Rapid Ransomware Detection Through Side Channel Exploitation

Michael A. Taylor
Darwin Deason Institute
Southern Methodist University
Dallas, TX, USA
taylorma@smu.edu

Eric C. Larson
Darwin Deason Institute
Southern Methodist University
Dallas, TX, USA
eclarson@smu.edu

Mitchell A. Thornton
Darwin Deason Institute
Southern Methodist University
Dallas, TX, USA
mitch@smu.edu

***Abstract*—A new method for the detection of ransomware in an infected host is described and evaluated. The method utilizes data streams from on-board sensors to fingerprint the initiation of a ransomware infection. These sensor streams, which are common in modern computing systems, are used as a side channel for understanding the state of the system. It is shown that ransomware detection can be achieved in a rapid manner and that the use of slight, yet distinguishable changes in the physical state of a system as derived from a machine learning predictive model is an effective technique. A feature vector, consisting of various sensor outputs, is coupled with a detection criteria to predict the binary state of ransomware present versus normal operation. An advantage of this approach is that previously unknown or zero-day version s of ransomware are vulnerable to this detection method since no apriori knowledge of the malware characteristics are required. Experiments are carried out with a variety of different system loads and with different encryption methods used during a ransomware attack. Two test systems were utilized with one having a relatively low amount of available sensor data and the other having a relatively high amount of available sensor data. The average time for attack detection in the "sensor-rich" system was 7.79 seconds with an average Matthews correlation coefficient of 0.8905 for binary system state predictions regardless of encryption method and system load. The model flagged all attacks tested.**

***Index Terms*—Ransomware Detection, Physical Sensor, Side Channel, Machine Learning**

I. Introduction

The global costs associated with ransomware are projected to exceed $20 billion dollars in 2021. Based on these projections ransomware will cost 57 times more in 2021 than it did in 2015 [1]. In the fall of 2020, during the COVID-19 pandemic, the Cybersecurity Infrastructure Security Agency (CISA), the Federal Bureau of Investigation (FBI) and U.S. Department of Health and Human Services issued a bulletin warning the health care industry of credible information they had obtained which indicated there would be increased ransomware attacks to hospitals and health care providers [2] [3]. Just prior to this bulletin the Duesseldorf University Hospital in Germany was forced to turn away a patient in need of urgent care due to a ransomware attack that had taken down 30 of their servers. The patient had to be re-routed to the nearest hospital over 30km away which resulting in her death [4]. This incident attracted world wide attention as it may have been the first instance of a cyber attack being directly responsible for a death [5]. The costs of ransomware are more often than not much higher due to operational downtime rather than the actual ransom that is being demanded. The costs of an attack associated with downtime is nearly six times higher now than it was in 2018 [6]. The proximate threat posed by ransomware, then, comes from downtime rather than data loss or data security. Even if the systems can be restored from backups the downtime for restoration can be economically disastrous or even life threatening when critical services are unavailable [7]. Colonial Pipeline, a company that transports more than 100 million gallons of gasoline daily, was the victim of a ransomware attack in May 2021 which forced a temporary shutdown of all operations. This attack is an example of how downtime in critical infrastructure is the consequence of a ransomware attack. Longer periods of downtime in critical infrastructure causes larger financial and operational losses in industries which are reliant upon the critical infrastructure. Critical infrastructure owned and operated by private sector companies are more likely to yield ransom payments as the company's projected losses due to downtown will more often than not far exceed the ransom being demanded. Unfortunately, over 85% of all critical infrastructure is owned and operated by the private sector. Brian Harrell, the former assistant secretary for infrastructure protection at the Department of Homeland Security, stated in an interview that he anticipates more attacks like the Colonial Pipeline attack to happen in the future. He further emphasized the need for rapid detection and recovery in critical infrastructure attacks when he stated "Attacks will happen, but how quick can you recover and restore critical services?" [8]. It has been suggested that ransomware may not always be about collecting money but instead a deliberate attack on a target's infrastructure and capabilities that can spread quickly. Ransomware attacks launched purely for destruction are effective due to how quickly they can take down entire networks of assets without the need to worry about being evasive [9]. Combating attackers who weaponize ransomware requires detecting potential attacks quickly without resource intensive and time consuming analysis. In support of this, we use the physical state of a system, as it is measured through existing physical sensors, to generate prediction models that quickly flag physical system states consistent with a ran-

978-1-6654-0285-9/21/$31.00 ©2021 IEEE

somware attack. Ideally this method would be effective alongside conventional methods of ransomware detection which are highly accurate but slower to complete analysis.

A. Physical Sensors

Most modern computer systems are comprised of sensors and associated processes that monitor the state of internal hardware components. These sensors continuously supply information that is communicated with other devices and subsystems for the intended purpose of ensuring that the system stays within specific operating specifications. If sensor data reveals that a system component is approaching a boundary of an operational specification, safety mechanisms are typically engaged to correct the internal environment so that system malfunctions can be prevented. Additionally, there are sensors that provide input to other subsystems such as internal power management units (PMU) to conserve power usage. Typically, computer system components are designed to be compact in size through the use of transistors with feature sizing in the nanometer scale. As a direct result, whenever computations become more complex, more stress is placed on a computer's hardware components. This increased stress occurs because a larger number of transistors are simultaneously switching in a circuit that correspondingly causes an increase in dynamic power consumption and results in more heat dissipation during heavy computational activity. Thus, monitoring the side channels of a system with embedded sensors that measure parameters such as temperature, power consumption, and battery voltage levels can give insight into the type of processing that is underway on a computer at a given time. Therefore, sensor data streams serve as side channels through which periodic observations can indicate when resource-intensive tasks, such as extensive file system I/O and encryption, are occurring. Because the silent phase of ransomware utilizes significant amounts of file system activity in combination with encryption, characteristic patterns present within a computer's sensor data may result in trends that are indicative of a ransomware attack.

A significant advantage of this approach, as compared to other side channel methods, is that the sensors and a means for querying them are natively provided. Thus there are fewer concerns in deploying and accessing sensors for the purpose of side channel exploitation. Furthermore, the trend has been that an increasingly diverse number of sensors are provided as integral components in modern computing devices. For example, a typical smart phone has many embedded sensors that could be used to support security applications including power monitors, accelerometers, ambient light sensors, antennas (including GPS receivers), fingerprint scanners, barometers, cameras, touchpad pressure sensors, and others. Even rack-mounted industrial servers contain a significant number of sensors that measure subsystem power consumption, temperature, and other environmental factors. All of these deployed sensors in modern computing devices provide a rich set of data sources that may be used to provide internal side-channel information for the environment in which a computing device is operating. Sensors have been used in other security-related applications in the past. As an example, in [10], sensors present in mobile computing devices have been used to provide a user demographic classification capability for mobile devices with embedded touchscreens [11].

B. Contributions

Current ransomware detection techniques generally fall into three different categories. Detection by signature, which detects various ransomware attacks by identifying specific parts of their code. This method is very effective and fast for detection of older attacks but does not offer protection against new or even modified attacks. Additionally, this method requires constant updates in order to detect new strains of ransomware. The second method is detection by abnormal traffic, which examines and analyzes traffic data, such as volume and timestamps, in order to find abnormalities. This method does not require prior knowledge of an attack, such as a signature, but the side effect is a high false positive rate. The high false positive rate can lead to losses due to downtime as accounts must be blocked when ransomware is believed to be detected. The third method is detection by data behavior, which monitors file execution in order to identify abnormal and potentially malicious behavior. This method does not require signatures and does not need to block accounts when malicious activity is believed to be detected. However, the potentially malicious file execution must be carried out for a long enough period of time that analysis can be carried out and malicious activity can be confirmed. Although reliable in ransomware detection this method often leads to some percentage of the system being encrypted by the attacker during analysis [12].

Instead of monitoring file system attributes, the victim host system behavior is monitored by taking advantage of the increasingly large number of onboard sensors. In this sense, this new method uses a physical side channel approach where the victim's files are not directly monitored, rather the behavior of the victim machine is monitored and onboard sensor provided data is used as side channel information that can indicate when an encryption operation is occurring. This monitoring can be accomplished through a background process that is loaded at boot time and thus continuously monitors the system for suspicious behavior. Once this suspicious behavior is detected, the user can be alerted and the suspicious processes can be suspended. The central difference between this approach and other previous approaches is that this approach uses secondary effects to detect the presence of malware rather than a direct effect, such as measuring increases in file entropy [13].

It is proposed that this new sensor-based detection methodology be used to complement more traditional signature-based approaches that are intended to prevent attack vector penetration. In contrast to prevention of attack vector penetration, the technique described here is designed to detect the presence of ransomware when penetration has been achieved. The side channel-based or sensor-based approach has an advantage in comparison to antivirus or IDS systems in that zero-day versions of ransomware can be detected since previously

captured malware signatures are not required. Furthermore, it is not necessary to monitor individual files and calculate entropy or other metrics that must be continually re-computed and compared with one another as is the case in the solution provided in [13].

An experimental prototype system based on sensor monitoring has been implemented and tested through the use of a variety of scenarios where simulated ransomware begins silently encrypting victim files. To evaluate this method, several different encryption methods were used from the Python Cryptography Toolkit that have been reported to be commonly used by adversaries during the development of ransomware [14].

II. Training Prediction Models

A. Test Systems

To train and evaluate the methodology, two computer systems were chosen: (1) an older laptop (Hewlett Packard ENVY m4-1015dx) with fewer onboard sensors as compared to the size of a sensor suite found in more modern systems, and (2) a more modern system (MacBook Air 13-Inch Mid 2013) that had several times the amount of sensors as compared to the older system. Access to the sensor output data was achieved through queries via the native operating systems and did not require the development of lower-level software. Because many third-party applications are developed that depend upon access to onboard sensor data, the means to access the sensors are generally available in most operating systems.

The Hewlett Packard m4-1015dx is the first system used in this experiment. When using Open Hardware Monitor the HP ENVY laptop returns 22 sensor values. The types of sensor values which are returned include clock, data, load, power, and temperature. While clock, data, and load are reported with the physical sensors they are considered ”probes” which are relaying performance metrics provided by the CPU [15]. Thus the HP ENVY laptop only contains nine sensor values which are of direct use in creating the prediction models in this experiment. It is important to note that of the nine sensors that are used in the evaluation of this system, eight of them directly measure aspects of the CPU thereby causing the predictive model to nearly entirely depend on CPU behavior. Therefore, the HP ENVY laptop can be seen as having a relatively “sensor-poor” environment for the evaluation of the sensor-based ransomware detection method.

The MacBook Air 13-inch mid-2013 model is the second system used in this experiment. When using Hardware Monitor for Mac the MacBook Air laptop returns 67 sensor values which are able to be used in this experiment. Unlike the Hewlett Packard ENVY m4-1015dx, only 17 of the available 67 MacBook sensors directly measure CPU activity. Therefore, the availability of sensors that measure other system components and parameters allow for the development of predictive models that are more holistic to the system. The MacBook Air laptop thus provides a more inclusive sensor-suite with regard to monitoring the entire system and thus enables our methodology to have access to a richer set of side channel data. Therefore, the MacBook Air 13-inch mid-2013 can be seen as having a relatively “sensor-rich” environment for the evaluation of the sensor-based ransomware detection method.

B. Simulating Ransomware Attacks

Physical sensor-based attack detection attempts to find a pattern in the physical state of a system that can identify the presence of an attack (many times referred to as a fingerprint). There are numerous variations of ransomware. Thus, training on the unique operation of a single variant would likely not be effective for detecting this general class of malware attacks. Ideally, the training data should be collected using a process that implements the most common and basic elements that are present in a variety of different ransomware attacks. The variation in ransomware attacks are usually due to the methods of infiltration, encryption, file system searching, file targeting, and the infiltration of additional attached systems. It is assumed that any variation in the infiltration method does not affect the detection process. Thus, the proposed method is designed to detect ransomware that has recently infiltrated a system. Additionally, the method an attacker uses for further propagating their attack to additional systems attached to the host is not a part of the attack that is considered in the detection method. For these reasons, the focus of the proposed method is on the use of sensor data that shows characteristic patterns with respect to the type of encryption used, the process of iterating through the directories in a host’s file system, and the targeting of files for encryption.

In this experiment a script was written that simulates the active encryption portion of a ransomware attack. The attack can be performed using many different choices of parameters based on criteria from all three of the previously noted areas of potential attack variation. The type of encryption is chosen from four different variations of AES encryption or a simple XOR encryption. The XOR encryption was included to simulate the behavior of more lightweight methods of encryption. The script accesses a host’s file system based on the particular directories that it finds and searches through. To help avoid detection, the script uses intentional, random delays while file searching is occurring. The simulation script selects one or more starting points in the file system that are most likely to contain a host’s personal and sensitive data. The script recursively traverses the directory and sub-directories of a starting point checking for files with the targeted file extensions. Targeted victim files are identified via the use of a list of file extensions that were historically targeted by multiple high-profile ransomware attacks. The script creates an encrypted version of the victim file, deletes the unencrypted version, and renames the encrypted file with the original unencrypted target file’s name. Prior to operation the script is either set to run continuously or to wait a different random amount of time between 1 and 60 seconds each time after encrypting a target file.

Randomly selecting parameters in the script creates a simulation of the active encryption phase of a ransomware attack variant. Although the methods used to infiltrate and propagate

a ransomware attack vary widely, the methods of actually finding and encrypting as many of the host's personal files as possible is more constricted. Repeated testing with the script was accomplished and intended to offer attack simulations that differ enough in their approach such that the collective training data is a generalization of the active encryption phase of a variety of different ransomware attacks.

C. Collecting Sensor Data

In implementation of our prototype detection system, sensor data is the input required to make predictions about the binary state of a system using machine learning algorithms. It is important that the methodology utilized in procuring sensor data is both quick and reliable. Many tools exist which allow users to simply monitor sensor data in a graphical user interface; however our prototype required the automated retrieval and parsing of sensor data at specific intervals in time. Both tools utilized in this experiment can be used to either write data to a file for future analysis or provide a feature vector of real-time sensor data to a prediction model. Writing data to a file allows for the direct comparison of machine learning algorithms in as far as how they would have predicted the state of the system given the same input data. Prior to reading any sensor values, a command can be issued that returns a comma separated list of sensor names and categories that can then act as a header for future sensor data. In order to access the sensor data a command is issued to the command line that returns a string of comma separated sensor values in the same order as the header string.

D. Building Training Data Sets

The training data needs to represent periods of operation in a system both while under attack from ransomware and while not under attack. The ransomware was implemented as a daemon that ran in two hour blocks with each block having a predefined encryption method for all attacks. During the two-hour collection period, the system is either in a state of "normal operation" or "under attack". Normal operation is any time the system is not being attacked with the simulated ransomware script while "under attack" is any time that the simulated ransomware script is active. For each simulated ransomware attack, the method of attack is different by randomly selecting one or more starting positions in the file system for recursive searches. Additionally, the script randomly determines if the attack will use detection avoidance through randomizing the timing of directory access and file encryption. Before the two hour training block initiates, the time required to encrypt all target directories with the selected encryption method is measured. The script then begins in a state of "normal operation" for the amount of time that was previously measured before launching the encryption attack. During the time that the script is logging the sensor data, it is also adding system state labels so that a supervised machine learning model can be utilized. After the simulated attack is completed the logging stops and system is decrypted. There is a waiting period of two minutes for the system to return to a state of normal operation before logging continues and a new cycle is begun. Each of the five encryption methods is run in 12 different two-hour test blocks. Each encryption method is then repeated after additional CPU load activity is initiated so that the training data can contain examples of sensor data with various amounts of background activity present in the runtime profiles. Test loads of 0%, 25%, 50%, 75%, and close to 100% system activity are each applied to the system in three separate two hour testing blocks for each encryption method. The complete training data set consists of 24 hours of regular training and 30 hours of simulated load training for each of the five encryption methods. The sensor readings and the CPU load of the system are polled every second during the training periods and labeled with a timestamp and either "normal operation" or "under attack." Data from each two-hour training block is collected in separate CSV files.

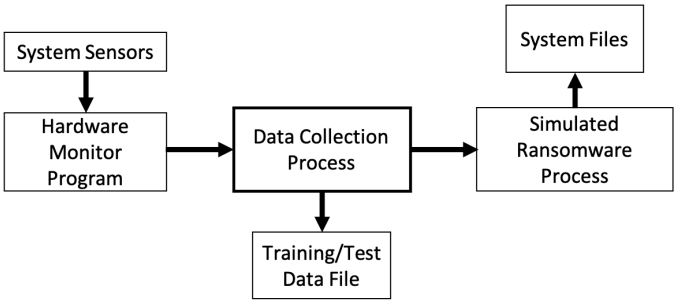

Fig. 1. Experimental Data Collection Diagram

E. Training Prediction Models

The collected training data was used to to create several different prediction models available in the Python Scikit-learn library [16]. In total, models were generated using 12 different machine learning algorithms that comprised 280 different combinations of parameter settings. Numerous combinations of methods were investigated including: data scaling method, feature selection method, prediction method, and moving average method (i.e., smoothing of the output predictions). The process of chaining methods together is often referred to as classification "pipelining" [17], [18]. For test deployment, each model was stored in its own Python pickle file for use in various online tests.

The prediction method includes two options, binary classification and ordinal regression. Binary classification models are trained with a dependent vector of binary values indicating true for "under attack" and false for "normal operation." Ordinal regression models are trained with a dependent vector consisting of values with seven being the highest and representing the highest likelihood of being "under attack" and zero being the lowest and representing the highest likelihood of being in "normal operation." Whenever the predicted value is greater than three, a prediction of "under attack" is made. Conversely, whenever the predicted value is less than or equal to three a prediction of "normal operation" is declared.

The data scaling methods investigated include four options: feature standardization, data normalization, feature min-max scaling, and no scaling.

The dimensionality reduction method includes seven variations including the use of PCA, feature selection, and no dimensionality reduction. Several principal component analysis data reductions are performed, using the cumulative explained variance to guide the number of components chosen. Three variations of PCA dimensionality reduction are investigated based upon if the components maintain at least 70%, 80%, or 90% of the total variance. Feature selection is performed such that only the top 50%, 70%, or 90% of features are selected using F-tests from the Scikit-learn library to analyze variance in the training data [16].

Once the features are scaled and selected, they are used to train one of twelve different classification or clustering techniques. For clustering techniques, the methods are used as unsupervised classification methods where each cluster is assigned as a particular class and the total number of clusters equals the total number of classes. Parametric and non-parametric classification methods are employed, as well as supervised and unsupervised methods. The following algorithms are investigated:

- **Parametric Trees Methods**: Decision Tree [19], Random Forest [20], Extremely Randomized Trees [21], One-versus-one Tree Ensemble, One-versus-rest Tree Ensemble
- **Parametric Methods**: Linear Regression, Logistic Regression, Support Vector Machine [22], Naive Bayes, Two Layer Neural Network
- **Non-parametric Method**: K-Nearest Neighbors
- **Unsupervised Method**: K-Means

Each classifier is trained to classify every second of data from the sensors streams, resulting in a binary stream of predictions. This output stream is smoothed using various moving average methods. The moving average method consists of five options including simple moving average with window sizes of two and four, weighted moving average with window sizes of two and four, and no moving average.

III. Testing Prediction Models

A. Building Test Data Sets

Test data was collected in one-hour test blocks in which a single ransomware attack would occur at a random time. The ransomware attack parameters were selected randomly for each attack. During training, the amount of time the system was recorded to be in the state of "normal operation" was similar to the amount of time the system was recorded in the state of "under attack" in order to create a balanced training set. However, during testing the use of a balanced set is not appropriate because most of the time a system would be in the "normal operation" state. Therefore, the testing was conducted in a manner such that there was a disproportionate amount of time the system was in the "normal operation" state. The purpose of this approach was to determine how many times the attacks would be correctly detected, the frequency of false positive predictions, and most importantly; how fast attacks were predicted. Each of the five encryption methods underwent 24 of the one-hour test blocks. Afterward, each of the five encryption methods was tested with additional CPU loads of 0%, 25%. and 50%. Each of the encryption methods was tested in six different one-hour time blocks with each of the three additional CPU load levels. During each one-hour test block of the additional CPU load testing, there was a single ransomware attack that occurred at a random time. During the collection of data for testing the method, the total system CPU load is recorded each time the sensor data is polled. This CPU load data is not recorded during training and not used as a feature in the prediction models.

The initial phase of test data collection represents a system which is sitting unused and only running regular background processes. This phase of testing was designed to determine the performance of the proposed method in favorable conditions for detection. Ideally the models should have a low number of false positive attack predictions while being able to quickly determine when the system is under attack. The second phase of test data collection represents a system which has different levels of user activity in addition to the regular background processes from the initial phase of testing. This phase of testing was designed to determine if the prediction models are able to perform in a more realistic scenario in which the physical state of the machine is more dynamic.

B. Predicting System States

This experiment tests twelve different machine learning algorithms. The test data is collected and stored in CSV files with labels that indicate what the actual state of the system was each time the sensor data was probed. The models for each of the different algorithms are all used to make predictions with the same data set. Any of the models which use sliding windows for their final prediction (such as when a moving average is used) have the initial prediction vector iterated through in order to generate a final prediction vector. The predictions are generated using the Python Scikit-learn library by loading the previously constructed model from it's saved file. Models which use data preprocessing (such as scaling or feature selection) also have the appropriate data structures loaded. The performance prediction is determined by comparing the actual system state vector from the test data to the final prediction vector generated from the prediction model.

C. Performance Evaluation Methodology

1) Binary Classification Evaluation: In this experiment machine learning algorithms are each used to make a prediction about the binary ransomware attack status of a system. The final prediction vector of a model and the actual system state vector are compared to obtain the distribution of the four types of binary classification. The distribution of the four classifications was used to compute five metrics which

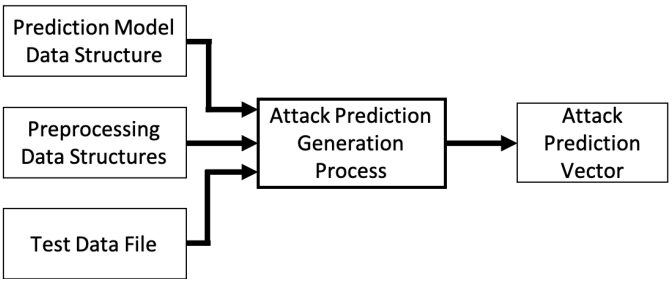

Fig. 2. Prediction Vector Generation Diagram

offer more insight into prediction performance: sensitivity, precision, specificity, fallout, and accuracy.

2) Matthews Correlation Coefficient (MCC): The Matthews correlation coefficient (MCC) takes into account true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are of considerably different sizes such as the data in this experiment.

The MCC is a correlation coefficient between the observed and predicted binary classifications. Values range between -1 and +1. A coefficient of +1 represents a perfect predictor, 0 represents the same as random prediction, and -1 indicates total disagreement.

3) Rate of Attack Recognition (RAR): When testing, an actual attack time series exists which defines the time periods during which there is an attack. Attack instances starts when a labeled system state of "normal operation" transitions to a labeled state of "under attack". Conversely, an attack instance ends when a labeled system state of "under attack" transitions to a labeled state of "normal operation". If a positive prediction exists during the period of time representing an attack instance then the attack is considered detected. Ideally there should exist at least one positive prediction during each attack instance which would result in a 1.0 or perfect rate of attack recognition.

4) Mean Time to Attack Recognition (MTAR): The initial labeled system state of "under attack" for each attack instance in the actual attack time series represents the time interval at which the attack began. The first instance of a positive prediction in the corresponding prediction time series at or after this initial "under attack" state and before the next "normal operation" system state represents the initial attack recognition. Ideally the initial "under attack" state itself would be a positive prediction, but in practice it is more likely that the sensors would need a small amount of time to reach the values at which positive prediction occurs. This metric counts the number of time intervals until the first positive prediction is recorded for every attack instance which was successfully recognized. Afterwards all values are averaged to determine the mean time to attack recognition.

IV. Experimental Results

A. Optimal Parameters for Algorithms

Stratified ten-fold cross validation is performed for each algorithm using the balanced training dataset in order to determine how well various test configurations are likely to perform. There exist a total of 280 different combinations of the test parameters that are all tested individually for each algorithm. The test parameters include the prediction method, data scaling method, dimensionality reduction method, and moving average method. For each combination the training data is separated into ten equal-sized subsets. Ten different MCC values are found by ten different tests in which each subset acts as the sole training data once and is part of the test data nine times. The ten MCC values are averaged to obtain the average MCC of the algorithm for a specific test parameter combination. The 280 average MCC values are ordered from greatest to least with the highest value belonging to the test parameter combination which is most likely to result in the highest performance during testing. The highest performing test parameter combination is used for the appropriate algorithm for the duration of testing instance.

TABLE I
Accuracy Analysis MTAR Results

HP ENVY m4-1015dx		MacBook Air Mid 2013	
Algorithm	**Avg MTAR**	**Algorithm**	**Avg MTAR**
Decision Tree	48.10	MLP	0.3891
K-Means	52.01	One-V-Rest	0.4044
Extra Tree	53.03	SVC	0.4050
Rand Forest	59.60	Decision Tree	0.4113
Log Reg	61.99	Rand Forest	0.4539
N Bayes	65.08	Extra Tree	0.4794
KNN	66.41	Log Reg	0.4924
MLP	69.47	Lin Reg	0.5600
One-V-One	69.81	One-V-One	0.5700
One-V-Rest	69.81	KNN	0.5812
SVC	69.81	N Bayes	1.511
Lin Reg	70.97	K-Means	54.31

B. Prediction Evaluation on Test Data

The test data analysis is performed for each of the twelve machine learning algorithms. There exists five different encryption modes with each having separate training and testing data. Every combination of encryption modes of every size from one to five has a prediction model trained, and the same combination of testing data is used to assess how well the model performs when making predictions for data it has been trained with. The prediction models from the sensor poor system all scored below 0.5 while all of the prediction models from the sensor rich system, except K-means clustering, score above 0.95. The prediction models used on the sensor poor system, with the exception of K-means clustering, were all able to detect every ransomware attack while maintaining low false positive rates. The accuracy scores for these prediction models were all above 0.9.

The majority of the prediction models for the sensor poor system took over one minute to make a prediction of "under attack." The sensor rich system is able use its prediction models to make the first "under attack" prediction in less than one second for all algorithms except two. The ten prediction models that had a mean time for attack recognition of less than one second were able to detect all of the ransomware attacks

TABLE II
ROBUSTNESS ANALYSIS MCC RESULTS

HP ENVY m4-1015dx		MacBook Air Mid 2013	
Algorithm	**Avg MCC**	**Algorithm**	**Avg MCC**
Log Reg	0.4952	Extra Tree	0.9980
One-V-One	0.4885	Log Reg	0.9980
One-V-Rest	0.4885	Rand Forest	0.9979
SVC	0.4885	KNN	0.9978
Extra Tree	0.4864	Lin Reg	0.9960
MLP	0.4747	One-V-Rest	0.9958
Rand Forest	0.4707	SVC	0.9958
KNN	0.4526	N Bayes	0.9839
Lin Reg	0.4433	Decision Tree	0.9559
N Bayes	0.4125	MLP	0.9378
Decision Tree	0.4013	One-V-One	0.9040
K-Means	0.1363	K-Means	0.7681

during the accuracy testing with an average MCC score of 0.989.

C. Robustness of Prediction Ability

The robustness analysis is performed for each of the twelve machine learning algorithms. Robustness, in the context of this analysis, is used to refer to the ability of an algorithm to perform well given encryption modes it has been trained to predict as well as encryption modes it has not been directly trained to predict. This analysis will convey whether models that have not been explicitly trained to detect certain encryption modes can still detect them with relative success. The algorithm with the highest MCC average in both the previous analysis and robustness analysis is selected and used in the remainder of the tests.

D. Effect of Training Time on Performance

The effect of training time on the prediction performance of the model was tested with the highest performing algorithm for each test system. Based on the outcome of the accuracy and robustness testing Logistic Regression was selected for further evaluation with the sensor poor system as it had the highest average MCC score during both tests. Likewise, the Extra Tree model was selected for further evaluation with the sensor rich system as it also had the highest average MCC score during both tests. Twelve prediction models where created for each system from training data collected over 24 hours. The first model was trained with only the first two hours of the data and subsequent models were trained with an additional two hours of data until all 24 hours worth of training data were utilized. The performance of the sensor poor system was highest after 4 hours of training with additional training causing the performance to decrease until leveling off around 18 hours. The sensor rich system showed very little change in performance after 4 hours with the MCC score only increasing to a maximum of 99.85% and the mean time to attack recognition only dropping by 0.1166 seconds at its peak performance after 20 hours of training. While the results of this experiment do indicate that there is small increases in performance with additional training time the models with the least amount of training time are still able to perform at a comparable level. This conclusion seems especially true in the sensor rich system which seems to indicate that systems with a large array of sensors are able to be effectively trained quickly.

E. Effect of User System Load on Performance

The previous testing demonstrated the viability and potential of the new ransomware detection method under favorable system conditions. Testing with additional system loads represent a more complex and realistic situation which in turn requires a more complex prediction model. Figure 3 shows the more complex model used for detecting ransomware when unknown additional system loads are present. This method uses a collection of five prediction models which implement the the same machine learning algorithms selected for each of the two systems during the training time testing which are trained with 0%, 25%, 50%, 75%, and 100% additional system loads present. Each of the five models provides a confidence score for a state of ”normal operation” and a state of ”under attack” from the same input vector of sensor data. In order to determine which of the five models to utilize the ransomware attack process is run with only regular background processes and monitored to determine a value to use as the additional CPU load the processes is likely to introduce on the system. During testing the current system CPU load is used to determine which of the models to use for a ”normal operation” confidence score and which of the five models to use for an ”under attack” confidence score. Given that no ransomware is attacking the system the behavior of the system would be best represented by the model trained with an additional CPU load closest to the current CPU load resulting in a higher ”normal operation” confidence score. However, given that there is currently ransomware attacking the system the current system CPU load minus the CPU load which the ransomware process is likely to place on the system would be used to select the prediction model to supply an ”under attack” prediction. The two confidence scores are compared and the higher score is used as the final system state prediction for the ensemble model. The ensemble model is first tested with no additional CPU load present on the system, similar to the previously conducted tests. However, unlike the previous tests which always used the prediction model generated with no additional system load this more complex model has to select a model which may not always be the right one. The performance of the model decreased due to the requirement of determining which models to use, but in the case of the sensor rich system there was still MCC scores above 90%. The ensemble prediction model was more effective with rising additional unknown CPU loads. All MCC scores for the sensor rich system were above 90% with additional CPU loads of 25% and 50%. Even with the additional requirement of detecting ransomware attacks with unknown additional system loads the new ransomware detection method was able to detect all attack instances with most mean time to attack recognition results just over 7 seconds on the system with a large array of sensors.

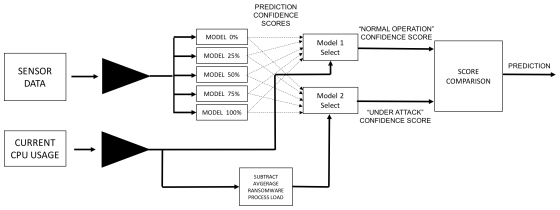

Fig. 3. Simulated Load Analysis Ensemble Predictive Model

V. Conclusion

System side channel data has commonly been used to attack systems in which an attacker has knowledge of how a process is physically carried out on a machine. Instead of attacking a system with side channel data we show that it is possible to defend a system by training machine learning algorithms to detect intricate patterns in the physical behavior of a system which correlate to a malicious process. Specifically, we show that ransomware attacks can be effectively detected in a rapid manner before significant amounts of data are encrypted during an attack. Once the machine learning algorithms have been trained and a predictive model has been generated, predictions about the state of a system are calculated quickly and with a low computational overhead.

Perhaps the most important aspect of this experiment is the speed in which an attack may be detected. Detection speed is a very important security concern in attacks on critical infrastructure as the paramount concern is reducing any downtime. Rapid detection could mean a significant reduction in downtime from an attack as less of the system would be corrupted. In experimental testing, the highest performing system had an average time to attack recognition which was as little as 1.3 seconds and never exceeded 13 seconds in even the lowest performing models. However, it has been found that even the best predictive models generate some false positive predictions. For this reason it is believed that the most effective method for implementing this technique would occur when more detailed analysis follows the indication of a positive prediction. When the quick acting predictive model indicates a positive prediction some degree of preventative measures would be taken to slow or stop the ransomware process from inflicting further damage. These preventative measure could include stopping the system for more in depth analysis in the most sensitive environments, briefly enforcing stricter firewall rules to defend against further attacks on network entities, or even simply notifying the user to potential risk.

While this experiment focused on detecting ransomware attacks with side channel data it may be possible to apply this method in a more broad manner to detect different processes which have predictable behavior. The predictive models only require training data which demonstrates the behavior of the system in a "normal" state and also demonstrates the behavior of the system when the target process is active. This idea opens up the possibility for advanced sensor based monitoring of a system which could include sensors that have been added to the system for the sole purpose of augmenting its ability to physically model a computational process.

References

[1] Safeatlast, ""22 Shocking Ransomware Statistics for Cybersecurity in 2021"," https://safeatlast.co/blog/ransomware-statistics/#gref.

[2] Bracken B, ""What's Next for Ransomware in 2021?"," https://threatpost.com/ransomware-getting-ahead-inevitable-attack/162655/, Dec 2020.

[3] Cybersecurity and Infrastructure Security Agency, ""Ransomware Activity Targeting the Healthcare and Public Health Sector"," https://us-cert.cisa.gov/ncas/alerts/aa20-302a, Oct 2020.

[4] Cimpanu C, ""First Death Reported Following a Ransomware Attack on a German Hospital"," https://www.zdnet.com/article/first-death-reported-following-a-ransomware-attack-on-a-german-hospital/, 09 2020.

[5] O'Neill P, ""Ransomware Did Not Kill a German Hospital Patient"," https://www.technologyreview.com/2020/11/12/1012015/ransomware-did-not-kill-a-german-hospital-patient/, 11 2020.

[6] Gately E, ""Cost of Downtime from Ransomware Nearly Doubles this Year"," https://www.channelfutures.com/mssp-insider/cost-of-downtime-from-ransomware-nearly-doubles-this-year, 11 2020.

[7] A. Ng. (2019) Ransomware froze more cities in 2019. next year is a toss-up. [Online]. Available: https://www.cnet.com/news/ransomware-devastated-cities-in-2019-officials-hope-to-top-a-re.peat-in-2020

[8] Cohen Z, Sands G, Egan M, ""Here's What We Know About the Colonial Pipeline Ransomware Attack"," https://www.cnn.com/2021/05/10/politics/colonial-ransomware-attack-explainer/index.html, 5 2021.

[9] J. Fuller. (2017) Cyber-security emergency: Weaponized ransomware attack spreading like wildfire. [Online]. Available: https://www.cnet.com/news/ransomware-devastated-cities-in-2019-officials-hope-to-top-a-re.peat-in-2020

[10] Alharbi A, Thornton M, ""Demographic Group Classification of Smart Device Users"," *IEEE Int. Conf. on Machine Learning and Applications*, pp. 481–486, Dec 2015.

[11] M. Taylor, K. Smith, and M. Thornton, "Sensor-based Ransomware Detection," in *Future Technologies Conference*, 2017, pp. 794–801.

[12] Spin Technology, ""Ransomware Detection Techniques: Which One Is The Best?"," https://spinbackup.com/blog/ransomware-detection-techniques-which-one-is-the-best/, 1 2021.

[13] Scaife N, Carter H, Traynor P, Butler K, ""Cryptolock (and Drop It): Stopping Ransomware Attacks on User Data"," *IEEE Int. Conf. on Dist. Computing Systems*, June 2016.

[14] Bonderud D, ""CryPy Ransomware Slithers Onto PCs With Unique, Python-Based Encryption"," https://securityintelligence.com/news/crypy-ransomware-slithers-onto-pcs-unique-python-based-encryption/, 2016, note: This is an electronic document. Date of publication: [October 17, 2016];.

[15] Bresink M, ""Hardware Monitor: Reference Manual"," https://www.bresink.com/osx/216202/Docs-en/index.html, 2017, note: This is an electronic document. Date of publication: [2017];.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python ," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[17] M. Chang, Q. Do, and D. Roth, "Multilingual dependency parsing: A pipeline approach," *AMSTERDAM STUDIES IN THE THEORY AND HISTORY OF LINGUISTIC SCIENCE SERIES 4*, vol. 292, p. 55, 2007.

[18] J. R. Finkel, C. D. Manning, and A. Y. Ng, "Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines," in *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2006, pp. 618–626.

[19] L. Breiman, *Classification and regression trees*. Routledge, 2017.

[20] ——, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[21] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, Apr 2006. [Online]. Available: https://doi.org/10.1007/s10994-006-6226-1

[22] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *Advances in neural information processing systems*, 1997, pp. 155–161.