

Ransomware Detection Tool Using Canary Files

Author: Nabila **Programme:** B.Com/BBA/BCA — Final Year Project, AI Foundation Programme **Institution:** SJ Global PU & Degree College, in collaboration with Aider Digital Pvt Ltd

Abstract

Ransomware has emerged as a significant cybersecurity threat because of its ability to encrypt or otherwise restrict access to a user's files, causing substantial data loss. Research on ransomware detection emphasizes identifying malicious activity at an early stage, ideally before extensive file damage occurs. Traditional detection approaches commonly rely on signatures, behavioral analysis, or machine-learning techniques; however, deception-based approaches using canary files (also called honeyfiles) provide an alternative mechanism for detecting suspicious file activity without requiring machine-learning-based ransomware classification as the primary defense.

This paper proposes a Ransomware Detection Tool Using Canary Files, designed as a lightweight early-warning system. The proposed system places specially prepared canary files within monitored locations and observes interactions with them. Unauthorized access, modification, renaming, or deletion of a canary file is treated as a strong indicator of ransomware activity, since legitimate users and applications have no reason to touch a file that exists only as a trap. The system extends the basic canary-file approach with two additional layers — AI-Based Risk Scoring and Real-Time Email Alerts — combining deception-based detection, event logging, risk assessment, and automated notification into a single, practical early-detection pipeline. **The system is scoped deliberately as a detection and alerting tool, not a defensive one** — it does not take any automated action on the monitored system itself; acting on an alert (isolating a machine, killing a process) is left to the human who receives it, since building a safe, reversible, OS-level automated response is a substantially harder engineering problem than detection and out of scope for this project.

Table of Contents

1. Introduction
2. Related Work — Ransomware: Background, Evolution, and Detection Techniques
3. Proposed System: Design and Methodology
4. System Workflow

- 5. Discussion — Expected Outcomes and Limitations
- 6. Conclusion and Future Work

References

1. Introduction

Ransomware has become one of the most serious cybersecurity threats, targeting individuals, organizations, and institutions by disrupting access to important files and data. A single ransomware attack can rapidly encrypt a large number of files, which makes **early** detection — catching the attack in its first seconds, not after it has already spread — the single most important factor in limiting damage. As ransomware techniques continue to evolve, security systems increasingly focus on identifying suspicious *behavior* rather than relying only on known malware signatures, which are trivial for a new or modified variant to evade.

1.1 Motivation — Why This Approach

Existing detection methods each carry a specific weakness that motivates this project:

- **Signature-based detection** only catches ransomware that has already been seen and catalogued elsewhere — it is blind to new or modified variants by definition.
- **Behavior-based and machine-learning detection** require either large labelled training datasets or careful heuristic tuning, and both approaches are computationally heavier than a small project can realistically deploy, and both can be fooled by ransomware that behaves innocuously at first.
- **Canary files (honeypots)** sidestep both problems. A canary file is a decoy that legitimate software and legitimate users have no reason to ever open. This means detection does not depend on recognizing *what ransomware looks like* — it depends only on recognizing that *a file which should never be touched, was touched*. That is a much simpler and much harder-to-evade signal than a signature or a learned pattern.

The objective of this project is therefore not to build a more accurate malware classifier, but to build a **tripwire**: a lightweight system that turns an inevitable ransomware action (touching files it doesn't know are fake) into an immediate, low-false-positive alert — and then to layer risk scoring and real-time notification on top of that tripwire so the detection is actually actionable, not just a log entry nobody sees until it's too late. The project stops deliberately at notification: it detects and alerts, it does not attempt to automatically intervene on the monitored system.

2. Related Work — Ransomware: Background, Evolution, and Detection Techniques

2.1 Early Ransomware — 1989

The history of ransomware traces back to 1989, with the AIDS Trojan (also known as PC Cyborg), distributed via floppy disks. It was one of the earliest examples of malware designed to restrict access to a user's data and demand payment for its release.

2.2 Development of Modern Ransomware

As internet connectivity grew, ransomware became more sophisticated. Attackers moved from weak, reversible encryption to strong cryptography and internet-based distribution, making unassisted recovery effectively impossible without the attacker's key [1].

2.3 CryptoLocker — 2013

CryptoLocker (2013) was a turning point: it used strong encryption to make victims' files genuinely inaccessible and is widely credited with popularizing the modern crypto-ransomware model that most current attacks still follow [1].

2.4 WannaCry — 2017

WannaCry (2017) demonstrated how ransomware could self-propagate between vulnerable systems at scale, without needing an individual victim to be tricked into running it. Its global impact underlined the importance of patching, network segmentation, and — critically for this project — detection mechanisms that work *before* mass encryption occurs, not after [1].

2.5 Modern Ransomware and the Case for Early Detection

Modern ransomware increasingly combines file encryption with data theft and extortion ("double extortion"), and attacks can cause severe damage within minutes of execution [3]. This trend is the core justification for this project's design priority: a detection system is only useful if it fires **before** the bulk of the encryption has happened, which rules out approaches that need to observe a large volume of suspicious activity before triggering.

2.6 Existing Detection Techniques

Technique	How it works	Key limitation
Signature-based detection	Compares files/programs against known malware signatures	Blind to new or modified variants that don't match an existing signature [1]
Behavior-based detection	Flags actions (rapid file modification, mass file operations) rather than known code	Can require a meaningful volume of malicious activity before it's confident enough to flag it [1]
File-system activity monitoring	Watches for abnormal volumes of file changes across the system	Broad monitoring scope can be noisy and resource-intensive [1]
Cryptographic/encryption-detection methods	Looks for statistical signs of encryption in file changes (e.g., entropy shifts)	Effective but reactive — by the time encryption is detectable this way, some files are already gone [4]
Canary files / honeyfiles	Decoy files that should never be touched; any interaction is a near-certain positive signal	Only useful if the decoys are well-placed and genuinely indistinguishable from real files — placement strategy matters a great deal

Prior work has specifically validated the honeyfile/honeypot approach for ransomware: Moore (2016) implemented and tested honeypot folders using Windows File Server Resource Manager and security-log monitoring, confirming that file-screening plus change monitoring on decoy locations is a practical, low-overhead way to catch ransomware activity early [2]. Cen et al.'s 2024 survey of early-detection research places deception-based methods alongside behavioral and ML-based approaches as one of the three main early-detection families, noting their comparatively low false-positive rate as their main advantage [3].

This project builds directly on that validated foundation, and adds two layers most academic honeyfile prototypes stop short of: automated severity scoring and real-time notification — closing the gap between “detected” and “a human knows about it immediately.”

3. Proposed System: Design and Methodology

The system is composed of four cooperating components:

3.1 Canary File Deployment

Specially prepared decoy files are placed in monitored locations chosen to sit directly in a ransomware's typical scan path (common document folders, Desktop, frequently-backed-up directories) while remaining outside the paths a normal user or legitimate application would ever open. Filenames are chosen to look valuable (e.g., resembling financial records or credentials) specifically because ransomware does not selectively skip files — it encrypts everything indiscriminately, which is what makes this approach reliable rather than probabilistic.

3.2 Event Capture and Logging

The system monitors the canary files for access, modification, renaming, and deletion events. Every detected interaction is logged with a timestamp, the affected file, and the type of activity, creating an auditable record independent of whatever the risk-scoring step decides.

3.3 AI-Based Risk Scoring

Rather than using AI/ML as the *primary* detector (which is where most academic prototypes place computational cost and complexity), this system uses it as a **secondary triage layer**: once a canary-file event has already occurred, the scoring component evaluates contextual signals (e.g., how many canary files were touched, how quickly, what kind of operation) to assign a severity score. This keeps the expensive part of the pipeline — the part that actually needs to be smart — working on a small number of already-suspicious events, not on the entire file system in real time.

3.4 Real-Time Email Alerts

When a risk score crosses a configurable threshold, the system sends an immediate email notification containing the detected event, timestamp, affected file, and calculated risk level, so a human can act without needing to be watching a dashboard. **This is the final step in the pipeline.** The system's responsibility ends at getting the right information to a human fast — it does not act on the monitored system itself. Automated defensive response (isolating a machine, killing a process, or similar) was considered during design and deliberately excluded: it requires OS-level intervention that carries real risk if it misfires, and is a substantially harder and riskier engineering problem than detection and notification, out of scope for a project of this size (see Section 5.2, Future Work).

4. System Workflow

1. Canary files are deployed across monitored locations.

2. A file-system monitor watches those specific files for any interaction.
3. On interaction, the event is logged (time, file, action type).
4. The event is passed to the Risk Scoring component, which assigns a severity score.
5. If the score exceeds the alert threshold → a real-time email is sent, giving a human the information needed to respond.
6. All events remain in the log for later review, regardless of whether they triggered an alert.

(A full component diagram and technical architecture is provided in the companion document, [Architecture_Paper.pdf](#).)

5. Discussion — Expected Outcomes and Limitations

5.1 Expected Outcomes

A working prototype should demonstrate: (a) reliable detection of simulated ransomware-style file operations (bulk rename/encrypt scripts against test canary files) within seconds of the first touch, (b) correctly scaled risk scores between an isolated single-file access and a rapid multi-file sweep, and (c) alert delivery without manual intervention.

5.2 Limitations and Open Risks — stated explicitly, not glossed over

- **False positives from legitimate software.** Backup tools, antivirus scanners, and search-indexing services sometimes enumerate or touch every file in a directory, including canary files. Placement locations and monitored operation types need to be chosen to minimize this, and it should be tested against at least one real backup tool before being called reliable.
- **Canary files only protect what they sit near.** This is a detection layer, not a prevention layer — ransomware that never touches a canary file's directory won't be caught by this mechanism alone. This should be stated as a scope boundary, not a hidden gap.
- **The “AI” in AI-Based Risk Scoring should be named precisely** in the final implementation write-up — whether it is a trained model, a weighted heuristic, or a rules engine changes what claims the paper can honestly make about it. Whichever it is, that should be stated plainly rather than left as “AI” for the reader to assume the stronger version.

5.3 Note on Terminology

This paper uses “Role-Task-Context-Format” (R-T-C-F) discipline in its own construction: every claim about existing techniques above is tied to a specific, verified source (see

References), and every claim about the proposed system is explicitly marked as *proposed* rather than *demonstrated*, since the tool has not yet been benchmarked. Distinguishing “what the literature shows” from “what this project proposes” is the same sourcing discipline taught in class, applied to research writing.

6. Conclusion and Future Work

This paper proposed a lightweight, deception-based ransomware detection tool built around canary files, extended with AI-assisted risk scoring and real-time alerting. The approach avoids the two biggest costs of mainstream detection methods — dependency on known signatures, and the training/tuning burden of full behavioral ML — while remaining grounded in a well-validated technique (honeyfiles) that prior research has already shown to work. The system is intentionally scoped as detection and notification only, not automated defensive response — a deliberate boundary, not an omission, since a safe, reversible, OS-level automated response is a substantially harder and riskier problem than detection, and out of scope for a project of this size.

Future work includes: building and benchmarking the prototype against real ransomware samples in an isolated test environment; measuring the false-positive rate against common backup and indexing software; tuning the risk-scoring thresholds empirically rather than by estimate; and, as a longer-term extension beyond this project’s scope, an automated protection response (e.g., process suspension or network isolation) with a clearly documented rollback path, only once the detection layer’s false-positive rate is known to be low enough to trust with an automated action.

References

- [1] P. O’Kane, S. Sezer, and D. Carlin, “Evolution of Ransomware,” *IET Networks*, vol. 7, no. 5, pp. 321–327, 2018. DOI: [10.1049/iet-net.2017.0207](https://doi.org/10.1049/iet-net.2017.0207)
- [2] C. Moore, “Detecting Ransomware with Honeypot Techniques,” *2016 Cybersecurity and Cyberforensics Conference (CCC)*, pp. 77–81, 2016. DOI: [10.1109/CCC.2016.14](https://doi.org/10.1109/CCC.2016.14)
- [3] M. Cen, F. Jiang, X. Qin, Q. Jiang, and R. Doss, “Ransomware Early Detection: A Survey,” *Computer Networks*, vol. 239, Art. no. 110138, 2024. DOI: [10.1016/j.comnet.2023.110138](https://doi.org/10.1016/j.comnet.2023.110138)
- [4] K. Begovic, A. Al-Ali, and Q. M. Malluhi, “Cryptographic Ransomware Encryption Detection: Survey,” *Computers & Security*, vol. 132, Art. no. 103349, 2023. DOI: [10.1016/j.cose.2023.103349](https://doi.org/10.1016/j.cose.2023.103349)

All four references above were independently verified against their publisher records (Wiley/IET, IEEE, Elsevier/ScienceDirect) before inclusion in this paper — no citation here is assumed or unverified.

Scope note (updated 27 Aug 2026): earlier drafts of this paper included a “Protection Mode” component that would take an automated defensive action (e.g., process suspension) above a second risk threshold. Nabila confirmed this was never part of what she intends to actually build — the project is detection and alerting only, since automated OS-level intervention is a substantially harder engineering problem than the rest of the system. Protection Mode has been removed from the proposed system throughout and repositioned as future work beyond this project’s scope, consistent with how every other unbuilt claim in this paper is already framed as proposed rather than demonstrated.