RPM: Ransomware Prevention and Mitigation Using Operating Systems' Sensing Tactics

Conference Paper · February 2023

CITATIONS
2

READS
174

4 authors, including:

Elias Bou-Harb
University of Texas at San Antonio
154 PUBLICATIONS **2,762** CITATIONS
SEE PROFILE

Chadi Assi
Concordia University Montreal
462 PUBLICATIONS **11,014** CITATIONS
SEE PROFILE

RPM: Ransomware Prevention and Mitigation Using Operating Systems' Sensing Tactics

Ricardo Misael Ayala Molina*, Elias Bou-Harb†, Sadegh Torabi‡, and Chadi Assi*
*The Security Research Centre, Concordia University, Montreal, Canada
†The Cyber Center For Security and Analytics, University of Texas at San Antonio, San Antonio, United States
‡The Center for Secure Information Systems, George Mason University, Fairfax, VA, United States

***Abstract*—Ransomware, an extortion type of malware, continues to create havoc targeting critical infrastructure and organizations at large, causing an estimated $20 Billion in direct and collateral damages in 2022. While significant efforts from both academia and industry are being pledged to address this debilitating and disrupting phenomena, the ransomware pandemic continues to expand rapidly in frequency, spread and stealthiness. To this end, in this work, we propose `RPM`, a `R`ansomware `P`revention and `M`itigation scheme. `RPM` is rooted in the proactive analysis of operating systems' API artifacts through the exploitation of a neat observation related to ransomware behavior, namely, activities generated prior to the actual execution of the malicious payloads. RPM employs OS-centric process hooking tactics to develop an offensive approach leveraging such sensing activities. To demonstrate the effectiveness of `RPM`, we empirically evaluated it using 100 of the most prominent ransomware samples. The results demonstrate very motivating accuracy metrics with low system footprint, asserting the rationale of the proposed scheme. We posture `RPM` as a strong step towards proactive mitigation, which aims at complimenting ongoing ransomware thwarting efforts.**

***Index Terms*—Ransomware, Operating systems, Host-based, Cyber Forensics**

I. Introduction

MALWARE, short for malicious software, has always been a key enabler to conduct many of today's malicious activities. While malware could manifest in various types including self-propagating worms, backdoors, spyware, trojans and others [1], a very specific type dubbed as ransomware started to re-gain traction in May 2017 with the eminent `WannaCry` attack [2]. While ransomware's history goes back to 1989 when a Harvard-trained evolutionary biologist Joseph Popp sent 20K infected diskettes labeled "AIDS Information: Introductory Diskettes" to attendees of the World Health Organization's international AIDS conference, encrypting their hard drives, ransomware's threat landscape has significantly evolved and intensified since 2013. Though such ransomware could be classified based on several criteria, including their functionality (i.e., locker-ransomware, which lock the screen of the victim to prevent access to data/system resources vs crypto-ransomware, which encrypt the data), their encryption routine, their underlying cyber-crime economic model, or their target hosts (i.e., OS type, mobile, IoT, etc.), the intended end-goal is quite the same, which is to prevent access to local and networked data and computing/processing resources for illicit financial gains. Indeed, the past two years have witnessed a momentous, unprecedented surge in ransomware attacks targeting highly diverse infrastructure.

For instance, very recently in 2022, Cisco systems fell victim to a foreign state-sponsored ransomware attack in which more than 50GB of classified documents, technical schematics, and source code were published to the dark web. Taiwanese computer manufacturer Acer was a victim of the `REvil` ransomware attack in the last quarter of 2021. This attack was particularly noteworthy due to its demand of $50M; the largest known ransom to date. Further, the Colonial Pipeline attack was arguably the most high-profile ransomware event of 2021. Colonial Pipeline is responsible for transporting nearly half of the U.S. East Coast's fuel. The ransomware attack was the largest cyber attack to target a critical infrastructure in U.S's history. Also in the U.S., in May of 2021, JBS, the world's largest meat processing plant, was hit by a ransomware attack that forced the company to stop the operations of all its beef plants. The cyber attack significantly impacted the food supply chain and highlighted the manufacturing and agricultural sectors' vulnerability to disruptions of this nature. Broadly, ransomware attacks hit more than 40% of fortune 500 companies in the past two years, including recent attacks on Garmin by state-sponsored actors, which paralyzed the avionics of several airlines, costing the company a postulated $10M.

While the direct cost of such attacks could be defined by the amount of ransom that the victims initially pay out, the corresponding collateral damage related to downtime, data recovery, lost of revenue, improvements to cyber defenses, and reputational damages is estimated to be close to $20B in 2022, with a predicted $40B by 2024. Thus, in face of such worrisome information and the ongoing global threat, it becomes intuitive, timely and highly imperative to devise, develop and evaluate empirically-driven methods and techniques to address the problem of ransomware.

Along this vein, research and development efforts addressing the ransomware phenomena have circulated around binary analysis, measurements and characterization, and network- and

host-based detection (see Section II). While such endeavors are very significant to comprehend and defend against the ransomware threat, efforts which offer proactive prevention methodologies seem to be quite limited. Indeed, while detection methods might be effective, nevertheless, in the context of ransomware, they might not be well-suited as they are reactive in nature, and typically have to maintain some sort of detection signatures. In this context, one should be aware that addressing the ransomware problem is quite different than addressing the broader malware issue. A core reason behind this could be illustrated when a ransomware lands at a host operating within a critical infrastructure such as a health entity. Here, a successful attack that was detected late would have already caused critical damage to the healthcare operations (e.g., threatening the survivability of critical patients on ventilators) and would have violated imperative compliance acts and regulations such as HIPAA (especially when the encrypted data is published in raw formats on the dark web to pressure the attacked entity to pay the ransom). In such context, detection would be unacceptably reactive, given that the tolerance for such a delay within this specific operating environment is nil.

This and related scenarios demonstrate the utmost need for proactive prevention methodologies in contrast to only detection. Furthermore, an effective ransomware solution ought to leverage notions and artifacts that do not require major and continuous updates, such as those generic malware solutions, which often carry (and have to maintain) a large set of detection signatures that are known to be easily circumvented with evolving malware.

Therefore, motivated by the severity of such ransomware attacks while realizing the gap related to the lack of proactive and prompt preventative measures, we propose herein **RPM**, a **R**ansomware **P**revention and **M**itigation scheme. Building upon our previous work [3], RPM is a host-based solution, given the momentous impact of the "last mile" on the success of the attack, and is postured as a *complementary*[1] (rather than a replacement) approach to already-available network- and host-based detection schemes. In this context, this work makes the following contributions:

1) We introduce RPM, a unique scheme which addresses the problem of ransomware prevention at the host. RPM deviates from the reactive process of detection, endeavoring to perform instantaneous halting of ransomware-affiliated processes, prior to any malicious payload execution. RPM is designed to operate in the background of an Operating System (OS), requiring very low footprint in terms of space and time complexities, while being easily deployable.
2) We exploit a very neat observation related to how ransomware probe its target host, prior to launching any malicious misdemeanors. We refer to these events as "paranoia", in which the ransomware assesses, using a manic/aggressive manner, if the host OS and related processes represent ideal grounds for execution. RPM embeds such rationale within its methodology to turn these paranoiac activities against the ransomware for prompt inference, prevention and thus mitigation. Particularly, RPM leverages OS process hooking techniques, intercepting, analyzing and accordingly reacting to the generated ransomware-sensing APIs.
3) We prototype RPM and empirically evaluate its effectiveness using recent ransomware samples. While we demonstrate that the obtained accuracy metrics are quite motivating, we also show that RPM possesses the capability to be lightweight and portable, given that it *(i)* does not require any modifications to the OS, *(ii)* does not depend on auxiliary software or specialized hardware support, and *(iii)* does not provision any signature detection models.

[1]We are not per say interested to compare RPM vis-à-vis against other host-based approaches, but rather make it operational along them to fill a gap.

The rest of this paper is organized as follows. In the next section, we elaborate on the related work. In Section III, we detail RPM's embedded approach and rationale. In Section IV, we prototype RPM and empirically evaluate it using recent ransomware binary samples, while reporting on its effectiveness and accuracy metrics. Lastly, in Section V, we summarize the contributions of this work while pinpointing a few endeavors which aim at paving the way for future work.

II. Related Work

Herein, we provide a review of recent and the most representative research endeavors addressing multidimensional aspects of the ransomware phenomena.

Forensic Analysis and Characterization. A plethora of ransomware investigative studies have relied on static or dynamic analyses. To this end, Nunes et al. [4] have recently performed dynamic analysis of contemporary ransomware to provide a longitudinal study of their evasive behaviors. They also motivated the need for more malware simulators for large-scale system calls' analysis. Alternatively, Zhang et al. [5] exploited opcode instructions from ransomware samples and applied a probabilistic language model rooted in N-gram analysis to convert them into sequences. A statistical technique was then applied on such sequences to generate feature vectors, which were fed to shallow learning techniques to enable the classification of ransomware families. From a network forensics perspective, Kurniawan et al. [6] investigated the `Cerber` ransomware, to highlight its *modus operandi*, while Quinkert et al. [7] proposed a network forensic approach to predict the identity of ransomware by exploring the domain names that they employ. In the area of ransomware characterization, Huang et al. [8] presented a 2-year measurement study of ransomware's threat landscape by focusing on payments, victims and operators. By combining an array of data sources, the

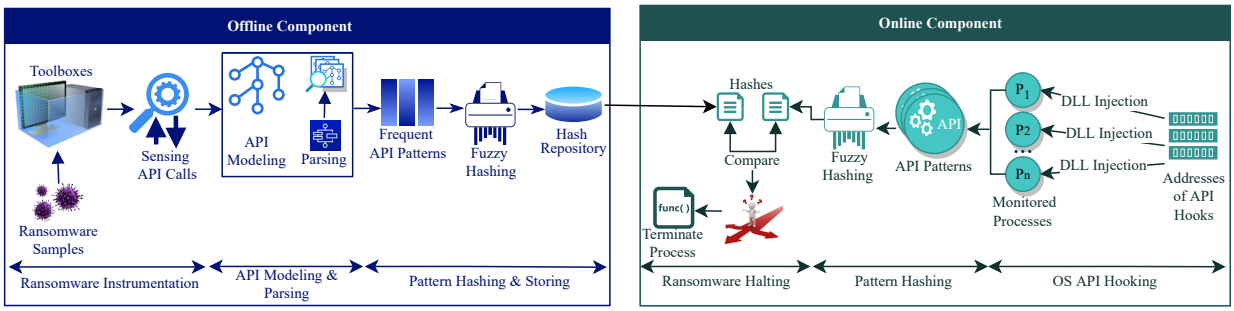

Figure 1. **RPM:** System Architecture and Components

authors demonstrated the burgeoning ecosystem and associated third-party infrastructure.

Network-based Detection Approaches. Given that ransomware traverse and exploit networks to propagate and reach the intended targets (i.e., via drive-by downloads, backdoor droppers, email attachments, etc.), a number of studies have been conducted to provide network-based resiliency. In this context, Almashhadani et al. [9] performed dynamic analysis of a large corpus of ransomware data to then introduce a flow-based fusion classifier to perform the network detection. The same authors in [10] have analyzed the network behaviors of the `Locky` ransomware, proposing several heuristics which work in-tandem to achieve the detection. In contrast, Kim et al. [11] leveraged network programmability capabilities, including Software-Defined Networking (SDN) and programmable forwarding engines, to devise and deploy adaptable network-based, learning detection algorithms. Additionally, Roy et al. [12] captured the network activities of legitimate users in an organizational network, while developing deep learning methodologies to detect ransomware-generated, network-wide encryption activities.

Host-based Detection Approaches. Arguably, the most effective approaches to ultimately mitigating the threat of ransomware are those which reside and operate at the targeted end-hosts. To this end, a number of noteworthy approaches have emerged. For instance, Kharaz et al. [13] presented a seminal work which monitors file systems' activities, namely, I/O requests of `Writes` and `Deletes`. By initially capturing distinctive I/O patterns generated by unique ransomware families and subsequently computing an entropy-based metric, the approach is able to distinguish between malicious and benign activities. Alternatively, Kolodenker et al. [14] designed and developed a system which tracks the use of symmetric session keys at the victim's machine. By employing crypto-function hooking techniques, the proposed approach was able to infiltrate the encryption session and hold the keys in an escrow, thus decrypting files that would otherwise only be recoverable by paying the ransom. In another context, Alam et al. [15] leveraged hardware performance counters to dynamically observe the hardware-related activities of a system. By capturing such low-level instructions and developing an autoencoder, the proposed approach was capable of fingerprinting benign activities while alerting about ransomware malicious activities.

In this work, we introduce RPM, which we posit as a complementary host-based solution. In contrast to the above-mentioned literature, RPM is unique (1) as it addresses the problem of ransomware prevention as opposed to detection, (2) does not require any complex (performance-impeding) API modeling schemes, without necessitating any modifications to the underlying OS or obtaining aid from auxiliary (or third-party) software or hardware. Additionally, (3) its underlying premise is distinctive, exploiting ransomware "paranoia", a set of fingerprinting activities that the ransomware conducts prior to executing its malicious payload, to offer a significantly proactive, lightweight solution while being highly portable between Linux variants and Windows OSs.

III. RPM: Proposed Approach

In this section, we present RPM and its related components, while elaborating on its inner machinery. Figure 1 holistically illustrates RPM, which depicts two core subsystems operating in an offline and online fashions, respectively.

A. RPM's Offline Component

The intuition behind RPM is to use ransomware's environment-sensing activities against them to perform prompt inference and mitigation. Recall that these activities are induced by almost all contemporary ransomware to avoid executing their malicious payloads in non-suitable environments (such as on virtualized systems or on erroneous architectures). These tactics, while they contribute to the efficiency and effectiveness of the attack, they are also typically referred to as evasion tactics as they ultimately aid the ransomware in remaining dormant and stealthy. For instance, a ransomware can sense a system's registry for any indicator of an installed anti-malware solution (to avoid it), or it can sense virtualized network interface cards to realize that it is running in a virtual setup (circumventing dynamic analysis).

To this end, the malware research literature [1] has amalgamated a large set of evolving sensing/evasion activities, most of which could be categorized as summarized in Table I. For

example, generic OS queries will verify if the total RAM is low, the number of processors is low, or if the host name is specific. Additionally, network-specific queries will examine whether the MAC address or adapter names are particular, or whether the network that the sample is running within belongs to some security container. Further, hardware-related queries will check whether the HDD has specific name or if the HDD Vendor ID has specific value, or if the audio device is absent. Indeed, such artifacts would return identifying and distinguishing output (between a suitable and an unsuitable execution environment) that the ransomware would typically leverage to sense a suitable environment for execution. Readers that are interested in more elaborative details related to these activities and their categories, along with their related *modus operandi* are kindly referred to [1]. In this vein, RPM exploits such *pre-attack*, ransomware-generated activities to fingerprint and proactively kill the system process forking the sensing activities, ultimately thwarting the ransomware.

Initially, offline, as depicted in Figure 1, RPM performs large-scale API monitoring (using old and recent ransomware samples) to extract and vet the list of monitored paranoiac activities based on the categories of Table I. This instrumentation procedure is achieved by running the ransomware samples in open-source and propriety sandboxing tools such as the Cuckoo sandbox and VIPRE's ThreatAnalyzer. Now given that each ransomware sample is expected to generate a large number of such API calls, there is a need to reduce the output to obtain the most generalizable and frequent sets. This would allow RPM to avoid carrying a large repository of repetitive and/or possibly obsolete API calls. To this end, RPM leverages all the previously extracted sensing APIs to build and parse a Frequent Pattern (FP) tree. In such a tree, each node after the root will represent a captured sensing API, which will be shared by the sub-trees beneath. Each path in the tree will represent sets of APIs that co-occur, in non-increasing order of frequency of occurrences. Thus, two samples that have several frequent APIs in common and are different just on infrequent APIs will share a common path on the tree. In this context, the FP tree-based mining algorithm, FP-growth, is employed by RPM for mining the complete set of generated frequent API patterns, in linear time [16].

For efficiency and practicality purposes, RPM also executes fuzzy hashing on each of the captured patterns, generating unique strings that provide similarity scores (rather than a binary output) when used in comparison operations. Herein, the capturing of the APIs, modeling and parsing the FP-tree as well as the hashing of the generated patterns will be executed offline. RPM also employs an update frequency mechanism which would be used to update such hashes (if needed) as new (raw) ransomware are harvested and analyzed.

B. RPM's Online Component

When deployed at the host, RPM aims at leveraging the previously generated paranoiac API patterns (namely, the computed hashes) to perform proactive inference of such activities to instantaneously halt the process initiating such activities. To enable such procedure in real-time, as illustrated in Figure 1, RPM exploits OS API hooking techniques (i.e., a set of methods to intercept and modify process behaviors). Specially, hooks are created for each of the ransomware-generated sensing APIs. Consequently, the memory addresses of such hooks are obtained, where RPM employs DLL injection to target possibly malicious, newly forked processes, by leveraging multi-threading. The latter procedure will enable RPM to capture all the generated APIs from (suspicious) processes. Once the APIs are captured, RPM computes their fuzzy hashes and compares them to those previously obtained from the offline module. If the similarity metric is beyond a pre-defined threshold, the process generating such APIs is deemed to be executing sensing activities, thus inducing RPM to execute a `TerminateProcess()` towards it, proactively halting the (ransomware) process.

Table I
RANSOMWARE SENSING CATEGORIES.

APIs Sensing Category	Synopsis
File System	Checks files and directories
Registry	Queries registry keys
Generic OS queries	Verifies target functions
Global OS objects	Checks mutexes and virtual pipes
UI artifacts	Validates open applets
OS Features	Checks stack and debug manners
Processes	Validates modules and libraries
Network	Queries network functions
CPU	Verifies processor instructions
Hardware	Checks emulated hardware devices
Firmware Tables	Checks memory areas

Assumptions and Considerations. We assume that RPM, when operating in an online fashion on the host, is deployed on vanilla OSs (i.e., freshly installed software and hardware drivers). Additionally, we also assume that RPM has knowledge about additional software/hardware modifications, which are part of the organizational policies (i.e., additional documents' processing software, specialized hardware drivers, etc.). These steps guarantee that RPM creates and employs a safelist of benign I/O- and CPU-bounded processes, thus reducing (or eliminating) the generation of false positives during real-time operations. We also assume that RPM has elevated privileges on the OS, which allows the usage of hooking techniques while accessing memory addresses.

Further, we also take into consideration the fact that some ransomware for one reason or another would not generate sensing activities, and thus will possibly circumvent RPM (leading to false negatives). We consider that such ransomware, which directly execute their malicious payloads, would have already been detected by complementary (typical) anti-viral or endpoint solutions, as they would have been successfully previously analyzed, say, through dynamic analysis, generating their attack signatures, which in turn would be used for their detection. Along this vein, it is important to note that RPM

is designed with a pluggable functionality, which generates attack signatures automatically when specific processes have been halted. Such signatures are envisioned to be fed to typical anti-virus solutions to aid in the mitigation of similar threats.

In relation to the offline module, particularly its update mechanism dealing with newly identified sensing activities, we notice that such sensing activities are relatively consistent between various contemporary ransomware. Having said that, and assuming we have a near real-time feed of ransomware samples (from publicly available sources or through collaborations with anti-virus companies running large-scale honeypots), the update mechanism could be quite frequent, thus reacting rather quickly to any evolving "in-the-wild" ransomware threat.

Lastly, we consider that the similarity matching threshold of the sensing API patterns (retrieved from the offline module and used in an online manner) to be a configurable parameter, set by the administrator of the OS whose deploying RPM. Here, one should balance between having a very loose threshold, thus informing RPM to generate a large number of alerts related to the terminated processes (as an indicator of ransomware activities) and between a very strict threshold, which would induce RPM only if almost an exact match has been observed.

While the former threshold will guarantee that all suspicious processes are terminated, it could generate a lot of noise that would overwhelm system administrators (who are already flooded with other alerts from Security Information and Event Management (SEIM) systems). Conversely, the latter threshold could be too rigid, possibly causing false negatives. A rule-of-thumb here is for system administrators to take into consideration the intrinsic natures of their realms when tuning the threshold, including the risk tolerance level, the criticality of the sector that their systems are operating within, and the resources available to perform post threat analysis.

IV. Implementation and Evaluation

We developed RPM's offline component in `Python` where a `Redis` in-memory database is used to store the generated hashes. RPM's online component is implemented in `C` and leverages a dictionary data structure as the safelist APIs. All the prototyped code, the leveraged techniques and the generated artifacts are native to any contemporary OSs, making RPM highly portable, without depending on auxiliary third party software or hardware support.

Complexity. Modeling and parsing the APIs occur in linear time, and the FP tree requires less than 50 MB of data to be stored, for every 20 various ransomware samples. For fuzzy hashing, RPM leverages the `ssDeep` tool. Further, the DLL injection procedure occurs in constant time, per thread.

Effectiveness. We initially executed various controlled tests to verify the effectiveness of RPM when deployed on real OSs. To this end, we collected and utilized 100 recent samples/variants from 5 currently "in-the-wild" ransomware families, including `Reveton`, `Locky`, `Teslacrypt`, `Yakes`, and `Cerber`.

(a) Successful attack without RPM

```
RPM
Listening to sensing APIs...
[+] Call to ntdll.dll!NtClose( Ret Addr: 0x75b4994a(N/A) no of Args 1)
[+] Call to user32.dll!GetSystemMetrics( Ret Addr: 0x696c20b7(N/A) no of Args 1)
[+] Call to user32.dll!GetSystemMetrics( Ret Addr: 0x696c215c(N/A) no of Args 1)
[+] Call to user32.dll!GetSystemMetrics( Ret Addr: 0x69713be7(N/A) no of Args 1)
[+] Call to user32.dll!GetSystemMetrics( Ret Addr: 0x69713b4e(N/A) no of Args 1)
[+] Call to user32.dll!GetSystemMetrics( Ret Addr: 0x697150fb(N/A) no of Args 1)
Process (pid 4798) was halted!
```

(b) Halted attack using RPM

Figure 2. Execution of the `Cerber` ransomware

These samples cover both crypto- and locker-ransomware. Windows 10 was used in these tests, given that it is currently the most targeted OS. Figure 2a shows how the `Cerber` ransomware attack was successful when RPM was not deployed while Figure 2b reveals the effectiveness of RPM in mitigating the ransomware process. We also experimented with the other ransomware families, but we omit the depiction of their similar results due to space limitations.

This demonstrates that RPM can be easily deployed and highly effective on real OSs. Furthermore, we ran additional experiments to capture how RPM behaves on vanilla OSs, *without* having access to any supplementary knowledge (i.e., by omitting the safelist APIs). Such experiments could be considered as stress testing for RPM, given that typically it operates using a safelist, as noted in Section III-B. We employed the 100 ransomware samples in conjunction with 20 applications (representing common organizational software, including browsers, file compression, office suit, media players, etc.) and executed them on the Windows OS while RPM was operating in an online fashion. We ran 5 iterations of this experiment, by using randomly-selected, different ransomware and benign applications each time and averaged the results. Figure 3 demonstrates the obtained key performance metrics, revealing that RPM successfully mitigated all 100 ransomware samples, while generating only a single false positive.

Upon further inspection, it was discovered that the false positive was related to how the compression software `WinRaR` operates, which apparently generates a number of sensing tactics, before it actually conducts its unzipping procedure. Nevertheless, on production systems, this could be easily remediated by either safe-listing `WinRaR` or just fine-tuning the similarity threshold. From this experiment, it can be deduced

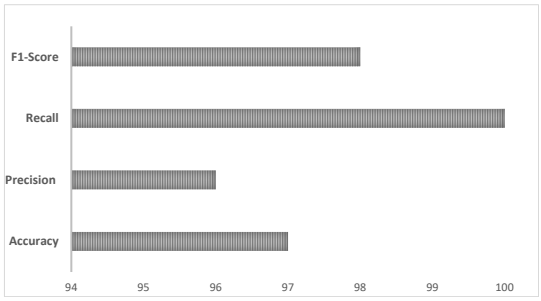

Figure 3. Generated accuracy metrics without a safelist

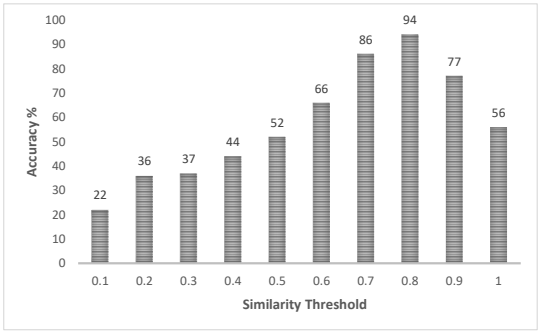

Figure 4. Impact of similarity threshold on the accuracy metric

that RPM, without auxiliary knowledge about its operating environment, is quite effective in distinguishing between benign and ransomware processes. This attests the unique rationale of employing OS sensing tactics as a key premise within RPM, to perform proactive mitigation of ransomware attacks, without relying on the typical (and specific) attack payload signatures.

Similarity threshold. We were also interested in exploring how the similarity threshold affects the accuracy metric. To this end, based on the previous 5 experimental iterations using random ransomware samples and benign applications, Figure 4 illustrates how the accuracy score fluctuates by modifying the similarity threshold, on average. It can be inferred that a reasonably strict threshold of around 0.8 yields the overall best accuracy. Nevertheless, as noted earlier in this article, system administrators have to take the final decision related to this threshold based on their intrinsic network dynamics and their risk tolerance levels, balancing false positives/negatives.

V. Concluding Remarks

In this paper, we introduced RPM as a complementary host-based ransomware preventative solution. RPM is based on the rationale that we can proactively prevent ransomware attacks by turning innate ransomware activities against them. RPM is based on two components, which work in-tandem to accomplish the intended goals. While we discussed a number of consideration and assumptions related to RPM and its *modus operandi*, we also empirically demonstrated its effectiveness and low system footprint on a real OS using an iteration of its prototype. As for future work, we are currently enhancing RPM's implementation to make it fully automated, increase its scalability and render it resilient to targeted service attacks. We also aim at evaluating it on various Linux-based systems along with experimenting it on the `Contiki` IoT-centric OS; given that IoT devices have been recently reported to also be an appealing attack vector to ransomware [17], as these IoT sensors continue to be deployed deep within critical infrastructure.

Acknowledgment

This work has been supported by Natural Sciences and Engineering Research Council of Canada (NSERC) and Concordia University. This work was also partially supported by a grant from the National Science Foundation (NSF) award #2104273.

References

[1] O. Or-Meir *et al.*, "Dynamic malware analysis in the modern era—a state of the art survey," *ACM Computing Surveys*, vol. 52, no. 5, pp. 1–48, 2019.

[2] H. Oz *et al.*, "A survey on ransomware: Evolution, taxonomy, and defense solutions," *ACM Computing Surveys*, vol. 54, no. 11s, pp. 1–37, 2022.

[3] R. M. A. Molina *et al.*, "On Ransomware Family Attribution Using Pre-Attack Paranoia Activities," *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 19–36, 2021.

[4] M. Nunes *et al.*, "Bane or boon: Measuring the effect of evasive malware on system call classifiers," *Journal of Information Security and Applications*, vol. 67, p. 103202, 2022.

[5] H. Zhang et al., "Classification of ransomware families with machine learning based on n-gram of opcodes," *Future Generation Computer Systems*, vol. 90, pp. 211–221, 2019.

[6] A. Kurniawan and I. Riadi, "Detection and analysis cerber ransomware based on network forensics behavior," *International Journal of Network Security*, vol. 20, no. 5, pp. 836–843, 2018.

[7] F. Quinkert *et al.*, "Raptor: Ransomware attack predictor," *arXiv preprint arXiv:1803.01598*, 2018.

[8] D. Huang et al., "Tracking ransomware end-to-end," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 618–631.

[9] A. O. Almashhadani *et al.*, "Mfmcns: a multi-feature and multi-classifier network-based system for ransomworm detection," *Computers & Security*, vol. 121, p. 102860, 2022.

[10] A. Almashhadani *et al.*, "A multi-classifier network-based crypto ransomware detection system: a case study of locky ransomware," *IEEE ACCESS*, vol. 7, pp. 47 053–47 067, 2019.

[11] H. Kim *et al.*, "Design of network threat detection and classification based on machine learning on cloud computing," *Cluster Computing*, vol. 22, no. 1, pp. 2341–2350, 2019.

[12] K. C. Roy and Q. Chen, "Deepran: Attention-based bilstm and crf for ransomware early detection and classifcation," *Information Systems Frontiers*, pp. 1–17, 2020.

[13] A. Kharaz et al., "UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 757–772.

[14] E. Kolodenker *et al.*, "Paybreak: Defense against cryptographic ransomware," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017, pp. 599–611.

[15] M. Alam et al., "Ratafia: ransomware analysis using time and frequency informed autoencoders," in *2019 IEEE Int. Symposium on Hardware Oriented Security and Trust (HOST)*, 2019, pp. 218–227.

[16] W. Gan *et al.*, "A survey of utility-oriented pattern mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1306–1327, 2019.

[17] C. Brierley *et al.*, "An IoT Bricking Ransomware Proof of Concept," in *Proc. of the 15th Int. Conf. on Availability, Reliability and Security*, 2020, pp. 1–10.