International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 04 | April – 2026 DOI:10.55041/ISJEM05985
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Ransomware Behaviour Detection and Response with Ml

Karishma. K
Dept. of cybersecurity
Dr. M.G.R. Educational and
Research Institute Chennai, India

Kiraan .K
Dept. of cybersecurity
Dr. M.G.R. Educational and
Research Institute Chennai, India

Fathima Nahal T.P
Dept. of cybersecurity
Dr. M.G.R. Educational and
Research Institute Chennai India

Mr. Sankara Narayanan S T
Dept. of cybersecurity
Dr. M.G.R. Educational and
Research Institute Chennai, India
sankaranarayanan.coedf@gmail.com

Dr. P. Dinesh Kumar
Dept. of cybersecurity
Dr. M.G.R. Educational and
Research Institute Chennai, India
dineshkumar.it@drmgrdu.ac.in

Kkarishmapawar2811@gmail.com kiraankannan3@gmail.com
Fathimanahal341@gmail.com

Abstract- It is evident that the ransomware remains one of those not-so-good types of malware that can lock up your files within a few seconds. The old signature tools are not capable of keeping up when newer variants emerge which are zero day. This article presents a lightweight and real-time Ransomware Behaviour Detection and Response framework developed in Python, based on three ML-style heuristics, Shannon entropy examination, fast modification rate, and suspicious file extensions. The Watchdog library monitors a folder that you have selected 24/7 and has a Tkinter GUI dashboard that provides you with a live log of activity and even allows you to download a report. Tests indicate it reaches 97 per cent detection and 2.1 per cent false positive which is significantly better than most conventional methods. It has a modular and scalable design and fits easily into enterprise endpoint protection pipelines**.**

Index Terms- Then, we have the topic of cybersec, endpoint protection, entropy analysis, file system observations, machine learning, malware elimination, Python, ransomware, real-time identification, Tkinter Watchdog.

I. INTRODUCTION

Basically, ransomware is this skill-inducing frightening computer software that simply encrypts all your files and demands you to pay a key. Ever since Crypto Locker struck in 2013 and WannaCry went viral in 2017, everyone (students, businesses, hospitals, governments) was hurting.

Most of the anti-malware software uses signature databases, which continuously keep on updating. When the malware is well known they're Marvelous, however when a brand new attack or a zero-day attack is presented, there is no signature present and you are merely left hanging.

Another trick that is employed instead of signature searching is the behaviour-based detection. It monitors the actual actions of a

program. Does it destroy files with madness? Has the effect of increasing file entropy? Does it change the names of files to strange names such as cryp.cyrp or lock? When it acts in a similar manner it is most likely ransomware- end of story.

Everything that we discuss in this paper is about a Python-built ransomware detection and response system, which is run in real-time. The three heuristics of our ML-inspired are the maximization of Shannon entropy, a fast file modification checking system, and suspicious extensions checks. Watchdog monitors the file system and a Tkinter interface informs you of what is going on.

The economic cost of ransomware has been increasing quite rapidly during recent years. Cybersecurity

headlines indicate that in 2021, the global losses reached the 20 billion USD and this is more than 57 times of the small expenses of 2015. Medical GPS Colonial Pipeline, JBS Foods, as well as KaseyaVSA almost all have demonstrated how ransomware could bring large-scale infrastructure destruction, decruitment of supply chains, and impact thousands of victims simultaneously. These events pointed to an essential vulnerability: even well-organized institutions with fully developed security organizations cannot prevent or promptly include ransomware with conventional walk members.

New families of ransomware have also become highly advanced and varied in their delivery manner. The first ones, such as Crypto Locker, had pretty basic key pairs of RSA-2048 and AES-256 with decryption keys being in attacker controlled servers.

The recent chapters of the ransomware story have introduced such additional elements as *double- and triple-extortion. They take your files first and lock them then demand triple-extortion with a threat of flooding your net in a DDoS attack otherwise. The barrier to entry is reduced by Ransomware-as-a-service (RaaS) which has resulted in such that even non-tech hackish people can purchase ready-to-go ransomware on cob in the black market. The resultant effect is that this creates a factory like pipeline wherein zero-day variants are found faster than the security vendors can fix it and so a purely reactive approach to defence can only be said to be more or less hopeless.

In response to this deluge, we should have a shift to behaviour-based and active endpoint security. Firewalls based on signature are based on a previous sample whereas the generic behaviours of ransomware are monitored using behaviour detection although the code may appear different. The paper I am developing is creating a minimal Python-based detection engine which can bind together three pieces of Python behaviour indicators or signals that are Entropy jumps, high rate of modifications, and suspicious file-extension modifications into one real-time workflow that can run without kernel modification or external feeds on a laptop device.

II. LITERATURE SURVEY

The old antiviruses were simply a habit tracker of familiar malware codes. Christodorescu et al. demonstrated that the ability to bypass those scanners with obfuscation raises researchers to the next level and encourages semantic and behaviours-based analysis.

Entropy has now become the application of choice to identify encrypted data blocks. It began with Shannon formula; AES-encrypted files were discovered to have on average 8.0 bits of entropy per byte by Kharraz and others; that is much more than what a typical document contains.

One of the early-warning tools used is a classic, CryptoDrop (Scaifeʻ et al.). It can detect file type changes, entropy peaks and similarity scores, as well as detecting all ransomware types it has tested and not generating big false positives.

Another red flag is an elevated modification rates. Continella et al. published ShieldFS at the time of writing to trape app writing patterns and mark abnormal bulk writes -encryption traffic that appears to look like nothing at all like normal disk usage.

A tinkering of files, file-extension, is a very inexpensive, simple hint. The majority of ransomware simply adds a suffix of either .crypt, .encrypted or .lock; the existence of such strings will allow you to identify the malware without having to perform a thorough crypto-analysis.

The classifiers have been trained on several features that have been dynamic and static by different groups. The logistic regression defined by the calls to Windows API implemented by Sgandurra et al achieved a rate of 96%+, but when Chen applied the random forests to filesystem log, it still offered high precision real-time.

Watchdog also provides a cross platform file observer; it is not noticed by either inotify or kqueue, and it is good at live monitoring. Tkinter would suit a small scale GUI dashboard well.

Nonetheless, users desire independent desktop programs that contain heuristics, user-friendly interface, and automatic reaction. That is what we will be making using entropy, modification rate and extension detection all in one deployable python module.

Ransom detection has also been covered by deep learning. CNNs that binarize images to grayscale images can reasonably end with an accuracy of 98%+ when in offline mode, but require GPUs, large label datasets, and continuous retraining- hence they are not able to support a laptop-level rule engine. Sequence based RNNs which model the sequence of API calls are also accurate but introduce latency, which kills real-time responsiveness.

There are also proper hybrid architectures which combine both a static and a dynamic analysis such as UNVEIL which pushes ransomware into a sandbox to

reveal its behaviour. That method is suitable to research work but not to production since it is based on controlled environments and it cannot be used to run continuous on a real user filesystem.

Our philosophy is to occupy that void: an incessant, live, user space monitoring which incurs little dependency and can be unceremoniously downloaded to an endpoint without pushing the operating system to its limits or requiring any special hardware.

III. PROPOSED METHODOLOGY

Our application is therefore simply a multi-layered user-space pipeline driver which runs as a user-space only application. It can be broken down into four key steps, namely picking a folder, configuring an observer, capturing real-time events, performing a multi-heuristic threat assessment, and then automatically responding to it, with the repelling of all such action being logged.

A user is first asked to choose a folder to open a file-dialog so as to lock down the spot. Watchman Observer then recursively walks through that folder and takes everything in it so that nothing is omitted.

Second, each event, create, modify, delete, move, entails a jump by the Ransomware Handler. It places an occurrence block in its path; it broadcasts the occurrence to the detection component but ignores the directory events since all other activities in the file are already under surveillance.

We have three parallel monorail heuristics. First, the extension checker scans a list of extensions that are scary such as crypted extensions such as .crypt, lock extensions such as .lock, encrypted extensions such as .encrypted, and other ransom extensions such as .ransom, etc. Second, the modification-rate monitor applies a 2 sec sliding window; anything changing 5 times in the sliding window is flagged by the monitor. Third, Shannon entropy analyser draws in the first 4,096 bytes, calculates the entropy, and when it moves more than 0.5 bits/byte above or below the previous value recorded it records a potential chiff.

Fourth, to automate everything we simply stamp a red flag on the event, pop up a warning to the user and display a fake ransom note informing the user on what was detected. Every event will receive a time stamp and plain text report.

A. Shannon Entropy Analysis

Written H (X) = -[?] p (x) log2 p (x), Shannon entropy calculates the randomness of a stream of bytes, or their information content. One hundred plain-text files, such as word documents, PDF files or large source codes, are typically between 3bits/byte and 6bits/byte owing to the non-uniform distribution of character patterns. By comparison, files encrypted using more modern ciphers such as AES -256 nudge nearly 8.0 bits per byte since the output appears as noise. Our system reads the initial 4,096 bytes of any new or modified file, calculates its entropy and when the difference between the previous value exceeds 0.5 bits per byte indicates that the file is likely being encrypted. It was set to this amount to ensure that there would be few false positives of legit compression at the expense of AES being detected as in the act of its course.

B. Rapid Change in the Rate of Modification Monitoring.

Tons of files get being encrypted as quickly as possible by ransomware in order to do as much damage as possible before being detected. Normal apps merely periodically write files think a doc editor, a compiler or an internet browser cache. Once ransomware runs full steam, it sprays numerous write events in hundreds of different paths in a matter of a couple of seconds. To track changes recent to the files, our monitor maintains a timestamped dictionary of the recent changes per file, a sliding window of two seconds, and considers a file that experiences over five distinct changes to the file as suspicious. This strikes ransomware which writes in blocky chunks rather than a single atomic write.

Checking suspicious file extensions involves detecting suspicious file extensions and blocking them from access.<|human|>C. Detection of Suspicious File Extension Detection Checks Suspicious file extensions are detected and prevented.

Many families of ransomware will also rename the encrypted output with a distinctive extension to ensure that the victim is aware that something has been happened. Our checker maintains a blacklist of extensions, such as .crypt, .lock, and .encrypted and .hacked and .ransom and enc and crypted and locky and cerber and zepto and wnry et al. On startup, the list is loaded using a config file and therefore new extensions could be added by the admins without any coding. Watchdog emits a create or rename event and we issue the path against the list when this occurs, a hit immediately raises a red flag of a threat irrespective of the entropy or the rate of modification: an extension

change is almost a certain signal that ransomware has completed encrypting a file.

IV. SYSTEM ARCHITECTURE

We divided the entire amount into five layers Presentation, Event Capture, Detection Engine, Response, and Reporting. The corresponding layers have defined job and communicate to one another through callbacks.

It is all Tkinter: you have a high title window in the top part, a side-bar where you are allowed to select folders and set up what is being monitored and in the main panel you see live totals and counter numbers of the suspicious files found and a scrollable pane along the bottom is the log window.

The Event Capture is run by Watchdog. Using one Observer thread, the folder tree is observed recursively without one causing any disturbance. The Ransomware Handler integrates into all events, sniffs filesystem events and passes them on to the Detection Engine.

Detection Engine two data structures, muscular over time, filemodcounts and fileentropyhistory, to record the latest file modification time and counter, as well as the latest entropy. With each and every pertinent event that occurs, they update, and therefore, the system profiles.

Response adds a suspicious counter in the GUI, adds one flagged line to the logs, displays a Tkinter warning box and presents a popup with a ransomware note to explain the threat.

Reporting allows you to save the complete log buffer that contains all the timestamps and also the detections down to a plain-text file; the report header counts the number of files scanned and the number of suspicious files detected.

The five-layer architecture is specifically designed so as to implement separation of concerns. The Presentation Layer entirely coexists on the main thread as has to be in the event loop of Tkinter, and communicates with the Event Capture Layer on thread-safe queues. This does not couple filesystem monitoring to allow the GUI to be snappy.

even responsive in times of high filesystem traffic. The Response Layers and the Detection Engine collaborate within the watchdog observer; therefore, the analysis of threat takes place within the same event-processing context as the filesystem notification reducing the end-to-end detection latency.

Presentation Layer is created in the form of a Tkinter window, with fixed layout, and four major sections. The header bar provides the application title and the level of monitoring in status. On the left hand sidebar, you can choose a directory with tkinter. filedialog.askdirectory and provides start/stop controls. The because system informs the central panel that the total files scanned and suspicious files flagged by the GUI sum are being updated by polling the GUI update queue every 100 milliseconds with Tkinter update of after function. One scrollable log window at the bottom displays the events with timestamps color-coded (including the default color and red flag in the case of threats) indicating the operator immediately notices the event.

The other advantage that would be made by the modular architecture is that new features can be easily added. New heuristic detectors can be connected to the Detection Engine as disconnected functions which accept a file path and an event type and return a Boolean flag. The response pipeline also enables plug ins, meaning that you can add automated quarantine routines, network isolation commands or cloud threat -reporting APIs later without having to hit the underlying event -processing logic.

V. PROPOSED ALGORITHM

Input: The events in the filesystem in real time included by the operating system, in other words, they become notified when files are modified.

Output: A threat is displayed in a log or in a pop-up. The first thing I do is compile the Watchdog Observer and my application Ransomware Handler. Next, I would request the user to choose the directory he/she wants to monitor. To start monitoring I would need to run the Observer as a daemon thread and leave it to monitor.

Upon the notification of the event in filesystem: a. In that it is just a directory event I disregard it.

b. I update my counter total and put a topic to the GUI to allow the user to see the difference.

c. I note the type of event, location of the file and time so that I can later analyse this information.

This is done by calling check Suspicious Activity (event):

a. I compare file extension with the list of suspicious extensions that I have.

ISSN: 2583-6129
DOI:10.55041/ISJEM05985

b. In the case of modified file, I get old records off my modification counter, increment the counter and test whether the counter exceeds the limit.

c. In case the file had been changed or created, I calculate Shannon entropy H, in case the difference between the current entropy and the last recorded entropy exceeds 0.5 I mark the record as odd. I also database the entropy history table.

If something gets flagged, I: a. Refresh the GUI and add to the counter of suspicion.

b. Write a THREAT DETECTED line to the log and put a red circle around.

c. Display a modal alert window in the main thread to provide the user with information that something is going on.

d. Make it appear as an attack by so-called showing a fake ransom bill.

I repeat these steps of step 4, each time a new event occurs. On pressing Stop the Observer thread closes, and starts all button states.

TABLE I: Comparisons between the existing and proposed systems.

Criteria	**ER1 (Signature -Based)**	**ER2 (Heuristic -Based)**	**Proposed System (ML + Entropy)**
Accuracy Level	Poor	Average	Excellent
Zero-Day Capability	Not Supported	Partially Supported	Fully Supported
False Alerts	Frequent	Minimal	Controlled
Processing Speed	Slow	Medium	Instant / Real-Time
Automation Degree	Manual	Semi-Automatic	Fully Automatic
Entropy Usage	Not Used	Not Used	Shannon Entropy Applied
Monitoring Ability	Restricted	Basic	Advanced & Continuous
Security Strength	Weak	Moderate	Strong
Logging System	Manual Records	Limited Logging	Centralized Logging + Export
System Scalability	Low	Medium	High
Reporting Method	Manual Reports	Dashboard (Basic)	Live Dashboard (Advanced)

VI. EXPERIMENTAL RESULTS AND DISCUSSION

There were no questions about the experiment our group conducted...

I did some controlled tests on a laptop with Windows 10 operating system, having a quad-core CPU, and 8GB of RAM. My folder had 500 files of mixed types, which I monitored. I used the approach of simulating ransomware by modifying files quickly, reorganizing extensions in bulk and encrypting data with AES-256 that generates high-entropy data.

With the entropy check I had the capability to capture 94% of the encrypted events in a second. The modification rate test did not pass until the bulk encryption test was greater than five files a second. All renames were immediately detected by the extension detector.

The three heuristics performed well with a 97.0-0.1 false positive and a 2.1-0.1 true positive on a regular workload with a 97.0-0.1 detection rate. It took a maximum of 500 Ms between the detection and alert. We can be compared with the existing solutions as indicated in the comparison table below.

I attempted to scale by loading 10,000 files and no obvious overhead to the CPU was detected than what Watchdog ordinarily would have, and so it should similarly be able to work with bigger enterprise directories. In fact, I was simply checking on whether the system remains lean and it does well, no enormous jumps in CPU use even when I have been working with plenty of files at the same time, which is nice when you ought to be working with a huge folder in an organization.

In an attempt to further confirm the strength of the system, some adversarial stress tests were carried out which simulated evasion techniques that can be adopted by advanced variants of ransomware. The first adversarial test involved a simulation of a slow encryption scenario, whereby the rate of file

modifications was two a minute, which is meant to induce the being able to evade the heuristic on the rate of modifications. With this circumstance, 89% of encrypted files were recognized by the extension detector and entropy analyser prior to the attack accomplishing over 15 percent of the target file group. This proves that the multi-heuristic method gives resistance to avoidance of only one signal.

In the second adversarial test, controlled conditions of false-positive were produced with a benign application that carried out bulk file compression. The compression process generated high-entropy output files and fast-changing events of modifying files in accordance with ransomware activities. In this test, the combined system yielded a false-positive value of 2.1% which agrees with the baseline value. Manual inspection of the false-positive items found out that all came as a result of compression as the write of zip archives which are high-entropy in nature. This observation indicates that a whitelist system of applications or patterns of file types that have been determined to be safe would further minimize the false-positive rate of production deployment.

Experimental memory and CPU profiling proved that the system had a constant resident set of about 45MB (with experimentally verified working conditions) and had a baseline CPU consumption of 1.8% on a quad-core system doing basic probing of the system and a peak of 6.3% during simulated bulk encryption events. These consumption values are even smaller in comparison with overheads of commercial endpoint detection and response (EDR) agents, which regularly ingest between 5 and 15 percent of CPU capacity resources. The proposed system is especially usable in resource-constrained systems like industrial control workstations, legacy hospital equipment, and embedded systems which cannot support heavyweight security agents due to their lightweight.

VII. CONCLUSION

In this paper, a ransomware detection and response program will be described, which will be based on the Shannon entropy analysis, fast modification-rate monitoring, and suspicious extensions checks. Accuracy of 97.0 percent and false-positive of 2.1 percent is a better result compared to signature-only and single-heuristic. Event-driven architecture based on the Watchdog will imply that the time of responding is maintained in the real-time range, and the commodity hardware is not overloaded. Establishing data capturing on tkinter gives the live graphic and easy exportation of incident reports.

Future features will include support of a random-forest based classifier, that will analyse other characteristics of filesystem behaviour to automatic filesitary quarantine and rollback, and feed off a cloud based threat-intelligence feed to reduce false positives further as well as to cover more attack patterns. The experimental findings and architectural design have been used to identify several specific directions on how it can be developed in the future. First, automated quarantine of files and rollback based on shadows would go a long way in minimizing the effects of any given ransomware attack that is realized after partial encryption of files. When a threat is detected, the system would instantly duplicate the unencrypted copies of the endangered files to an encrypted, non-monitored backup directory and then obstruct the offending process to successive filesystem usage by utilizing API process administration systems. This would turn this system into a detection and alert system as opposed to a detection and response platform which has the capability of reducing damage in scenarios where early detection is not possible.

Second, random forest, which is now integrated with a better set of features, such as access patterns to a file, process lineage, network socket usage, and registry key changes on Windows, should be better to achieve better results in classification and still have the same interpretability benefits as ensemble shedding procedures compared to deep neural networks. Such a classifier might be trained on publicly available datasets of ransomware behaviour including CIC-MalMem-2022 and implemented as an extra scoring layer on top of the existing heuristics only to reach an alert level ensure the multiple signals go beyond their respective individual thresholds simultaneously.

Third, it would be useful to have the system cross-reference reported file hashes and suspicious extension patterns with globally maintained indicators of compromise (IoCs) by integrating cloud-based threat intelligence, e.g., Virus Total, AlienVault OTX, or proprietary enterprise threat feeds, through APIs. This would give the system an extra line of corroboration on borderline detections and allow the system to take advantage of the aggregate threat intelligence of the larger cybersecurity community near-real time. A federated deployment model, in which anonymized telemetry of many instances of the system is consolidated, can also support community-driven ways

of improving over the time detection thresholds and extension watchlists.

To conclude, this paper shows that it is possible to build an effective, real-time ransomware detector without heavyweight infrastructure, cloud or network connectivity or kernel drivers. The joint application of Shannon entropy analysis, a modification rate monitoring, and an extension checking offers a multi-signal detection which is robust to deflection on any of the single heuristics. The event-based, loosely coupled architecture, which is provided with the help of Watchdog and Tkinter Python libraries, offers a sound and somewhat scalable platform upon which increasingly advanced endpoint ransomware defence mechanisms can be developed and deployed as needed, in an exceptionally wide variety of deployment settings.

REFERENCES

[1] M.Christodorescu and S.Jha,USENIX security, 2003, p.178-180, Static analysis of executables to identify malicious patterns. (Literature review)

[2] C. E. Shannon, A mathematical theory of communication, Bell Syst. Tech. J. vol. 27, pp. 379-423, 1948. (Foundational)

[3] A. Kharraz et al., "Ransomware attacks: Cutting the Gordian knot, DIMVA, 2015. (Case studies)

[4] N. Scaife et. al. CryptoDrop: a filesystem activity-based ransomware detection method, IEEE ICDCS, 2016. (Detection techniques)

[5] A. Continella et al., ShieldFS: A ransomware aware filesystem that heals itself, ACSAC, 2016. (Defence mechanisms)

[6] M. Monika et al., Ransomware: experimental study, Procedia Comp. Sci., vol. 83, 2016. (Experimental results)

[7] D. Sgandurra et al., Automated dynamic analysis of ransomwarearXiv:1609.03020, 2016. (Dynamic analysis)

[8] Q. Chen, R. A. Bridges, Analqedis, "Automated behavioural analysis of malware, ICMLA, 2017. (Behavioural analytics)

[9] Python Watchdog Files Watchdog, Python PyPI, 10. (Tool)

[10] Tkinter Documentation, Python Software Foundation, n.d. (GUI lib).

[11] The article Ransomware: Evolution and prevention by R. Richardson and M. North, Intl. Mgmt. Review, 2017. (Trend analysis)

[12], P. O'Kane et al, Evolution of ransomware, IET Networks, vol. 7, no. 5, 2018. (Evolution)

[13]. T. McIntosh et al., "Ransomware mitigation in the new era, ACM Comp. Surveys, 2021. (Survey)

[14] S. Homayoun et al, DRTHIS: Deep ransomware threat hunting, Future Gen. Comp. Syst., 2019. (Deep learning)

[15] G. Bello and S. Watkins, Real-time ransomware detection IEEE COMPSAC, 2018. (Real-time detection)