**Abstract**

Ransomware is a type of malware that infects a device and spreads rapidly to encrypt files or to lock the device, restricting the users from accessing them. The attackers then demand payment, in exchange for providing the decryption key or unlocking the device. Additionally, they may steal critical files from the victim’s machine and threaten to publish them online in case the recovery of the system and the files is possible. Lately ransomware have grown to become a serious threat to many organizations and individuals, resulting in high financial, legal, and sometimes social costs. However, the current approaches for detecting malware and specifically ransomware are not effective, especially when ransomware since must be detected as early as possible. In this dissertation I propose a method to detect ransomware using a combination of features extracted from the phases before and after encryption has occurred, by training a machine learning-based anomaly detection model. To train the model I monitored a test system for several days, to baseline a normal system behavior and I deployed a self-developed ransomware along with the model on the system. The results show that the model can detect ransomware activity most notably before encryption has occurred and supplementarily during encryption and exfiltration.

**INTRODUCTION**

**WHAT ARE RANSOMWARE:**

As mentioned before, ransomware is a type of malicious software, or malware that threatens to permanently block users from accessing their files or systems and may additionally publish the victim’s data unless a ransom is paid, in what is called “double extortion”. Ransomware have generally been around since 1989, when the first incident classified as a ransomware was documented. With the passage of time, ransomware attacks have evolved, and many families and strains have been created.

Overall, ransomware can be split into two broad categories based on the way they operate: Crypto-Ransomware or Encryptors and Locker Ransomware.

*Crypto-Ransomware* or *Encryptors* are the most common types of ransomware and are designed to encrypt data, information and files on the victim’s device. Although the victims would be able to see the encrypted assets and use the device, they would not be able to access the data since the data are encrypted. After the encryption, the ransomware notifies the users that their data have been encrypted and prompts them to make a payment until a specific deadline to retrieve them.

*Locker Ransomware* work similarly to encryptors, but with the difference that they do not encrypt, destroy, or alter the users’ data, but they lock the users out of their systems. Usually, the victim is allowed to interact with the system minimally; he can view only the lock screen or interact only with the screen displaying the ransom demand. Similarly, to encryptors, a timer with a deadline may be displayed to pressure the victim into paying.

Another notable mention is *Scareware.* Scareware, as the name suggests, works by scaring the victim into paying the attackers an amount of money. Victims may be bombarded by messages and pop-ups saying that their systems have been infected by a ransomware, that their files have been encrypted or that they have been lost access to their systems, and that the only way to regain access is to pay an amount of money. In reality though, this type of malware does not actually affect the machine and counts only on the fear and panic of the user to pay the ransom.

The severity of ransomware though is better understood if we even shortly examine their impact.

**CONSEQUENCES OF RANSOMWARE**

Ransomware tend to cause catastrophic consequences upon the victims. Individuals infected with ransomware may temporarily or permanently denied access to the systems, where important and sensitive data may be stored. Financial information, personal documents, university assignments, and other assets of the individual may become inaccessible to their owners or may be published, until the ransom is paid. Furthermore, if the ransom is paid the individual can have major financial costs, as the ransom is usually high.

The more severe consequences though are related to businesses and organizations. While ransomware can of course target both individuals and organizations, they tend to target mostly organizations, as the risks for them are higher and much more critical. Additionally, attackers can request more money from the organization than from individuals.

For all those reasons, detection of ransomware attacks and more importantly early detection is crucial. The consequences of an attack can lead to catastrophic consequences to a variety of direct and indirect stakeholders and detection of the attack will mitigate those risks or alleviate its impact.

**HOW RANSOMWARE WORK**

Ransomware, as any kind of malware must take certain steps to operate. They are typically spread to the victims via phishing emails, malicious downloads, or by exploiting vulnerabilities in the system and deceiving the victim into downloading them and running the malicious payload. To fully comprehend the way ransomware operate it is useful to group and their map their actions to a framework similar to the one presented by Stamper, who used the Cyber Kill Chain as a starting point.

**Reconnaissance**

First and foremost, the attackers must conduct reconnaissance activities to find out information about the victims as ransomware are usually targeted and do not rely purely on the random spread of the malware. Although some malware exhibit worm behavior to propagate, they usually target and spread to systems that satisfy a certain number of criteria that match the desired victims. For example, they may have a list of IPs to or not to attack, or systems that reside into specific regions as was done by Sodinokibi, a ransomware that avoided attacking computers from countries that were formerly part of the USSR <https://www.acronis.com/en-gb/blog/posts/sodinokibi-ransomware/>, by checking the locale settings. In this stage, ransomware may also conduct network scans and view the running processes to uncover any useful information.

**Weaponization**

When a target is identified, the attackers craft the attack vectors that will be used to conduct the attack. At this point, they craft specialized emails that will be sent to the targets and the seemingly benign, but in reality, malicious links and attachments that will be contained in the emails. They can also upload their malicious files to websites, so the users can download them following a supply chain compromise or phishing attack.

**Delivery**

Ransomware are typically delivered to the victims via Phishing emails, when a user clicks on the crafted malicious links or attachments, or when a user is redirected to websites containing the malicious files in a drive-by compromise. For example, Ryuk uses leverages phishing campaigns which redirect unaware users to documents hosted in lefitimate hosting services, such as Google Drive. Additionally, they may be dropped by other malware, if the victim has been infected by another malware cooperating with the ransomware. Furthermore, the delivery can be conducted by Supply Chain attacks, where a trusted third party is compromised and used to distribute malware.

**Exploitation and Privilege Escalation**

Many of the actions that ransomware need to take (deletion of shadow copies, halting of certain processes and utilities, interaction with the Windows registry, access to specific directories and files) require elevated privileges. To get elevated privileges, the malware may exploit vulnerabilities, such as zero-day vulnerabilities, may use code injection techniques to inject malicious code into normal elevated processes, or if exploitation is unsuccessful, they may directly request from the user to give elevated privileges to the process using the Windows UAC prompt. While vulnerability exploitation is the most preferred way to gain elevated privileges, ransomware such as Sodin, uses UAC prompts to request elevated privileges, if exploitation is not possible.

**Installation**

Ensuring that the malware has sufficient privileges to perform its actions, it unpacks itself and decrypts its payload so that its files and are accessible to be executed and used for the attack. Then, to ensure that only one instance of the malware is active on the victim’s machine it may use a mutex and check. Next, the ransomware check if the machine is included in a whitelist, that is machines that should not be attacked, as explained in the Reconnaissance phase. If the machine is included in the whitelist, then the attack is aborted.

Subsequently, ransomware tend to terminate processes that may detect malicious activity, inhibit file encryption and file access and help recovery of the system. Antivirus software, and Database Management Systems (DBMSs) such as mysql are terminated, as they might detect the ransomware or prevent it from accessing and encrypting files on the device.

As attackers want to be able to gain a strong foothold into the device and remain there, the ransomware tries to gain *persistence*. To do that, it is common that ransomware modify the Windows Registry to add themselves as an AutoRun program. For example, this behavior is exhibited by Ryuk and Wannacry.

**Command and Control**

As with many other malware families ransomware may utilize a Command and Control server to gather information, coordinate the attack, update the malware and exfiltrate data in case of a double extortion attack. At this stage, the victim establishes a connection with the C&C server(s), which may be done via custom protocols, encrypted channels, the TOR network, or may blend with legitimate traffic, for example by using HTTPS to transmit all the necessary information in an encrypted fashion. In the early stages of the attack, the ransomware may send to the C&C important information about the victim’s device, such as OS, Machine Name, basic Network topology and other information that will help the proper configuration and success of the attack and also the propagation of the malware to other systems. Moreover, the C&C server may transmit the encryption keys that will be used for the encryption of the files, or the encryption of the key that will be used to encrypt the files, if hybrid encryption is used.

**Self-Propagation**

While self-propagation is not a necessary step to conclude the attack ransomware that want to maximize the number of victims may be designed to spread automatically to other machines. It is common that ransomware propagate within the LAN after the infected machine performs network scans to identify potentially vulnerable hosts. For example, Ryuk reads the victim’s ARP cache and send Wake-on-Lan messages to each entry, so that when they are online it can send an ICMP ping to all addresses in a range of subnets. After identifying potential victims, the ransomware attempts to spread to those machines using a variety of methods such as mounting shared resources and copying itself to the new victim.

**Actions on Objectives**

At this phase, the stage is almost set for the attack to perform its main function; encrypt and potentially exfiltrate the data. The few parts remaining relate to the *discovery* of the directories and files that the ransomware wants to encrypt. The ransomware iterates through all directories in the machine to discovered target files and some access also network attached drives on the LAN.

It is important to note that ransomware do not blindly encrypt all the files they can find in each directory. They usually target specific file extensions or directories that contain critical data and can disrupt the normal operation of the organization. Ransomware usually target file types that may be used to store information that are critical for the users or for the organization such as *documents* that may contain sensitive personal and financial information, databases, configuration files and other assets that are crucial for the operation of business processes. On the other side, ransomware may avoid certain directories and files that may inhibit the rest of the attack or are easy to recover. For that reason, many file types are not targeted by ransomware, such as critical Windows files, web browsers and executables, as they are used by the user to minimally interact with the device, to get notified about the infection, to pay the ransom and can be easily and quickly recovered.

Once the desired files have been located, ransomware utilizing double extortion exfiltrate the data to the C&C server(s). Ransomware have been observed to exfiltrate data at various points in the kill chain, for example, before or after encryption. Nevertheless, exfiltration occurs after the directories are iterated and the target files have been located. For the data exfiltration attackers may use existing C&C channels, over network mediums such as Wifi connections, Bluetooth or other radio frequency channels, over Web Services and Cloud accounts. For example, they may use Google Drive, or Cloudfare to blend their activity with the legitimate activity of hosts.

Reaching this stage, ransomware are ready to encrypt the discovered files. Although there exist many methods to encrypt the files, modern ransomware use hybrid encryption where the file is encrypted using a symmetric cipher, such as AES, and then the symmetric key itself is encrypted using an asymmetric public key, saved and potentially transmitted to the C&C, who owns the private key. That asymmetric key, and perhaps the symmetric key itself, may have been transmitted by the C&C server at an earlier stage or they may have been hardcoded in the ransomware itself. It is known that symmetric ciphers are faster than asymmetric ones, so as attacks at this stage depend on their speed, attackers prefer to speed up this process as much as possible, by using symmetric ciphers and multithreading for file encryption.

To show that a file is encrypted, attackers may choose to rename the encrypted files by appending an extension. For instance, Babuk adds the *.babyK* extension and Wannacry appends the *.WNCRY.*

Having encrypted all the desired files, ransomware typically delete all backups, by scanning for directories containing the word ‘Backup’ or similar and interacting with known backup utilities such as Windows Shadow copies. Additionally, recovery tools are disabled, such as Windows Recovery Environment, so that recovery of the system is impossible.

**Extortion**

To conclude the attack, the attacker notifies the user that his files are encrypted by placing a ransom note in each encrypted directory and awaits payment. That ransom note explains to the victims that their files are encrypted, demands a ransom payment for their recovery and for them not to be published, assures them that there is no way of recovering the files unless the ransom is paid and provides information related to the payment of the ransom.

Of course, this abstract kill chain cannot encapsulate and accurately describe all of the ransomware that are met in the wild but succeeds in providing sufficient and detailed description and enumeration of the most common steps observed in ransomware and sometimes the order that they happen in. Therefore, it is a very valuable tool when analyzing ransomware behavior and especially when building defense mechanisms to detect ransomware, as this work aims to do. Throughout the rest of this work, I use this kill chain to understand what features can be constructed to detect ransomware activity and what activity they aim to catch.

**RANSOMWARE DEVELOPMENT**

To test my model and its detection capabilities, it was essential to simulate an attack. For that reason, I chose to develop a ransomware which exhibits the major attributes of a ransomware attack, which were described in a previous section. I developed this ransomware using Python 3.9, as Python offers a wide variety of tools useful for the development of a ransomware and interaction with the subsystem. Of course, the ransomware could be written in any other language but Python offers great capabilities for the development of such a script.

Since for this work, we are not interested in the first phases of a ransomware attack (Reconnaissance, Weaponization, Delivery) I focused on the steps taken by common ransomware from the Exploitation phase onwards. Therefore, I assume that the attackers have managed to deliver the payload to the target machine and have managed to unpack and decrypt its contents. Below, I describe the core actions taken by this ransomware to understand the way it works and other assumptions that I have taken.

First and foremost, the most important assumption taken is that this ransomware tries to conduct the attack on its own, without the help of any other child processes. It doesn’t spawn any other processes and doesn’t rely on the coordination between other processes that may have been delegated certain aspects of the attack.

Once the script is executed it first checks if it has elevated privileges (exploitation phase). If it does not, then it tries to elevate its privileges with a UAC prompt by directly asking the user to grant those privileges. If the user grants them, then the process is restarted with elevated privileges. This choice was made as the feature that I chose to use to capture the privileges of the process is not specific to exploitation techniques or UAC prompt requests. On the contrary, it is technique-ignorant and generally checks if a process is run in elevated mode. This way, it can capture any technique used by an attacker to elevate the privilege of a process.

Having acquired elevated privileges the script tries to gain *persistence* (installation phase) by modifying the Windows Registry and adding itself as an Autorun program. Specifically, it creates itself in the

“HKEY\_CURRENT\_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Rans” key.

Afterwards, it traverses all the directories of the system to locate target files. At this point it is worthy to note, that instead of traversing all the directories starting from the C:/ directory, the ransomware was configured to start the search from a certain directory that mimics the root of the directories. Starting from that directory it searches all the subdirectories in a depth-first manner, to locate all the files that it wishes to encrypt and the directories in which they are stored.

As many ransomware do, this ransomware does not blindly target all files indiscriminately. On the contrary, it is configured to encrypt only a specific set of file types, by examining their extensions. Specifically, the file extensions targeted are listed in the table below. Overall, the ransomware aims to encrypt text documents, Microsoft Office and LibreOffice documents, images, videos, audio files, and compressed files, to compromise documents that may contain personal, private and sensitive information of the victim and other indirectly linked stakeholders, and any other data and information that may disrupt the normal operation of the victim’s activities. Additionally, since (as will be explained shortly) the ransomware plans to exfiltrate the data to a C&C and threaten the victim to publish them unless the ransom is paid, files that contain such information have more value to the attackers and can be used as a leverage against the victim.

The malware then proceeds to gather information about the device, such as its OS, its version and release, its hostname, and its architecture, to send to the C&C. Although this information will not be used in any other way by the C&C to tailor the attack, it is gathered to mimic the communication of the victim with the C&C.

Shortly after the victim transmits the victim information to the C&C, it generates a symmetric key using the AES-128 bit cipher in CBC mode utilizing the Fernet python package. This key is the key that will be used to encrypt all the target files. Moreover, as was explained earlier, AES was used as the preferred cipher as it is a strong encryption algorithm, and additionally it is symmetric, which means that it will be able to encrypt a big amount of files in a short period of time, in comparison to asymmetric ciphers, directly corresponding to robust ransomware observed in the wild.

When the key is generated, it is sent to the C&C server so the C&C can provide the decryption key once the ransom is paid, and the victim can decrypt their assets. WRITE ABOUT ENCRYPTING WITH ASYMMETRIC KEY! When the key is transmitted, the ransomware connects with another C&C server that is in charge of data exfiltration. This server instead of communicating over a custom protocol (as the previous one) is a configured HTTP(S) server. Data exfiltration to an HTTP/S server is a very common and crucial method when performing data exfiltration attacks, because it can give a significant amount of cover due to the likelihood that users of the machine are already communicating with such protocols over their normal operation of their device. Specifically, data exfiltration over HTTP/S can blend with normal browsing behavior of the users, or Web API usage from legitimate applications. Therefore, the ransomware connects to the HTTP server on the alternative HTTP port 8080 and send the files with HTTP POST requests.

Once all the files have been exfiltrated to the HTTP C&C server, the encryption of the files is triggered. To encrypt a file, the malware reads all the data of the file in byte format (as not all data stored in a file is expected to be in text format), uses the symmetric key it generated to encrypt them, overwrites all the data of the file with the encrypted data and finally closes the file.

Finally, after encrypting all the desired files, the ransomware visits all the directories that it iterated through during the discovery phase and drops the ransom note informing the victim of the attack and the steps that they need to take to recover their files.

**FEATURES USED. PRESENT AS A TABLE. DESCRIPTION SHOULD BE IN THE DATASET CREATION SECTION OR FEATURE USE**

“NET”: Especially during the exfiltration phase, the ransomware is expected to send an abnormal amount of data to a remote host. For that reason, it is important to capture the number of bytes sent from a certain process to any host and add it to the feature sent. Therefore, essentially, this feature holds the aggregate of bytes sent from a certain process in a one-second period. This feature is expected to have a minimum of 0, where no bytes were sent by that process for one second and the maximum can be any number of bytes.

“WORKING\_HOUR”: The strength of anomaly detection systems does not lie only on the detection of anomalies, but its ability to detect anomalies based on the temporal context. To illustrate, it is expected that an organization will have significant amount of web traffic circulating during the working hours of the day, while in the non-working hours that traffic is expected to be much lower. For that reason, a temporal feature is used to indicate whether the observed behavior happens during working hours or not and therefore learn the different patterns emerging during those two periods.

“FILES\_WRITTEN”: During the encryption phase, ransomware are expected to write data to many files. This feature then stores the number of files that had at least one write operation by a specific process during the target one-second period.

“AVG\_WRITTEN\_ENTROPY”:

**METHODOLOGY:**

It is clear that the defense industry and research space need a system that can detect ransomware attacks as early as possible, while taking special care of false positives. The main objective of this work is to construct a system that can detect ransomware attacks on a device as early as possible on the ransomware kill chain, and simultaneously minimizes the number of false positives generated.

To achieve that I propose to create a comprehensive dataset comprised of features that correspond to the various stages of a ransomware infection. Using that dataset, I can model the normal operation of a system and use that to detect significant deviations and classify them as ransomware attack incidents. To make that classification, I further propose to train a machine learning-based anomaly detection model on those data, specifically an Auto-Encoder, as it has shown its capabilities in such tasks.

To create that dataset, I suggest identifying the various points in the abstract ransomware kill chain and their effects on a machine that are indicators of a ransomware infection. Combining them I will create an accurate depiction of the normal operation of the machine and deviations from it will thus indicate a potential ransomware attack.

Subsequently, an Auto-Encoder should be trained on those data to mathematically uncover the patterns and trends of a normal behavior.

To test the detection capabilities of this model, I advise to develop a ransomware that performs the main actions of ransomware following the presented kill chain. By deploying the trained detection model and the ransomware in a test system, I can monitor the behavior of the system’s processes and use the inferred features to test if the model can detect the ransomware infection, at what stage of the attack it is captured and finally the number of false positives and false negatives that are generated.

**DATASET CREATION MAYBE COMBINE WITH THE DATASET COLLECTION SECTION**

To create the dataset I must first think of the ways that the behavior of ransomware may deviate from that of normal processes. Despite their limitations, the previous works presented have been instrumental, as they have help uncover the potential points in a ransomware infection that can be indicators of compromise and provide intuition to explore more features. Firstly, as the features used during static analysis of ransomware have significant drawbacks, I pursue a purely behavior-driven detection method. In this work I focus on behaviors and patterns that are generated by ransomware during execution from the first stages of an attack until the attack is completed. Secondly, I focus on the steps taken by ransomware once they are inside a system and are executed.

Before diving into the exact features that will be used, it is important to reiterate that no feature can detect ransomware attacks on each own, but different features are combined to create a comprehensive *set* of features capable of describing and ultimately detecting a ransomware infection.

Initially, many actions performed by ransomware require elevated privileges and for that reason during the first steps of the execution, ransomware processes try to gain elevated privileges. While it is hard to enumerate and capture all methods that a process might gain such privileges, I propose to monitor the status and abstractly infer the privileges of each active process on the system. That way, regardless of what method a process might leverage to escalate its privileges, eventually this privilege escalation will be noticeable by the change of the process’s status.

Progressing in the ransomware kill chain, ransomware try to gain persistence by modifying the Windows Registry AutoRun keys. Specifically, ransomware may use the following subkeys:

* “HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run”
* “HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce”
* “HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Run”
* “HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce”

From these keys, the first two require elevated privileges to be modified, while the last two can be modified with normal user.

Some ransomware may rely on other processes to add the main malware to the AutoRun keys, or may want to trigger the execution of the ransomware every time a user logs on the device. In such scenarios, the process that modifies the registry and the main body of the malware have different Process IDs and their cooperation cannot be easily traced. To accommodate that, I advocate to include another feature which states whether a process exists already in the Registry AutoRun keys.

Subsequently, ransomware try to terminate certain processes that inhibit file access, encryption or that relate to defense systems such as antivirus software, firewalls etc. As this is a key step in a successful attack, I suggest monitoring the use of certain applications and tools that kill processes, such as “TASKKILL”. Normal applications use such tools and commands very rarely and such events should be included as a part of the feature set that can differentiate normal from ransomware activity.

Moving forward, ransomware are expected to traverse an abnormal amount of directories in a very short period of time when trying to discover files that they wish to encrypt. Therefore, the volume of directory traversal and file enumeration efforts should be monitored.

After locating the target files, ransomware usually will try to exfiltrate them to a remote location. This activity provides another point that can be used to detect an infection. By monitoring the volume of egress traffic initiated by a process we can detect data exfiltration efforts, regardless of the type of C&C or exfiltration server employed.

When ransomware move on to the encryption phase, they can be detected using a variety of features. Although encryption itself is not easily traceable, one can infer use of encryption by inspecting the side effects of certain activities. For instance, ransomware are anticipated to write to many files in a short amount of time, especially since attacks as usually sped up when they reach the encryption phase. Thus, one should track the number of files that a process writes to. However, not all file types should be considered in the calculation of that feature. Instead, a detection system should monitor mainly specific file types that are common targets of ransomware attacks. Overall, ransomware aim to encrypt text documents, Microsoft Office and LibreOffice documents, images, videos, audio files, and compressed files, to compromise documents that may contain personal, private and sensitive information of the victim and other indirectly linked stakeholders, and any other data and information that may disrupt the normal operation of the victim’s activities. These files that contain such information have more value to the attackers and can be used as a leverage against the victim to pay the ransom.

Additionally, in information theory, “Shannon Entropy” is a measure of uncertainty or randomness of a variable and can indicate how “random” or “unpredictable” the data of a file are. This measure takes values between 0 and 8, where 0 indicates no randomness and 8 indicates complete randomness in the given sequence. Therefore, as encryption generates seemingly random sequences of data, it in consequence produces high entropy data and thus high entropy can be used as a feature to infer encryption. Because ransomware are expected to write high entropy data to a large number of files during a given time period, it is then advisable to calculate the average entropy of the files that a process wrote to, additionally to the number of files that were modified by that process.

Furthermore, I advocate that a ransomware may exhibit different writing patterns when writing to files. Specifically, the average number of write operations per file can be a sign of infection. One may understand that, if they think that a process may write many times to a file as they modify that file. On the other hand, ransomware are expected to write to files only once, when they overwrite the contents of the files with the encrypted data and never write any data to them again. Supplementarily to the previous two features, this can help distinguish normal from ransomware activity.

One reasonable question that one may raise is that all these features correspond to behaviors seen by benign processes as well. For instance, a web browser process sends data to other hosts as a part of their normal operation and compression applications may write to files with a high data entropy, due to the nature of compression.

While that is true in some sense, I advocate that while many applications may exhibit behaviors similar to a ransomware for some features, the *combination* of all the features is what will definitely differentiate normal from suspicious behavior. For example, a ransomware is expected to send some amount of data over the web and have elevated privileges, and have accessed AutoRun Registry keys in the past *and* has written to many files with high entropy etc. Therefore, the *combination* of those otherwise normal feature values, will eventually classify a behavior as ransomware and not normal.

Moreover, during the exploration of those features I did not focus on ransomware that self-propagate as they are not a crucial characteristic of ransomware. Similarly, C&C communication is not an indispensable part of a ransomware attack so I chose not to focus on features that detect such traffic except from data exfiltration activities.

**DATA COLLECTION**

To gather the necessary data, the test system must be monitored for a specific period and across different time contexts. In this work, I used the Process Monitor utility (or procmon), an advanced tool for Windows that can monitors and records changes made in the system by all processes in terms of file system, registry, network, and general process activities. Additionally, I utilized a self-developed script to supplement Process Monitor and infer the privileges of active processes.

Firstly, because a good anomaly detection system should detect anomalies based on the time context as well, I keep track of the time of day that a process is running. If the time is between 09:00 to 17:00 I record this as being a working hour and any time outside of that I record it to be a non-working hour.

I used procmon and extracted the volume of the egress network traffic from the “Detail” column, information useful to detect any data exfiltration and C&C communication attempts.

Then, I monitored *any* interactions of the processes with the AutoRun registry keys, to detect any attempts of processes to gain persistence. It is prudent to monitor any interaction with these keys as different ransomware may initially simply check if they exist already in the registry by relying on another process to make that modification.

To supplement that, I monitor if a process is already registered as an AutoRun program, by another process, or by any other means. To do that, I utilize procmon’s “Command Line” column which lists the command that was issued to execute that process, strip any command line arguments that may have been for execution and check if the remaining executable paths exist in the registry AutoRun keys.

Additionally, when a process traverses directories and enumerates the files in that directory, it generates “QueryDirectory” calls. Therefore, I record such calls made by each process by using the “Operation” column. As mentioned before, a large volume of such calls can be indicative of a ransomware trying to discover files to encrypt by traversing the directories of a system.

Next, I track the number of files written by a process in a certain amount of time, by recording the “WriteFile” operations and using the “Path” column to infer the file that the process wrote in. Using those two pieces of information I can get the number of files that a process wrote to for example, when a ransomware writes the encrypted data in a file. It must be noted, that I do not blindly record write operations in any file type, but I focus on file types that are usually targeted by ransomware, such as documents, media, compressed files and others, as explained earlier. A large number of write operations in a short amount of time can signify a ransomware infection.

An important feature is the average written entropy. As described before, a large entropy of a file can be an indication that a file is encrypted or compressed, due to the nature of these methods. To capture the average *per-file* written entropy by a process I calculate the entropy of the files that were identified in the previous feature, and I use the following formula to calculate their data’s entropy. Finally, I average the calculated entropies of all the files, to get the average entropy of all the files that were written by each process. MENTION CHOICE OF 4 DEFAULT ENTROPY. EITHER HERE OR IN ANOTHER SECTION

Usually, the use of encryption also means the usage of DLLs related to cryptographic functions. Using the process activity utilities of procmon I keep track of efforts of processes to load cryptographic DLLs, by monitoring any “Load Image” operations. In detail, I take into account only DLLs that contain the word “crypt” inside, such as the “bcrypt.dll”, the “crypt32.dll” and other DLLs that are part of the various Windows cryptography APIs. Undoubtedly, attackers may bypass the use of such APIs by statically linking encryption DLLs. As Stamper explains though Stamper, API and DLL monitoring should be used as a *part* of a larger feature set.

While procmon is a tool that provides access to granular information, it has its limitations. An important feature to keep track of, is whether a process is running with elevated privileges. To get hold of that information, I created the following *Powershell* script influenced by [CITE SCRIPT INFLUENCE SOURCE HERE], that iteratively enumerates all the processes running in a system and their privileges.

At the end, by running the script simultaneously with procmon I get all the active processes in the system and their privileges, gathering all the necessary information to create the dataset.

In summation, the final dataset is comprised of the following features:

|  |  |  |
| --- | --- | --- |
| **FEATURE** | **DESCRIPTION** | **TYPE** |
| WORKING\_HOUR | The process runs during working hours | Binary |
| NET | Bytes sent by a process | Numerical |
| AUTORUN\_EDITS | Interaction with the Autorun Registry keys | Binary |
| EXISTS\_IN\_AUTORUN | The process exists as an AutoRun program in the registry | Binary |
| CRYPT\_DLLS\_LOADED | The process uses any DLLs associated with cryptographic primitives | Binary |
| FILES\_WRITTEN | Number of files written | Numerical |
| AVG\_WRITTEN\_ENTROPY | The average entropy of files written | Numerical |
| AVG\_WRITES\_FILES | The average number of write operations per file | Numerical |
| ELEVATED | Process running with elevated privileges | Binary |
| QUERY\_DIRECTORY | The number of “QueryDirectory” operations performed by a process when enumerating the files in a directory | Numerical |
| KILL\_DEFENSE | The process attempts to terminate processes related to defense mechanisms | Binary |

**DETECTION MODEL**

For an accurate and reliant detection system, equally important to the features used is the model that is deployed. In this work I propose to use an Auto-Encoder to learn a representation of the operation of normal applications. After the normal behavior has been baselined, the same model will be used to infer if the behavior of a process is an anomaly or not.

Although more sophisticated models and variations of the Auto-Encoder exist, I suggest that an Auto-Encoder suffices for the scope of this work and can produce the intended results.

As was explained earlier, an Auto-Encoder consists of two main parts, an Encoder, and a Decoder. The Encoder learns to encode the input data to an efficient representation of fewer dimensions and the encoder learns to recreate the input data from that compressed representation.

In this work, the Auto-Encoder will learn the patterns and trends of normally behaving processes, by learning to compress the input features into a latent representation and it will learn how to effectively reconstruct the original features from this latent vector. By minimizing the reconstruction loss during training the Auto-Encoder will effectively encapsulate and baseline the normal behavior of the system, and it will be able to reconstruct accurately any other normal behaviors observed.

For the Auto-Encoder to learn accurately the intricacies of a normal operation and to work appropriately it is important that the training dataset and validation datasets include exclusively samples from normal applications. Thus, it is crucial to ensure that the device that is used to collect and create the dataset is not infected with any kind of malware and specifically with ransomware, as the inclusion of ransomware samples in the training dataset will force the model to learn how to compress and reconstruct effectively malicious events as well.

That being said, when there is an ongoing ransomware attack the Auto-Encoder will not be able to reconstruct the original input from the latent representation, as it has not been trained on any ransomware infected instances. Generally, the output of the Auto-Encoder will be the reconstruction loss of the input given. It is expected that normal inputs, that is samples that correspond to normal behavior, will have a low reconstruction loss and abnormal inputs, will result in a generally higher reconstruction loss. Using that reconstruction loss we can infer if the observed behavior corresponds to a normal operation of the system or an ongoing ransomware attack.

Since the output of the system is simply the reconstruction loss of the input against the output of the Autoencoder, it is important to provide a threshold above which any observed sample will be classified as an anomaly. Although this is a process that will be described in a later section and belongs to the hyperparameter tuning phase of the system, it must be said that there are many choices for thresholds that can influence the efficiency of the model.

**BASELINING NORMAL**

Additionally to the features used and the chosen model, the quality of the training data is of vital importance. For the system to be reliable, the data that will be collected must reflect a real use of the system where plenty and realistic interaction of the user with the system takes place.

The target device should be monitored for a significant period while the device is used for normal activities performed by the user. As the goal is to baseline the normal behavior, the user should use the device normally, performing a diverse set of activities that they usually do throughout the day, such as browsing the web, editing documents, viewing media, modifying files and any other activities that they normally do in the system.

Additionally, the device should be monitored across different times of the day, to capture the behavioral patterns in both working and non-working hours of the day. This will increase the expressive capabilities of the model and it will enable it to be used across different times of the day.

It is imperative that while the system if monitored to baseline normal, the device is not infected with any kind of malware and most importantly with ransomware. As the Auto-Encoder that will be trained is used to capture and model only the normal behavior, the training data should not contain any malicious processes that can reduce the perception and detection capabilities of the system.

**PREPROCESSING**

Before any training occurs, it is important to preprocess and transform the dataset into a format suitable for use by Neural Networks. To begin with, some of the features used are not numerical but categorical and specifically binary (0 and 1). For that reason it is considered best practice to use One-Hot encoding for those features. So every binary feature in the dataset is transformed into two features where each feature holds the value of the original feature encoded in one-hot format.

It is also important that the dataset does not include duplicate values, as the model may overfit and be heavily influenced to learn mostly these duplicate values. For that reason, I discarded all the duplicate samples and kept the unique samples only.

Next, the dataset is split into two non-overlapping datasets, one used solely for training and the other one used for validation with a ratio of 80/20.

It can be seen that the numerical features used have different scales. For example, the “NET” feature ranges from 0 to tens of thousands, while “AVG\_WRITTEN\_ENTROPY” is bound between 0 and 8 (by the definition of Shannon Entropy). Therefore, it is extremely beneficial to normalize or standardize the dataset, where normalization means that the resulting features will be casted to the range between [0,1] and standardization means that additionally the mean and standard deviation of each feature will be 0 and 1 respectively. During validation it was found that normalization works better for this case, so eventually each feature was transformed to have a range of 0 to 1.

What must be highlighted here is that only the training sample was normalized using its own statistics. To ensure that there is no data leaking to the validation and test datasets, both sets are normalized using the statistics of the training set. Therefore, while the training set is expected to have min and max values of 0 and 1, that is not true for the validation and test sets, although their scales will be similar.

**EXPERIMENTS**