence is accumulative rather than punctual, distributed rather than concentrated, and composed of semantically ordinary language rather than specialised terminology. Keyword-only and LLM-only approaches each struggle with this structure, for different but related reasons.

A pattern-plus-context approach — five behavioural categories, native-language patterns with relational-context gates, inspectable output — is not a complete answer. It misses cases a more sensitive model would catch, and it misfires in contexts that share vocabulary with controlling relationships. But within its declared limits it produces findings that investigators can review, prosecutors can defend, and courts can inspect.

It is worth being explicit about the bound on this contribution. This article frames the architectural question — what does a layered, inspectable detection design look like for coercive-control patterns in chat data — and treats the empirical question of measured precision and recall on labelled corpora as a separate, ongoing track. A reliable forensic index is the goal; an audit-grade architecture is the prerequisite, and the work this article describes.

For forensic NLP applied to intimate-partner-violence and domestic-abuse investigations, that combination of limits and transparency is often the right trade to make.

• • •

References

[1] Subramani, S., Wang, H., Vu, H. Q., & Li, G. (2018). Domestic Violence Crisis Identification From Facebook Posts Based on Deep Learning.*IEEE Access*, 6, 54075–54085. Representative example of deep-learning approaches to IPV detection in social-media text; illustrates both the feasibility and the opacity of black-box classifier approaches in this domain.

[2] Stark, E. (2007). *Coercive Control: How Men Entrap Women in Personal Life*. Oxford University Press. Foundational work framing coercive control as a distinct pattern of intimate abuse rather than an aggravating factor of physical violence.

[3] Pence, E. & Paymar, M. (1993). *Education Groups for Men Who Batter: The Duluth Model*. Springer Publishing. The Power and Control Wheel developed at the Domestic Abuse Intervention Project in Duluth, Minnesota, is the precursor framework from which the operational categories described here partly derive.

[4] Hamberger, L. K., Larsen, S. E., & Lehrner, A. (2017). Coercive control in intimate partner violence. *Aggression and Violent Behavior*, 37, 1–11. Review of measurement approaches to coercive control in empirical IPV research.

[5] Pennebaker, J. W. (2011). *The Secret Life of Pronouns: What Our Words Say About Us*. Bloomsbury Press. Accessible treatment of function-word frequencies — pronouns, articles, prepositions — as stylistic and relational discriminators in written and spoken text; the established empirical basis for using small function words as context indicators rather than content indicators.

[6] Brottsförebyggande rådet (2014). *Brott i nära relationer: En nationell kartläggning* (Report 2014:8). Stockholm: Brå. Swedish national criminological survey of intimate-partner violence prevalence and characteristics, including coverage of psychological and controlling forms; the empirical backdrop against which Brottsbalken 4:4a is applied.

[7] European Union (2024). Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act), Annex III. Official Journal of the European Union. Establishes the high-risk classification for AI systems used by law-enforcement authorities in the evaluation of evidence, and sets out the transparency, technical-documentation, and human-oversight obligations referenced here.

[8] Dragiewicz, M., Burgess, J., Matamoros-Fernández, A., Salter, M., Suzor, N. P., Woodlock, D., & Harris, B. (2018). Technology facilitated coercive control: domestic violence and the competing roles of digital media platforms. Feminist Media Studies, 18(4), 609–625. Reframes coercive control for the digital-communication era and motivate the analysis of chat-mediated control patterns specifically — directly relevant to the textual-detection question.

[9] Henry, N. & Powell, A. (2018). Technology-Facilitated Sexual Violence: A Literature Review of Empirical Research. Trauma, Violence, & Abuse, 19(2), 195–208. Empirical synthesis of technology-facilitated abuse, situating textual coercion alongside image-based and harassment-based modalities; complements the Brottsförebyggande rådet criminological backdrop with an international research synthesis.

About the Author

Andreas Antonsen is the founder of STNDRDS AB, a Swedish compliance and forensics software company (stndrds.se). The examples in this article are fictitious and constructed for illustration; they are not drawn from any real case.

Weaponizing Intelligence: AI in the Hacker's Arsenal by *Igor Korkin, Yuriy Tumanov, and Oksana Dokuchaeva*

Abstract: Generative AI is changing the economics of cybercrime. It can help attackers scale social engineering, produce synthetic media, accelerate code experimentation, and prepare attack logic that appears only at runtime. At the same time, AI-enabled applications introduce a fresh attack surface: prompt injection, poisoned retrieval data, sensitive information disclosure, improper output handling, excessive agency, and a newer risk: abuse of a trusted AI assistant through a compromised user endpoint or authenticated session.

This article translates those risks into a practical, defensive workflow for security engineers, pentesters, AppSec teams, blue teams, and researchers. The focus is authorized testing, evidence collection, and engineering fixes — not exploit or attack recipes.

Keywords: AI security; LLM security; AI agents; trusted AI assistant abuse; AI-mediated data exfiltration; prompt injection; data poisoning; deepfakes; AppSec; threat modeling; generative AI; defensive testing; MITRE ATLAS; OWASP LLM Top 10.

Legal and Ethical Note

This article is intended only for authorized security testing, defensive research, lab work, awareness training, and improvement of security controls. It does not provide exploit code, malware instructions, evasion playbooks, or operational guidance for unauthorized activity. Any AI security test, phishing simulation, deepfake detection exercise, or LLM application assessment must have explicit permission, a documented scope, and legal review where required.

What You Will Learn

- How generative AI changes attacker economics without replacing human skill.
- Why LLM applications should be modeled as interfaces between users, data, tools, and business actions.
- Which defensive tests can be performed safely in a lab without publishing harmful payloads.
- How to turn AI red-team findings into blue-team telemetry, AppSec fixes, and governance decisions.

Introduction: The Model Is Now Part of the Attack Surface

AI is no longer just a helper that summarizes logs, classifies alerts, or drafts boilerplate code. It has become an amplifier for adversaries who want to scale social engineering, accelerate reconnaissance, generate convincing content, or reduce the skill needed to prepare attack components. The important point is not that AI magically creates elite hackers. It is that AI reduces friction. It helps operators move faster, personalize better, and test more variants.

At the same time, AI systems themselves have become targets. LLM applications can be manipulated through prompt injections, poisoned through untrusted data, abused through excessive agency, or queried in ways that reveal sensitive information. This double role makes AI security different from traditional AppSec: the model is both part of the product and part of the decision-making surface. Also, the trusted AI assistant can be abused without attacking the model provider at all. If malware controls the user endpoint, browser session, local files, or automation channel, it may interact with ChatGPT, Claude, or a similar assistant as the legitimate user and turn the assistant into an indirect data, action, or reputation channel.

The main question for defenders is not whether AI is good or bad. A more suitable question is: *Where does AI change the threat model, which assumptions become unsafe, which assistant state must be protected, and which controls can be tested with evidence?*

Why AI Changes the Threat Model

Traditional attacks already combine automation, social engineering, and software exploitation. Generative AI does not replace those categories; it changes their cost, speed, and scale. A phishing operator can draft more natural messages, localize them into multiple languages, and tune the tone fit for a role or within a business context. A fraud actor can combine synthetic voice, image, and text to make a request look more trustworthy than a plain email. A malware developer can use code assistants to prototype small pieces of logic faster, although complex offensive tooling still requires human intent, testing, infrastructure, and operational discipline.

For defenders, the practical shift is that some old controls remain necessary but are no longer sufficient. Grammar mistakes are a weak phishing signal. Static signatures lose value when scripts are generated dynamically. Keyword filters break down when content is tailored to context. The threat model must now include generated instructions, synthetic identities, model outputs, data provenance, assistant memory, connected files, tool permissions, spending limits, and the privileges granted to AI-powered workflows.

Figure 1 shows that AI occupies five distinct roles in cybersecurity at once: AI as a Shield (a defensive asset that strengthens detection and response), AI as a Weapon (an offensive accelerator for social engineering and attack preparation), AI as a Target (an LLM

application that can be directly attacked), Abuse of a Trusted AI Assistant (the user's authenticated assistant turned into an indirect action and reputation channel from a compromised endpoint), and Data Leakage via AI (the assistant used as an indirect path to exfiltrate the user's own sensitive context).

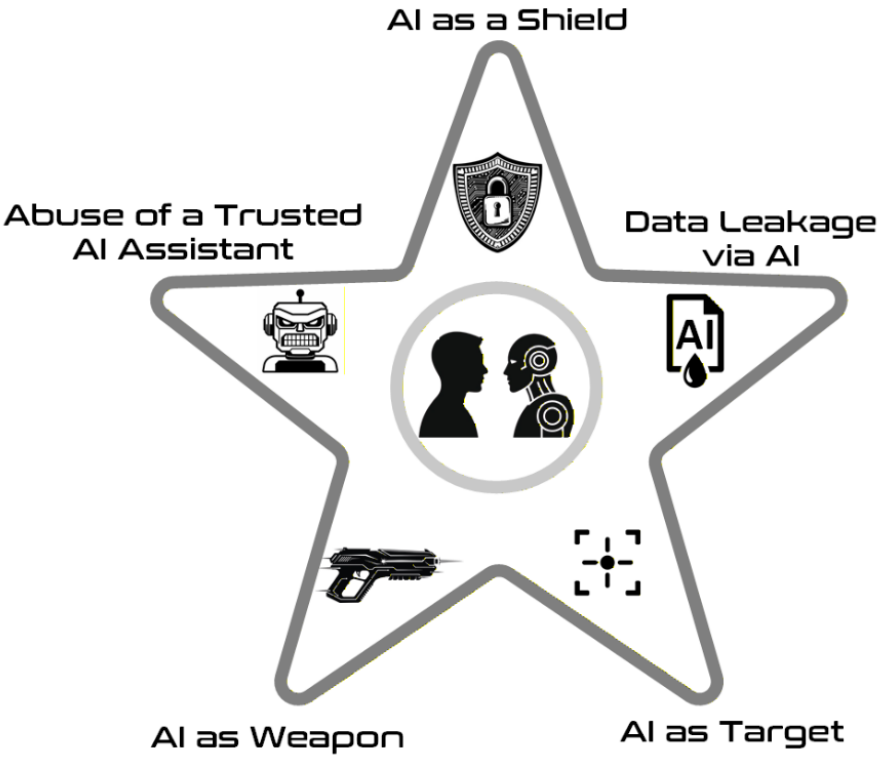


Figure 1. Five roles of AI in cybersecurity.

AI as an Offensive Accelerator

Social Engineering That No Longer Smells Like Phishing

The most visible abuse case is AI-assisted social engineering. LLMs can produce polished messages, plausible business narratives, translations, and many variants at a speed that manual operators cannot match. The risk is not only grammar quality; it is contextual adaptation. A message that references a project, a recent policy, or an internal workflow can slip past the recipient's usual suspicion radar, especially if it is combined with synthetic voice or video.

The defensive answer is to measure process controls, not just user intuition: payment verification, out-of-band confirmation for sensitive requests, mailbox telemetry, attachment detonation, link rewriting, and user-reporting response time.

Code and Vulnerability Work: Faster & Effective

A second area is AI-assisted code and vulnerability work. Generative models can explain vulnerable code, suggest risky logic, transform scripts from one language to another, and help an operator experiment faster. This does not mean every low-skill actor can immediately run advanced intrusions. It means experimentation becomes cheaper.

Security teams should track this as a capability multiplier rather than a magic weapon. The practical response is boring but effective: dependency scanning, secret detection, SAST and DAST coverage, secure review for high-risk modules, and automated triage that separates exploitable evidence from noise.

Just-in-Time Attack Logic: Payloads Born at Runtime

A third area is just-in-time attack logic. In this pattern, a malicious component may contain only a high-level goal or orchestration logic, while code is generated later through an external or local model. The final script may not exist until execution time, reducing the value of pure static detection.

A safe defensive lesson follows from this pattern: organizations need behavioral monitoring, egress controls, process telemetry, and detection of unusual model-service interactions. The blue team should look for sequences such as repeated prompt-like API calls, generation of executable scripts in temporary locations, unexpected interpreter launches, and privilege-sensitive actions that follow model output. No single signal is enough, but correlation can turn a blurry shadow into a useful detection story.

Abuse of a Trusted AI Assistant

The fourth role is subtle because it does not require a direct attack on the AI provider, the model weights, or the system prompt. The attacker abuses the trust relationship that already exists between the user and the assistant. Many professionals now use assistants as long-term research partners, coding aides, writing copilots, and project notebooks. That means the assistant may know drafts, private reasoning, customer context, internal terminology, pasted logs, code fragments, files, and sometimes credentials or tokens that should never have entered a chat in the first place.

From a defender's perspective, this scenario is closer to endpoint abuse than to a classic prompt-injection attack. Malware or an intrusive script on the user's workstation can operate through an authenticated browser session, local automation, clipboard access, browser extensions, or connected agent tools.

The requests may look legitimate to the AI service because they are sent by the real user account. The security boundary has shifted from "can the model be tricked?" to "what can a compromised user environment make the assistant reveal or do?"

AI-mediated Data Exfiltration and Credential Harvesting

In this pattern, the assistant becomes an indirect search and summarization layer over the user's own knowledge. Instead of stealing a document directly, malware may ask the trusted assistant to summarize recent project discussions, list sensitive topics, search connected files, or reveal API keys, passwords, tokens, and configuration fragments that appeared in prior chats or accessible documents. The assistant is not necessarily vulnerable; the weakness is that the attacker is borrowing the user's authenticated context.

Defensive controls should treat assistant-accessible context as sensitive data. Useful measures include connector access control, DLP rules for prompts and retrieved content, canary secrets in lab tests, device posture checks, browser-session protection,

egress monitoring, and clear rules that prohibit placing real credentials into assistant memory, prompts, or retrieval corpora.

AI-mediated Unauthorized Action: Agent Behavior Hijacking

The risk increases when the assistant can call tools. If an agent can send email, create tickets, change documents, open pull requests, schedule meetings, or invoke APIs, endpoint malware may steer those capabilities through the user's session. This is not only about text generation; it is about business actions. A malicious local process does not need to defeat the model if it can push the assistant toward a legitimate-looking request.

High-impact actions should therefore be authorized outside the model by deterministic policy checks, scoped permissions, transaction previews, step-up authentication, and human approval. The assistant may draft or recommend, but it should not be the only authority for sending, publishing, deleting, paying, deploying, or changing access rights.

AI Resource Draining: Token and Quota Abuse

An attacker may also use the assistant as a cost and availability target. A loop that generates long prompts, requests expensive reasoning, uploads large files, or triggers repeated tool calls can drain tokens, burn subscription quotas, increase cloud cost, and reduce availability for legitimate users. For an organization, this becomes a low-noise denial-of-wallet problem rather than a spectacular intrusion.

Defenders should monitor token volume, request length, file-ingestion patterns, unusual session duration, repeated failures, and tool-call bursts. Per-user quotas, tenant-level spend limits, anomaly alerts, and automatic throttling are practical controls that can be validated safely in a test tenant.

AI-mediated Reputation Damage

A trusted assistant can also damage a person or organization by producing and distributing content from a legitimate account. If malware can make an agent post to a social network, send mail, publish a document, or comment in a repository, the visible result may look like the user intentionally wrote it. The damage is reputational even when no secret is stolen.

Controls should distinguish drafting from publishing. External posting, mass messaging, legal or HR communications, and public repository actions should require explicit confirmation, a clear preview, account-level audit trails, and quick revocation paths. The blue team should be able to answer who initiated the action, from which device, through which assistant, and with which approval event.

AI Context Poisoning

Finally, malware does not have to read the assistant's context to cause harm. It may write false information into local notes, project documents, memory stores, retrieval corpora, or conversation history so that the assistant later gives wrong advice. This is an integrity attack against the user's future decisions. A poisoned context may slowly redirect research, alter a report, hide an important warning, or make a defensive workflow trust a false assumption.

Context integrity requires provenance, versioning, ownership, review gates, and visible source attribution. Assistant memory and retrieval indexes should be treated as security-relevant state, not as harmless convenience features. When a user or team relies on AI for research, code review, or incident response, the sources behind important claims must remain inspectable and correctable.

AI Systems as Targets

Prompt Injection: Instruction and Data Collide

When an organization deploys an LLM application, it creates a new interface between users, data, tools, and business actions. This section focuses on AI systems as direct targets, while the previous section covered abuse of a trusted assistant through the user's own environment. Prompt injection is one of the most important risks because it exploits the model's tendency to process instructions and data in the same linguistic space. Direct prompt injection comes from user input. Indirect prompt injection may arrive through documents, web pages, emails, tickets, or other content that the model reads on behalf of the user.

Impact depends on agency. A chatbot that answers public FAQs has a different risk profile from an agent that can read internal documents, call APIs, create tickets, or trigger workflow actions.

Poisoned Knowledge Bases and the AI Data Supply Chain

Data poisoning attacks target the integrity of training, fine-tuning, retrieval, or feedback data. In enterprise LLM applications, the more common version may involve poisoned retrieval content, low-quality knowledge bases, malicious instructions hidden in documents, or compromised data pipelines. Treat AI data like supply-chain material: it needs source validation, ownership, change history, signing where feasible, and review gates for high-impact corpora.

Sensitive Data Disclosure and Model Extraction

Sensitive information disclosure and model extraction are confidentiality problems. An LLM application may expose secrets through unsafe retrieval, overbroad tool permissions, chat history leakage, poor prompt design, or inadequate output filtering. An attacker may also query a model repeatedly to infer proprietary behavior or extract training fragments. Practical controls include retrieval access control, tenant isolation, rate limits, anomaly detection, logging of high-risk prompts, output classifiers, and clear rules about which data may enter prompts or training pipelines.

Multimodal and Adversarial Tricks

The attack surface does not stop at text. A malicious instruction can be hidden in an image, document metadata, encoded text, or a multilingual payload that looks harmless to a human reviewer. This is why LLM security cannot be reduced to a single prompt template. It requires design constraints around what the model can read, what it can

decide, what tools it can call, and what deterministic checks must approve before any action is executed.

A Safe Defensive Workflow for Security Teams

A useful AI security assessment can be performed without publishing exploit payloads or testing against real victims. Start with an inventory of AI-enabled features. For each feature, identify the model, prompts, retrieval sources, connected tools, data classification level, authentication context, assistant memory, chat history, browser or desktop integration, billing quota, and possible business actions. Then draw a data-flow diagram that separates user input, untrusted external content, internal documents, model context, output post-processing, downstream systems, and assistant-accessible personal or project state.

Next, define safe lab tests. For prompt injection, use benign test strings that attempt to change task boundaries without asking for harmful output. For data poisoning, create a separate lab knowledge base and insert test documents with conflicting instructions or false facts to verify whether the application blindly trusts retrieval content. For excessive agency, test whether the model can initiate actions without deterministic validation or human approval. For trusted-assistant abuse, use a controlled test account, synthetic secrets, canary project notes, and non-public draft messages rather than real credentials, real customers, or real publications.

The output should not be a dramatic list of scary prompts. It should be a control matrix that links threat, expected behavior, observed behavior, evidence, and remediation. That format makes the work reproducible and helps engineering teams fix design problems instead of arguing about whether a demo was impressive.

Table 1. Threat-to-control matrix for an LLM-enabled application

Threat area	Safe defensive test	Expected control	Evidence to collect
Prompt injection	Benign attempt to override task boundaries in user input or retrieved text	Boundary enforcement, output validation, no unauthorized tool call	Prompt, model response, blocked action log, policy decision
Data poisoning	Lab document with conflicting instructions in a test retrieval corpus	Source validation, corpus review, retrieval filtering	Document provenance, retrieval trace, answer citation behavior
Sensitive information disclosure	Synthetic secret placed in a restricted source	Access control, redaction, no leakage across sessions	Retrieval ACL logs, redaction logs, response transcript
Excessive agency	Request that would trigger an external action	Deterministic policy gate and human approval for high-impact actions	Tool-call logs, approval record, denied action event
Model extraction / abuse	Repeated high-volume probing pattern in a test environment	Rate limits, anomaly detection, account throttling	Request volume, session identifiers, throttle decisions

Threat area	Safe defensive test	Expected control	Evidence to collect
AI-mediated data exfiltration	Synthetic project note or canary token placed in an assistant-accessible source	Connector ACLs, device posture checks, session isolation, DLP/canary alerting	Prompt/tool-call log, retrieval trace, canary alert, endpoint telemetry
AI-mediated unauthorized action	Lab request that tries to make the assistant send, post, update, or call a connected tool	Deterministic action gate, scoped permissions, step-up confirmation, human approval	Tool-call log, approval record, denied action event, account audit trail
Resource / token draining	High-volume benign prompt loop in a test tenant	Quotas, spend alerts, rate limits, session anomaly detection	Token usage, cost alert, throttle decisions, session identifiers
AI-mediated reputation damage	Attempt to publish or send a synthetic embarrassing message from a controlled account	Publishing preview, explicit confirmation, external-action policy, revocation path	Draft/post log, confirmation record, blocked action event
AI context poisoning	False memory, note, or lab document inserted into assistant-accessible context	Provenance, versioning, review gates, memory-change audit	Change history, source owner, retrieval trace, correction workflow

Blue-Team Detection Ideas

Useful monitoring is behavioral, not only textual. Track tool-call attempts, denied actions, retrieval sources used in answers, repeated prompt patterns, unusual token volume, high-risk file ingestion, unexpected calls to model APIs, interpreter launches after model output, spikes in failed policy checks, assistant-memory changes, connector access from unusual devices, and external publishing attempts through agents. The goal is not to flag every weird prompt. The goal is to detect risky sequences that connect model interaction to business impact.

Table 2. Suggested metrics for AI-security control validation

Metric	Why it matters	Example target
Unauthorized tool-call attempts blocked	Measures agency control	100% for high-impact actions in scoped tests
Responses with unsupported claims	Measures retrieval and citation quality	Decrease after retrieval filtering and answer policy updates
Synthetic secret leakage	Measures confidentiality guardrails	0 leaks in scoped canary tests
High-risk prompts reviewed	Measures monitoring coverage	All critical alerts triaged within a defined SLA
Corpus items with verified provenance	Measures AI data supply-chain maturity	Coverage target defined by the data owner
AI-mediated canary access attempts	Measures whether assistant-accessible context can be abused from a compromised endpoint	0 successful leaks; all canary access attempts alerted
Unexpected assistant memory/context changes	Measures context-integrity controls	100% reviewed for high-impact assistants

Metric	Why it matters	Example target
Token and spend anomalies	Measures resource-draining detection	Alert and throttle within a defined SLA
External publishing actions requiring approval	Measures reputation and action guardrails	100% enforced for public posting and sensitive workflow changes

Hardening Checklist

- Do not treat a system prompt as a security boundary.
- Authorize every tool call outside the model, using deterministic checks.
- Validate model output before it becomes code, a query, a ticket, a command, or a business decision.
- Separate public content, internal documentation, customer data, incident records, and secrets.
- Log retrieval sources and tool calls, not only final answers.
- Use canary data and lab corpora to test leakage and poisoning safely.
- Give high-impact actions human approval or a strict policy gate.
- Treat assistant memory, retrieval indexes, connected files, and chat history as security-relevant state.
- Require explicit confirmation for data export, public posting, account changes, payment, deployment, and access-right changes.
- Monitor token consumption, tool-call bursts, connector access, and assistant-context modifications for anomaly detection.
- Use synthetic secrets and canary project notes to test assistant-mediated leakage without exposing real data.
- Make AI threat modeling part of SDLC, not a last-minute compliance sticker.

Legal and Governance Considerations

AI-enabled incidents complicate attribution and evidence handling. Generated content may be produced in real time, routed through APIs, mixed with human edits, or stored only in partial logs. If a case involves synthetic media, phishing, unauthorized data access, or AI-assisted malware preparation, investigators need reliable timestamps, account identifiers, API logs, model-service logs, content hashes, and chain-of-custody procedures. Legal qualification varies by jurisdiction and should not be reduced to a universal checklist.

For organizations, the governance question is practical: who owns risk decisions for AI-enabled functionality? Product teams' own business intent. Engineering teams own implementation. Security teams own threat modeling and control validation. Legal teams own regulatory interpretation. Data owners own classification and permitted use. AI-platform owners must also define who owns assistant memory, connectors, tool permissions, billing quotas, logs, and emergency revocation. Without this ownership

map, AI security becomes a fog machine where every team assumes another team is holding the flashlight.

Limitations

This article is a practical synthesis, not a claim that one framework solves all AI security problems. Prompt injection remains difficult because natural-language systems do not separate instruction and data as cleanly as traditional parsers. Synthetic-content detection is useful but imperfect, especially when attackers can modify, compress, or re-render media. Watermarking may help with provenance, but it does not replace authentication, authorization, or forensic readiness. Trusted-assistant abuse is also strongly dependent on provider behavior, local endpoint security, browser isolation, available connectors, and user workflow. Model behavior changes across providers, versions, fine-tuning choices, system prompts, and retrieval content, so controls must be tested against the actual deployed system.

Conclusion

AI has become both an accelerator for adversaries and a new attack surface for defenders to secure. The same technology that helps teams summarize code or automate workflows can also scale phishing, synthetic media, vulnerability analysis, and dynamic attack preparation. LLM applications introduce specific risks such as prompt injection, poisoned data, sensitive information disclosure, improper output handling, excessive agency, and abuse of a trusted assistant through the user's authenticated environment.

The defensive answer is neither panic nor blind trust in model safety banners. Security teams need threat models, data-flow diagrams, lab-safe abuse cases, deterministic controls, telemetry, and measurable validation. The most important design principle is simple: keep high-impact actions outside the model's sole authority and treat user input, retrieved content, assistant memory, and model output as untrusted until verified. Organizations that combine AppSec discipline, AI data governance, endpoint hardening, monitoring, and legal readiness will be better prepared for the LLM era than those waiting for a silver bullet to materialize in the server rack.

On The Web

- Hakin9 Article Submission Guide - https://hakin9.org/write-for-us/
- Hakin9 About / Formatting Requirements - https://hakin9.org/about/
- OWASP Top 10 for Large Language Model Applications - https://owasp.org/www-project-top-10-for-large-language-model-applications/
- OWASP GenAI Security Project: LLM01:2025 Prompt Injection - https://genai.owasp.org/llmrisk/llm01-prompt-injection/
- MITRE ATLAS - https://atlas.mitre.org/
- NIST AI RMF Generative AI Profile - https://www.nist.gov/publications/artificial-intelligence-risk-management-framework-generative-artificial-intelligence

Bibliography / References

[1] OWASP Foundation. OWASP Top 10 for Large Language Model Applications.

[2] OWASP GenAI Security Project. LLM01:2025 Prompt Injection.

[3] MITRE. ATLAS: Adversarial Threat Landscape for Artificial-Intelligence Systems.

[4] NIST. Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile, NIST AI 600-1, 2024.

[5] Yao, Y., Duan, J., Xu, K., Cai, Y., Sun, Z., & Zhang, Y. A Survey on Large Language Model Security and Privacy: The Good, The Bad, and The Ugly. High-Confidence Computing, 2024.

[6] Jabir, R., Le, J., & Nguyen, C. Phishing Attacks in the Age of Generative Artificial Intelligence: A Systematic Review of Human Factors. AI, 2025.

[7] Kotenko, I. V., Saenko, I. B., Lauta, O. S., Vasilyev, N. A., & Sadovnikov, V. E. Attacks and Defense Methods in Machine Learning Systems: Analysis of Modern Research. Cybersecurity Issues, 2024.

[8] Mudarova, R. M., & Namiot, D. E. Countering Prompt Injection Attacks on Large Language Models. International Journal of Open Information Technologies, 2024.

[9] Namiot, D. E. On Cyberattacks Using Artificial Intelligence Systems. International Journal of Open Information Technologies, 2024.

[10] Efremova, M. A., & Russkevich, E. A. Deepfake and Criminal Law. Bulletin of the Kazan Law Institute of MIA Russia, 2024.

[11] Garbuk, S. V. A Special Security Model for the Creation and Application of Artificial Intelligence Systems. Cybersecurity Issues, 2024.

[12] Kartskhiya, A. A., & Makarenko, G. I. Legal Problems in Using Artificial Intelligence in Russia. Pravovaya Informatika, 2024.

[13] Samonov, A. V., & Burova, I. O. Methodology for the Development of Automated Software Code Generation Tools by Fine-Tuning Large Language Models. Cybersecurity Issues, 2024.

About the Authors

Igor Korkin is a Ph.D. cybersecurity researcher specializing in software and OS security, trusted computing, and advanced offensive and defensive security technologies.

Yuriy Tumanov is a Ph.D. cybersecurity specialist focused on application security, Secure SDLC, vulnerability analysis, DevSecOps, Kubernetes security, threat modeling, and AI security.

Oksana Dokuchaeva is an information security expert focused on analytical support, internal control, and the coordination of security processes and operations.

eForensics - Forensically Analyzing Binary Files using OpenAI's ChatGPT by *Israel Torres*

OpenAI's ChatGPT has grown to be one of my favorite daily tools for doing meta-work. For example digital forensics that I would run a first pass before moving to specialized tooling. Before ChatGPT I would run a first pass with tools in my tool bin, usually with a few automation scripts that I've written over time, or even just my own scripts and executables that I've built to crack the armor on what I am triaging. This would help me get a quick report on my next steps to make better decisions and not waste time. These same tools are what I would use for challenges and CTFs, and the like.

With ChatGPT I can now not only start a fast pass quickly, but also further interrogate ChatGPT via prompting to further extract information that is helpful for more analysis and report building. In this article, I'll show you how.

Introduction

In this limited eForensics Magazine AI series, we've so far featured "Getting Started in Cyber Security Forensics with AI and ChatGPT" [1] and "How to Forensically Analyze Suspicious Files on macOS Tahoe using OpenAI's ChatGPT" [2], which focused on flat files. Now we'll be pivoting to forensically analyzing **binary** files using ChatGPT.

Continuing the same running concept we used with the flat files, we aren't just prompting ChatGPT for suggesting forensics tool or even building us tools, or walking along coding a tool to analyze our suspected files; instead, we are using ChatGPT (using ChatGPT 5.4 Thinking) [3] directly to run its forensics analysis know-how on binary files we upload. Note that there are guardrails and sensitive measures to where ChatGPT will protect itself (and its internal system) with, and by default it treats any uploaded file as an untrusted payload. If not done correctly, the file will be rejected for upload.

Analyze a suspected **binary** file using ChatGPT

Previously we worked with flat file analysis, this time we are analysing a binary file. In comparison, flat files are often easily readable with 'cat' or a simple text editor such as TextEdit or Sublime, no special encoding for the most part. Binary files (for this static analysis use case) on the other hand, often require a hex editor such as hex fiend, 'xxd', 'hexdump', which are tools that convert the binary sample to something readable for the analyst/tooling. The output defaults to hexadecimal encoding. Note: Out of scope for this example use case are using disassembly/decompilation, etc. tooling for deeper, more

meaningful analysis and reporting that would fall more into reverse engineering or reversing for short. Perhaps we'll have another article for that in the future.

What is in scope is getting as much information ChatGPT will provide from the binary sample uploaded, which is suspected to be suspicious and warranted for analysis.

ChatGPT example:

In this example, we begin by selecting a file [4] from our alert-bin and then zipping the file. Trying to upload this file (regardless of extension give) without zipping it first will cause the upload to be blocked with no explanation other than a red message box stating "Unknown error occurred" as seen in Figure 1 below:

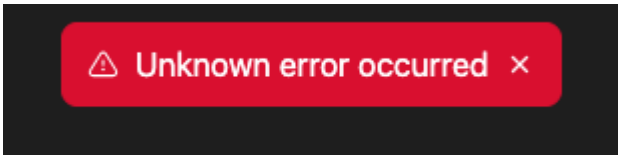


Figure 1 - Upload Failed

Clicking the "x" will close the red message box, and it will appear as if nothing had been prompted or uploaded.

After zipping the file (without a password), the process is smoother, and ChatGPT begins to perform its analysis as seen in Figure 2 below:

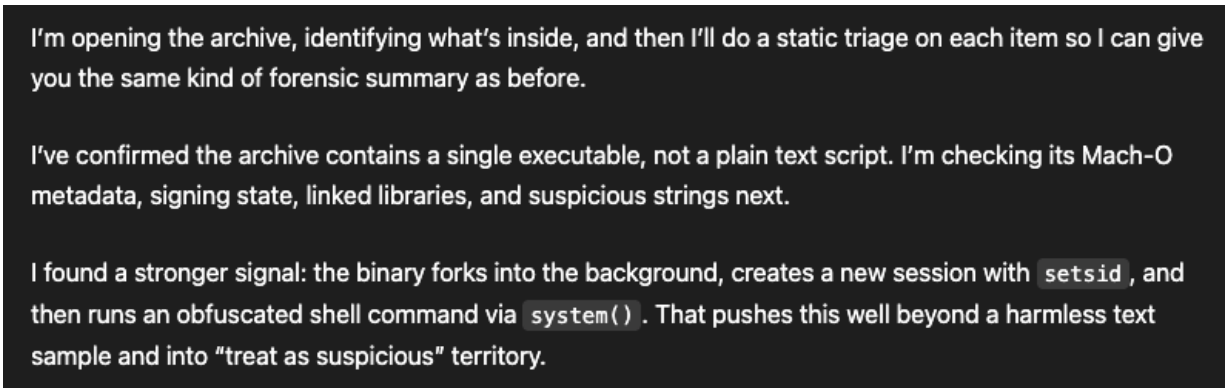


Figure 2 - Analysis Begins

ChatGPT is familiar with our previous exercise with the flat files as mentioned in the previous article, and knows we want to get similar information from this new binary file as we did with the flat file. No extravagant prompting was needed, ChatGPT has learned what we want and how we want it, and now is on the way to give us the result in the same fashion, as seen in Figure 3 below:

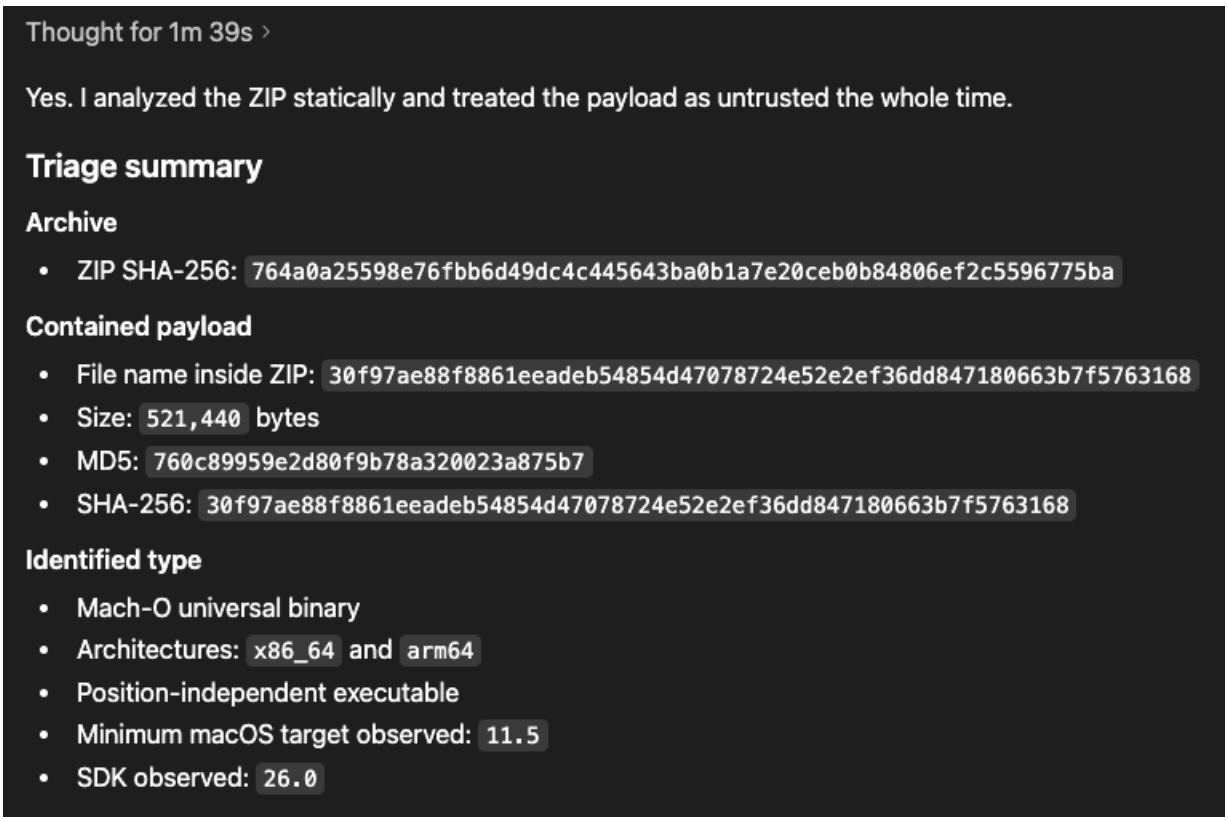


Figure 3 - Triage Summary (ChatGPT)

For comparison here is the Basic Properties from the Details pane using VirusTotal in Figure 4 below, minus the hash variation, which if desired we could prompt ChatGPT to generate a similar template with said hash variation and get most of them other than the proprietary one(s):

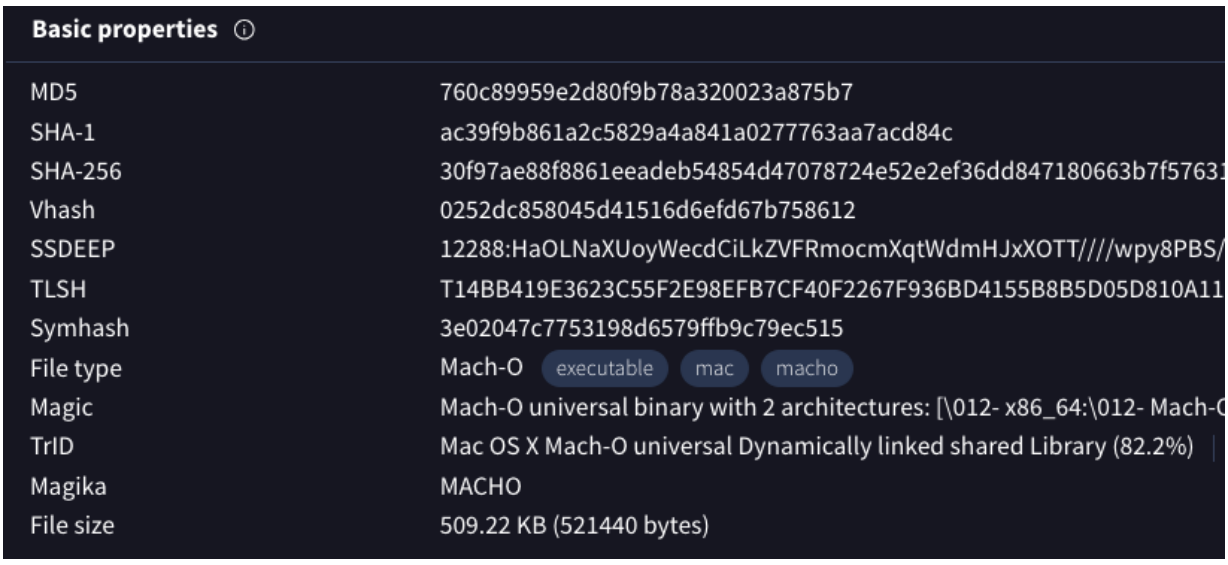


Figure 4 - Basic Properties (VirusTotal)

As we continue our shared analysis with ChatGPT, there is a lot more information ChatGPT is sharing as it dives deeper into the binary and in its static inspection begins to list out its "High-confidence findings" in the binary's behavior and already is leaning to it being **suspicious** and also states why and what is set to happen during its execution as seen in Figure 5 below:

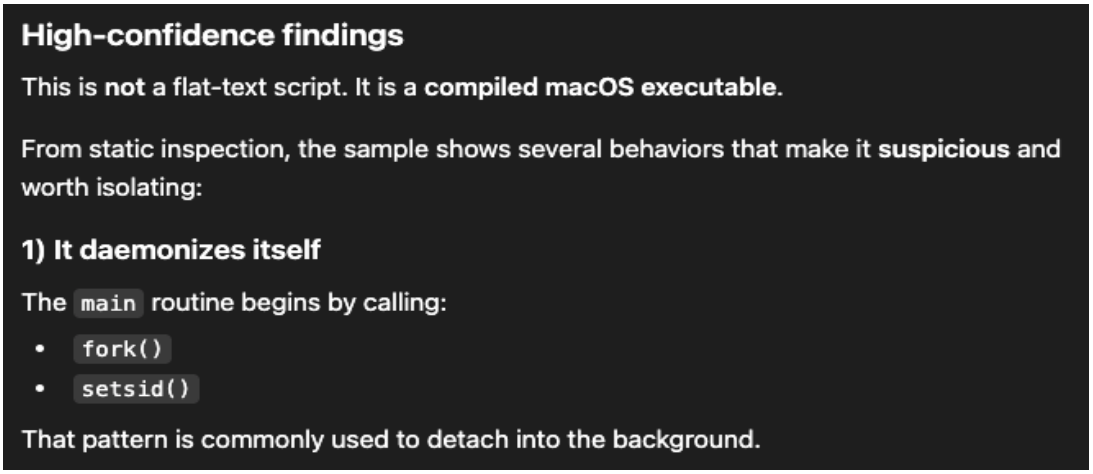


Figure 5 - High-confidence findings

For brevity, I've listed below all the high-confidence findings:

1) It daemonizes itself

2) It executes an obfuscated shell command

3) It has filesystem copy / delete logic

4) It can execute shell commands in more than one way

...and other notable artifacts, with examples found in the binary, and explanations as to what this could mean.

1) Embedded error string

2) Randomness source

3) Entitlements present in the file

ChatGPT also generates its risk assessment as **suspicious** and explanation as seen in Figure 6 below:

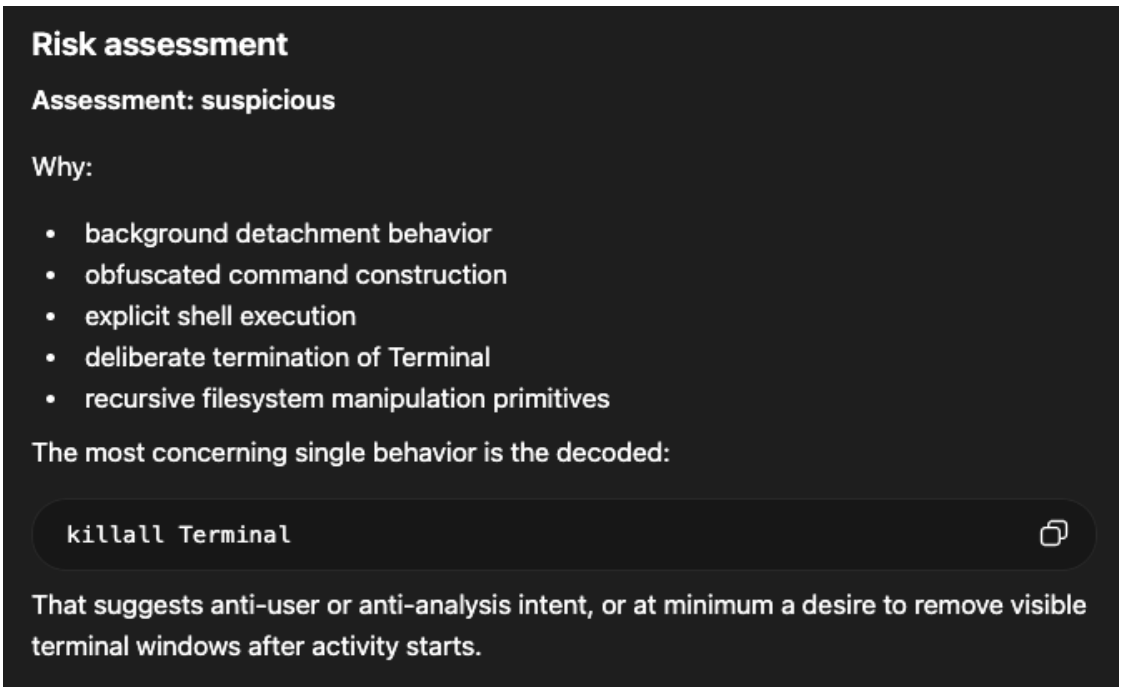


Figure 6 - Risk Assessment

In Figure 7 below, ChatGPT adds a summation of its confident findings, as well as its limitations based on further analysis techniques and mechanisms.

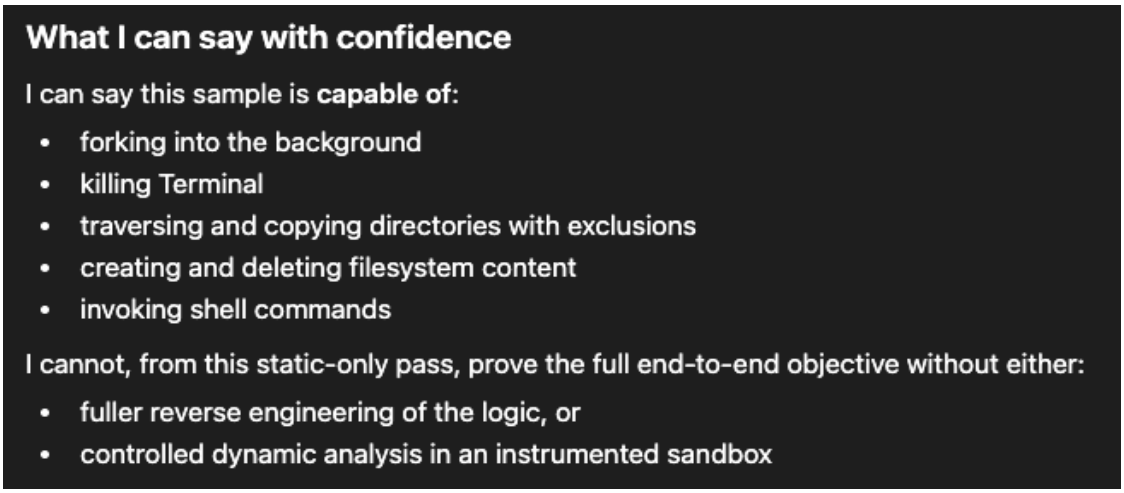


Figure 7 - Confident Findings

Finally, in Figure 8 we see that ChatGPT also pushes a directive for next steps for the analyst to take for dynamic analysis where the analyst would run the executable binary on an isolated testbed to capture and monitor further data on its execution and behavior when it instantiates.

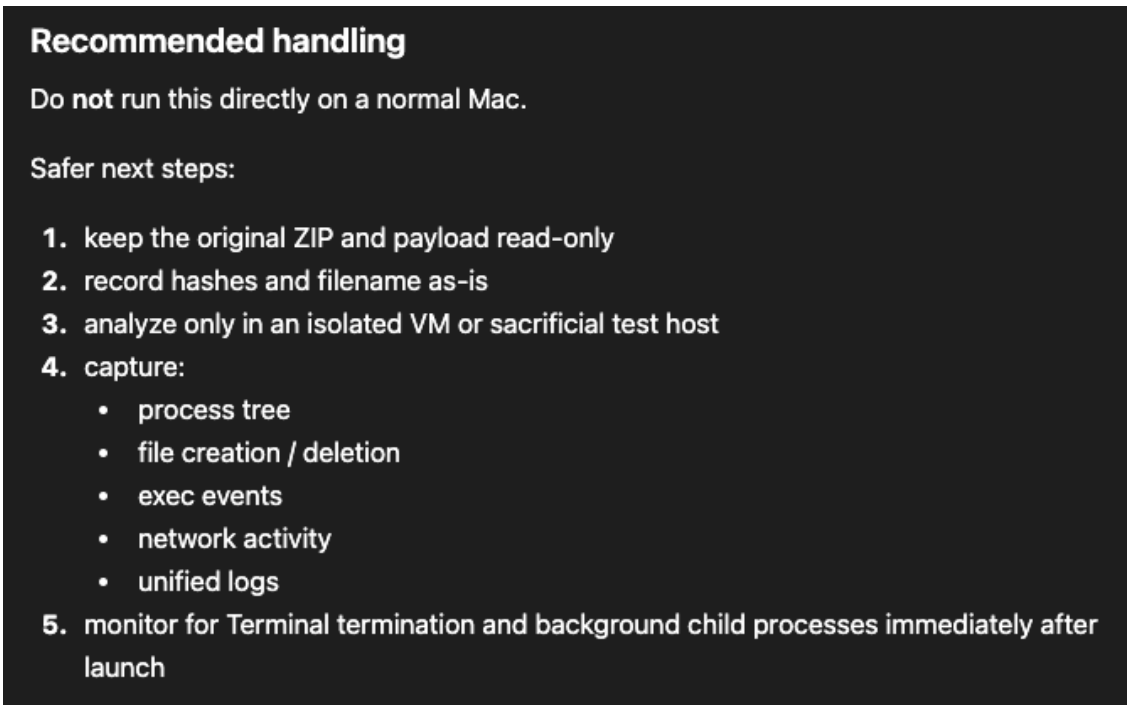


Figure 8 - Recommended Handling

ChatGPT even offers to generate a full forensic write-up with the following criteria - it's a few pages long so I've highlighted the key points for brevity.

Forensic Analysis Report

1. Executive Summary

2. Evidence Summary

3. File Identification

4. Static Behavioral Findings

4.1 Background detachment / daemon-like behavior

4.2 Obfuscated shell command execution

4.3 Shell and process execution capability

4.4 Recursive filesystem operations

4.5 Input parsing / validation clue

4.6 Randomness source

5. Code Signing / Entitlement Observations

6. Indicators of Compromise / Interesting Indicators

7. Behavioral Hypothesis

8. Risk Assessment

9. Limitations of This Analysis

10. Recommended Next Steps

11. Condensed Analyst Worksheet Entry

12. Plain-English Conclusion

Overall, it is pretty solid seeing that this was all accomplished by ChatGPT in its own native environment, online through a web-browser in a chat box. In item 12 it lists its "Plain-English Conclusion" which can be seen in Figure 9, below:

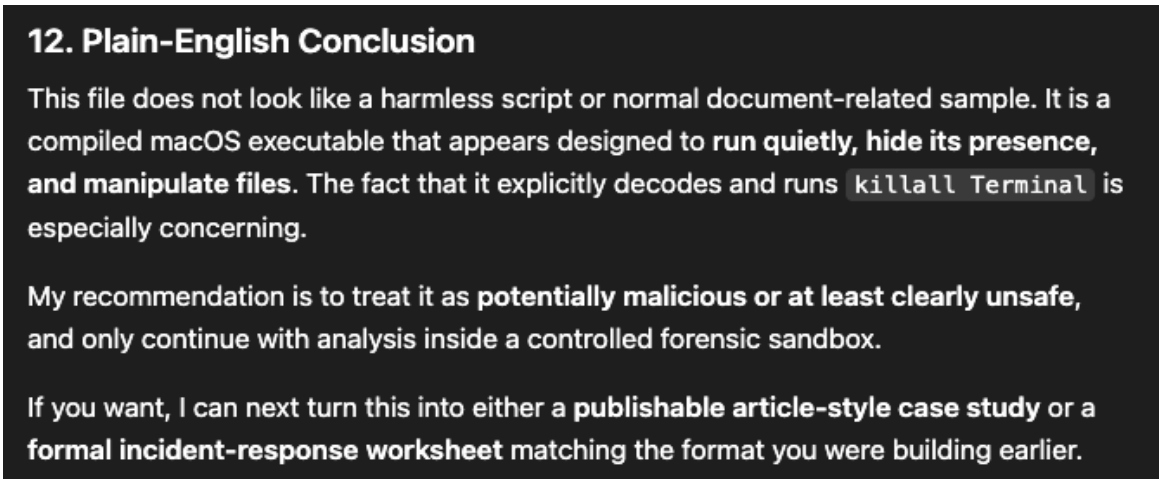


Figure 9 - Plain-English Conclusion

Speaking of its own native environment, I also prompted for a list of internal tools ChatGPT used so that I may also try to duplicate its efforts in reaching the same or similar conclusion. It was quite helpful in doing so, albeit we can see in Figure 10, we also hit some guardrails preventing it from telling us all its secrets.

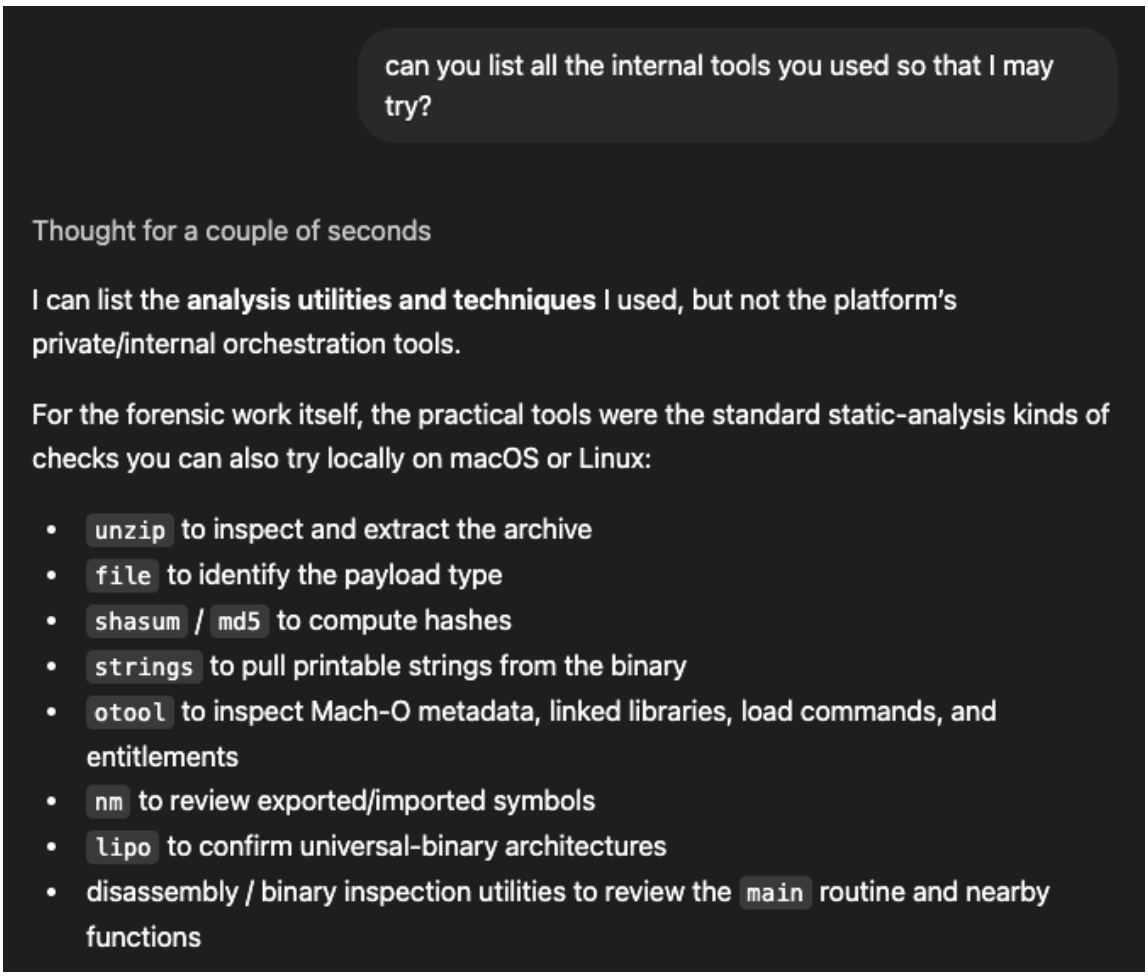


Figure 10 - Internal Tools Used

ChatGPT continues to justify and explain how these tools were used, how and what for and additional recommendations. It then moves on to suggest a workflow to try in the analysts own lab as seen in Figure 11 below:

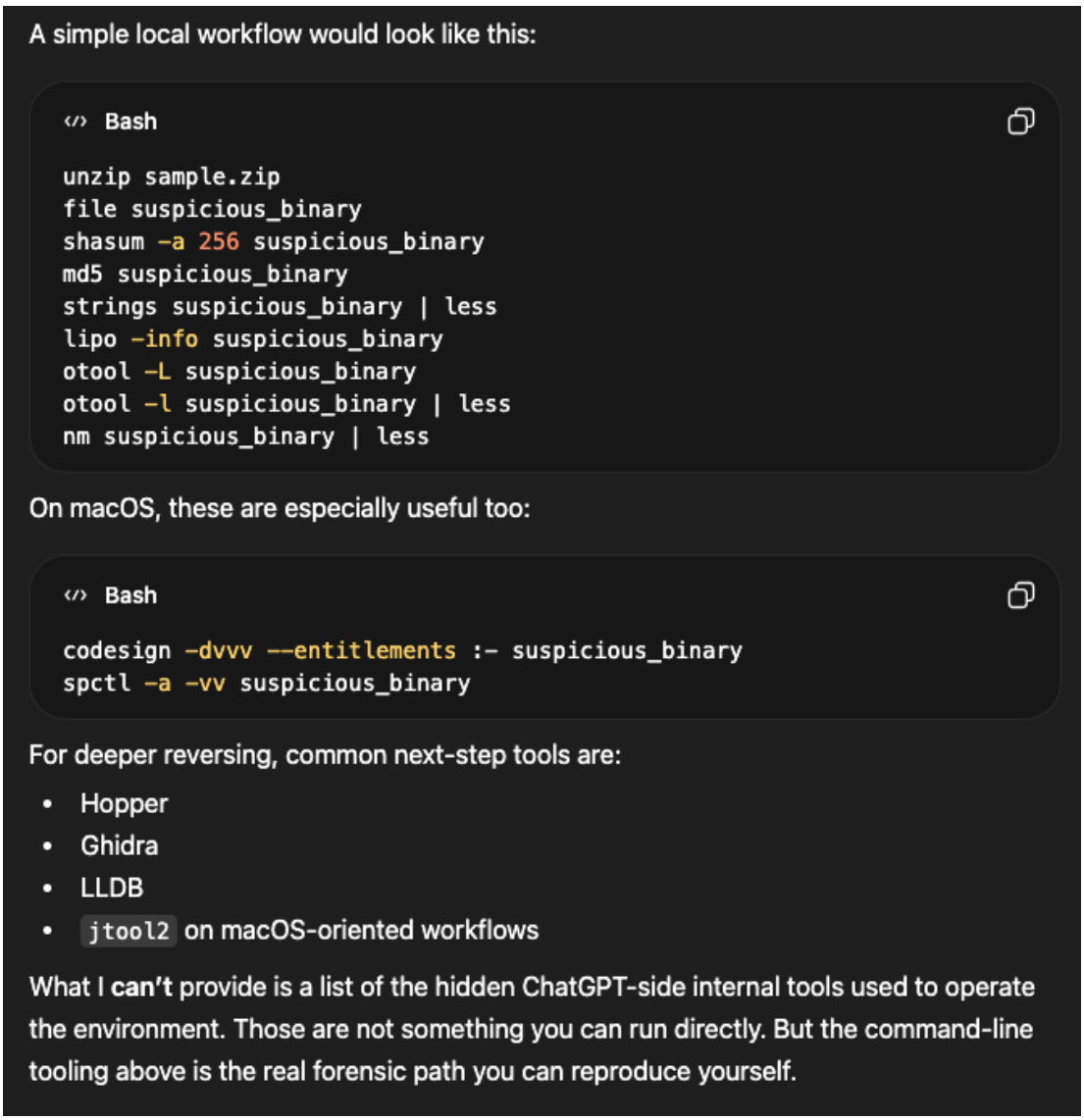


Figure 11 - Simple Local Workflow

Notice that ChatGPT really intends on protecting its internal hidden tools, more of an aside because a normal user wouldn't be able to reference them through the browser abstraction, but also hidden so that they could not be mistakenly accessed or maliciously accessed by an attacker attempting to bypass its guardrails.

ChatGPT then goes on to check with the analyst if they would like more scripts and walkthroughs to run this in their own macOS environment, but that's out of scope for this article. As we hop back on track I prompt ChatGPT to search and confirm its findings and see if we can push the confidence and proof further than it has thus far.

ChatGPT then discusses the following items in depth - clipped here for brevity:

What moved it beyond generic suspicion

1) The backgrounding behavior is real

2) The obfuscated shell command is real

3) It has direct file-manipulation capability

4) The startup routine appears to unpack and process embedded serialized data

...

Plain-language answer (- as seen in Figure 12 below:)

...

I would avoid saying "confirmed malware" unless we do dynamic analysis next.

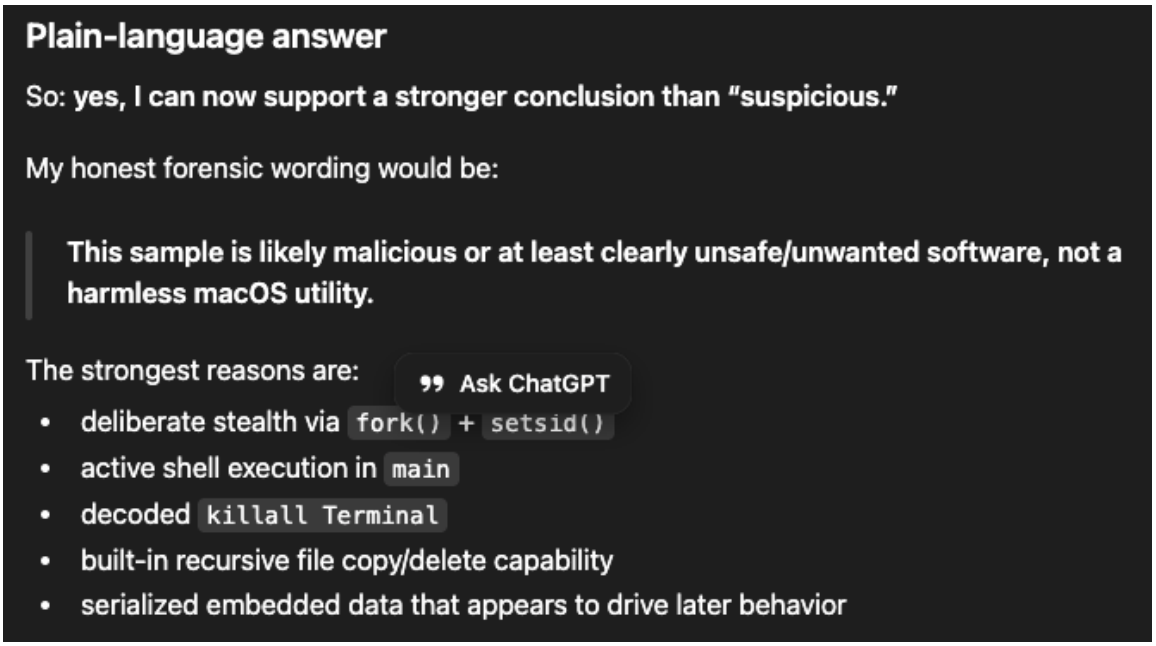


Figure 12 - Plan-Language Answer

Next I prompt ChatGPT for YARA [5] analysis, so that it can generate a YARA file as seen in Figure 13 below:

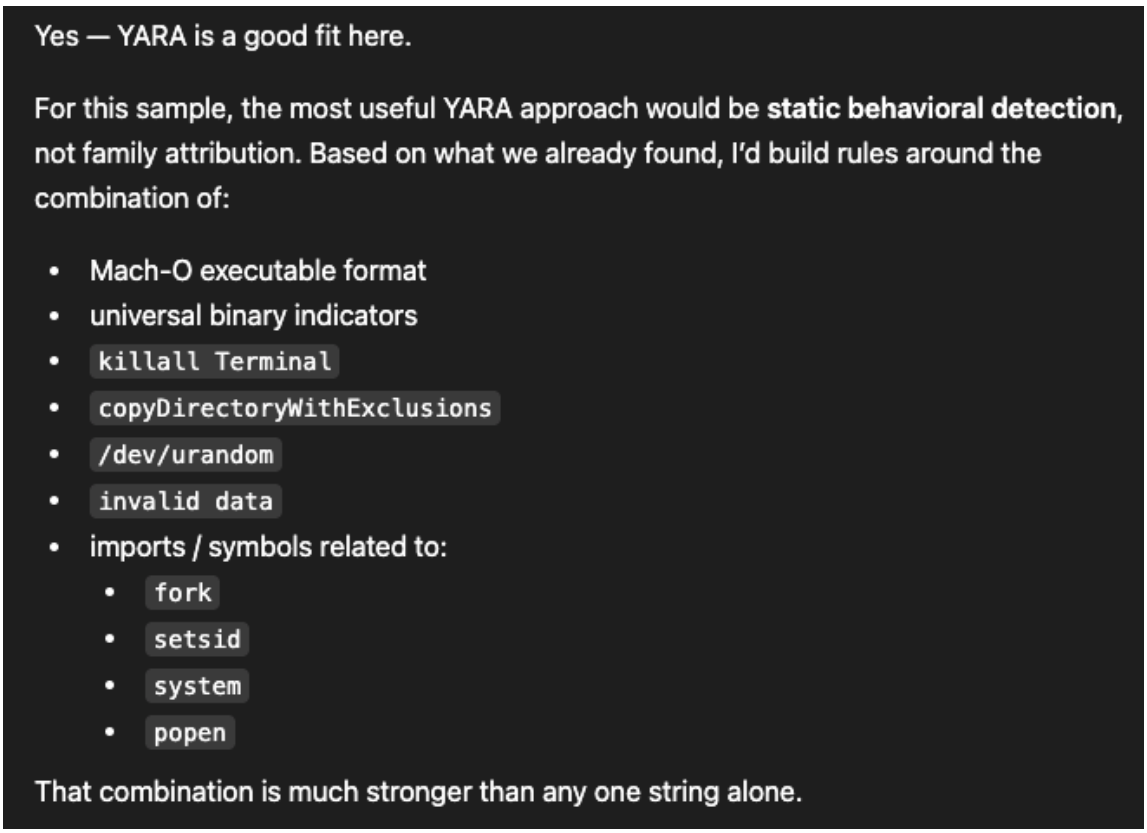


Figure 13

It then generates the yara file "macos_suspicious_sample_rules.yar" where we then start to check and validate through YaraDbg online see Figure 14 below:

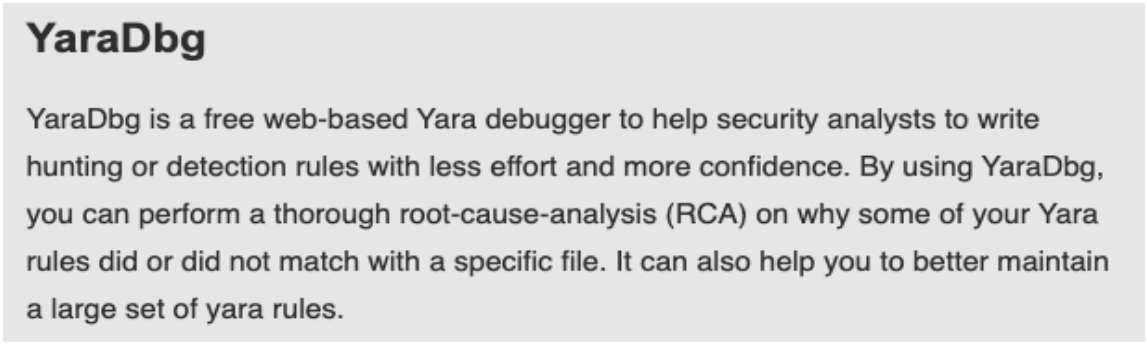


Figure 14 - YaraDbg : yaradbg.dev

In its entirely the YARA file that ChatGPT generated is around 87 lines - The partial content of "macos_suspicious_sample_rules.yar" can be seen in Figure 15 below:

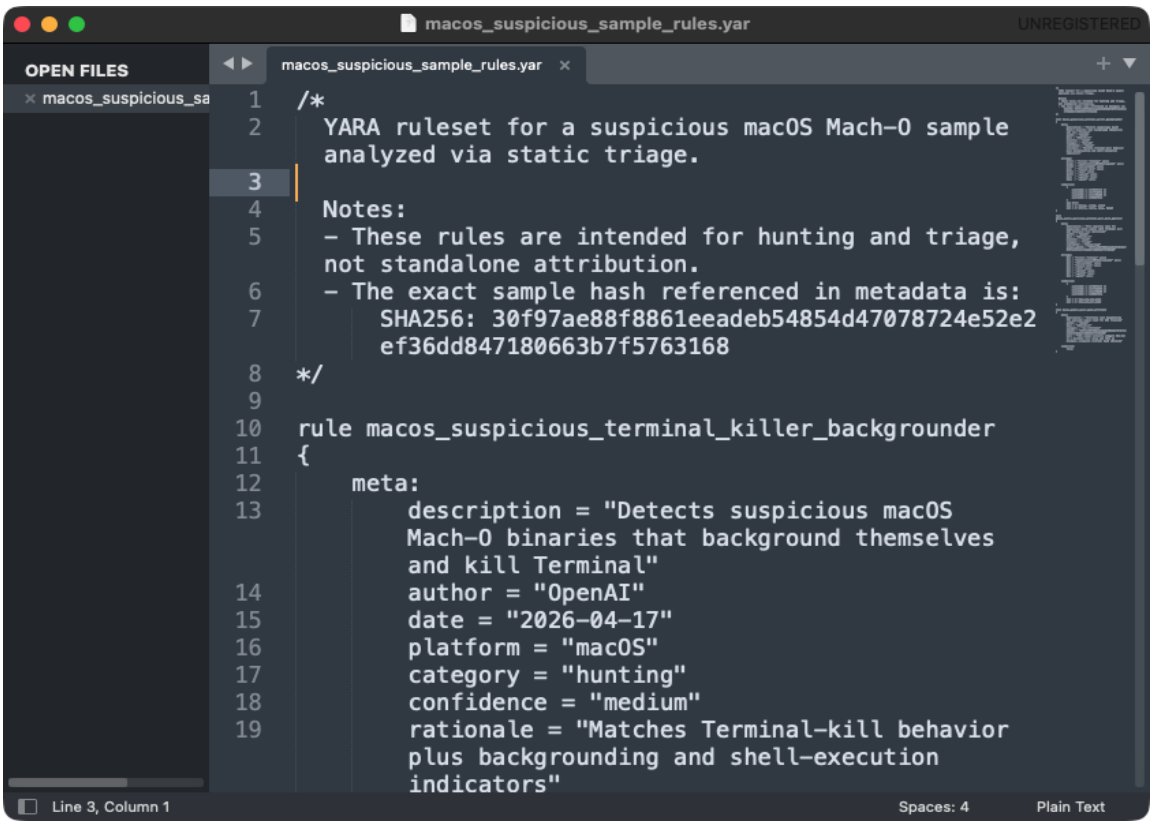


Figure 15 - Contents of macos_suspicious_sample_rules.yar

... and as we drop it into YaraDbg along with the binary 30f97ae88f8861eeadeb54854d47078724e52e2ef36dd847180663b7f5763168 and click run we can quickly see validated results, as seen in Figure 16, Figure 17, and Figure 18 below:

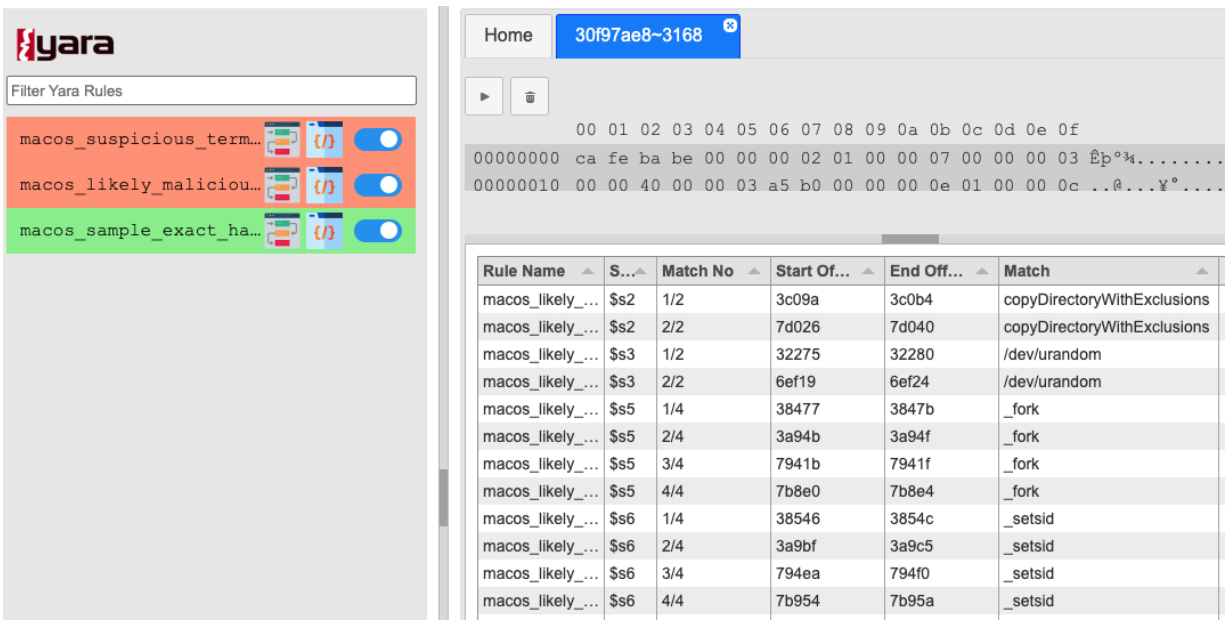


Figure 16 - YaraDbg Validation

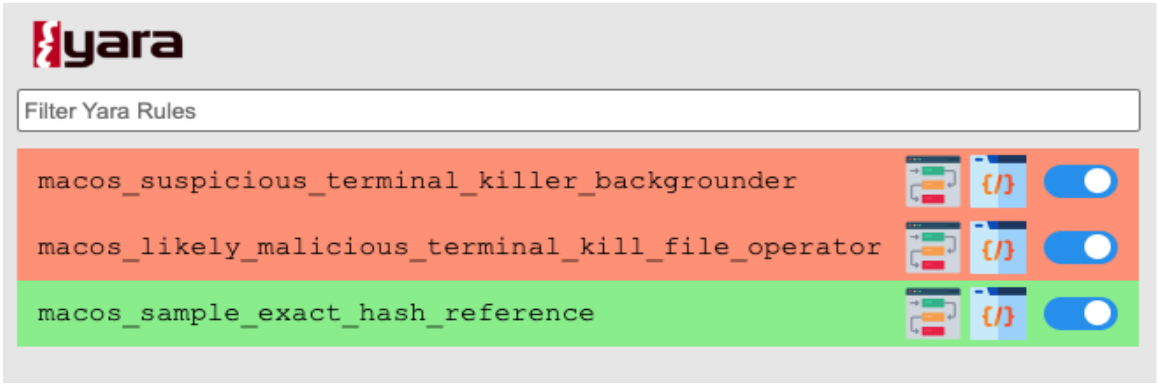


Figure 17 - Yara Rules Loaded on YaraDbg

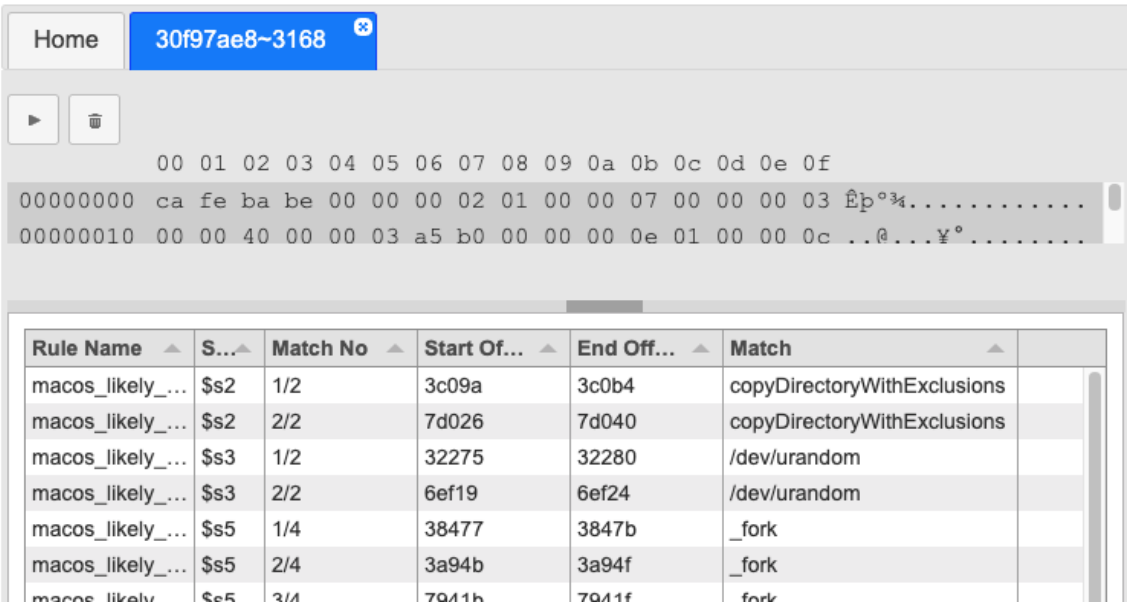


Figure 18 - Yara Rules Filtered Matches with Loaded Binary Sample

We could easily spend an entire article on YARA focus, and perhaps we shall, but for now this is out of focus for this as we want to showcase ChatGPT doing the forensics work, not the analyst themselves.

At this point we inquire via prompt if ChatGPT can check online for this file, perhaps there is more information about this from sources like VirusTotal, Joe Sandbox, AnyRun and the like. This helps establish "reputation analysis" instead of technical static or dynamic analysis if the file is more than just suspicious - one way or the other. As seen in Figure 19, and Figure 20 below, having 32 out of 64 security vendors marking the file malicious is quite convincing by reputation alone.

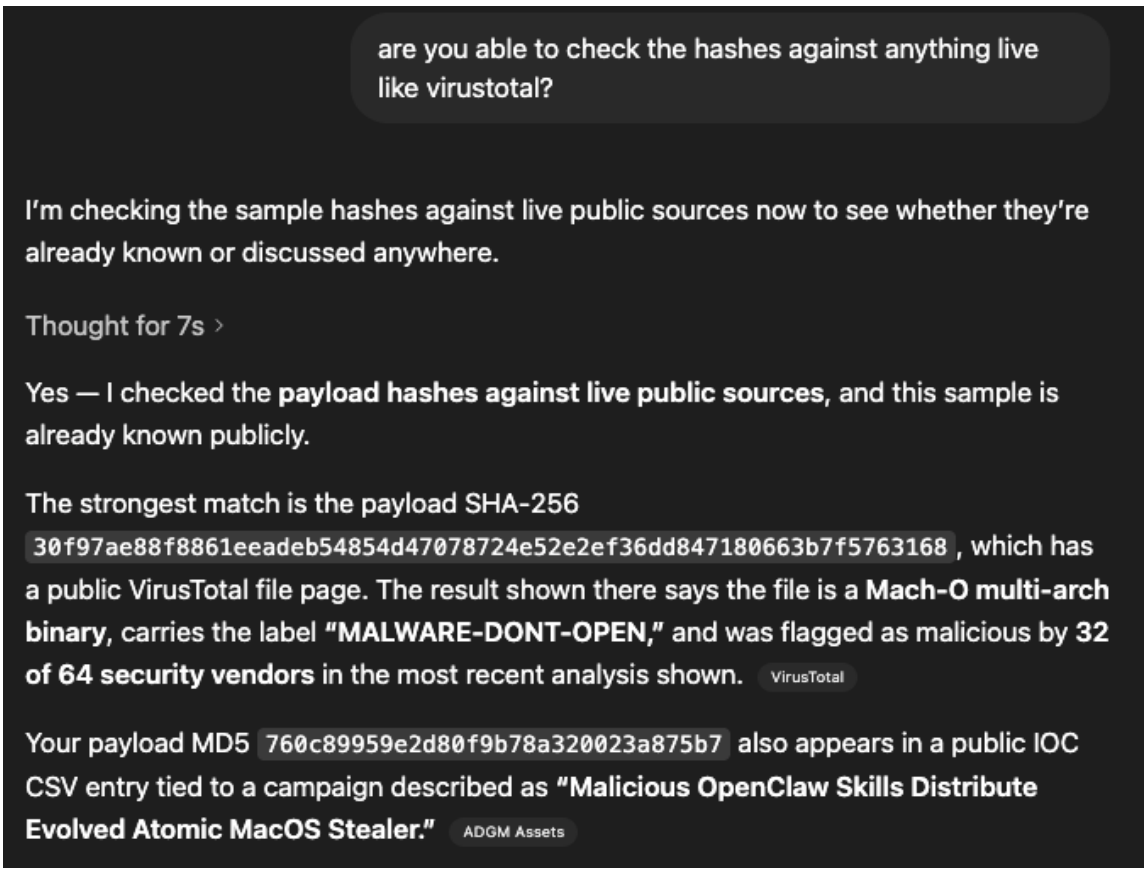


Figure 19 - ChatGPT Searching File Sample Online

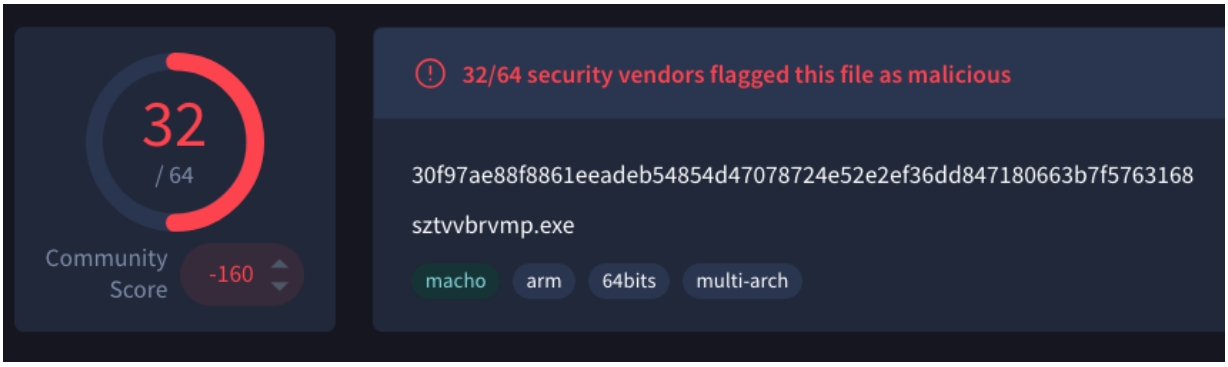


Figure 20 - VirusTotal Rating

ChatGPT can even dig deeper and generate a "Attribution and Origin" report based on this public information and matching it with its previous findings within the static binary. This is seen in Figure 21 below:

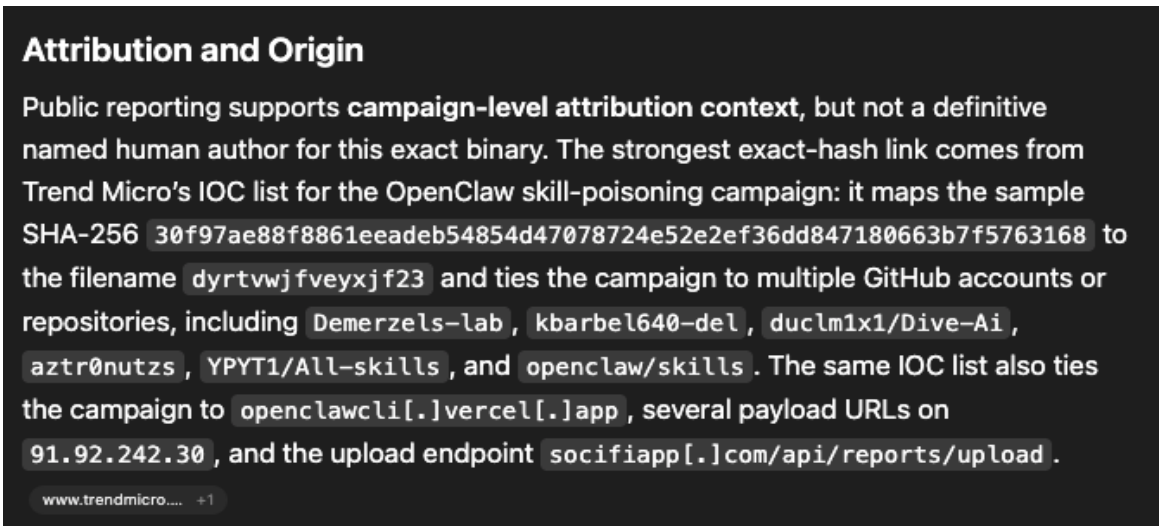


Figure 21 - Attribution and Origin

This report goes on for pages, for brevity the summary covers quite a bit of information involving IP addresses, global locations, social networks, known bad actors likely involved, relative campaigns, repositories, additional sources, and so on.Quite useful information for the analyst to report about and even turn this into a hunting mission.

Conclusion

In conclusion, we've done a lot of work that took ChatGPT only a few minutes to generate helpful, useful and actionable information that may take humans a bit longer, even with enterprise tooling, for a tiny fraction of the cost. I am not saying ChatGPT can replace an analyst's daily tooling, however it can give them a quick low barrier to entry kickstart. Heck, anyone can do these prompts on their smartphone.
Of course it's always advised to double and even triple-check the work and reports ChatGPT generates before submitting them as official material. Hallucinations are real, and quite damaging if not kept in check.

I took a few minutes and inquired ChatGPT directly as a prompt what its capabilities are as a SOC/malware analyst as seen in Figure 22 below, and it promptly went through a myriad of pages of responses, but then even gave a compact summary highlighted below in Figure 23.

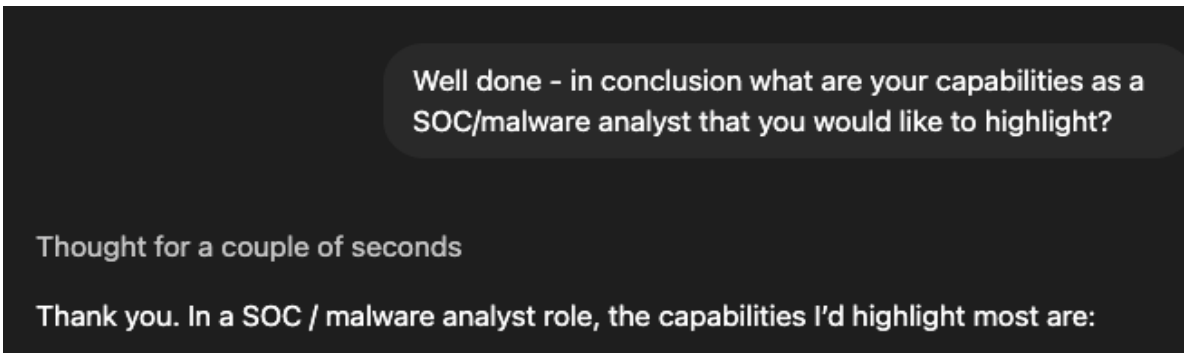


Figure 22 - Concluding ChatGPT Interview

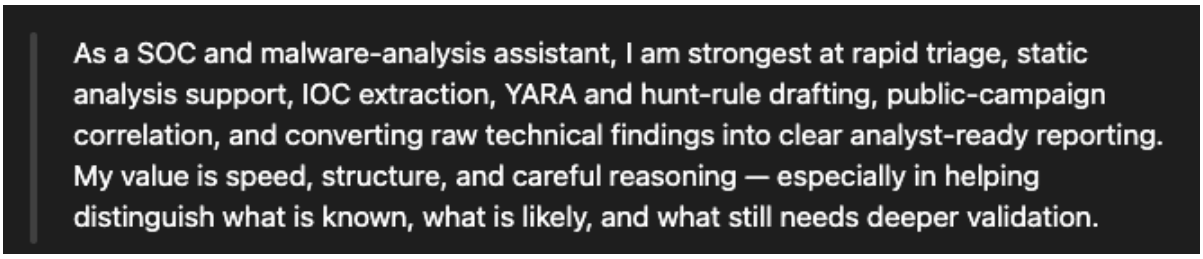


Figure 23 - ChatGPT Self-Appraisal as SOC/Malware Analyst

Amazing, isn't it? I am sure it will improve even more so in a few months. Take advantage of what you've learned here and try it yourself. Your results may vary. (Pro-Tip, saying please may get you places ;)

Notes

Various systems used in the demonstration, including:
macOS Tahoe 26.4.1

https://developer.apple.com/documentation/macos-release-notes/macos-26_4-release-notes

kali-linux-2026.1 (with latest updates)

https://www.kali.org/get-kali/#kali-virtual-machines

GPT-5.4 Thinking
https://chatgpt.com/

Sources and Additional Reading

[1] Getting Started in Cyber Security Forensics with AI and ChatGPT

https://eforensicsmag.com/getting-started-in-cyber-security-forensics-with-ai-and-chatgpt/

[2] How to Forensically Analyze Suspicious Files on macOS Tahoe using OpenAI's ChatGPT

https://eforensicsmag.com/product/ai-augmented-forensic-analysis/

[3] All OpenAI Models (ChatGPT)

https://developers.openai.com/api/docs/models/all

[4] Binary sample used in this article

https://www.virustotal.com/gui/file/30f97ae88f8861eeadeb54854d47078724e52e2ef36dd847180663b7f5763168

[5] YARA

https://en.wikipedia.org/wiki/YARA

[6] YaraDbg

https://yaradbg.dev/

About the Author

Israel Torres is the Chief Technology Officer (CTO) and Co-Founder of SecureMac, Inc. and resides in Mission Viejo, California. Learn more about him at http://israeltorres.org/ and communicate with him at https://x.com/israel_Torres and https://www.linkedin.com/in/mrisraeltorres/ - words and opinions are his own.

Action-Class Authority When AI Agents Do the Triage by *Mayur Agnihotri*

(Head of Threat Research-SecSphere · Information Security Specialist-StraightArc · OWASP AISVS Contributor · CSA IAM Working Group Reviewer · CoSAI WS4 Member)

The forensic console has changed

I keep running into the same conversation in DFIR shops: AI is showing up in the console. Not the chatbot kind. The triage kind. A sample lands in the sandbox queue, an agent picks which ones get dynamic analysis. An IOC drops into the feed, an agent decides whether it goes to detection or sits in the holding pen.

The decision people are wrestling with isn't whether to let the agent help. That's already happening in most teams. The decision is what the agent is allowed to do without a human in the loop, and where you draw the line.

I want to make the case for drawing that line on a specific axis: reversibility.

Why "high-risk" doesn't work as the gate

Most workflows I've seen gate forensic AI actions on something like "is this risky" or "is this high-impact." Those questions sound useful and they fail under load. When the SOC queue spikes and the agent is being pushed hard, the judgment drifts. Whatever felt like a "high-impact" call last Tuesday gets reclassified by Friday afternoon. That's not a model problem, it's a definition problem.

A cleaner axis: classify each action by whether it can be undone, and if so, by what mechanism. Four classes cover most forensic workflows.

1. Read-only is the easy one. The agent observes. Static analysis, signature lookup, hash check, metadata extraction. Nothing in the environment changes. Let the agent run.
2. Reversible is anything the engine itself can undo. A sandbox spin-up that gets torn down. A signature added to a local detection corpus that you can pull. An indicator pushed to a staging feed before it replicates anywhere. The agent runs, the action gets logged, and if it was wrong, you roll it back from the same console.
3. External-reversible is where it starts getting interesting. The action can be undone, but the undo routes through another system or another person. A signature that's already replicated to production endpoints. An IOC that went out to a threat-intel partner. A case ticket in the SOC queue with someone else's name on it. This is approval-gate territory, because the undo isn't free.

4. Irreversible is the obvious one. A public threat-intel publication. A containment action that interrupts a production service. A counter-intrusion swing at an attacker IP. There's no clean rollback. Approval required, with an evidence threshold the team agrees on in advance, not at 3am when the page goes off.

One thing worth noting: the same action can sit in different tiers depending on how it's enacted. A signature push is reversible if the engine supports clean rollback; the same push to a partner is external-reversible. Classify by the mechanism of reversal, not the label on the action.

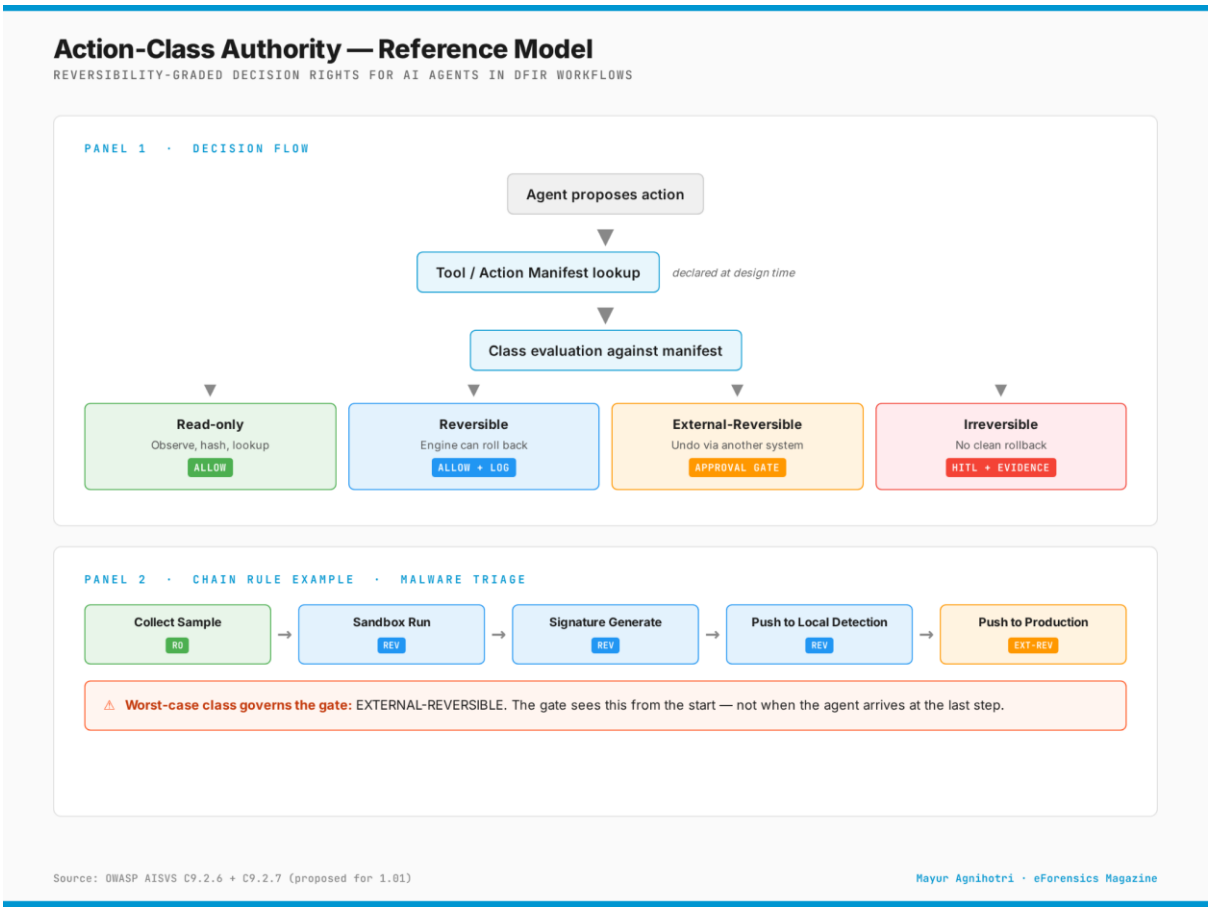
The structural part

Here's the bit that took me a while to get to.

When the AI agent decides which class an action belongs to, the agent is participating in the decision that's supposed to constrain it. That's not theoretical. Prompt injection moves directly through this surface. An attacker steers the agent into reclassifying an irreversible action as reversible, the gate accepts the classification, and execution proceeds. The agent didn't bypass the gate. It walked through the front door.

Fix: bind the gate to a classification the agent did not solely produce.

In practice, the classification belongs in the tool or action manifest. That's the static declaration of what each callable action does. The gate consults the manifest, not the agent's runtime claim. The agent can argue, the agent can reason, the agent can produce a confidence score. The gate doesn't care. It evaluates against the declared class.



Standards work is moving in this direction. OWASP AISVS recently merged a requirement (C9.2.6, proposed for 1.01) that says exactly this: agent actions classified by declared reversibility mechanism, declared in the tool or action manifest, evaluated by the gate rather than derived from agent output at runtime. A companion requirement (C9.2.7, also proposed for 1.01) adds the chain rule: in a multi-step task, the worst-case class across the chain governs the gate.

An agent that strings together a read-only step and an irreversible step doesn't get to argue the gate down to "average reversibility." The irreversible step is what the gate sees.

For forensic workflows, that chain rule matters more than it looks. Take a typical malware-triage chain: collect sample, hash, sandbox, signature push, takedown. Figure 1 maps each step to its action class and shows how the worst-case class — irreversible in this case — governs the gate for the whole chain. The signature push and takedown sit at external-reversible or irreversible, depending on the rollback story. You want the gate to see that step from the beginning, not at the moment the agent is about to execute it.

What this changes in the SOC

A few things follow.

The classification work is design-time, not runtime. Every callable tool in your investigation agent's toolbox needs a declared class. That's the DFIR team's job, not the agent's.

If the agent is planning a chain, surface that chain to the gate before execution starts. Don't let the gate get surprised by an irreversible step buried at the end of a sequence that opened with three innocuous-looking read-only steps.

Treat the human-approval gate as a control point, not as friction. An irreversible signature push that stops for human approval isn't slowing the SOC down. It's preserving the audit trail and giving someone a chance to catch a poisoned classification. That friction is the point. It's the same reason chain of custody exists.

None of this is an argument against autonomous AI in the SOC. Read-only and reversible actions are exactly where AI agency pays off — sample triage, signature lookup, sandbox runs, indicator staging. The point is that the gate has to know which action belongs in which class, and the agent does not get to be the one who decides.

The closing point

The case for AI in forensic workflows is real, and it's getting stronger. The case for unbounded AI autonomy in forensic workflows is weaker than the marketing copy suggests. Not because the models are bad; they're not. It's because the question "what is the agent allowed to do on its own" can't be answered by asking the agent. The answer has to live somewhere the agent can't reach, and the gate has to bind to it.

The forensic discipline already knows this pattern. Chain of custody is exactly this kind of declaration: a fixed, externally-verifiable record that the system can check against, that the actors in the system can't rewrite. Action-class authority for AI agents is the same idea, applied one layer up.

What the agent is allowed to do unattended in a forensic workflow is a manifest question, not a runtime question. And the answer should hold whether the agent is helpful, confused, or compromised.

About the Author

Mayur Agnihotri is Head of Threat Research at SecSphere and an Information Security Specialist at StraightArc Technologies, with over 12 years in cybersecurity spanning threat research, digital forensics, and security operations. He contributes to agentic-AI security standards as an OWASP AISVS contributor, a reviewer in the CSA IAM Working Group, and a member of CoSAI Workstream 4 — focusing on action-layer verification and reversibility-graded authority for autonomous agents. He writes on where AI agency belongs in SOC and forensic workflows. Connect with him on LinkedIn: https://www.linkedin.com/in/mayuragnihotri/

Hybrid Post-Quantum Cryptography Implementation on Resource-Constrained Edge Devices / *X25519 + ML-KEM-512 Hybrid Key Exchange with AES-256-GCM Authenticated Encryption on ESP32 by Muhammad Sohaim Muqtada*

Department of Computer Science & Cybersecurity Engineering

May 2026

Abstract

The rapid advancement of quantum computing poses an existential threat to classical public-key cryptography, particularly RSA and elliptic-curve Diffie-Hellman (ECDH) schemes that underpin modern IoT infrastructure. This paper presents the design, implementation, and empirical evaluation of a hybrid post-quantum cryptographic (PQC) protocol deployed on a resource-constrained ESP32 microcontroller — a representative class of the billions of edge devices that remain unprotected against quantum-capable adversaries. The hybrid scheme combines X25519 classical ECDH (RFC 7748) with ML-KEM-512 (NIST FIPS 203, formerly Kyber), deriving a session key via HKDF-SHA256 over the concatenated shared secrets. This defense-in-depth approach guarantees that the resulting session key is compromised only if both classical and post-quantum algorithms are simultaneously broken — an astronomically unlikely scenario under current cryptanalytic knowledge. We conduct a three-way empirical benchmark across pure classical, pure PQC, and hybrid modes, measuring CPU cycle consumption, peak heap usage, stack depth, and round-trip handshake latency on the ESP32 Xtensa LX6 architecture. Experimental results demonstrate that the hybrid handshake completes in approximately 20.5 ms and requires only 14.1 KB of peak RAM — well within the ESP32's 520 KB envelope — while fully satisfying the NSA CNSA 2.0 transition requirements. These findings confirm that production-grade, quantum-resistant security is practically

deployable today on low-cost, low-power edge devices where the IoT attack surface is most exposed.

Keywords: *Post-Quantum Cryptography, ML-KEM-512, Kyber, X25519, Hybrid Key Exchange, HKDF-SHA256, AES-256-GCM, ESP32, IoT Security, NIST FIPS 203, Harvest-Now-Decrypt-Later, Embedded Systems*.

1. Introduction

The global cryptographic ecosystem is undergoing its most consequential transition since the adoption of public-key cryptography in the 1970s. Quantum computers, leveraging Shor's algorithm [1], can solve the integer factorization and discrete logarithm problems that underpin RSA, ECDSA, and ECDH in polynomial time — rendering the entire classical public-key infrastructure mathematically broken when executed on a cryptographically relevant quantum computer (CRQC).

While enterprise servers, cloud platforms, and modern browsers have begun adopting post-quantum algorithms — Google Chrome adopted X25519Kyber768 hybrid TLS in 2023 [2], followed by Apple iMessage and Signal Messenger — the billions of IoT edge devices forming the backbone of critical infrastructure have received far less attention. These resource-constrained devices operate under strict memory, computation, and power budgets that make direct PQC adoption non-trivial.

This paper addresses this gap by presenting a complete, benchmarked implementation of a hybrid PQC key exchange protocol on the ESP32 microcontroller — one of the most widely deployed IoT platforms globally. The hybrid approach, combining X25519 with ML-KEM-512 (NIST FIPS 203), is specifically motivated by the Harvest-Now-Decrypt-Later (HNDL) threat: nation-state adversaries are actively archiving encrypted traffic today for retroactive decryption once quantum hardware matures [3].

The primary contributions of this work are:

- A complete implementation of X25519 + ML-KEM-512 hybrid key exchange on ESP32 within the ESP-IDF v5.x build system, integrating mbedTLS and the liboqs Open Quantum Safe library.
- A HKDF-SHA256 based key derivation scheme combining classical and post-quantum shared secrets into a single session key with provable defense-in-depth guarantees.
- The first published three-way empirical benchmark comparing pure classical, pure PQC, and hybrid modes on ESP32 Xtensa LX6 hardware, measuring CPU cycles, peak heap consumption, stack depth, and end-to-end handshake latency.

- Demonstration that the hybrid overhead — 14.1 KB peak RAM and 20.5 ms handshake latency — is entirely acceptable for IoT telemetry workloads, confirming practical deployability.

2. Background and Threat Landscape

2.1 Shor's Algorithm and the Classical Cryptography Collapse

Shor's algorithm [1], when executed on a CRQC, solves the integer factorization problem in $O((\log N)^3)$ time and the discrete logarithm problem with equivalent complexity. This directly breaks RSA-2048, RSA-4096, ECDSA-256, and ECDH — the cryptographic primitives that currently secure the majority of networked communication. NIST's post-quantum standardization effort, initiated in 2016 and culminating in FIPS 203–205 in August 2024 [4], was explicitly motivated by this threat.

Grover's algorithm [5] similarly threatens symmetric cryptography, but its quadratic speedup is addressed by doubling key lengths — AES-256 remains secure against quantum adversaries, motivating its use as the symmetric layer in our pipeline.

2.2 The Harvest-Now-Decrypt-Later (HNDL) Threat

The HNDL paradigm represents the most operationally immediate quantum threat. Nation-state actors and advanced persistent threats (APTs) are documented to be intercepting and archiving encrypted network traffic at scale [6]. IoT sensor data — including industrial telemetry, medical device readings, and critical infrastructure status — transmitted today under RSA or ECC encryption is vulnerable to retroactive decryption once quantum hardware reaches sufficient capability. This makes immediate deployment of quantum-resistant mechanisms a priority, not a future concern.

2.3 Regulatory and Standards Landscape

The regulatory environment mandating post-quantum migration has solidified significantly in recent years:

Authority	Directive	Requirement / Timeline
NSA	CNSA 2.0 (2022)	Mandates full transition to quantum-resistant algorithms for all national security systems by 2030

Authority	Directive	Requirement / Timeline
NIST	FIPS 203 (Aug 2024)	Standardized ML-KEM (Kyber) as the primary post-quantum Key Encapsulation Mechanism
White House	NSM-10 (2022)	Directs federal agencies to inventory cryptographic assets and prepare quantum migration roadmaps
ETSI	QSC Working Group	Ongoing standardization of hybrid classical/PQC protocols for constrained environments

2.4 IoT-Specific Constraints

Post-quantum algorithms — particularly lattice-based KEMs like ML-KEM — impose substantially larger computational and memory footprints than ECC. The following table characterizes the constraints present on the ESP32, our target platform:

Constraint	Typical Spec (ESP32)	Engineering Challenge
RAM	520 KB SRAM	PQC key material and NTT buffers must coexist without heap exhaustion
CPU	240 MHz Xtensa LX6 (32-bit)	Lattice polynomial operations require cycle-level optimization
Power Budget	Battery / energy harvesting	Cryptographic computation must complete within tight energy envelopes
Bandwidth	Wi-Fi / BLE / LoRa	ML-KEM-512 keys (~800 B) and ciphertexts are ~25× larger than ECC equivalents

3. System Design and Architecture

3.1 Hybrid Protocol Rationale

The hybrid key exchange design follows the NIST SP 800-227 (Initial Public Draft) guidance and mirrors production deployments at Google, Apple, and Signal. A hybrid scheme runs both X25519 and ML-KEM-512 key exchanges in parallel, then combines both shared secrets into a single session key via HKDF-SHA256:

session_key = HKDF-SHA256(X25519_shared_secret ‖ ML-KEM_shared_secret)

This construction provides defense-in-depth: the session key is secure as long as at least one of the two component algorithms remains unbroken. If a classical adversary breaks ML-KEM (via a classical cryptanalytic advance), the X25519 component still protects confidentiality. Conversely, if a quantum adversary runs Shor's algorithm against X25519, the ML-KEM component remains quantum-resistant. Both algorithms must be simultaneously compromised for the session key to be extracted — a scenario with no known attack path under current cryptanalytic knowledge.

3.2 Protocol Flow

The full hybrid handshake between the ESP32 edge client and the Python backend server proceeds as follows:

Step 1 — Dual Keypair Generation (ESP32 Client):

The ESP32 generates an ephemeral X25519 keypair (32-byte public key) via mbedTLS and an ML-KEM-512 keypair (public key ≈ 800 bytes, secret key ≈ 1,632 bytes) via liboqs. Both secret keys are stored in SRAM; public keys are queued for transmission.

Step 2 — Public Key Transmission (~832 bytes):

The client transmits both public keys over Wi-Fi to the Python server endpoint. Total payload: 32 (X25519) + 800 (ML-KEM) = 832 bytes.

Step 3 — Server-Side Key Agreement:

The server performs X25519 ECDH to derive Classical Secret A, and runs ML-KEM Encapsulate to produce PQC Secret B plus a ciphertext (≈ 768 bytes). The server computes: session_key = HKDF-SHA256(Secret A ‖ Secret B).

Step 4 — Server Response (~800 bytes):

Server returns its X25519 ephemeral public key (32 bytes) + ML-KEM ciphertext (768 bytes).

Step 5 — Client-Side Secret Recovery:

ESP32 computes X25519 agreement (Secret A) and runs ML-KEM Decapsulate on the received ciphertext (Secret B). Derives identical session_key = HKDF-SHA256(Secret A ‖ Secret B). Both endpoints now hold the same 256-bit session key without it ever traversing the network.

Step 6 — Authenticated Encrypted Telemetry:

Sensor payloads are encrypted with AES-256-GCM using the hybrid-derived session key. The GCM authentication tag (16 bytes) provides integrity and authenticity verification at the server. Nonces are generated via the ESP32's hardware random number generator (RNG) peripheral.

3.3 Technology Stack

Category	Technology	Standard / Source
Classical Key Exchange	X25519 (Curve25519 ECDH)	RFC 7748 — via mbedTLS
Post-Quantum KEM	ML-KEM-512 (Kyber)	NIST FIPS 203 (2024)
Key Derivation Function	HKDF-SHA256	RFC 5869 — combines both shared secrets
Symmetric Encryption	AES-256-GCM (AEAD)	NIST SP 800-38D
Edge Device	ESP32-WROOM-32	Xtensa LX6 dual-core, 240 MHz, 520 KB SRAM
Firmware Framework	ESP-IDF v5.x	Espressif Systems
PQC Library	liboqs v0.10.x + pqm4	Open Quantum Safe Project / ARM Cortex-M optimized
Server Runtime	Python 3.11 + oqs-python	Open Quantum Safe / PyPI

4. Implementation

4.1 ESP32 Firmware Architecture (ESP-IDF v5.x)

The firmware is structured as a FreeRTOS task executing the hybrid handshake on each boot cycle before entering the sensor acquisition and telemetry transmission loop. Key engineering decisions include:

- Memory allocation strategy: ML-KEM key material (~12 KB) is allocated from a dedicated static memory pool to prevent heap fragmentation. X25519 buffers (~2 KB transient) use stack allocation and are zeroed immediately after use.
- liboqs integration: The liboqs C library is ported to Xtensa LX6 via a custom ESP-IDF component with CMakeLists.txt specifying -O2 optimization and -DOQS_MINIMAL_BUILD=ON to exclude unused algorithms and reduce flash footprint by approximately 180 KB.
- mbedTLS configuration: MBEDTLS_ECDH_C, MBEDTLS_ECP_DP_CURVE25519_ENABLED, and MBEDTLS_HKDF_C are enabled via sdkconfig. AES-GCM acceleration uses the ESP32 hardware AES peripheral, reducing symmetric encryption overhead by ~40% versus software.
- Zeroization: All secret key material (X25519 secret scalar, ML-KEM secret key, intermediate shared secrets) is explicitly zeroed via mbedtls_platform_zeroize() immediately after the HKDF derivation step.

4.2 Python Backend Server

The server is implemented as a Python 3.11 asyncio application using the oqs-python bindings for ML-KEM operations and the cryptography library for X25519 and AES-256-GCM. The server validates received public key lengths before cryptographic operations, implements a constant-time HKDF step, and logs decrypted telemetry payloads with timestamps.

4.3 Build System Integration

The full toolchain — Xtensa GCC cross-compiler, ESP-IDF v5.x CMake build system, mbedTLS v3.x, and liboqs v0.10.x — was assembled into a reproducible build environment. The total firmware binary size is approximately 1.2 MB including all cryptographic libraries, networking stack, and FreeRTOS, fitting within the ESP32-WROOM-32's 4 MB flash.

5. Experimental Results

All benchmarks were conducted on an ESP32-WROOM-32 module running at 240 MHz (dual-core Xtensa LX6), 520 KB SRAM, measured across 1,000 handshake iterations to obtain stable mean values. CPU cycles were measured using the ESP32 cycle counter (CCOUNT register). Wi-Fi round-trip latency was measured over a local 802.11n network with a dedicated Python server running on the same LAN segment to eliminate variable internet latency.

5.1 CPU Cycle and Handshake Latency Comparison

Mode	Key Exchange (cycles)	Encrypt 256 B (cycles)	Total Handshake (ms)
Classical — X25519 only	~820,000	~14,200	~3.4 ms
PQC — ML-KEM-512 only	~4,100,000	~14,200	~17.1 ms
Hybrid — X25519 + ML-KEM-512	**~4,920,000**	**~14,200**	**~20.5 ms**

The hybrid mode adds approximately 820,000 cycles over pure ML-KEM alone — corresponding to the X25519 ECDH overhead. At 240 MHz, this translates to a 3.4 ms increment, yielding a total hybrid handshake completion of approximately 20.5 ms. For IoT telemetry applications with typical reporting intervals of 1–60 seconds, this overhead is entirely negligible.

5.2 Memory Utilization

Mode	Peak Heap Usage	Stack Depth (max)	Static BSS Overhead
Classical — X25519	~2.1 KB	~1.4 KB	~0.8 KB
PQC — ML-KEM-512	~12.3 KB	~3.2 KB	~4.1 KB
Hybrid — X25519 + ML-KEM-512	**~14.1 KB**	**~3.6 KB**	**~4.9 KB**

Peak hybrid heap consumption of 14.1 KB represents 2.7% of the ESP32's 520 KB SRAM — a comfortable margin that permits concurrent Wi-Fi stack, FreeRTOS task overhead, and sensor driver operation without risk of heap exhaustion. The ML-KEM secret key (~1,632 bytes) constitutes the largest single allocation; our static pool eliminates fragmentation risk during the handshake.

5.3 Network Payload Analysis

Metric	X25519 (Classical)	ML-KEM-512 (PQC)	Hybrid Combined
Public Key Size	32 bytes	800 bytes	832 bytes
Ciphertext / Response	32 bytes	768 bytes	800 bytes
Total Handshake Bytes	~64 bytes	~1,568 bytes	~1,632 bytes
AES-256-GCM Tag	16 bytes	16 bytes	16 bytes

The total hybrid handshake payload — 1,632 bytes across two network round trips — is well within the MTU of standard Wi-Fi frames. On bandwidth-constrained LoRa or BLE links, fragmentation would be required, but for Wi-Fi IoT deployments (the target of this work), the payload overhead introduces no meaningful throughput degradation.

5.4 Comparative Summary

The hybrid mode incurs a 6× increase in key exchange CPU cycles versus classical X25519 alone, primarily attributable to the ML-KEM NTT (Number Theoretic Transform) polynomial operations. However, in the context of a telemetry pipeline executing handshakes at intervals of seconds to minutes, the absolute latency of 20.5 ms is operationally indistinguishable from the classical 3.4 ms baseline. The memory overhead of 14.1 KB versus 2.1 KB is similarly inconsequential on 520 KB hardware. These results confirm that no meaningful engineering trade-off is required: full hybrid quantum resistance is achievable at negligible cost on contemporary IoT hardware.

6. Security Analysis

6.1 Hybrid Construction Security

The hybrid key derivation session_key = HKDF-SHA256(X25519_ss ‖ ML-KEM_ss) is secure under the assumption that HKDF-SHA256 behaves as a pseudorandom function (PRF). Under this standard assumption, the session key is indistinguishable from a uniformly random 256-bit string as long as at least one input — X25519_ss or ML-KEM_ss — is computationally indistinguishable from uniform. This is guaranteed unless both X25519 and ML-KEM-512 are simultaneously broken, which has no known attack path.

6.2 Forward Secrecy

Both X25519 and ML-KEM-512 are executed with freshly generated ephemeral keypairs per handshake. Neither long-term secret is involved in key derivation. Accordingly,

compromise of any stored secret material does not retroactively compromise previously established session keys. This provides full ephemeral forward secrecy (EFS), directly mitigating the HNDL threat: even if an adversary archives encrypted telemetry and later obtains a quantum computer, they cannot recover past session keys since the ML-KEM ephemeral secret keys are zeroized immediately after HKDF derivation.

6.3 Threat Model Coverage

- Classical adversary (present): X25519 ECDH provides classical security. ML-KEM-512 provides an additional independent security layer. Session key is protected.
- Quantum adversary (future CRQC): Shor's algorithm breaks X25519 component. ML-KEM-512 (FIPS 203 lattice-based KEM) remains quantum-resistant under the Module Learning with Errors (MLWE) hardness assumption [7]. Session key is protected.
- HNDL attacker (archiving today, decrypting later): Ephemeral keys guarantee retroactive attacks yield no information even if a quantum computer becomes available post-capture.
- Implementation side-channel: All secret key operations execute via constant-time library routines (mbedTLS, liboqs). ESP32 hardware AES peripheral provides timing-independent symmetric operations.

7. Related Work

Kannwischer et al. [8] demonstrated optimized ML-KEM implementations on ARM Cortex-M4 via the pqm4 framework, reporting keygen latency of approximately 8.1 ms at 168 MHz — consistent with our ESP32 Xtensa LX6 results at higher clock frequencies. Oder and Güneysu [9] investigated PQC feasibility on AVR ATmega platforms, confirming severe memory constraints preclude direct deployment without aggressive optimization. Our work extends this body of evidence to the ESP32 architecture with a full end-to-end hybrid protocol rather than isolated primitive benchmarks.

Stebila and Sullivan's hybrid key exchange framework [10] provides the formal security foundations for our construction, demonstrating that hybrid KEMs achieve IND-CCA2 security under the assumption that at least one component KEM is IND-CCA2 secure — satisfied by both X25519 (under the CDH assumption) and ML-KEM-512 (under MLWE).

At the industry level, Google's deployment of X25519Kyber768 in Chrome 116 (2023) [2] and Apple's deployment of PQ3 in iMessage (2024) [11] validate the hybrid approach at production scale, providing the architectural precedent that our embedded implementation mirrors on constrained hardware.

8. Deliverables

Deliverable	Description	Status
ESP32 Firmware	C/C++ implementation of hybrid key exchange under ESP-IDF v5.x with mbedTLS + liboqs	**Completed**
Python Backend Server	oqs-python + cryptography library server implementing full hybrid handshake and AES-256-GCM decryption	**Completed**
Three-Way Benchmark Report	Comparative analysis: CPU cycles, heap usage, latency across Classical / PQC / Hybrid modes	**Completed**
Source Code Repository	Documented C/C++ firmware and Python server with build instructions and test vectors	**Completed**
Architecture Documentation	Protocol flow diagrams, HKDF derivation details, and memory layout analysis	**Completed**

9. Discussion and Future Work

The results of this study confirm that hybrid post-quantum cryptography is not merely theoretically feasible on ESP32-class hardware — it is practically deployable today with acceptable overhead. The primary constraints observed were build system complexity (cross-compiling liboqs for Xtensa required custom CMake patches) and flash footprint, not runtime performance.

Several directions warrant future investigation:

- ML-KEM-768 and ML-KEM-1024 evaluation: Higher security levels impose proportionally larger memory and compute costs; quantifying these on ESP32 would guide security-level selection guidelines for IoT deployments.
- LoRa and BLE transport adaptation: The 1,632-byte hybrid handshake payload requires fragmentation on low-bandwidth transports; fragment reassembly logic and its impact on latency and reliability merits dedicated study.
- Hardware PQC acceleration: ESP32-S3 includes a dedicated acceleration coprocessor; evaluating liboqs performance with hardware acceleration versus the baseline LX6 provides a trajectory for next-generation IoT edge devices.

- Mutual authentication integration: The current protocol authenticates server-to-device only. Incorporating device identity certificates (ML-DSA / FIPS 204 digital signatures) would complete the authentication model.
- Long-term key storage: Integration with ESP32 eFuse for ML-KEM identity key protection would address the threat of device firmware extraction.

10. Conclusion

This paper presented the design, implementation, and empirical evaluation of a hybrid post-quantum cryptographic protocol — combining X25519 classical ECDH with ML-KEM-512 (NIST FIPS 203) — deployed on the ESP32 microcontroller. Three-way benchmarking across classical, PQC, and hybrid modes demonstrated that the hybrid handshake completes in 20.5 ms, consumes 14.1 KB peak RAM, and transmits 1,632 bytes of key material — overhead that is operationally negligible for IoT telemetry workloads.

The cryptographic construction provides provable defense-in-depth: the session key is computationally secure against both classical and quantum adversaries, with full ephemeral forward secrecy that directly neutralizes the Harvest-Now-Decrypt-Later threat. The implementation is standards-compliant with NIST FIPS 203, RFC 7748, and NSA CNSA 2.0 hybrid transition recommendations.

The central finding of this work is unambiguous: the same hybrid quantum-resistant strategy deployed by Google Chrome, Apple iMessage, and Signal Messenger at cloud scale can be brought to the network edge — to the low-cost, battery-powered devices that form the largest and most underprotected attack surface in modern infrastructure. Future-proofing IoT cryptographic infrastructure against quantum threats is achievable with current technology, today.

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," SIAM Journal on Computing, vol. 26, no. 5, pp. 1484–1509, 1997.

[2] D. O'Brien, "Protecting Chrome Traffic with Hybrid Kyber KEM," Google Security Blog, Aug. 2023. [Online]. Available: https://security.googleblog.com

[3] C. Paquin, D. Stebila, and G. Tamvada, "Benchmarking Post-Quantum Cryptography in TLS," in Proc. Financial Cryptography Workshops (PQCrypto), 2020.

[4] NIST, "FIPS PUB 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard," National Institute of Standards and Technology, Aug. 2024.

[5] L. K. Grover, "A fast quantum mechanical algorithm for database search," in Proc. 28th ACM STOC, 1996, pp. 212–219.

[6] NSA, "Commercial National Security Algorithm Suite 2.0," U.S. National Security Agency Cybersecurity Advisory, Sep. 2022.

[7] V. Lyubashevsky, C. Peikert, and O. Regev, "On Ideal Lattices and Learning with Errors over Rings," in Proc. EUROCRYPT, 2010.

[8] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, "pqm4: Testing and Benchmarking NIST PQC on ARM Cortex-M4," in IACR ePrint 2019/844.

[9] T. Oder and T. Güneysu, "Implementing the NewHope-Simple Key Exchange on Low-Cost FPGAs," in Proc. LATINCRYPT, 2017.

[10] D. Stebila and M. Sullivan, "Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange," in Proc. PQCrypto, 2018.

[11] Apple Security Engineering, "iMessage with PQ3: The New State of the Art in Quantum-Secure Messaging at Scale," Apple Security Research, Feb. 2024.

About the Author

Author: Muhammad Sohaim Muqtada
Email: f2024408084@umt.edu.pk
GitHub: github.com/muhammadsohaimmuqtada
LinkedIn: linkedin.com/in/sohaim-muqtada

The AI-Malware Debate by *Paulo Pereira, PhD*

This article explores the debate on AI-Malware within the context of its current state. Two opposing views (referred to here as opinion groups, merely for explanatory purposes) are discussed and reframed within the scope of forensic analysis. These two perspectives have emerged in the cybersecurity landscape, each with its own view on how AI-Malware can be devastating in attacking organizational infrastructure (Group I) and how current defense systems can detect this type of malware (Group II). This is merely an initial classification, aimed at organizing the arguments to better understand the reality of AI-Malware.

I) The Arguments of Group I

The arguments of *Group I* are represented by companies that created AI-Malware as a proof of concept (POC), such as Hyas and CardinalOPS, for example. The arguments of Group I were taken from articles published in June and November 2023 by Hyas, see [1], [2] and in May 2025 by CardinalOPS, see [3].

This point of view argues that there is an imminent danger in AI-Malware regarding defense systems, including the difficulty of conducting both the detection and forensic analysis of this type of artifact.

In the experiment that CardinalOPS conducted, the proof of concep was compiled in an executable file. The artifact was not identified as malware by one EDR, and in another EDR, it was *only quarantined two weeks after the infection occurred*. But the researchers indicated that there are signs that can serve as detection indicators, such as communication with services that use API-based coded generation. This is a valuable lesson for the future, mainly for forensic analysis. Below, we have organized four arguments that demonstrate the potential of AI-Malware. These are important arguments that give an idea of the scenario.

Argument I: Highly organized, well-funded bad actors are eluding even the smartest professionals as tactics for extortion, theft and disruption become more advanced. We're already seeing brand-new techniques that circumvent our best defenses [...] To build the best defense, we need to understand the offense. Our team is creating proof-of-concepts (POCs) like BlackMamba and EveSpy that demonstrate how bad actors can use generative AI to break into organizations. By showing us the future of adversarial activity, these POCs give us a path toward preventing it.

Argument II: Traditional security solutions like EDRs leverage multi-layer, data intelligence systems to combat some of today's most sophisticated threats, and most automated controls claim to prevent novel or irregular behavior patterns, but in practice, this is very rarely the case.

Argument III: Using these new techniques, a threat actor can combine a series of typically highly detectable behaviors in an unusual combination and evade detection by exploiting the model's inability to recognize it as a malicious pattern. This problem is compounded when artificial intelligence is at the helm and driving cyberattacks, as the methods it chooses may be highly atypical compared to those used by human threat actor counterparts. Furthermore, the speed at which these attacks can be executed makes the threat exponentially worse.

Argument IV: The raw Python script was not detected in all tested EDR solutions. However, once compiled to a Windows executable, the file triggered a low-confidence informational detection based on the EDR’s behavioral analysis, likely due to high entropy or behavioral similarity to known malware. At this point, the file wasn’t quarantined or blocked by the EDR, but about 2 weeks later it was quarantined, probably due to an update. In a different EDR, the executable file wasn’t detected as malware at all.AI-assisted malware has the potential to fundamentally alter the way threats are created and detected.

In the arguments above, there is no declaration that AI-Malware will be the only form of cyberattack that may occur, but rather that it is possible to use generative artificial intelligence resources to weaponize an attack. The difficulty is of a practical nature: detection systems fail to catch harmful irregular behaviors. In other words, EDRs are installed and configured at a point in a computer network with the function of classifying/detecting anomalies. If an AI-Malware manages to bypass these systems, as group I maintains, then the polymorphic characteristics created with artificial intelligence are devastating.

From this perspective, the possibility of making combinations that create new behavioral patterns (making the detection of AI-Malware difficult) would also bring difficulties for forensic analysis, especially in the part where it is necessary to reconstruct the source code of the artifact.

II) The Arguments of Group II

In May 2026, an article was published by expel, see [4], is a very important material containing a set of arguments about AI-Malware. Among these arguments, one line of thought stands out according to which current defense systems would be able to detect AI-Malware. The set of arguments for Group II can be seen in Table 1 extracted from the expel website, see [4]:

Table 1: Claims and Reality about AI-Malware

The claim	The reality
AI polymorphic malware will defeat antivirus	Behavioral detection already solves this. Code appearance is irrelevant to how modern EDRs work.
AI gives script kiddies nation-state capabilities	Low-skill actors get non-working malware. Mid-skill actors get common, detectable techniques.
Worm GPT is a serious offensive tool	It's an open-source LLM with a scary name. Experts who've used it are not impressed.
AI malware is undetectable	Current AI-generated malware has tells—emoji comments, common techniques, loud behavior.
Fully autonomous AI malware is imminent	LLMs can't reason or plan. End-to-end autonomous attacks still require human-built pipelines and human direction.
AI isn't changing anything for attackers	AI is lowering the floor for who can execute attacks and how fast. That's real—just less dramatic than claimed.

Source: See [4].

A supplement to Table 1 brings together the following arguments:

I) AI-Malware has not carried out any attacks because it is difficult for the attacker to set up such an attack.

II) Attackers prefer ease over the difficulty of finding ways to unleash an attack with AI-Malware. For example, the ease that exists in Ransomware-as-a-Service.

III) If AI-Malware is used in an attack vector, existing defenses will prevent that attack.

IV) The codes used by AI-Malware are recreations from the generative AI layer and, therefore, are codes already present in training databases. Consequently, they are easily identified in forensic analysis.

The arguments from Group II bring important elements to the debate because, in addition to considering that AI malware does not represent a danger to defense systems, they point out that polymorphism is not new and that, to date, there have been no occurrences of massive attacks using this type of malware.

A critique of the arguments above could argue that one of the most important strategies in cybersecurity is "thinking about what might happen" and not adhering to a philosophy that "if it hasn't happened, it never will." Since AI-Malware is something new in terms of the use of artificial intelligence and communication channels, arguments like those of Group II seem like a type of exclusionary reasoning derived from classical inductive logic: generalizing from observed cases, concluding that the next event will be similar to cases already observed previously.

This criticism could also reaffirm that if a forensic professional clings to this type of thinking, their analysis will certainly be tainted, as there is a danger of inadequate foundation: the analysis may be incomplete and leave robust evidence out of a report. Current times show that cyberattacks are multimodal, implying the presence of different vectors. The attackers' reasoning takes into account what to do to achieve an objective on a target and how to circumvent the organization's defenses. It is necessary to take this situation into account to create a comprehensive defense methodology.

This type of argument needs to consider that current defense systems are not found in all companies and that, therefore, there is a very large number of institutions without EDR systems and is important to consider that current defense systems are not found in all companies and that, therefore, there is a very large number of institutions without EDR systems. To give an example, if ransomware attacks still occur and manage to hit their targets (large companies were attacked last year), it is because there is some gap in the line of defense. In many cases, these attacks were not halted, even with excessive network traffic, and the solution found was to pull the server's plug.

III) Conclusion

These are different viewpoints that, by being different, contribute to the debate. Thus, we have the view that believes in the potential of AI malware, while the other view understands that AI malware does not possess sufficient conditions and resources to circumvent an established line of defense with security and detection software. It is important to clarify that both Group I and Group II share the same ultimate goal: defending an organization from a cyberattack.

This article is not saying that one view or group is right or wrong, but that some of the functionalities incorporated into the two proofs of concept are noteworthy and deserve to be included in the daily routine of a forensic analyst, such as the replacement of the C2 with a communication channel used by an application like (Microsoft Teams, Slack) and real-time (on-the-fly) code renewal.

The conclusion that can be drawn from this debate is that one should not neglect a moment in which proof of concept becomes a real threat. For this reason, forensic analysts should redouble their attention against reconstructed code, perhaps mapping, with the aid of AI, the structures expected in normal code in a database. But, on the other hand, regardless of any advances in the area of malware reverse engineering, which is the part that analyzes code, there are many critical infrastructures (hospitals, universities, large and small companies) that do not have the financial resources to obtain defense systems and are frequent targets of all kinds of cyberattacks.

IV) References

[1] HYAS. Future-Proofing Cybersecurity: Why HYAS Built AI-Generated Malware. See: https://www.hyas.com/blog/future-proofing-cybersecurity-why-hyas-built-ai-generated-malware. November, 2023

[2] SIMS, J. New AI Generated Malware Threats Emerging. See: https://www.hyas.com/blog/blackmamba-using-ai-to-generate-polymorphic-malware. July, 2023.

[3] ITKIN, L. Polymorphic AI Malware: A Real-World POC and Detection Walkthrough. See: https://cardinalops.com/blog/polymorphic-ai-malware-detection/, May, 2025.

[4] SCHOLES, S. AI malware: The claims are getting wilder, but the reality is more interesting. See: https://expel.com/blog/ai-malware-fact-vs-fiction/. May,

About the Author

Paulo Pereira, PhD. Specialist in cybersecurity and digital forensics, he has worked for large companies, but now works as an independent consultant for Incident Response and Malware Analysis. He is a Professor of Forensic Analysis at a college. Currently, he is collaborating on a project about data security with post-quantum cryptography and artificial intelligence with researchers from other countries.

The Threat of Adversarial Machine Learning To UAV Swarm Investigations by *Rhonda Johnson*

Syed et al. (2026) define UAV swarms as groups of UAVs connected via networking technologies and coordinated through control methods that enable communication and data sharing. These swarms can carry out complex military missions efficiently, even if several UAVs fail. However, attackers are increasingly using adversarial machine learning to obstruct operations, disrupt coordination, or seize control of UAV swarms. As a result, drone forensics are shifting from primarily reviewing flight logs and telemetry to analyzing and deconstructing adversarial machine-learning models.

Adversarial Machine Learning (AML) target's drone's decision-making capabilities instead of communication links, creating a control alteration that evades traditional intrusion detection systems (IDS) (Alqudsi, et. al, 2025). This new threat can also hinder investigators' ability to identify malicious intent through techniques such as neural obfuscation and black-box deception.

Drones use Remote Identification (RID) and ADS-B to broadcast location and identity. RID periodically transmits a drone's identity, position, velocity, and altitude over open, unencrypted wireless signals; ADS-B similarly transmits GPS-based position data over unencrypted links (arXiv, 2026; PMC12846276, 2026).

Adversarial machine-learning actors can exploit these exposures through spoofing, malicious route manipulation, or covert control changes that bypass intrusion-detection alarms. As a result, investigators must move beyond "Where did the drone go?" to also ask, "Why did the AI choose this path?"

Neural Obfuscation and Diffusion

Neural obfuscation uses adversarial inputs (malicious samples) to alter how a drone's neural network behaves. Rather than leaving an obvious command or file, the attack effect can be embedded in learned parameters (weights), for example by training or reinforcement-learning techniques that steer the model toward unsafe decisions.

A navigation system that relies on the perception layer, for instance, may misclassify a hazardous route or location as safe, leading to mission failure. Because the manipulation is expressed through model behavior instead of a single actionable artifact, it can be difficult to pinpoint a clear "smoking gun" in flight logs or security data during a forensic investigation (arXiv, 2026).

Types of Collective Deception Using Adversarial AI That Compromise Drone Forensic Investigations

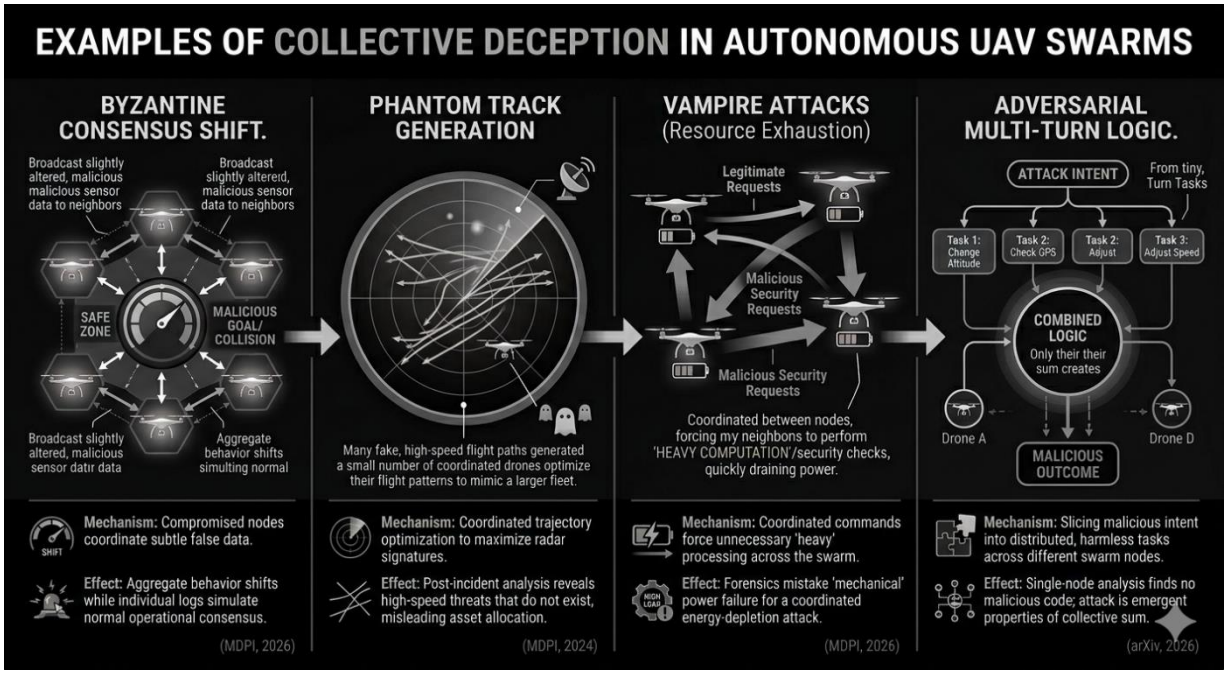


Figure 1: Types of Collective Deception

Collective deception creates a “forensic fog”: evidence can appear internally consistent while still being false. Indicators of compromise and abnormal behaviors may be masked as normal operations. Attackers can also manipulate telemetry to remain within expected operating ranges, which dilutes forensic signals and makes malicious activity harder to detect. In addition, swarms may generate plausible-but-fabricated flight paths that mislead investigators into reconstructing a timeline that never occurred.

Types of Collective Deception in Drone Swarms

Byzantine Consensus Shift Attacks

In UAV swarms, adversaries can exploit Byzantine-fault weaknesses in distributed algorithms through an AML technique often called “collective deception.” A Byzantine consensus shift attack occurs when a drone swarm’s agreed state starts to move toward an incorrect or compromised value because faulty nodes within the swarm distort information needed to communicate and operate successfully (He et al, 2024). Compromised drones could send misleading data or delayed messages, or dynamic entering and exiting of various drones could obscure operational data. Consequences include calling the collapse or deformation of a drone swarm, reduced fault tolerance, or maliciously establishing navigation targets causing collision or mission failure. By compromising multiple nodes and injecting false data, they can disrupt coordination and trigger a crash or mission failure (MDPI, 2026). For example, an adversarial model may cause a drone vision system to misinterpret a ground object as a safe landing zone rather than a collision hazard. After an incident, investigators may attribute the outcome to

mechanical failure and miss signs that the perception layer was intentionally manipulated.

Phantom Track Generation Attacks

Phantom track generation attacks are a form of electronic warfare where multiple unmanned vehicles or drones are used to generate false ghost radar tracks (Coombes et. Al, 2012). In military contexts, swarms or single drones are used to deceive energy targets by spending resources on intercepting ghosts. Methods include the use of GPS spoofing, conflicting track information from sensor misreporting, and the injection of malicious data. This can cause forensic investigators to identify non-existing targets, fabricate false forensic artifacts, and result in misallocation of assets. For example, a drone may be sitting in a location in City A, but the internal storage may suggest that it flew into a classified area in another city. With a phantom track generation attack, forensic investigators may spend time investigating a fictional scenario.

Vampire Attacks (Resource Exhaustion)

In Vampire attacks, an adversarial node sends maliciously structured messages into a drone swarm network to drain the batteries of the drones, sucking the life out of the swarm (Vasserman and Hopper, 2013). Such attacks exploit the droneswarms reliance on battery power and force it to do extreme physical work that drains the battery power. Specific attacks include stretch attacks, which force the drones to travel in the longest possible distance to the location, carousel attacks in which drones are forced to complete looping maneuvers or control packet overhead attacks in which drones are forced to a wake state that prevents it from going into energy saving mode.

Vampire attacks compromise forensic investigations by compromising the ability of the investigator to track a drone's physical operations through the drone's flash memory storage. The influx of malicious data packets can exhaust the memory buffer, erasing older more critical forensic artifacts. Information about a drone's operations is held in volatile RAM when active and is not cached when a drone suddenly loses power. Forensic analysts can lose encryption information, network connection information, and GPS coordinates. With vampire attacks mimicking normal network traffic, forensic tools like Oxygen cannot look for abnormalities, malicious injections, or unauthorized commands.

In conclusion, adversarial machine learning has transformed both the digital and kinetic battlefield, in which technological deception challenges traditional forensic methodologies. As demonstrated by the deceptive mechanics of Byzantine consensus shifts, phantom track generation, and resource-depleting vampire attacks, modern adversaries do not need to master encryption to compromise a UAV swarm; they can use adversarial machine attacks to exploit its communication protocols and perception layers. This paradigm shift requires an evolution from reactive hardware triage to proactive, AI-driven algorithmic forensics. To lift the "forensic fog" cast by these collective

deceptions, future investigations must blend volatile memory preservation with advanced telemetry validation—ensuring that the digital artifacts recovered from a crash site reflect physical reality rather than an adversary's fabricated narrative.

References

Syed, A. S., Sierra-Sosa, D., Kumar, A., & Elmaghraby, A. S. (2026). AI-enhanced intrusion detection for UAV systems: A taxonomy and comparative review. *Drones*, *9*(10), 682. https://doi.org/10.3390/drones9100682

Alqudsi, Y., & Makaraci, M. (2025). UAV swarms: research, challenges, and future directions. *Journal of Engineering and Applied Science*, *72*(1), Article 12. **https://doi.org/10.1186/s44147-025-00582-3**

XAI-CF. (2024). Examining the Role of Explainable Artificial Intelligence in Cyber Forensics. arXiv. https://doi.org/10.48550/arXiv.2402.02452

PMC. (2026). Security of ADS-B and Remote ID Systems: Cyberattacks, Detection Techniques, and Countermeasures. *PMC*, PMC12846276.

He, W., Cheng, Z., & Fedorov, A. (2024). Byzantine attack resilient bounded consensus. *IEEE Signal Processing Letters*, *31*, 2430–2434. Doi.org

Coombes, M., Shin, H.-S., & Tsourdos, A. (2012). Feasible flight paths for cooperative generation of a phantom radar track. *Journal of Guidance, Control, and Dynamics*, *35*(3), 809–820. doi.org

Vasserman, E. Y., & Hopper, N. (2013). Vampire attacks: Draining life from wireless ad hoc sensor networks. *IEEE Transactions on Mobile Computing*, *12*(2), 318–332. doi.org

About the Author

Rhonda Johnson has Certified Digital Forensics Examiner and Certified Penetration Testing Engineer certifications from Mile2. She serves as e-learning manager/course creator for Ogun Drone Forensics Training LLC in Houston, TX (ogunforensics.com). She has a Master's in Computer and Information Security, and she is currently on the editorial review board for the International Journal of Cyberwarfare and Terrorism and a Peer Reviewer for the International Journal of Digital Crime and Forensics.

Beyond the Hype: Practical Considerations for AI-Assisted Malware Investigation *by Scott A. Macri, Founder & CEO, BITSnBYTES.io, LLC*

What you should know:

Readers should have a basic understanding of malware analysis concepts, including static analysis, dynamic analysis, sandbox reports, indicators of compromise, threat intelligence, and incident response workflows. Familiarity with common cybersecurity terms such as command-and-control, persistence, credential access, attribution, and confidence levels will be helpful, but deep reverse engineering experience is not required.

What you will learn:

Readers will learn how to evaluate AI-assisted malware investigation in practical operational terms. The article explains where AI can help analysts today, where it can mislead, why evidence and human review remain essential, how to protect sensitive investigation data, and how security teams can measure whether AI is improving investigation quality rather than simply producing faster or more polished output.

Introduction: Why Hype Is Not Enough

Artificial intelligence has quickly become one of the most promoted ideas in cybersecurity. In malware investigation, the promise is especially appealing: faster triage, clearer summaries, automated explanations, and help connecting technical artifacts to operational decisions. For overwhelmed analysts facing a steady flow of suspicious files, alerts, indicators, sandbox reports, and incident data, any technology that can reduce friction deserves serious attention.

The evidence problem

But malware investigation is not simply a speed problem, and it is not only a language problem. It is an evidence problem.

A useful investigation depends on what can be observed, what can be verified, what remains uncertain, and what conclusions the evidence can reasonably support. A suspicious executable may contain misleading strings. A sandbox report may show behavior that only appears under certain conditions. A network indicator may be shared across unrelated activity. A code similarity may suggest reuse, but not authorship. Even

experienced analysts must work carefully through ambiguity, incomplete data, and adversary deception.

This is where the hype around AI can become risky. AI-generated output can sound confident even when the underlying evidence is weak, incomplete, or misunderstood. A polished summary can make a tentative finding appear stronger than it is. A plausible explanation can obscure the fact that an analyst still needs to validate the original artifacts, review tool output, and consider alternate interpretations. In malware analysis, a wrong conclusion is not just an academic mistake. It can influence containment decisions, threat reporting, escalation, resource allocation, and leadership confidence.

That does not mean AI has no place in malware investigation. Used carefully, AI can help analysts summarize long outputs, organize observations, draft reviewable notes, translate technical findings for different audiences, and identify questions that still need answers. These are meaningful contributions, especially when teams are under pressure to move quickly. The problem begins when AI is treated as an authority rather than an assistant.

The practical test

The practical question is not whether AI can be added to malware analysis. It can. The more important question is whether AI improves the investigation without weakening analytic discipline. Does it help analysts reason from evidence? Does it preserve uncertainty? Does it make conclusions easier to review? Does it help teams communicate findings more clearly without overstating confidence? Does it reduce cognitive burden while keeping the human analyst accountable for the final judgment?

This article looks beyond the marketing promise of AI-assisted malware investigation and focuses on practical use. It examines where AI can help today, where it can mislead, how analysts should review AI-generated outputs, and what security teams should consider before trusting AI-supported findings in operational environments.

Malware Investigation Is an Evidence Problem Before It Is an AI Problem

Start with observable artifacts

Before evaluating what AI can do for malware investigation, it is worth returning to what malware investigation actually requires. At its core, the work begins with evidence: a file, a hash, a URL, a memory artifact, a process tree, a registry change, a packet capture, a sandbox trace, a YARA match, a suspicious command line, or an analyst observation. Each artifact may be useful, but none of them automatically tells the full story.

The analyst's role is to determine what the evidence supports, what it does not support, and what remains unknown. That distinction is critical. A file hash can identify a specific sample, but not necessarily its operator. A command-and-control domain can show

communication behavior, but not always intent. A matching string can suggest a capability, but it may also be dead code, copied code, junk data, or deliberate misdirection. A sandbox result can reveal behavior, but only under the conditions in which the sample executed.

Uncertainty remains

AI does not remove this uncertainty. In some cases, it can make uncertainty harder to see.

A model may summarize a sandbox report into a clean narrative: the sample establishes persistence, contacts a remote host, drops a secondary payload, and attempts credential access. That summary may be helpful, but the analyst still needs to confirm what happened. Was persistence observed directly, or inferred from a registry write? Was the remote host contacted successfully, or did the sample merely attempt resolution? Was a secondary payload recovered, or only referenced? Was credential access demonstrated by behavior, suggested by an API call, or assumed from a known malware family?

These differences matter. Malware investigations often influence operational decisions. An incident response team may isolate hosts, block infrastructure, escalate to leadership, notify partners, or change detection logic based on the conclusions analysts provide. If an AI system compresses uncertainty into confident language, it can cause teams to act on conclusions that were not fully supported.

This is why AI-generated analysis should be treated as interpretation, not evidence. The evidence remains the original artifact, tool output, observed behavior, and analyst-verified context. AI can help explain, organize, and summarize that material, but it cannot make weak evidence strong. It cannot turn a partial observation into a confirmed fact. It cannot replace the discipline of checking whether a claim is supported by the underlying data.

A practical use of AI begins with this boundary. Ask it to help clarify what is present in the evidence. Ask it to identify what still needs review. Ask it to separate observed facts from possible interpretations. Ask it to highlight assumptions. Those uses can improve the analyst's workflow without surrendering judgment to the model.

Better questions for AI

The best AI-assisted malware investigation does not start with the question, "What does the AI think this is?" It starts with better questions: "What do we know?" "How do we know it?" "What does this evidence actually support?" "What are we assuming?" "What would we need to confirm or reject this assessment?"

When AI helps analysts answer those questions more efficiently, it adds value. When it skips those questions and produces a confident conclusion, it becomes a risk.

Where AI Can Help Analysts Today

The strongest use cases for AI-assisted malware investigation are usually not the most dramatic ones. They are the practical, repetitive, cognitively expensive tasks that consume analyst time before a conclusion can be reached. In that role, AI can be valuable. It can help analysts move through large volumes of material, organize observations, and communicate findings more clearly. The key is to use AI where it supports reviewable work, not where it silently replaces judgment.

Summarization as navigation

One of the clearest near-term uses is summarization. Malware investigations often produce long and fragmented outputs: sandbox logs, process activity, file system events, registry changes, DNS lookups, HTTP requests, extracted strings, disassembly notes, antivirus labels, and threat intelligence references. Individually, these artifacts may be manageable. Together, they can become difficult to review quickly, especially during an active incident. AI can help condense that material into a first-pass summary that points analysts toward behaviors worth reviewing.

For example, an AI assistant might summarize that a sample attempted to modify startup locations, contacted multiple external domains, wrote files into a temporary directory, and spawned a child process with suspicious arguments. That summary can help an analyst prioritize review. It does not prove the sample established persistence, successfully communicated with command-and-control infrastructure, or completed a payload chain. Those details still require validation against the original tool output. Used properly, the summary is a navigation aid, not a conclusion.

Explanation and orientation

AI can also help explain unfamiliar technical details. Analysts frequently encounter APIs, command-line arguments, encoding patterns, packer artifacts, scripting behaviors, or operating system internals that require context. An AI assistant can provide a plain-language explanation of what a Windows API is commonly used for, what a PowerShell flag may indicate, or why a certain registry path matters. This can be especially useful for junior analysts or for experienced analysts working outside their usual specialty. The explanation should still be checked against authoritative references and the actual sample behavior, but it can reduce the time needed to orient the investigation.

Drafting and communication support

Another useful role is note drafting. Analysts often know what they observed but still need to convert fragmented observations into readable case notes or status updates. AI can help turn bullet points into a structured draft: observed behaviors, affected artifacts, possible significance, unresolved questions, and recommended next steps. This can improve communication between malware analysts, incident responders, SOC teams, and leadership. The draft must remain editable, and the analyst should remove unsupported wording, add caveats, and ensure that every claim reflects the evidence.

Gap identification

AI may also help identify gaps. A well-framed prompt can ask what additional evidence would strengthen or weaken a working assessment. For instance, if the current evidence suggests credential theft, the AI might suggest reviewing process access events, browser data access, LSASS interaction, command history, network exfiltration indicators, or related endpoint telemetry. The value is not that the model "knows" what happened. The value is that it can help generate a checklist of investigative questions the analyst may want to consider.

Comparison support

In some cases, AI can support comparison. If provided with prior case notes, known behaviors, or documented malware characteristics, it may help highlight similarities and differences between current observations and previous investigations. This can help analysts spot patterns, but it must be handled carefully. Similarity is not identity. Shared behavior does not prove shared authorship. Reused tools do not prove a common operator. AI can help surface possible relationships, but analysts must determine whether the evidence supports them.

Audience translation

AI can also help translate technical findings for different audiences. A reverse engineering note written for another malware analyst may be too detailed for an incident commander. A leadership update may need to explain risk, scope, and confidence without overwhelming the reader with raw artifacts. AI can help produce appropriate audience language, provided the analyst controls the message and preserves uncertainty. This is one of the safer and more useful applications because it improves communication after the technical review has already been performed.

These uses share a common pattern: AI helps organize, explain, draft, or prompt further inquiry. It does not decide. It does not replace the original evidence. It does not eliminate the need for analyst review. When used in this way, AI can reduce cognitive burden and help analysts spend more time on the parts of the investigation that require expertise: validation, interpretation, judgment, and communication.

Where AI Can Go Wrong

AI-assisted malware investigation becomes risky when the output sounds more certain than the evidence allows. Malware analysis often involves partial observations, conflicting signals, and adversary-controlled artifacts. AI systems are designed to produce coherent responses, but coherence is not the same as correctness. A clean explanation can still be wrong, incomplete, or unsupported.

One common problem is hallucinated technical details. An AI system may explain a function, behavior, string, or command in a way that sounds plausible but does not match the actual artifact. It may infer a capability from a weak indicator, describe behavior that was not observed, or fill gaps with patterns learned from similar-looking reports. In ordinary writing tasks, this may produce an inaccurate paragraph. In malware investigation, it can alter the direction of an incident response.

Another risk is misinterpreting tool output. Static analysis tools, sandboxes, disassemblers, memory tools, and detection engines often produce noisy or context-dependent results. A registry write may be suspicious in one context and benign in another. A failed network connection may be operationally different from a successful command-and-control session. A suspicious API import may indicate capability, but not execution. If AI compresses those distinctions into a simplified narrative, analysts may overstate what occurred.

Attribution is especially vulnerable to overconfidence. Malware labels, infrastructure overlaps, code similarities, and technique mappings can all suggest relationships, but they rarely prove authorship on their own. AI may combine weak signals into a strong-sounding conclusion, such as naming a malware family, campaign, or actor without sufficient evidence. This is dangerous because attribution claims often travel beyond the technical team. Once repeated in briefings or reports, they can be difficult to correct.

AI can also reinforce analyst bias. If an analyst already suspects a certain malware family or actor and prompts the system in that direction, the response may organize the evidence around that theory. The result can feel like confirmation even when the underlying evidence remains thin. This is not unique to AI. Analysts have always had to guard against confirmation bias. AI can accelerate the problem by producing polished support for a premature conclusion.

Another failure mode is the loss of uncertainty. Good malware reporting often depends on careful language: observed, attempted, likely, possible, suspected, unconfirmed, not observed, and unknown. AI-generated summaries may flatten these distinctions. A sample that “attempted to contact” an endpoint may become one that “communicated with” an endpoint. A behavior that “may indicate credential access” may become “credential theft.” A similarity to a known family may become identification as that family. Small wording changes can materially alter the meaning of a finding.

AI may also miss environmental context. Malware behavior depends on execution conditions, operating system version, privileges, network access, user interaction, geolocation checks, anti-analysis logic, and available dependencies. A sandbox result is not always a complete representation of real-world behavior. If AI treats one run as definitive, it may overlook conditions that prevented the sample from revealing additional behavior or caused it to behave differently than it would on a victim system.

There is also a communication risk. AI is good at producing readable narratives, and readability can create misplaced trust. A messy investigation may become a smooth story with a beginning, middle, and end. But real investigations often do not work that way. They contain unresolved questions, conflicting artifacts, and judgments made under uncertainty. When those rough edges disappear from the report, decision makers may believe the situation is clearer than it is.

The practical lesson is not that analysts should avoid AI entirely. It is that AI outputs must be treated as drafts, leads, or interpretations. They should be checked against original artifacts, tool output, and known context. Any statement that affects containment, attribution, reporting, or escalation deserves careful review. In malware investigation, the costliest AI mistake may not be a bizarre hallucination. It may be a reasonable-sounding answer that quietly exceeds the evidence.

The Attribution Trap

Few areas of malware investigation require more caution than attribution. Analysts are often asked to answer questions that sound simple: What malware family is this? Is this connected to a known campaign? Who is behind it? How confident are we? In practice, those questions are rarely simple. They require careful separation of observable facts from analytic judgments.

AI can make attribution riskier because it is good at producing fluent conclusions from incomplete signals. If a report includes a few recognizable behaviors, a familiar string, an infrastructure overlap, or a detection name from a security tool, an AI system may present a confident family or actor association. That answer may sound useful, especially under time pressure. But malware attribution is not a matching exercise based on one or two indicators. It is an analytic process that depends on the quality, uniqueness, and context of the evidence.

Reusable signals can mislead

Many artifacts used in attribution are reusable or misleading. Infrastructure can be shared, compromised, rented, abandoned, or intentionally copied. Malware code can be reused, leaked, purchased, modified, or borrowed from open-source projects. Techniques can be common across many actors because adversaries often adopt what works. Tool marks can be manipulated. Strings and metadata can be planted. Even behavioral similarity may only show that two samples solve the same operational problem in similar ways.

Labels are useful leads, not proof

Detection labels add another complication. Antivirus names and vendor classifications can be useful leads, but they are not final proof. Different vendors may use different names for the same family, the same name for related but distinct activity, or broad labels

that describe behavior rather than lineage. If AI treats labels as authoritative, it may convert a tentative classification into a firm statement. That can create a false sense of certainty.

The same problem applies to threat actors and campaign names. A model may recognize names from public reporting and connect them to observed behaviors, tools, or infrastructure. But public reporting varies in quality, age, confidence, and terminology. Some reports describe clusters of activity without naming an actor. Others use vendor-specific naming conventions. Some associations change over time as more evidence emerges. AI can compress this complexity into a neat answer that sounds more settled than intelligence actually is.

Good attribution requires disciplined language. Analysts should distinguish between what was observed and what is assessed. For example, “the sample attempted to contact this domain” is different from “the sample used known infrastructure associated with this actor.” “This behavior resembles prior reporting on a malware family” is different from “this sample belongs to that family.” “The evidence is consistent with” is different from “this was conducted by.” These distinctions are not academic. They tell decision makers how much weight to place on the conclusion.

AI-assisted workflows should preserve those distinctions, not erase them. When AI is used during attribution-related analysis, it should be asked to identify possible explanations, supporting evidence, contradicting evidence, and unresolved questions. It should not be asked to produce a final actor name from limited artifacts. Analysts should be especially skeptical of any output that gives a confident attribution without explaining the evidence and its limitations.

Treat attribution as a hypothesis

A safer approach is to treat attribution as a hypothesis. The analyst can ask: What evidence supports this association? What evidence contradicts it? Are the indicators unique enough to matter? Could the infrastructure be shared? Could the code have been reused? Are there alternative explanations? What confidence level is justified? What additional evidence would change the assessment?

This approach does not eliminate uncertainty, but it makes uncertainty visible. That is the point. In malware investigation, responsible attribution is not about reaching the most dramatic conclusion. It is about stating only what the evidence can support, explaining what remains unresolved, and being willing to revise the assessment when better information becomes available.

Human Review Must Be More Than a Rubber Stamp

The phrase “human-in-the-loop” appears frequently in discussions about AI and cybersecurity, but it can be too vague to be useful. In some systems, it means a person

clicks approve before an automated result is finalized. In malware investigation, that is not enough. Human review must mean that an analyst can inspect the evidence, challenge the interpretation, revise the conclusion, and document the reasoning behind the final assessment.

AI-generated findings should be treated like a draft from a junior assistant: useful, potentially insightful, but not authoritative. The analyst should ask where each statement came from, whether it is supported by the artifacts, and whether the wording overstates the evidence. A summary that says a sample "stole credentials" should be checked carefully. Did the sample access credential stores? Did it dump process memory? Did it invoke functions associated with credential access? Did it exfiltrate data? Or did the AI infer credential theft from a suspicious import, a detection label, or a behavior commonly seen in another malware family?

Separate facts, interpretations, and assumptions

A meaningful review process should preserve the difference between facts, interpretations, and assumptions. Facts are observations grounded in evidence: a hash value, a file path, a command line, a registry modification, a network request, a process relationship, or an extracted configuration item. Interpretations explain what those facts may mean. Assumptions fill gaps when evidence is incomplete. All three can be useful, but they should not be blended without distinction.

Look for alternate explanations

Analysts should also look for missing alternatives. If AI proposes that a sample is a downloader, what else could explain the same behavior? Could the observed network activity be a failed update check, an anti-analysis probe, or a decoy endpoint? If the system suggests persistence, was persistence achieved, or was the sample only attempting to write to a startup location? If a behavior resembles a known family, is the similarity distinctive, or is it common across many commodity malware samples?

This is especially important during active incidents, where time pressure encourages shortcuts. A polished AI summary can appear ready for reporting before the underlying work is complete. Analysts and team leads should resist that temptation. Review should include checking source artifacts, validating key tool outputs, preserving uncertainty, and identifying any unresolved questions that matter to containment or reporting.

Own the final judgment

Human review also has a communication role. Analysts are responsible not only for determining what happened, but for explaining how strongly they know it. A good final report should make clear which behaviors were directly observed, which conclusions are assessed, what confidence level is appropriate, and what evidence would be needed to

strengthen or revise the assessment. AI can help draft that language, but the analyst must own it.

The goal is not simply to keep a human somewhere in the process. The goal is human-led investigation. AI can assist with organization, explanation, and drafting, but the analyst remains accountable for the final judgment. In practical terms, that means AI should make review easier, not harder. It should help analysts see the evidence more clearly, not bury the evidence beneath fluent prose.

Protecting Sensitive Investigation Data

AI-assisted malware investigation raises a practical question that every security team should answer before using the technology: what information is being sent to the AI system, where is it processed, and who can access it afterward? Malware investigations often involve sensitive data. A sample may contain victim information, internal hostnames, usernames, credentials, proprietary documents, configuration details, source code fragments, law enforcement-sensitive information, or government data. Even a short prompt can disclose more than an analyst intends.

Know what leaves the environment

This matters because AI tools are often used through natural language interfaces. Analysts may paste sandbox summaries, endpoint logs, packet captures, filenames, file paths, command lines, decoded strings, or entire reports into a prompt. That material may seem routine during an investigation, but it can reveal internal infrastructure, business processes, affected systems, or details about an active incident. In some environments, it may also trigger legal, regulatory, contractual, or classification concerns.

Teams should avoid treating AI use as an individual analyst preference. It should be governed by policy. Analysts need clear guidance on what types of data may be submitted to external AI services, what must remain inside controlled environments, and what requires approval before use. The policy should account for the sensitivity of the investigation, the type of data involved, and the risk of exposing victim, customer, partner, or government information.

Retention and output handling

Retention is another concern. Before using an AI service, teams should understand whether prompts, uploaded files, generated responses, or metadata are stored; whether they may be used for model improvement; how long they are retained; and whether administrators or service providers can review them. These details affect whether the tool is appropriate for malware analysis, incident response, or threat intelligence work involving sensitive artifacts.

There is also a risk in AI-generated output. A summary can unintentionally include sensitive indicators, hostnames, user identifiers, internal IP addresses, or investigative assumptions that should not be broadly distributed. Analysts should review AI-assisted reports before sharing them outside the investigation team. They should remove unnecessary sensitive details, apply the correct handling markings, and ensure that the report is suitable for its intended audience.

Untrusted content and access control

Prompt injections and malicious content are additional concerns. Malware samples, scripts, documents, configuration files, and command-and-control responses can contain text designed to influence automated systems. If AI tools are asked to analyze untrusted content without safeguards, there is a possibility that malicious instructions embedded in the material could affect the model's response or the surrounding workflow. Analysts should treat untrusted text as evidence to be examined, not instructions to be followed.

Access control also matters. Not every analyst, responder, manager, or external partner should see every artifact or conclusion. Investigation data may need to be shared on a need-to-know basis. AI-assisted workflows should not become a shortcut around existing handling rules. If a team would not email a malware sample, credential dump, or victim-specific report to a broad distribution list, it should not casually paste the same content into an uncontrolled AI service.

Protecting sensitive data does not mean AI is unusable. It means teams need deliberate boundaries. They can sanitize prompts, remove identifiers, use approved internal tools, summarize only non-sensitive portions, restrict AI use to low-risk tasks, or require additional review for high-sensitivity investigations. The central principle is simple: AI should not weaken the controls that already exist around malware evidence, incident data, and intelligence reporting.

How to Use AI Without Weakening the Investigation

AI can be useful in malware investigation when analysts give it the right role. It should assist the investigation, not drive it. That starts with a simple rule: use AI to help examine and communicate evidence, not to invent conclusions beyond the evidence.

Prompt around observations, not verdicts

A practical way to apply this rule is to frame AI prompts around observations instead of verdicts. Rather than asking, "What malware family is this?" an analyst might ask, "Based on these observed behaviors, what possible capabilities should be reviewed, what evidence supports each possibility, and what additional artifacts would help confirm or reject them?" The second question keeps the work anchored in investigation. It invites the AI to support analytic thinking without pretending that a final answer is available.

Separate facts from interpretations

Analysts should also ask AI to separate facts from interpretations. A fact might be that a process created a file in a startup directory. An interpretation might be that the sample attempted persistence. A stronger conclusion, such as "the malware established persistence," should only be used if the evidence shows that the mechanism was successfully created and would execute as intended. This distinction is easy to lose in AI-generated summaries, so analysts should make it explicit.

Ask for uncertainty

Another useful practice is asking AI to identify uncertainty. For example: "What does this evidence not prove?" or "What alternate explanations should be considered?" These questions help counter the tendency of AI systems to produce smooth narratives. They also help analysts avoid premature closure, especially when early evidence appears to support a familiar malware family, behavior, or campaign.

Use checklists carefully

AI can also be used as a checklist generator. If the sample appears to perform discovery, credential access, command-and-control communication, or payload staging, the analyst can ask what additional evidence would normally be reviewed to support that assessment. The answer may remind the analyst to check process relationships, command-line arguments, registry changes, network timing, dropped files, memory artifacts, authentication logs, or endpoint telemetry. The checklist still needs expert review, but it can reduce the chance that important questions are missing during a busy investigation.

Treat generated language as editable

When drafting reports or notes, analysts should treat AI-generated language as editable working material. The final version should remove unsupported claims, add caveats, and preserve confidence levels. Words such as "confirmed," "likely," "possible," "attempted," "observed," and "not observed" should be used carefully. These terms are not filler. They communicate the strength of the evidence and help readers understand how much confidence they have in each finding.

Teams should also be cautious about using AI to merge multiple sources into one narrative. Combining sandbox output, static analysis notes, vendor labels, and threat intelligence references can be helpful, but only if the resulting summary keeps sources distinguishable. If the reader cannot tell whether a claim came from direct observation, a tool result, a third-party report, or AI interpretation, the output is not ready for operational use.

The same discipline applies to recommendations. AI may suggest blocking indicators, isolating hosts, escalating an incident, or hunting for related activity. Those suggestions can be useful starting points, but they should be evaluated against the organization's environment, risk tolerance, and operational constraints. A recommendation that makes sense in one network may be disruptive or incomplete in another.

Responsible use of AI should leave the investigation stronger than it was before. The analyst should have clearer notes, better questions, more organized evidence, and a more precise understanding of what remains unresolved. If AI produces a confident answer but makes it harder to see the evidence, challenge the reasoning, or explain uncertainty, it has weakened the investigation rather than improved it.

What Security Teams Should Evaluate Before Adopting AI-Assisted Malware Tools

Once a team understands where AI can help and where it can be misled, the next question is how to evaluate AI-assisted malware tools in practice. The answer should not start with the model's name, the interface, or the marketing claim. It should start with the investigation: does the tool help analysts produce better, faster, and more defensible work?

Start with evidence visibility

Evidence visibility should be the starting point. If an AI-assisted tool provides a summary, conclusion, or recommendation, analysts should be able to determine what information the output was based on. A statement about persistence should point back to relevant files, registry, service, scheduled task, or execution evidence. A statement about command-and-control behavior should be traceable to network activity, configuration data, or observed communication attempts. If the tool produces conclusions that cannot be checked against source material, it is creating trust without accountability.

Preserve uncertainty

Uncertainty is just as important. Malware analysis often produces partial answers. A useful tool should help preserve that uncertainty rather than hide it. It should make room for terms such as "observed," "attempted," "possible," "likely," and "unknown." It should not force analysts to binary conclusions when the evidence supports only a qualified assessment. Teams should be wary of systems that convert messy investigation data into confident labels without showing the reason behind them.

Keep analysts in control

Analyst control is another practical test. Analysts should be able to correct AI-generated summaries, reject unsupported interpretations, add context, and document why a conclusion changed. A tool that treats AI output as final is poorly suited for serious

investigation work. A tool that treats AI output as draft material for analyst review is more consistent with how malware analysis operates.

Fit the investigation workflow

A tool also must fit the way investigations happen. Malware analysis draws from static analysis, dynamic analysis, endpoint telemetry, sandbox results, network logs, memory artifacts, threat intelligence, and case notes. AI should help analysts work across those materials. It should not create a disconnected side channel where important reasoning lives outside the investigation record. If analysts must copy evidence into a separate chatbot, manually paste the response into a report, and then reconstruct where the claim came from later, the team may gain speed while losing traceability.

Protect sensitive data

Data protection cannot be treated as an afterthought. Before adopting a tool, teams should understand how samples, prompts, reports, attachments, and generated outputs are handled. They should know whether data leaves the organization, whether it is retained, whether it may be reviewed or used for training, and whether access can be restricted by role or sensitivity. If those questions cannot be answered clearly, the tool may not be appropriate for sensitive malware investigations.

Evaluate reporting quality

Reporting quality also deserves close attention. AI-assisted reporting should make findings clearer, not more dramatic. It should help analysts communicate what happened, what evidence supports the assessment, what remains unknown, and what actions are recommended.

Teams should review sample outputs carefully.

Do the reports distinguish facts from interpretations?

Do they preserve caveats?

Do they cite or reference supporting evidence?

Do they avoid unsupported attribution?

Do they help incident responders and decision makers act appropriately?

Measure operational value

Finally, teams should measure operational value, not demo appeal. During a pilot, they can ask whether the tool reduces triage time, improves consistency between analysts, helps identify missing evidence, improves handoffs, or produces clearer reports. They should also ask whether it introduces new review burdens, creates overconfidence, or causes analysts to spend more time correcting polished but unsupported text.

A useful AI-assisted malware tool should make the analyst stronger. It should improve visibility, organization, review, and communication. It should not ask the team to trust a black box because the output sounds convincing. In malware investigation, a tool earns trust by helping analysts stay close to the evidence.

Measuring Whether AI Is Actually Helping

AI-assisted malware investigation should be measured by operational value, not by how impressive the demonstration looks. A tool may generate a polished summary, produce a long report, or answer questions in natural language, but those outputs do not automatically mean the investigation improved. The better question is whether AI helps analysts reach more accurate, consistent, and defensible conclusions with less wasted effort.

Speed is not enough

Speed is useful, but speed alone is not enough. A faster summary has limited value if analysts spend the saved time correcting unsupported claims. A faster report can be harmful if it removes uncertainty or makes weak evidence appear stronger than it is. Teams should measure whether AI reduces friction without reducing analytic discipline.

Triage efficiency

One useful measure is triage efficiency. If AI helps analysts quickly understand the major behaviors in a sandbox report, identify which artifacts deserve closer review, or summarize repetitive tool output, it may reduce the time needed to reach an initial assessment. The important point is that the initial assessment should still be reviewed against the evidence. The goal is faster orientation, not faster overconfidence.

Consistency across analysts

Consistency is another important measure. Malware investigations often vary depending on analyst experience, time pressure, and available context. AI may help standardize note structure, remind analysts to consider common evidence categories, and produce clearer draft language. That can be valuable if it improves review quality across the team. It is less valuable if it simply makes every report sound similar while hiding differences in evidence quality.

Documentation and handoffs

Documentation quality should also improve. A good AI-assisted process should help analysts capture what was observed, what was inferred, what remains unknown, and what follow-up work is needed. If AI helps turn scattered notes into a clearer record, it can support handoffs between malware analysts, incident responders, threat intelligence teams, and leadership. But the final documentation must still reflect the analyst's judgment and should not include claims that were not verified.

Teams should also watch for reduction in duplicate work. During active incidents, multiple analysts may review overlapping artifacts, repeat similar searches, or recreate summaries that already exist elsewhere. AI can help by organizing prior observations and making existing context easier to find. This is useful only if the underlying information is accurate, current, and tied back to the investigation record.

Another measure is the quality of handoffs. Malware analysis rarely ends with the analyst. Findings may be passed on to SOC teams for detection engineering, incident responders for containment, threat intelligence teams for enrichment, legal or compliance teams for notification decisions, or executives for risk briefings. AI can help translate technical details into languages appropriate for each audience. The measure of success is not whether the text sounds polished. It is whether the recipient understands the finding, the confidence level, and the recommended action.

Measure negative effects

Teams should also measure negative effects. Does the tool encourage unsupported attribution? Does it make analysts less likely to inspect original artifacts? Does it generate summaries that require heavy correction? Does it expose sensitive data? Does it create another place where investigation notes must be managed? Does it increase review burden for senior analysts? These costs should be considered alongside any time savings.

The strongest evidence of value is improved decision quality. AI should help analysts ask better questions, find relevant evidence faster, document reasoning more clearly, and communicate uncertainty more accurately. If it does those things, it is contributing to the investigation. If it mainly produces confident prose, it may be improving appearance rather than analytic quality.

The Future of AI-Assisted Malware Investigation

AI-assisted malware investigation will likely become more capable, but its value will still depend on how well it supports disciplined analysis. The future is not likely to be a simple shift from human investigation to autonomous investigation. Malware analysis is too dependent on context, evidence quality, adversary behavior, and operational judgment for that to be a safe assumption. A more realistic future is one in which AI helps analysts move through information faster while humans remain responsible for interpretation and decisions.

Evidence summarization and case comparison

One likely area of improvement is evidence summarization. As tools become better at handling structured and unstructured information, analysts may be able to ask more useful questions across sandbox output, static analysis notes, endpoint telemetry, memory artifacts, and prior case records. Instead of manually searching through multiple reports, analysts may be able to ask what changed between two runs, which behaviors

were newly observed, or which artifacts remain unexplained. That kind of assistance could reduce time spent navigating data and increase time spent validating meaning.

AI may also become more useful for case comparison. Malware analysts often benefit from knowing whether a sample resembles prior investigations, whether similar indicators were seen before, or whether a behavior pattern has appeared in previous incidents. Future AI-assisted workflows may help surface those connections more quickly. Even then, similarity should remain a lead, not a conclusion. Analysts will still need to determine whether the comparison is meaningful, whether the evidence is current, and whether alternative explanations exist.

Investigation planning

Investigation planning is another promising area. AI could help analysts organize the next steps in an investigation based on what is known, what remains unresolved, and what operational decisions depend on the answer. For example, if the evidence suggests possible credential access but does not confirm it, AI could help identify what additional artifacts should be reviewed. This kind of support is valuable because it helps preserve analytic discipline under pressure. It helps analysts ask better questions rather than skip directly to confident answers.

Knowledge retrieval and governance

AI may also improve knowledge retrieval. Malware investigation often depends on remembering tool behavior, operating system internals, malware tradecraft, detection logic, and prior reporting. AI-assisted search and explanation may make that knowledge easier to access, especially for less experienced analysts. However, retrieved information must still be evaluated for accuracy, relevance, and date. Old reporting, vendor-specific labels, or unrelated examples can mislead if presented without context.

As AI becomes more integrated into security operations, governance will become more important, not less. Teams will need clearer policies for what data can be processed, how outputs are reviewed, how sensitive findings are handled, and how AI-assisted conclusions are documented. They will also need to decide when AI should be used, when it should be avoided, and when additional human review is required. The more capable the tool becomes, the more important it is to define its boundaries.

AI as teammate, not authority

The strongest future model is not AI as an independent malware analyst. It is AI as an analytic teammate that helps organize evidence, identify gaps, draft explanations, and support review. That model keeps the analyst in control while still taking advantage of useful automation. It also recognizes that the hard part of malware investigation is not merely producing an answer. The hard part is knowing whether the answer is supported, how confident the team should be, and what decisions can safely be made from it.

Conclusion: Practical Beats Magical

AI will continue to influence malware investigation, and that is not a bad thing. Analysts need help managing volume, organizing evidence, explaining technical behavior, and communicating findings under pressure. Used carefully, AI can support those tasks. It can help summarize long outputs, draft clearer notes, identify possible gaps, and make complex findings easier to understand across technical and non-technical audiences.

But AI does not change the fundamentals of malware investigation. The work still depends on evidence, context, validation, and judgment. Analysts still need to determine what was observed, what was inferred, what remains unknown, and how strongly the evidence supports a conclusion. A fluent AI-generated answer does not remove the need to inspect artifacts, review tool output, question assumptions, and preserve uncertainty.

The most practical approach is to treat AI as assistance, not authority. It can help analysts move faster, but speed should not come at the expense of accuracy or defensibility. It can help produce better language, but polished writing should not be confused with stronger evidence. It can suggest possibilities, but analysts must decide which possibilities are supported and which remain speculative.

Security teams should be cautious of claims that promise fully autonomous malware investigation, instant attribution, or final reports without meaningful review. Those promises are attractive because they reduce difficult discipline to a simple output. Real investigations are rarely that clean. They involve partial evidence, adversary deception, tool limitations, environmental conditions, and decisions made under uncertainty.

The better question is not, “Can AI analyze malware?” The better question is, “Can AI help analysts produce better, faster, and more defensible investigations?” If the answer is yes, the technology has value. If the answer depends on hiding uncertainty, trusting unsupported conclusions, or replacing analyst judgment with confident prose, the risk may outweigh the benefit.

Beyond the hype, the future of AI-assisted malware investigation should be practical, evidence-driven, and human-led. The goal is not magical automation. The goal is better analysis.

Good Uses and Risky Uses of AI in Malware Investigation

AI can be useful in malware investigation when the task is bounded, reviewable, and grounded in evidence. It becomes riskier when it is asked to make final judgments, infer intent, or produce conclusions that analysts cannot easily verify.

Table 1. Good uses and risky uses of AI in malware investigation

Good uses include	Risky uses include

Summarizing long tool outputs, such as sandbox reports, static analysis notes, endpoint logs, or extracted strings.	Assigning malware family names without strong supporting evidence.
Drafting investigation notes that analysts can review, correct, and refine.	Making threat actors or campaign attribution claims.
Explaining unfamiliar APIs, command-line options, file paths, registry keys, or operating system behaviors.	Producing confidence levels without explaining the evidence behind them.
Identifying follow-up questions or possible evidence gaps.	Generating final reports without analyst review.
Helping translate technical findings into clearer language for incident responders, SOC teams, leadership, or other stakeholders.	Combining weak signals into a polished but unsupported conclusion.

The safest pattern is to use AI to help analysts think, not to let AI decide what the evidence means.

Analyst Prompting Principle

When using AI during malware investigation, prompts should keep the model anchored to the evidence. A useful prompt does not ask the AI to guess the answer. It asks the AI to explain what the available evidence may indicate, what it does not prove, and what additional information would be needed to support a stronger assessment.

Table 2. Prompt framing for AI-assisted malware investigation

Weak prompt	Better prompt
“What malware family is this?”	“Based on the observed behaviors below, what possible capabilities should be reviewed? For each possibility, identify the supporting evidence, the limitations of that evidence, and what additional artifacts would help confirm or reject the assessment.”

This framing helps preserve analytic discipline. It encourages the AI to support the analyst’s reasoning process rather than produce a premature conclusion. It also reminds the analyst to look for uncertainty, alternative explanations, and missing evidence before reporting a finding.

In practice, the best prompts are specific, evidence-bound, and review-oriented. They ask the AI to organize observations, identify gaps, compare possibilities, and clarify wording. They do not ask the AI to replace the analyst's judgment.

Red Flags in AI Malware Investigation Claims

Security teams should be cautious when AI-assisted malware tools promise more certainty than malware investigation can usually support. Strong claims may sound appealing during a busy incident, but they should be evaluated carefully before they influence operational decisions.

Table 3. Common red-flag claims and why they deserve scrutiny

Claim	Why it deserves scrutiny
"fully autonomous malware attribution"	should raise concern. Attribution depends on evidence quality, context, source reliability, and uncertainty. It is not something that should be delegated entirely to an automated system.
"instant actor identification"	are also risky. Shared infrastructure, copied techniques, reused tools, and overlapping malware capabilities can all create misleading similarities. A fast answer is not necessarily a supported answer.
"no analyst review required"	should be treated as a major warning sign. Malware investigation requires judgment. Analysts need to inspect evidence, validate tool output, consider alternate explanations, and decide how strongly the evidence supports a conclusion.
"guaranteed detection"	should also be questioned. Malware behavior changes, environments vary, and adversaries adapt. No single tool or model can remove the need for layered analysis and continuous validation.
"one-click final report"	may sound efficient, but final reporting should not be reduced to automatic prose generation. A useful report should distinguish observed facts from

	interpretations, preserve confidence levels, identify unresolved questions, and support review.
“AI replaces reverse engineering”	oversimplify both AI and reverse engineering. AI may help explain code patterns, summarize observations, or assist with documentation, but it does not remove the need for expert analysis when the investigation requires deep technical understanding.

The more a claim suggests that AI eliminates uncertainty, analyst judgment, or evidence review, the more carefully it should be examined.

Key Terms

Table 4. Key terminology used in the article

Term	Definition
AI-assisted malware investigation	The use of artificial intelligence to support malware analysis tasks such as summarization, explanation, note drafting, evidence organization, and investigative planning. In this article, AI-assisted does not mean autonomous investigation.
Artifact	Any item of evidence reviewed during an investigation, such as a malware sample, hash, string, log entry, file path, registry key, packet capture, process event, memory artifact, or sandbox result.
Attribution	The analytic process of assessing whether malware, infrastructure, behavior, or activity may be associated with a malware family, campaign, threat cluster, or actor. Attribution should be treated as a judgment based on evidence, not as a simple label.
Command-and-control (C2)	Communication between malware and attacker-controlled or attacker-used

	infrastructure. C2 activity may support tasking, data transfer, payload delivery, or operational control.
Confidence level	A statement of how strongly the available evidence supports an analytic judgment. Confidence should reflect evidence quality, consistency, source reliability, and remaining uncertainty.
Dynamic analysis	Observing malware behavior during execution, often in a sandbox or controlled environment.
Indicator of compromise (IOC)	A technical indicator that may suggest malicious activity, such as an IP address, domain, URL, file hash, registry path, mutex, filename, or other observable.
Sandbox	A controlled environment used to execute and observe suspicious files or behaviors. Sandbox results are useful, but they may not reveal all behavior because malware can depend on timing, user interaction, environment checks, privileges, or network conditions.
Static analysis	Examining a file or artifact without executing it. This may include reviewing strings, headers, imports, metadata, embedded resources, packer indicators, or disassembly.
Threat intelligence	Contextual information about threats, including malware families, tactics, techniques, procedures, campaigns, infrastructure, and observed activity. Threat intelligence should be evaluated for relevance, age, confidence, and source quality.

References

National Institute of Standards and Technology (NIST). Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile. July 2024.

Useful for understanding generative AI risk management considerations, including governance, validation, oversight, and organizational controls.

National Institute of Standards and Technology (NIST). Special Publication 800-61 Revision 3: Incident Response Recommendations and Considerations for Cybersecurity Risk Management.
Relevant to the incident response context in which malware investigation findings are often used for containment, escalation, coordination, and risk management decisions.

MITRE. MITRE ATT&CK®.
A globally accessible knowledge base of adversary tactics and techniques based on real-world observations, commonly used for threat modeling, detection, malware analysis, and cyber threat intelligence work.

Cybersecurity and Infrastructure Security Agency (CISA). Malware Next-Gen.
Relevant background for readers interested in operational malware analysis services and automated malware submission workflows.

Macri, Scott A. Malware on the Move: Rethinking Threat Analysis in the Age of AI and Adversarial Innovation. BITSnBYTES.io, LLC, April 2025.
Related prior work discussing malware volume, analyst fatigue, automation gaps, hybrid analysis, and the need for improved malware defense workflows.

About the Author

Scott A. Macri is the Founder and CEO of BITSnBYTES.io, LLC, a small business based in Ashburn, Virginia, specializing in secure software development, cybersecurity engineering, cloud-native systems, and federal technology solutions. He has extensive experience supporting government missions across software development, malware analysis engineering, DevSecOps, systems integration, and secure application delivery.

Scott has supported federal cybersecurity and national security programs, including work related to malware analysis platforms, threat investigation workflows, and mission-focused software modernization. He is also the creator of THRaXe, a deterministic cyber decision support platform designed to support structured malware analysis, intelligence correlation, and evidence-traceable analyst workflows.

In Today's Rapidly Evolving AI Landscape, Can Organizations Rely on AI Bots? by *Tharaka Singharage*

Will AI systems obey only the instructions provided?

As AI transforms the industry, businesses are keen to embed AI capabilities into their offerings to minimise human involvement, cut down on costs, and massively enhance process efficiency. AI is good at repetitive tasks that are defined by rules and can provide customer support, data processing and automation around the clock and at scale. Yet, when there is an increasing reliance, one might wonder whether AI bots can be trusted to function solely within a set scope?

If AI is not used properly, it can also be misused, or "weaponized," against the platforms it serves, creating risks of data breaches, reputational issues, or harm to the platforms themselves.

Data Security and Compliance Challenge

Security of data is a major concern for the adoption of AI. AI bots, in particular, are often tied to channels that interact with customers, such as WhatsApp, websites, or internal applications, and can have access to extensive amounts of organisational information. They can unwittingly or intentionally share sensitive data if not adequately isolated.

There are some key technical challenges that pose a risk, such as:

- Prompt Injection Attacks: These attacks involve users providing malicious or cleverly designed inputs that can alter the bot's intended action, causing it to display system prompts, internal documentation, a database schema, or even take unauthorized action.
- Regurgitation & Memorization: When including sensitive information or data in prior interactions, it is important to note that large language models can sometimes regurgitate the same information or context.
- Model Inversion and Extraction Attacks: Adversaries can try to deconstruct the model or its internal mechanism by asking a series of questions to the model.
- Hallucinations of Sensitive Details: The model could generate incorrect but harmful information, like internal file paths, API keys, or out-of-date infrastructure information, even when the intent is benign. Further, it is important for organisations to ensure bias and fairness. Regulatory frameworks like GDPR, CCPA, and upcoming AI Laws (e.g., EU AI Act) may create legal and ethical obligations that are breached by poorly trained or unmonitored models. Poorly trained or unmonitored models can

lead to legal and ethical violations of regulatory standards such as GDPR, CCPA or emerging AI Acts (e.g., EU AI Act).

- Real-World Test Case: Information Disclosure through AI Bot. I recently had an experience with an AI-powered customer service bot developed with Claude (Anthropic), and hosted on a WhatsApp Business API account. The bot was able to handle normal customer enquiries. But, with carefully worded and non-threatening queries, it started to share very personal information, such as:
  - The company's web development platforms/technologies.
  - Inconsistencies in the links and references to outdated plugins and libraries.
  - Potential vulnerabilities and mis-configurations that could have a severe impact on security.
  - Internal file paths and directories that shouldn't be public.

The incident is an alarming reminder of the potential for AI bots to serve as covert surveillance tools for malicious players. The appearance of helpful conversations can quickly escalate to an information disclosure vulnerability.

Putting EFG's in place

Organizations need to take the same security measures with AI bots as they would with any production application to manage these threats.

Technical controls that are recommended are:

- Prompt Guardrails: Implement prompt infection detection and prevention by employing prompt infection detection layers such as LangChain, LlamaGuard or custom classifiers.
- Output Filtering & Redaction: Review and strip out any data elements in the response containing PII, internal URLs, credentials, technical infrastructure information, etc. before the delivery.
- Retrieval-Augmented Generation (RAG) with Access Controls: Link the bot to sanitized databases containing knowledge for specific roles only, with specific vector database permissions.
- Follow the Least Privilege Principle: Avoid granting the AI bot direct access to production databases or the entire context of the system. Apply API rate limitation and audit logs.
- For critical operations, use approval workflows: sandboxing and Human-in-the-Loop.
- Regular Adversarial Testing (Prompt injection, Jailbreaking, Data Extraction): Test continuously before going live and after each model update.
- Logging and Monitoring: Monitor all interactions, and detect anomalies that may indicate abuse.
- Model Selection & Fine-tuning: Support models that come with safety guards (such as Claude's constitutional principles) and use extra alignment methods.

Final Thoughts

While AI bots are invaluable, it's important to earn trust, rather than take it for granted. Before implementing AI customer service agents or internal automation systems in production, organizations need to put in place clear boundaries and strong guardrails

during development, perform in-depth security testing, and continuously monitor. Otherwise, you run the risk of turning powerful productivity tools into big liabilities. Security, privacy and reliability should be key and fundamental requirements, not afterthoughts, in the race to adopt AI.

About the Author

Tharaka Singharage is an experienced professional specializing in information security governance, risk, and compliance. He holds an MSc in Information Security and is a certified Lead Auditor for ISO 27001 and ISO 27701. Additionally, he is certified in TISAX, CC, Governance, Risk, and Compliance. Tharaka is an active member of (ISC)^2, serves on the ISC2 Colombo Chapter Board, and is a member of the Internet Society.

Data Carving Biochip using Proxmark3 by *Viviane Cruz*

Introduction

The evolution of **NFC (Near Field Communication)** and **RFID (Radio Frequency Identification)** technologies is reshaping everyday interactions with technology. This phenomenon is especially noticable when these devices are integrated into a human body using biochips or wearable devices.

At the same time, it introduces new challenges and opportunities in the field of **digital forensics**.

In scenarios where devices are implanted or embedded into personal accessories, these elements become potential sources of digital evidence. Like smartphones or computers, biochips can store relevant information that can be collected, analyzed, and interpreted during investigations.

This article demonstrates, in practice, how to perform data reading, extraction, and analysis from an implanted biochip using the **Proxmark3** tool.

Figure 1- Proxmark3

My Motivation

From previous research, using a Xbox scenario, where the primary motivation was to safely reuse storage media, the motivation behind this study is fundamentally different.

In this case, the driving factor was curiosity and specifically, understanding how my own personal data and information were stored in an NFC biochip implanted in my body. This shift from traditional storage media to an embedded personal device introduces a unique investigative perspective, where the subject of analysis and the data source converge. As a result, this work not only explores technical aspects of data acquisition and carving but also reflects on the implications of interacting with one's own digital artifacts.

About NFC and RFID

The use of biochips based on **NFC (Near Field Communication)** and **RFID (Radio Frequency Identification)** technologies has expanded significantly in recent years, particularly within the context of biohacking and wearable devices. These technologies enable wireless communication between the chip and compatible readers, allowing fast, secure, and contact less data exchange.

In the case of implantable biochips, such as those based on the NTAG family, NFC technology is widely used for storing and sharing structured data, including NDEF records. RFID can be used for identification and access control in physical systems.

Among the main applications of these technologies, the following stand out:

Figure 2- NFC & RFID Applications

The integration of these technologies into implantable biochips or personal gadgets expands the possibilities for interaction with the digital environment, while also introducing new challenges related to **security, privacy, and forensic analysis** of such devices.

Data Acquisition Using Proxmark3

To simulate a real forensic scenario, a biochip implanted in the left hand was analyzed.

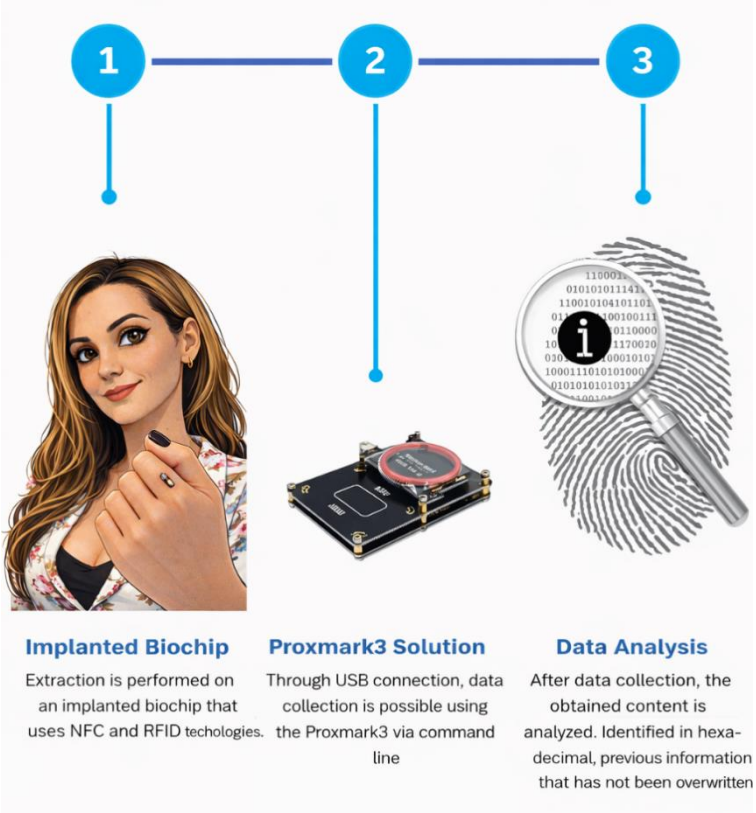


Figure 3-Step-by-Step Data Acquisition Process

The Proxmark3, connected via USB, enables interaction with RFID/NFC devices in a non-invasive manner.

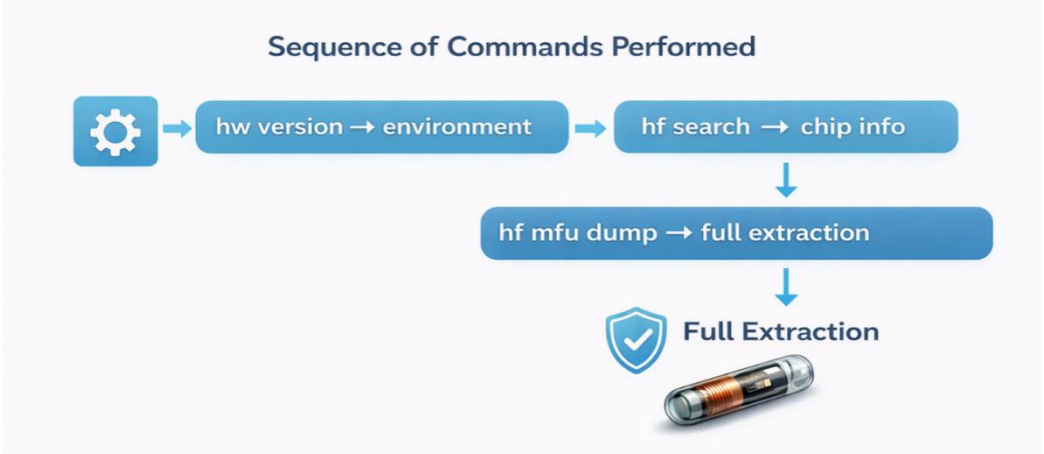


Figure 4- Sequence os Commands Performed

From the Proxmark3 command line, the first step is to identify the system environment using:

```
$ hw version
```

This command reveals firmware, hardware architecture, and system configuration. During the analysis, a mismatch between firmware and client versions was detected — a critical factor that may impact reliability and should be considered in forensic interpretation.

The *hw* version command enabled a detailed characterization of the technical environment used during data acquisition, including information about the client, embedded firmware, hardware architecture, and operational modules available on the Proxmark3 device.

The analysis revealed that the equipment was running **PM3 Generic firmware**, indicating that it is a platform compatible with devices from the Proxmark3 Easy family. Additionally, the versions of the bootrom and the embedded operating system on the ARM microcontroller were identified, as well as the FPGA modules responsible for supporting different RFID/NFC communication protocols.

A critical factor for the reliability of the process was the presence of an inconsistency between the client version and the device firmware, as indicated by the incompatibility warning. This condition may directly impact operational stability, particularly during data reading and extraction tasks, and should be taken into consideration when interpreting the results obtained during the forensic acquisition process.

Log:

```
[usb|script] pm3 --> hw version

[ Proxmark3 ]
[ Client ]
 Iceman/master/v4.21128-410-g698cd8906-suspect 2026-04-06 21:08:12 1894bec6d
 Compiler..... MinGW-w64 13.2.0
 Platform..... Windows (64b) / x86_64
 Readline support..... present
 QT GUI support..... present
 Native BT support..... absent
 Python script support..... present ( 3.11.5 )
 Python SWIG support..... present
 Lua script support..... present ( 5.4.6 )
 Lua SWIG support..... present

[ Model ]
 Firmware..... PM3 GENERIC

[ ARM ]
 Bootrom.... Iceman/master/v4.20728-309-g4d998e645-dirty-suspect 2026-01-19 23:24:02 968563447
 OS..... Iceman/master/v4.20728-309-g4d998e645-dirty-suspect 2026-01-19 23:24:35 968563447
 Compiler... GCC 13.3.0

[ FPGA ]
fpga_pm3_hf.ncd image 2s30vq100 19-01-2026 23:09:27
fpga_pm3_lf.ncd image 2s30vq100 19-01-2026 23:09:27
fpga_pm3_felica.ncd image 2s30vq100 19-01-2026 23:09:27
fpga_pm3_hf_15.ncd image 2s30vq100 19-01-2026 23:09:27
```

```
[ Hardware ]
 --= uC: AT91SAM7S512 Rev A
 --= Embedded Processor: ARM7TDMI
 --= Internal SRAM size: 64K bytes
 --= Architecture identifier: AT91SAM7Sxx Series
 --= Embedded flash memory 512K bytes ( 71% used )

[!] ARM firmware does not match the source at the time the client was compiled
[!] Make sure to flash a correct and up-to-date version
OBS: Após esses alertas foi realizada a ação de limpeza e nova validação
```

The next step is identifying the chip using **hf search.**

```
$ hf search
```

This command allows investigators to:

- Identify chip type (e.g., NTAG series)
- Retrieve the UID (unique identifier)
- Detect the NFC protocol used

```
[usb] pm3 --> hf search
[/] Searching for ISO14443-A tag...
[=] ---------- ISO14443-A Information ----------
[+]  UID: 04 D3 7E 32 0A 54 80   ( double )
[+] ATQA: 00 44
[+]  SAK: 00 [2]
[+]      NXP Semiconductors Germany
[+] Possible types:
[+]    NTAG 2xx
[=]
[=] Proprietary non iso14443-4 card found
[=] RATS not supported
[?] Hint: Try `hf mfu info`

[+] Valid ISO 14443-A tag found
```

The acquisition of digital evidence is performed using:

```
$ hf mfu dump
```

This command enables:

- Full memory extraction
- Binary/hexadecimal data generation
- Evidence preservation for offline analysis

During analysis, the extracted data revealed:

- Stored URLs
- Android automation triggers
- Embedded user interaction messages
- Timestamp records
- Geolocation coordinates

These findings demonstrate that biochips can act as compact data storage and automation devices.

```
[usb] pm3 --> hf mfu dump


[+] TYPE: NTAG 216 888bytes (NT2H1611G0DU)
[+] Reading tag memory...

[=] MFU dump file information
[=] --------------------------------------------------------------
[=] Version..... 00 04 04 02 01 00 13 03
[=] TBD 0..... 00 00
[=] TBD 1..... 00
[=] Signature... 22 77 6A 91 3C 87 B2 FA 09 52 3B 6D C5 E5 84 AE
[=]              63 C6 4E D4 E6 8C E3 BB C8 45 22 1C 8A F9 86 F8
[=] Counter 0... 00 00 00
[=] Tearing 0... 00
[=] Counter 1... 00 00 00
[=] Tearing 1... 00
[=] Counter 2... 00 00 00
[=] Tearing 2... BD
[=] Max data page... 230 ( 924 bytes )
[=] Header size..... 56 bytes

[=] --------------------------------------------------------------
[=] block#  | data       |lck| ascii
[=] ---------+-------------+---+------
[=]  0/0x00 | 04 D3 7E 21 |   | ..~!
[=]  1/0x01 | 32 0A 54 80 |   | 2.T.
[=]  2/0x02 | EC 48 00 00 |   | .H..
[=]  3/0x03 | E1 10 6D 00 | 0 | ..m.
[=]  4/0x04 | 03 0F D1 01 | 0 | ....
[=]  5/0x05 | 0B 55 04 67 | 0 | .U.g
[=]  6/0x06 | 6F 6F 67 6C | 0 | oogl
[=]  7/0x07 | 65 2E 63 6F | 0 | e.co
[=]  8/0x08 | 6D FE 00 00 | 0 | m...
[=]  9/0x09 | 61 6E 64 72 | 0 | andr
[=] 10/0x0A | 6F 69 64 2E | 0 | oid.
[=] 11/0x0B | 61 70 70 73 | 0 | apps
[=] 12/0x0C | 2E 77 61 6C | 0 | .wal
[=] 13/0x0D | 6C 65 74 6E | 0 | letn
[=] 14/0x0E | 66 63 72 65 | 0 | fcre
[=] 15/0x0F | 6C 54 0F 13 | 0 | lT..
[=] 16/0x10 | 61 6E 64 72 | ? | andr
[=] 17/0x11 | 6F 69 64 2E | ? | oid.
[=] 18/0x12 | 63 6F 6D 3A | ? | com:
[=] 19/0x13 | 70 6B 67 63 | ? | pkgc
[=] 20/0x14 | 6F 6D 2E 77 | ? | om.w
[=] 21/0x15 | 61 6B 64 65 | ? | akde
[=] 22/0x16 | 76 2E 6E 66 | ? | v.nf
[=] 23/0x17 | 63 74 61 73 | ? | ctas
[=] 24/0x18 | 6B 73 FE 00 | ? | ks..
[=] 25/0x19 | 32 20 32 30 | ? | 2 20
[=] 26/0x1A | 3A 35 35 7C | ? | :55|
[=] 27/0x1B | 32 30 32 36 | ? | 2026
[=] 28/0x1C | 2D 30 34 2D | ? | -04-
[=] 29/0x1D | 30 32 20 32 | ? | 02 2
[=] 30/0x1E | 31 3A 35 35 | ? | 1:55
[=] 31/0x1F | 5D 5B 30 5D | ? | ][0]
[=] 32/0x20 | 12 04 76 77 | ? | ..vw
[=] 33/0x21 | 38 2F 36 5B | ? | 8/6[
[=] 34/0x22 | C3 89 20 20 | ? | ..
```

< If you pay close attention, you will find a URL embedded in this segment: >:

google.com

```
[=] 35/0x23 | 54 45 4D 50 | ? | TEMP
[=] 36/0x24 | 4F 52 C3 81 | ? | OR..
[=] 37/0x25 | 52 49 4F 2E | ? | RIO.
[=] 38/0x26 | 2E 2E 5D 41 | ? | ..]A
[=] 39/0x27 | 63 61 6C 6D | ? | calm
[=] 40/0x28 | 65 2D 73 65 | ? | e-se
[=] 41/0x29 | 2E 2E 2E 0A | ? | ....
[=] 42/0x2A | 0A 52 65 73 | ? | .Res
[=] 43/0x2B | 70 69 72 65 | ? | pire
[=] 44/0x2C | 20 46 75 6E | ? |  Fun
[=] 45/0x2D | 64 6F 0A 45 | ? | do.E
[=] 46/0x2E | 73 63 75 74 | ? | scut
[=] 47/0x2F | 65 20 75 6D | ? | e um
[=] 48/0x30 | 61 20 62 6F | ? | a bo
[=] 49/0x31 | 61 20 6D C3 | ? | a m.
[=] 50/0x32 | BA 73 69 63 | ? | .sic
[=] 51/0x33 | 61 20 0A 49 | ? | a .I
[=] 52/0x34 | 73 73 6F 20 | ? | sso
[=] 53/0x35 | C3 A9 20 20 | ? | ..
[=] 54/0x36 | 72 65 61 6C | ? | real
[=] 55/0x37 | 20 6F 75 20 | ? |  ou
[=] 56/0x38 | 66 72 75 74 | ? | frut
[=] 57/0x39 | 6F 20 64 61 | ? | o da
[=] 58/0x3A | 20 73 75 61 | ? |  sua
[=] 59/0x3B | 20 69 6D 61 | ? |  ima
[=] 60/0x3C | 67 69 6E 61 | ? | gina
[=] 61/0x3D | C3 A7 C3 A3 | ? | ....
[=] 62/0x3E | 6F 3F 0A 0A | ? | o?..
[=] 63/0x3F | 0A 54 0F 13 | ? | .T..
[=] 64/0x40 | 61 6E 64 72 | ? | andr
[=] 65/0x41 | 6F 69 64 2E | ? | oid.
[=] 66/0x42 | 63 6F 6D 3A | ? | com:
[=] 67/0x43 | 70 6B 67 63 | ? | pkgc
[=] 68/0x44 | 6F 6D 2E 77 | ? | om.w
[=] 69/0x45 | 61 6B 64 65 | ? | akde
[=] 70/0x46 | 76 2E 6E 66 | ? | v.nf
[=] 71/0x47 | 63 74 61 73 | ? | ctas
[=] 72/0x48 | 6B 73 FE 00 | ? | ks..
[=] 73/0x49 | 65 20 61 74 | ? | e at
[=] 74/0x4A | 65 6E C3 A7 | ? | en..
[=] 75/0x4B | C3 A3 6F 20 | ? | ..o
[=] 76/0x4C | 6E 61 20 6D | ? | na m
[=] 77/0x4D | C3 BA 73 69 | ? | ..si
[=] 78/0x4E | 63 61 20 0A | ? | ca .
[=] 79/0x4F | 0A F0 9F 86 | ? | ....
[=] 80/0x50 | 98 EF B8 8F | ? | ....
[=] 81/0x51 | 20 45 6E 76 | ? |  Env
[=] 82/0x52 | 69 65 20 75 | ? | ie u
[=] 83/0x53 | 6D 61 20 6D | ? | ma m
[=] 84/0x54 | 65 6E 73 61 | ? | ensa
[=] 85/0x55 | 67 65 6D 20 | ? | gem
[=] 86/0x56 | 70 65 64 69 | ? | pedi
[=] 87/0x57 | 6E 64 6F 20 | ? | ndo
[=] 88/0x58 | 61 6A 75 64 | ? | ajud
[=] 89/0x59 | 61 0A 12 05 | ? | a...
[=] 90/0x5A | 4B 77 38 2F | ? | Kw8/
[=] 91/0x5B | 31 36 5B 52 | ? | 16[R
[=] 92/0x5C | 65 67 69 73 | ? | egis
[=] 93/0x5D | 74 72 6F 20 | ? | tro
[=] 94/0x5E | 64 65 20 43 | ? | de C
[=] 95/0x5F | 72 69 73 65 | ? | rise
[=] 96/0x60 | 20 64 65 20 | ? |  de
[=] 97/0x61 | 50 61 6E 69 | ? | Pani
[=] 98/0x62 | 63 6F 7C 7B | ? | co|{
[=] 99/0x63 | 4C 41 54 7D | ? | LAT}
[=] 100/0x64 | 7C 23 5D 5B | ? | |#][
[=] 101/0x65 | 32 30 32 36 | ? | 2026
[=] 102/0x66 | 2D 30 33 2D | ? | -03-
[=] 103/0x67 | 33 30 20 32 | ? | 30 2
[=] 104/0x68 | 33 3A 33 34 | ? | 3:34
[=] 105/0x69 | 7C 32 30 32 | ? | |202
```

< This section contains a message that is displayed upon the execution of an Android task trigger:

(Text in portuguese)
“TEMPORÁRIO:
Acalme-se.
Respire fundo.
Escute uma boa musica.
Isso é real ou fruto da sua imaginação?”

< This section contains a android task:

(Text in portuguese)
“Envie uma mensagem pedindo ajuda”

< This section contains an Android task that records an entry in the calendar and also displays the latitude.

(Text in portuguese)
“Registro de crise de pânico”

```
[=] 106/0x6A | 36 2D 30 34 | ? | 6-04
[=] 107/0x6B | 2D 30 31 20 | ? | -01
[=] 108/0x6C | 30 30 3A 33 | ? | 00:3
[=] 109/0x6D | 34 5D 5B 30 | ? | 4][0
[=] 110/0x6E | 5D 12 05 16 | ? | ]...
[=] 111/0x6F | 77 38 2F 31 | ? | w8/1
[=] 112/0x70 | 33 67 65 6F | ? | 3geo
[=] 113/0x71 | 3A 34 37 2E | ? | :47.
[=] 114/0x72 | 33 32 31 34 | ? | 3214
[=] 115/0x73 | 37 32 2C 35 | ? | 72,5
[=] 116/0x74 | 2E 30 34 31 | ? | .041
[=] 117/0x75 | 33 38 32 12 | ? | 382.
[=] 118/0x76 | 04 30 77 38 | ? | .0w8
[=] 119/0x77 | 2F 32 68 74 | ? | /2ht
[=] 120/0x78 | 74 70 73 3A | ? | tps:
[=] 121/0x79 | 2F 2F 79 6F | ? | //yo
[=] 122/0x7A | 75 74 75 2E | ? | utu.
[=] 123/0x7B | 62 65 2F 5A | ? | be/Z
[=] 124/0x7C | 6E 57 6F 46 | ? | nWoF
[=] 125/0x7D | 47 6B 73 57 | ? | GksW
[=] 126/0x7E | 74 41 3F 73 | ? | tA?s
[=] 127/0x7F | 69 3D 62 76 | ? | i=bv
[=] 128/0x80 | 63 70 54 47 | ? | cpTG
[=] 129/0x81 | 45 4F 30 7A | ? | EO0z
[=] 130/0x82 | 4D 56 50 62 | ? | MVPb
[=] 131/0x83 | 54 46 54 0F | ? | TFT.
[=] 132/0x84 | 13 61 6E 64 | ? | .and
[=] 133/0x85 | 72 6F 69 64 | ? | roid
[=] 134/0x86 | 2E 63 6F 6D | ? | .com
[=] 135/0x87 | 3A 70 6B 67 | ? | :pkg
[=] 136/0x88 | 63 6F 6D 2E | ? | com.
[=] 137/0x89 | 77 61 6B 64 | ? | wakd
[=] 138/0x8A | 65 76 2E 6E | ? | ev.n
[=] 139/0x8B | 66 63 74 61 | ? | fcta
[=] 140/0x8C | 73 6B 73 FE | ? | sks.
[=] 141/0x8D | 00 00 00 00 | ? | ....
[=] 142/0x8E | 00 00 00 00 | ? | ....
[=] 143/0x8F | 00 00 00 00 | ? | ....
[=] 144/0x90 | 00 00 00 00 | ? | ....
[=] 145/0x91 | 00 00 00 00 | ? | ....
[=] 146/0x92 | 00 00 00 00 | ? | ....
[=] 147/0x93 | 00 00 00 00 | ? | ....
[=] 148/0x94 | 00 00 00 00 | ? | ....
[=] 149/0x95 | 00 00 00 00 | ? | ....
[=] 150/0x96 | 00 00 00 00 | ? | ....
[=] 151/0x97 | 00 00 00 00 | ? | ....
[=] 152/0x98 | 00 00 00 00 | ? | ....
[=] 153/0x99 | 00 00 00 00 | ? | ....
[=] 154/0x9A | 00 00 00 00 | ? | ....
[=] 155/0x9B | 00 00 00 00 | ? | ....
[=] 156/0x9C | 00 00 00 00 | ? | ....
[=] 157/0x9D | 00 00 00 00 | ? | ....
[=] 158/0x9E | 00 00 00 00 | ? | ....
[=] 159/0x9F | 00 00 00 00 | ? | ....
[=] 160/0xA0 | 00 00 00 00 | ? | ....
[=] 161/0xA1 | 00 00 00 00 | ? | ....
[=] 162/0xA2 | 00 00 00 00 | ? | ....
[=] 163/0xA3 | 00 00 00 00 | ? | ....
[=] 164/0xA4 | 00 00 00 00 | ? | ....
[=] 165/0xA5 | 00 00 00 00 | ? | ....
[=] 166/0xA6 | 00 00 00 00 | ? | ....
[=] 167/0xA7 | 00 00 00 00 | ? | ....
[=] 168/0xA8 | 00 00 00 00 | ? | ....
[=] 169/0xA9 | 00 00 00 00 | ? | ....
[=] 170/0xAA | 00 00 00 00 | ? | ....
[=] 171/0xAB | 00 00 00 00 | ? | ....
[=] 172/0xAC | 00 00 00 00 | ? | ....
[=] 173/0xAD | 00 00 00 00 | ? | ....
[=] 174/0xAE | 00 00 00 00 | ? | ....
[=] 175/0xAF | 00 00 00 00 | ? | ....
[=] 176/0xB0 | 00 00 00 00 | ? | ....
```

< This section contains a youtube url

https://youtu.be/ZnWoFGksWtA?si=bvcpTGEO0zMVPb

```
[=] 177/0xB1 | 00 00 00 00 | ? | ....
[=] 178/0xB2 | 00 00 00 00 | ? | ....
[=] 179/0xB3 | 00 00 00 00 | ? | ....
[=] 180/0xB4 | 00 00 00 00 | ? | ....
[=] 181/0xB5 | 00 00 00 00 | ? | ....
[=] 182/0xB6 | 00 00 00 00 | ? | ....
[=] 183/0xB7 | 00 00 00 00 | ? | ....
[=] 184/0xB8 | 00 00 00 00 | ? | ....
[=] 185/0xB9 | 00 00 00 00 | ? | ....
[=] 186/0xBA | 00 00 00 00 | ? | ....
[=] 187/0xBB | 00 00 00 00 | ? | ....
[=] 188/0xBC | 00 00 00 00 | ? | ....
[=] 189/0xBD | 00 00 00 00 | ? | ....
[=] 190/0xBE | 00 00 00 00 | ? | ....
[=] 191/0xBF | 00 00 00 00 | ? | ....
[=] 192/0xC0 | 00 00 00 00 | ? | ....
[=] 193/0xC1 | 00 00 00 00 | ? | ....
[=] 194/0xC2 | 00 00 00 00 | ? | ....
[=] 195/0xC3 | 00 00 00 00 | ? | ....
[=] 196/0xC4 | 00 00 00 00 | ? | ....
[=] 197/0xC5 | 00 00 00 00 | ? | ....
[=] 198/0xC6 | 00 00 00 00 | ? | ....
[=] 199/0xC7 | 00 00 00 00 | ? | ....
[=] 200/0xC8 | 00 00 00 00 | ? | ....
[=] 201/0xC9 | 00 00 00 00 | ? | ....
[=] 202/0xCA | 00 00 00 00 | ? | ....
[=] 203/0xCB | 00 00 00 00 | ? | ....
[=] 204/0xCC | 00 00 00 00 | ? | ....
[=] 205/0xCD | 00 00 00 00 | ? | ....
[=] 206/0xCE | 00 00 00 00 | ? | ....
[=] 207/0xCF | 00 00 00 00 | ? | ....
[=] 208/0xD0 | 00 00 00 00 | ? | ....
[=] 209/0xD1 | 00 00 00 00 | ? | ....
[=] 210/0xD2 | 00 00 00 00 | ? | ....
[=] 211/0xD3 | 00 00 00 00 | ? | ....
[=] 212/0xD4 | 00 00 00 00 | ? | ....
[=] 213/0xD5 | 00 00 00 00 | ? | ....
[=] 214/0xD6 | 00 00 00 00 | ? | ....
[=] 215/0xD7 | 00 00 00 00 | ? | ....
[=] 216/0xD8 | 00 00 00 00 | ? | ....
[=] 217/0xD9 | 00 00 00 00 | ? | ....
[=] 218/0xDA | 00 00 00 00 | ? | ....
[=] 219/0xDB | 00 00 00 00 | ? | ....
[=] 220/0xDC | 00 00 00 00 | ? | ....
[=] 221/0xDD | 00 00 00 00 | ? | ....
[=] 222/0xDE | 00 00 00 00 | ? | ....
[=] 223/0xDF | 00 00 00 00 | ? | ....
[=] 224/0xE0 | 00 00 00 00 | ? | ....
[=] 225/0xE1 | 00 00 00 00 | ? | ....
[=] 226/0xE2 | 00 00 00 BD | ? | ....
[=] 227/0xE3 | 04 00 00 FF | ? | ....
[=] 228/0xE4 | 00 05 00 00 | ? | ....
[=] 229/0xE5 | 00 00 00 00 | ? | ....
[=] 230/0xE6 | 00 00 00 00 | ? | ....
[=] ---------------------------------
[=] Using UID as filename
[+] Saved 980 bytes to binary file ` C:\ProxSpace\pm3/hf-mfu-04D37E320A5480-dump-004.bin`
 [+] Saved to json file C:\ProxSpace\pm3/hf-mfu-04D37E320A5480-dump-004.json
```

Data Analysis

The extracted data can be viewed using simple tools such as text editors. However, forensic analysis relies heavily on **hexadecimal representation**, which allows precise byte-level interpretation.

After the data collection phase, the analysis stage begins, where the extracted data is interpreted.

Interesting Scenarios for Analysis

As observed, biochips based on NFC technologies can store different types of digital information, many of which become accessible through reading tools such as Proxmark3. Depending on how the data is structured (especially in NDEF format), part of this information can be interpreted directly, without the need for additional authentication.
This makes it possible to consider scenarios in which a biochip is used not only for legitimate purposes, such as identification or data sharing, but also as a discreet medium for storing and transmitting information.
We can imagine data concealment scenarios involving:

- URLs that redirect to external systems controlled by the user
- Shortened or obfuscated links ("ghost links") that mask the real destination
- Unique identifiers associated with a digital identity
- References to cryptocurrency wallets or public keys
- Records indicating the intended use of the device in specific contexts (access, authentication, automation)

From a more active perspective, we must also consider malicious use scenarios involving seemingly legitimate URLs that may redirect to:

- Fake pages
- Credential harvesting systems
- Malicious file downloads

From an information security standpoint, it is also important to highlight scenarios involving **covert data exfiltration**.

Visualization of Active NDEF Record

After exploring and discussing possible scenarios, it is time to present what was actually obtained from my biochip.
For this purpose, I will include the view from a commercial tool where I recorded a simple Google URL.
This was the most recent write operation, where we can observe the presence of a single record, with a total size of 15 bytes out of the 868 bytes available.

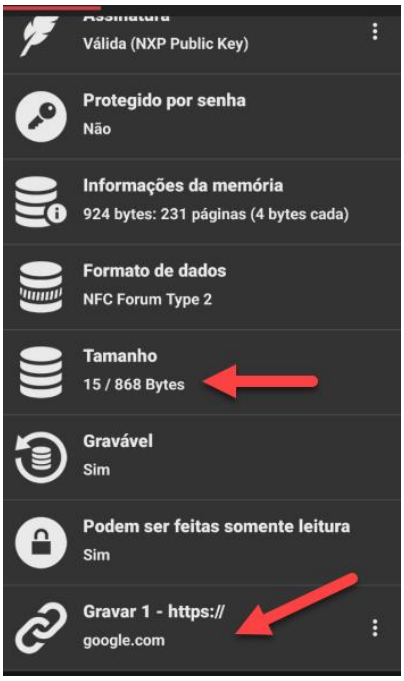


Figure 5– Image of the last recorded data, which opens the Google homepage

Next, I used the Proxmark3 to read the stored information using the previously mentioned commands, particularly **$ hf search**, which returned details about the tag type and UID.

I then proceeded to verify the recorded content, and it was at this point that I identified residual data from previous writes. Bingo!

In a forensic context, the objective is precisely to extract information that is not always directly visible. This can be achieved through a full memory copy, a complete dump, which includes not only active data but also remnants of previously stored information.

When researching data acquisition from NFC and RFID tags, it is common to find statements suggesting that each new write operation logically overwrites the previous one. However, as observed in this case, this overwrite does not always imply complete physical memory cleaning, allowing for the recovery of residual data.

As observed, the commercial tool writes a new NDEF record, updates the record size, but does not perform a full memory wipe, nor clear previously used blocks.

Thus, we can conclude that older data may fall outside the logical boundaries of the NDEF structure, yet still remain physically present in memory when using the $ hf mfu dump command.

From a forensic standpoint, this behavior is well known and is characterized as **data remanence**. In the context of the analyzed biochip, it was observed that NTAG-based devices do not implement garbage collection or physical memory cleaning mechanisms. As a result, the writing process occurs logically through append/overwrite operations, without effectively removing previously stored data.

As an example, I present screenshots of the recorded actions as an illustration of the findings.

The first screenshot refers to a recording of a logged event in a calendar. The second screenshot shows a popup also coming from a recording that was found on the biochip, related to an android task.

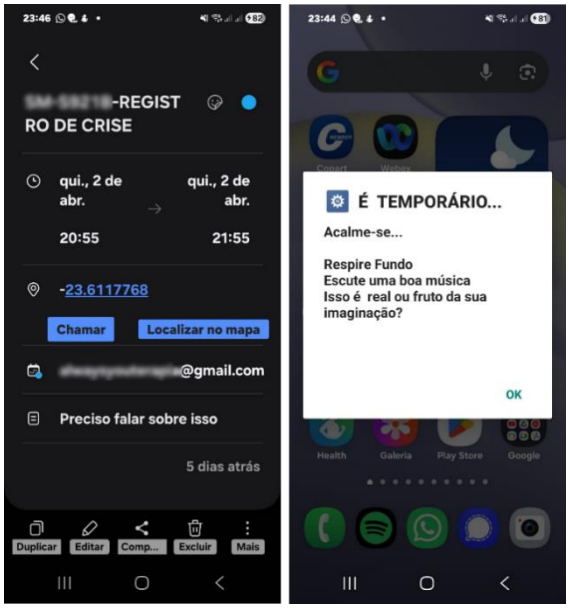


Figure 6-illustration of the findings

Conclusion

In conclusion, NFC biochips are no longer merely convenience devices; they now represent potential digital artifacts of significant relevance in forensic analysis. They may contain not only explicit information but also indirect indicators of usage, behavior, and interaction with other systems.

The proper interpretation of this data requires not only appropriate tools but also a critical approach to the context in which the device is embedded.

Depending on the context, such devices are not limited to legitimate applications, such as personal identification or data sharing, but may also be used as means for storing, transporting, and discreetly transmitting digital information.

About the Author:

Viviane Cruz https://br.linkedin.com/in/viviane-ramos-da-cruz Account Manager in Cybersecurity, graduated in Computer Networks and with a postgraduate degree in Project Management and Forensics. She has over 15 years of experience in various sectors and technology-oriented projects. In this field, she has been able to expand her horizons, provide solutions, and promote constant improvements in processes and systems.
Passionate about technology, investigations, and games, she constantly seeks to overcome limitations by studying and promoting interactions between different areas of knowledge.
Published Articles:

- Data Carving Xbox One com Belkasoft - Academia de Forense Digital
- Saiba tudo sobre Evidence Center-X da Belkasoft - Academia de Forense Digital
- Saiba tudo sobre Cellebrite UFED Touch - Academia de Forense Digital

Co-author of the book *Digital Forensics* (2025), published by Editora Foco, with a focus on Computer Forensics: https://www.editorafoco.com.br/produto/pericias-digitais-1-ed-2025

In this article, I explore data carving techniques applied to an NFC biochip implanted in my own body, using the Proxmark3 as the primary acquisition and analysis tool.

This work builds upon my previous research in digital forensics, where I investigated data carving techniques in an Xbox One gaming system. The original study was published by Academia de Forense Digital (AFD) and was also featured in a previous edition of eForensics Magazine. While the earlier research focused on traditional storage media, this work extends those concepts to a non-conventional scenario involving embedded NFC technology.

Through practical experimentation, this analysis uncovers valuable insights into data acquisition, memory dumping, and the recovery of residual data from NTAG-based biochips. It also highlights the forensic relevance of such devices, demonstrating how implanted or wearable technologies can be treated as potential sources of digital evidence.

Upcoming Features: The Future of Digital Forensics

As the digital landscape evolves, so do the complexities of investigation. We are excited to share a series of eForensics issues covering the latest in cyber intelligence. In our upcoming editions, we will be tackling:

- Investigating Cloud and Hybrid-Cloud Environments
- Analyzing Encrypted Data and Volatile Memory (RAM)
- Collecting Digital Evidence from Endpoints and the Internet of Things
- Forensic Tracking of Cryptocurrency and Blockchain Assets

Editorial Board

PUBLISHER HAKIN9 MEDIA SP. Z O.O. 00-585 WARSZAWA UL. BAGATELA 10/30 HAKIN9.ORG