No Strings on Me: Linux and Ransomware

GIAC (GCUX) Gold Certification

Author: Richard Horne, richard.horne90@gmail.com
Advisor: *David Hoelzer*

Accepted: September 11th 2020

Abstract

Ransomware poses an ever-increasing threat to businesses and organizations as it continues to evolve and change. Many organizations are forced to pay for solutions to this growing problem with expensive and out-of-date signature-based solutions. As the possibility looms for ransomware to impact all operating systems and businesses alike, organizations will need to focus on early detections and warnings to stay ahead of its spread. This paper aims to examine the probability of detecting ransomware throughout its lifecycle within Linux environments. In conjunction with detections, the ultimate goal of the ideas presented is to provide security teams with a more reliable and cost-effective method to detect, react, and neutralize Linux ransomware variants.

1. Introduction

Within the last decade, ransomware has become one of the fastest and most popular ways for malicious actors to target businesses. This increase in pointed ransomware attacks was underlined by the United States Federal Burau of Investigation when they stated:

> Ransomware attacks are becoming more targeted, sophisticated, and costly, even as the overall frequency of attacks remains consistent. Since early 2018, the incidence of broad, indiscriminate ransomware campaigns has sharply declined, but the losses from ransomware attacks have increased significantly, according to complaints received by IC3 and FBI case information (FBI, 2019).

Ransomware infiltrates businesses via various means and can often go undetected for some time. Many groups experience the unfortunate notification when any preliminary indicators have failed and hopes for early containment are past due. Businesses and organizations that have fallen victim to ransomware are at the mercy of encrypted files and attackers who metaphorically “call the shots." When files become encrypted, targeted companies see their technological infrastructure hanging by a mere thread as their property's decryption is nearly impossible without the appropriate key. As those who become infected are issued an individual ransom for the said key, demands for overwhelmingly large sums of anonymous bitcoin are offered as the only means of repair. The monetary compensation, as an attacker might say, for the inconvenience of the attack, allows businesses to take back what they owned in the first place, or suffer the consequences of losing seemingly everything they have worked to develop.

Targeted ransomware attacks provide a means for prominent statements of distaste to be issued by hacktivists and others. Often, attacks such as these provide onlookers insight into the views of malicious actors and the businesses they deem as the "evil du jour." With many ransomware strains on the loose and hundreds of infections being reported or sold on the black market, one might ask where are the strains that impact Linux? Before answering this question, it is important to understand the potential impact of a Linux based ransomware variant. The Linux Foundation stated that:

> Linux has since become the world's most dominant operating system, with massive adoption in almost every sector, including finance, government, education, and even film production. It is also the operating system of choice to support cutting-edge technologies such as the Internet of Things, cloud computing, and big data (The Linux Foundation, 2020).

With such a glaringly obvious attack vector in so many different industries, the eventual ransom of any organization has the potential to impact Linux directly.

As integral parts of the way we live online and in an ever-growing cloud-based world, Linux devices hold the potential to be the next large-scale ransom-based attack that we as a security community are not yet ready to deal with. As this idea is one that hopefully will never come to fruition, it is crucial that those who monitor and protect networks consider what impact a widely spread Linux ransomware variant would have on the way business is done. We must consider how the internet is connected via Linux, and how the cloud age we live in would be crippled. As major cloud providers such as AWS, Oracle, and Microsoft base entire workloads and services on Linux, the magnitude of such an incident becomes apparent.

In an attempt to avoid such a widespread and monolithic problem, the ideas presented herein attempt to move away for the dependency on signature-based antivirus alerts and expand to a more ecosystem-centric approach. It is believed that ransomware detections cannot rest solely on the backs of those that write and provide paid-for security products, but instead should be enriched by environmental awareness and correlation. With this new leans of sight, opensource tooling and long-trusted best practices can be used to gain visibility that currently escapes Security Operation Teams. It is believed that these key concepts will only be viable in situations where security teams are involved and exercise patience, understanding, and foreknowledge about their own Linux systems. In such circumstances, the ideas presented will have the most proliferating impact for lasting success.

Richard Horne - Richard.horne90@gmail.com

1.1. Example of Ransomware and Its Impact

Within the last two years, an outstanding example of pervasive ransomware occurred within the state of Colorado, USA. The Colorado State Department of Transportation (CDOT) was infected with a ransomware variant known as SamSam. While this strain of ransomware is primarily known to impact Windows operating systems, the impact of the SamSam infection, as well as CDOT's unfortunate plight, demonstrates the need for security teams everywhere to be prepared contextually for ransomware derived attacks.

SamSam's impact on CDOT resulted in a dual wave infection of nearly two thousand workstations and servers (Chuang, 2020). As a stop-gap for daily business operations, employees within CDOT were forced to use pen and paper to perform daily duties while information security members worked to isolate and contain the spreading infection. As CDOT's Information Security group scrambled to contain the spread and impact of SamSam's first wave, their efforts were quickly undermined by another variant of the same ransomware strain that was spreading throughout many of the newly sanitized zones of their network. The Governor of the State of Colorado, who was overwhelmed and did not have the resources to deal with such a pervasive and undermining attack, issued a verbal decree indicating a state of emergency. This issuance made by the Governor allowed the Colorado National Guard to assist CDOT in all reparation efforts from that point forward. With two ransomware strains moving quickly through their network, the physical setback for the department of transportation required a multimillion-dollar emergency budget and hundreds of staff-hours by all parties involved. Nearly two weeks of downtime for thorough sanitization by CDOT employees and National Guard members finally resulted in both infections' remediation. Additionally, weeks of staff-hours were added to the seemingly unceasing efforts to continue to monitor after the dust had settled on the department's infrastructure (Willis, 2018).

From the beginning, it was clear from Colorado's state officials that CDOT refused to pay the ransom that the attackers requested. As CDOT grappled with its multiple setbacks and the aftermath of such events, this attack shook the state of Colorado and sent a rippling warning to state and local government agencies. The ransomware in this attack proved to be real, and it came with a deadly and expensive price tag. Although this strain is effective against Windows operating systems, as stated before, this example's importance cannot be understated.

Like many ransomware attacks, CDOT was initially made aware of the infection and worked tirelessly to avoid paying the ransom. CDOT relied heavily on backups taken from before the attack, but unbeknownst to them and many others in similar circumstances; backups contained the infection which had been dormant for days. When the initial containment efforts were underway, and business started to return to normal, but the infection returned and was on the loose once more. CDOT, at this point in time was forced to watch as their efforts and progress faded back to ground zero.

For CDOT and many others, having antivirus tooling in place is simply not enough. Antivirus and anti-Malware software is sold on the principle that it will catch ransomware, malware, and viruses alike. Why were their tools unable to assist or alert them to the spread of a standard ransomware stain sooner? Ironically, in this scenario for CDOT, an infection that had not only spread once but was spreading for a second time still was overlooked by their antivirus software. As companies and businesses spend billions each year on security tooling, why do our sophisticated and modern firewalls, antivirus, EDR toolsets, or IDS / HIDS, not alert us to the spread of ransomware before it is too late (Morgan, 2019)? The answer, as proven by the plight of CDOT, remains true to this day. Signatures and detections prove slow and useless when working with ransomware and many other viruses. Companies that produce antivirus software products will always be behind when it comes to catching and thwarting malicious software that has not been previously cataloged.

1.2. Detecting Ransomware Differently

Like many other strains or variants of malware, ransomware has historically been tracked and traced based on its actions taken during infection or its various code structures when reverse-engineered. These literal bits of information may ultimately account for a strain's name, a designation within a malware family, and detection signatures created or used by vendors. Antivirus vendors create signatures based on unique, characteristic traits and actions for each strain of ransomware. As these signatures have been developed by household and enterprise brands such as: Malwarebytes, Microsoft, McAfee, ClamAV, Avast, and or Norton/Symantec, they have only ever focused on catching ransomware after it has been well researched. Of specific note within the Colorado Department of Transportation example is that their antivirus vendor was engaged and required time to custom fabricate a signature while the infection was still spreading. As stated in the Colorado Sun's review of the incident, they stated: "One of the problems with SamSam was that the strain was so new, CDOT's anti-malware software didn't detect it" (Chuang, 2020). As a way of support and more importantly for the vendor, as a means to benefit from the incident, the security company that CDOT employed later used the CDOT SamSam malware sample to build a signature to protect its other customers from the strain that plagued CDOT (Chuang, 2020).

Antivirus companies require extended periods of time before they feel confident that they can identify a strain of ransomware without any false positives or erroneous alerts. As they create signatures and reverse engineer, as seen with CDOT, high-risk situations of infection are in progress around the globe. One fundamental issue with this approach has been that ransomware historically avoids detection due to its customizable coding. A variant is easy to create and will change quickly as attackers revise source code. CDOT's variant of SamSam was able to quickly change even though its core code structure remained the same. A method of weekly or monthly vendor-made security patches for antivirus products may work for home users, who are not commonly targeted by malicious actors. However, for those large organizations that are commonly targeted, who is left holding the bag of responsibility for their best interest?

For this reason alone, a different approach needs to be developed and accepted as the first line of defense when accurately alerting to the presence of ransomware, rather than waiting for detections, alerting, and blocking to come from the same vendors and applications that are notoriously latent. Alerting should be done separately so that isolation can take place first and foremost. Taking the act of alerting away from these vendors will not only allow for other options to fill the space left in their absence, but will provide faster response times and more room for stopping mass infections and moving towards cleaning small groups of isolated devices whenever an infection occurs.

As the idea of signatureless ransomware detection is explicitly presented for Linux operating systems, a particular focus will be given to providing the structure and lattice for the successful move from vendor-centric environments to that of a vendor-less security ecosystem. Ultimately when focusing on the possible attack rather than having blind trust in vendors, security teams can take back the security they need and desire.

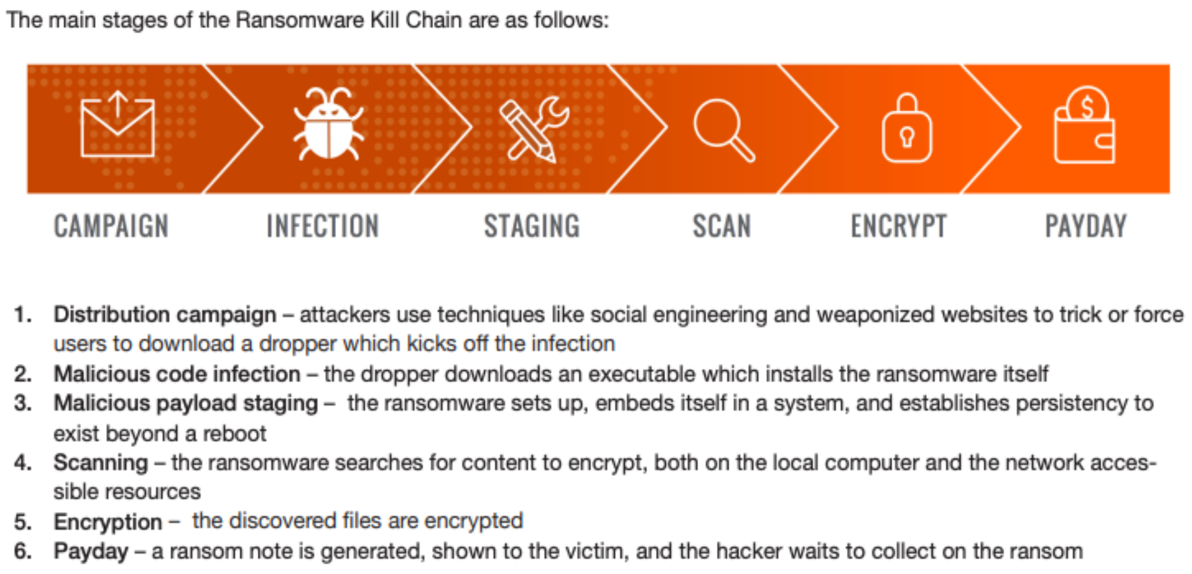
1.2.1. Ransomware Kill Chain

Kill chains are commonly used in a way to designate the critical path that attackers and or attacker tools must take in order to be successful when performing an attack. As the attack progresses, the theoretical idea is that at any point in the various stages, if a countermeasure is employed, the entire attack is thwarted and must be started again (Lockheed Martin, 2011). This rinse and repeat model becomes difficult for attackers as mitigations and permanent solutions used by defenders require malicious actors to go back to the drawing board and develop a new attack strategy over and over again. As the number of attempts increases, the cost required for each successful attack of a malicious nature becomes higher and higher.

Although the Cyber Kill Chain is designed to break down an attack from start to finish, the same type of idea can be used when looking at ransomware directly. Attackers must develop their campaign, work out staging and payout options before attempting infection. As ransomware requires time to become fully active inside of one's environment after infection, the malicious code will take specific actions to determine where it is, how it might spread, and what its next steps will be when communicating with its command and control mechanism (Exabeam, 2016). An example of the

Richard Horne - Richard.horne90@gmail.com

Ransomware Kill Chain is as follows:



The main stages of the Ransomware Kill Chain are as follows:

1. **Distribution campaign** – attackers use techniques like social engineering and weaponized websites to trick or force users to download a dropper which kicks off the infection
2. **Malicious code infection** – the dropper downloads an executable which installs the ransomware itself
3. **Malicious payload staging** – the ransomware sets up, embeds itself in a system, and establishes persistency to exist beyond a reboot
4. **Scanning** – the ransomware searches for content to encrypt, both on the local computer and the network accessible resources
5. **Encryption** – the discovered files are encrypted
6. **Payday** – a ransom note is generated, shown to the victim, and the hacker waits to collect on the ransom

Figure 1: Ransomware Kill Chain

As one focuses on the impact of ransomware on Linux specifically, it is essential to break this kill chain down into applicable and controllable sections that pertain to the experiment at hand. Since there are no reporting, logging, or alerting mechanisms possible inside of the areas of "Campaign" or "Payday," one must focus on those segments found in-between these two kill chain bookends. From a host and network-level perspective, all-important detections will be based on the phases of "Infection," "Staging," "Scanning," and "Encryption." As these areas are within the network domain that a security team has direct access to, a crucible of intense review should be placed around what can be uncovered during these parts of the Ransomware Kill Chain. In order to create this crucible of alerting, many of the advantages that will be explored within the Linux operating system will rely on tools that are native to Linux. Developing a tooling system to catch ransomware should be inherent to the operating system itself and rely on detecting deviations from the norm. In order to do this, it will be fundamentally important that a solid understanding of the environment be known prior to development and alerting on any deviations that may or may not be indicative of ransomware. As there are many parts to the development of ransomware, different ideas and probable detection means will be explored in section 4.1.1.

1.2.2. Examples of Linux ransomware

Ransomware specifically developed and created for Linux is somewhat of a rarity. Over the past ten years, only a handful of Linux ransomware variants have been created and successfully employed. Of those that have been created, the following three may be considered to be the most noteworthy. As these are just examples of the impact that has already been seen on Linux specific devices, the importance of an early warning system cannot be understated.

Lilu or Lilocked	• Debuted July 2019 • Thought to have infected 6,000+ Linux based web servers • Notably skipped critical files while focusing on items with file extensions based in HTML, SHTML, CSS, JS, INI, and PHP • Infection thought to be from Exim exploit or outdated versions of WordPress (Balaban, 2020)
Erebus	• Debuted September 2016 • Notably infected Nayana, a South Korean Web Hosting Company who attempted to recover by paying the ransom demands of one million dollars. • Known to target roughly 433 different file extensions • Infection thought to be from malvertisements (Trend Micro, 2017)
KillDisk	• Debuted December 2015 • Most notably infected the Ukrainian power grid and various financial groups • Designed to demand a ransom falsely due to recovery not being an option that is coded into the ransomware's functionality.

	• Permanently damages operating systems and causes boot issues (ESET, 2017)

2. Linux Tooling

Linux tooling is a versatile and seemingly endless repository of programs and tools designed to integrate intimately with Linux for the needs of security and administrators alike. Due to the different distributions of Linux, some security and admin tools may be more apt at defending or alerting than others. As the problems of Linux ransomware have been explained, the goal of developing an easy detection system for Linux was paramount. The chosen tools were a means of leveraging already existing applications that were native and or widely known, and that would provide security teams with starting positions that could be developed further in the future. The SIEM, which will be exposed later, provides the basis for detecting and correlating events that help security groups stay ahead of ransomware inside of their environments.

All of the following sections will outline the steps that were taken to create a detection mechanism for the purpose of detecting ransomware. Samples of configurations and actual steps taken may be found in the Appendix. As each configuration shows plausibility, it is fundamentally crucial that when working to implement the same type of alert, attention is given to the environment in which the tooling will reside. One size does not fit all in these situations, but with the right understanding, modifications are simple. This type of alerting can be effective and efficient in all environments.

2.1.1. Test environment

The test environment for the work exposed from this point forward should demonstrate that catching ransomware is possible and applicable to any environment that runs or maintains Linux devices. The primary objective was to use a stripped-down version of any Linux server distribution, with only widely available tools and free versions of other programs such as Splunk. Oracle Linux was selected as the primary distribution for the ransomware to be installed on. Oracle Linux 7 was installed and was not allowed to update or install any tools other than the Splunk forwarder, auditd, and sysstat daemons. A true "sheep" was desired for the ransomware to infect. Forwarding configurations were made to the Oracle 7 image to allow the independent logs for audit and sysstat to be logged and consumed by a standard Fedora Desktop 32 image running Splunk. The Fedora box will be acting as a Search Head and log collection platform. Splunk Enterprise was chosen, as this application is already found in many security operation centers, and can leverage many of the ideas and concepts explicitly needed for correlation and monitoring. As many of the Splunk configurations are found as flat files, modifications can quickly be made within said files for quick tuning and modification. Because the basic structure of Splunk does not change, whether it is Splunk Enterprise or the free trial, all examples will functionally be the same in this experiment.

Of specific note, the exercise performed does not include any of the added functionality or customization of Splunk's Enterprise Security (ES) application or add-on. All of the actions performed were individually done and customized outside of the ES app. As ES is frequently employed in an Operations Center as a SIEM, it does not offer any improved functionality over the base Enterprise version of Splunk when performing searching or working with correlation searches. Thus, ES was found inutile for this exercise. Splunk's alerting functionality is also the same inside and out of ES. Those that are familiar with Splunk's ES tool, will note that the default notable index provided when ES is installed is not commonly found when ES is absent. As this index was not present inside of the test environment, one that is functionally the same was created for ease of use as well as understanding to familiar parties.

2.1.2. Audit

Linux auditing under the auditd daemon is a means to track a given application or process from start to finish. According to Red Hat:

> The Linux Audit system provides a way to track security-relevant information on your system. Based on pre-configured rules, audit generates log entries to record as much information about the events that are happening on your system as possible. This information is crucial for mission-critical environments to determine the violator of the security policy and the actions they performed. Audit

does not provide additional security to your system; rather, it can be used to discover violations of security policies used on your system (Red Hat, 2020).

As the goal of this research is ultimately to detect what is taking place with ransomware on a Linux endpoint from start to finish, audit logs will provide critical insight into any and all activity. Audit, by default, will monitor and log activity associated with, but not limited to:

- file access
- system calls
- commands run via the command line
- security alerts
- events
- processes
- network activity

Each of these log entries will be logged for future consumption within /var/log/audit/audit.log. As logs are written, the Splunk forwarder automatically monitors this file path and will send them to the Splunk Search Head for further correlation and manipulation.

The auditd configuration is the keystone of what is being ignored and what is being logged on the Linux endpoint. An example of the audit configuration for this exercise can be found in the Appendix. As custom configurations for any given environment can be very complicated, all detailed explanations are beyond the scope of this paper. Suffice it to say that with patience and tuning, a very robust and all-inclusive audit ruleset can be made for any environment.

2.1.3. Sysstat and host performance monitoring

Sysstat will be paramount in this scenario because ransomware inherently performs encryptions. Cryptographic functions usually are very process-intensive and will require as many hardware resources as possible. A spike in the CPU, for example, should indicate that encryption is underway. When accounting and logging for these changes in the CPU, those familiar with the endpoint will be able to tell if the OS or the kernel is under heavy stress, and activity is expected. If there is a deviation, or if ransomware is performing cryptographic functions, these oddities may be a sign of nefarious actions. Calls to the encryption libraries might also be seen via auditd, but the kernel reporting back frequent totals for CPU and memory usage will be a baseline that will permit an environmental accounting of how the CPU is doing historically.

Along with sysstat's ability to primarily look at usage totals for the CPU, it will also play an important task in baselining the activity taking place on the network interfaces. This aspect is not incredibly helpful for this testing due to the fact that only one interface is being used. However, in a typical environment, logs generated based on network interface usage are found in the directory /var/log/sa/sar*. In addition to these logs types that are being consumed by the Splunk forwarder, there have not been any specific modifications made to sysstat other than cron frequency. Since this is a config file, a reference will also be found within the Appendix.

3. Splunk: Bringing It All Together

3.1. Power of big data

For those that are familiar with the ideas of big data, this section might act as a refresher and can willingly be overlooked. For those that are unfamiliar with the ideas of large data sets, the idea of big data is critical in determining what is suspicious and what is not in the case of ransomware. Large databases allow anyone to apply correlating ideas to our host logging and see in real-time the results of these correlated hypotheses. Splunk has long been an easy entry point into the market of ingesting logs in real-time and is apt for the job of our test scenario. Splunk consumes almost all log formats and then indexes them in a time-sequential order as close to real-time as possible. As a log is created on an endpoint, that log can be, within seconds, forwarded, indexed, and searched using the Splunk user interface. Additional customization to how telemetry data is stored in indexes can be given at any point in Splunk's lifecycle. As data is separated and partitioned according to user preference and need, pairing otherwise complex log formats together becomes easy. From a security standpoint, the consumption and ingestion of all logs allow for our two Linux devices configured with audit and sysstat to report their status on an ongoing basis.

Statistically speaking, as more events of what would be considered normal are created, indexed, and become searchable inside of Splunk, the better baseline operators and detection creators will have at finding deviations that can be alerted on. Security Teams need to be able to collect, store, and analyze this data to be able to make sense of what is going on, not only on the endpoint but amongst network peers. Having all of these logs and other data points in a single place with a robust search language to detect and find malicious events is where Splunk is able to flex its muscles. Splunk provides a time structured data format, with a robust implementation of its own Splunk Processing Language that allows for a mixture of bash and SQL-like queries to be made on virtually all data types.

3.2. Correlation and hierarchal security models

Splunk allows analysts to search and combine multiple data and information types into precise results. As this concept has been done in tools such as ELK Stack as well, this is not new to the world of big data. Nevertheless, as an essential element in our test scenario, correlation, and log normalization stand as the panicle preparatory steps needed for success when combining the various log types from different disparate log sources to try and produce meaningful information. It is crucial that we are able to perform a succinct correlation between two meaningless sources, that when brought together, form a compelling picture of what is taking place on the endpoint. To further this meaning, overlaying the urgency of a given alert to the results that we have requested Splunk to report on further extends what can be done with the data as it pertains to ransomware or other security relevant logging. Ultimately the goal of correlation and hierarchal data is to structure like data together in a precise and accurate manner allowing Splunk's alerting mechanism to freely search over collected data quickly and efficiently.

Richard Horne - Richard.horne90@gmail.com

4. Splunk Crons and Alerting

4.1.1. Detections / Alerting

One of the functionalities of Splunk is that it allows for data to be continuously ingested. As data flows in and is indexed, Splunk also performs as close to real-time alerting on the newly consumed data. In the test scenario, as the events of every action on the endpoint from audit and sysstat data are received and indexed by Splunk, the saved searches that create alerts are commissioned to run every few minutes. Splunk employs the standard cron job model that many *nix-based operating systems are accustomed to. Splunk's alerting mechanisms are configured to take action and perform further enrichment.

Each of the actions Splunk may take is fundamentally important in helping generate meaningful alerts. As experience has proven, the goal of this experiment should be to stay away from just another set of ransomware signatures in an application that metaphorically mimics an antivirus. Alerting actions, such as the following, provide the distinct separation from antivirus software that is sought for. All actions that Splunk provides enhance future correlation, and will add to a "bubbling up" effect which will be discussed in the following section. Splunk actions include but are not limited to:

- Firing an alert for a positive search result and then indexing its metadata.
- Rewriting summary data to a separate index for further correlation.
- Urgency identifiers that allow for a correlative data point.
- Identification of consecutive and similar events.
- Engaging a script that can be used for external communications to ticketing systems.

In taking what we know about the Ransomware Kill Chain, Splunk's functionalities, and Linux's Tooling, the following table outlines a few of the key concepts or ideas that can be used to create Splunk alerts. This table focuses explicitly on the lifecycle of generic ransomware and is intended to provide layers upon layers of possible true positive matches. It is essential that we remember that no single signature or alert should stand as a catch-all for determining ransomware. Table 1 includes a number of probable means to isolate and trigger based on ransomware activities within an enterprise environment.

Infection stage:	• Possible creation and running of processes inside of the /tmp directory • A new process that has never been seen on the endpoint prior that has external network connectivity requests • Files being renamed multiple times in the same directory tree. • Large process tress where the Parent Process is terminated prior to the child processes • Escalation of privileges or attempts to gain sudo access • Use of the strings cmd to attempt to encode communications prior to or during infection.
Staging stage:	• File names generated with entropy or that are in a quick succession • Modification of boot options • Attempted communications with DNS names where entropy is found or directly with bare IPs • Creation of .sh files inside of non-user-based home directories • Use of the chmod cmd to change executable files to overly permissive rights • Change in distribution Yum or Apt repositories
Scanning stage:	• Quick and successive directory enumeration

	• High memory usage for short or sustained periods of time. Focus being given to outliers and events that fall outside of the typical day to day operations. • External as well as internal network communications • The use of wget or curl cmds • Enumeration of shared web, database, or file storage directory trees
Encryption stage:	• Files created with odd file extensions • Multiple modifications within a single directory • Multiple copies of the same file in multiple directories • Possible calls for encryption libraries • Removal of files from directories using wildcards or without confirmation • The use of chmod with wildcards

Table 1 Possible Kill-chain events for Splunk based detections

As the number of possible alerts for a specific topic may seem significant, it is fundamentally important to have a comprehensive list of all possible characteristics for ransomware or any other malicious code. When the number of probable scenarios increases, the likelihood of coverage also intensifies. The counter-intuitive nature to this idea is offset as alerts feed directly to operations teams contributing to the eventual "bubbling up," which will be discussed next.

4.1.2. "Bubbling up" and searching over your notable index data again

The power of correlation in Splunk is the ability to historically look back over time and compare any and all results that have individually already triggered. One of the essential ideas that is needed to separate from the traditional model of Antivirus and Antimalware is to be able to tell if something odd occurred in the recent past. Having a contextual awareness of if an event that did occur applies to what is happening right now strengthens any security posture as it links all events historically together. If a Security Operations team can draw a correlative conclusion on all facts that have occurred and provide those to an analyst, that individual can draw either a positive or negative conclusion about the situation.

Splunk's indexing power plus the search action of writing metadata back to the indexer at the time of a triggered event might be a bit repetitive, but streamlines our abilities in detecting trends of ransomware over time and will provide factually relevant information for further correlation. As an example, the following search pictured in Figure 2, reviews data that has already been triggered and written to our notable index.

```
index=notable Alert_Name!=*Notable*
| eval original_host=coalesce(HOSTS, host,orig_host)
| stats count by original_host, Alert_Name
| stats values(Alert_Name) count by original_host
| where count>=3
| collect addtime=true index=notable sourcetype=collect source="Detection:Notable of Notables
for single host" marker="Alert_Name="Notable of Notables for single host"
```

Figure 2: Notable of Notable Search

This search reviews the data that is in our notable index. It removes any Notable events that have occurred prior and then filters based on hostname. In addition to the detections metadata that is re-indexed, our search also provides results where three or more detections have triggered. As the alert pictured above is for testing purposes, customization for individual needs as threshold settings typically should be tailored to each environment.

The correlative power that is found within this notable search would be a monumental and challenging task to ask of operations staff. To perform this manually would require a typical Security Operations staff member to comb through thousands of events each day. While leveraging Splunk, these correlations become automatic, not only saving time but achieves the sought-for result of this exercise "critical events that are bubbled to the top." As those most critical of critical events bubble up, the difference between ignorance in a real compromise or decisive actions becomes clear and concise. Pictured in Figure 3 is a visual representation of how a bubbling would be tracked based on the normal flow of host telemetry.

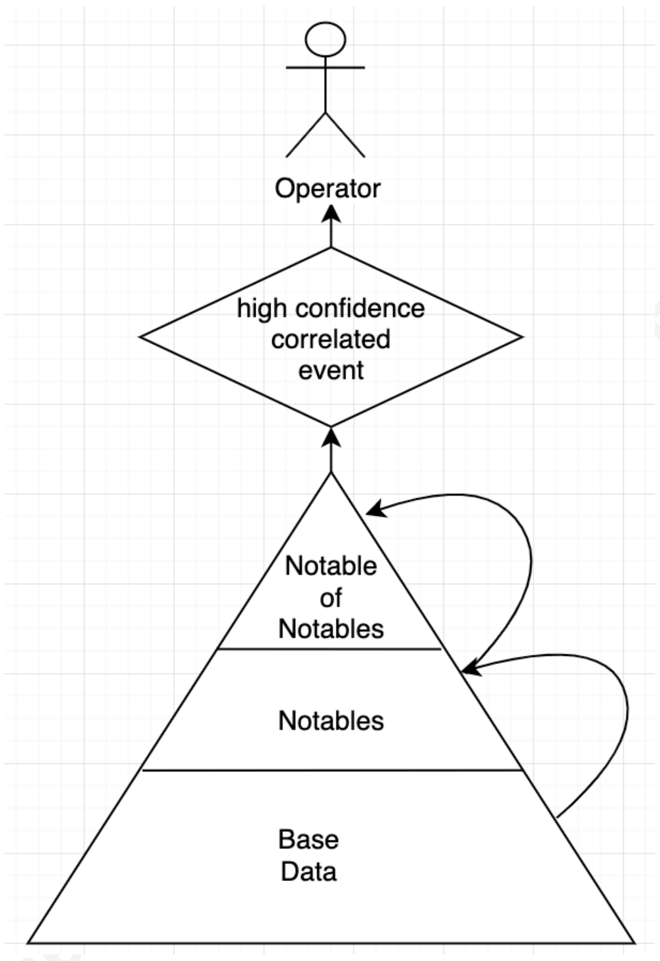


Figure 3: Bubbling up function of Splunk

4.1.3. The end goal - Increasing probability and confidence

To make such an endeavor a reality for many organizations, it will be paramount that ardent attention is given to the concepts and ideas presented pertaining to log ingestion and parsing. Many security groups have and will continue to use the mindset that one large detection with multiple aspects will be able to catch all malware. Others will continue with the methodology that Splunk is an event collector and that all events can be indexed via Splunk. This is possible, but not nearly as effective as using many small detections that are hyper-focused to create the coverage that is needed to be near the theoretical 100% that is the goal for any organization.

When the events and variables that correlate are garbled with background network noise and other host-level logging, the theory of effective alerting is unequivocally undermined. Data flow with any SIEM or related product is easily polluted when careful consideration of data is overlooked. Security teams must avoid at all costs the lackadaisical idea of ingesting data because “it goes into Splunk easily.” In order to make ransomware detection a real possibility while using Splunk, the utmost diligence must be given to ensure that end to end data flow is precise and that detections, alerting, and the notable of notable reviews are clear and accurate. Security teams must abide by the principle that only meaningful data should be ingested into Splunk. Avoiding data ingestion of all log types will pollute indexes and generate events that do not match log parsing rulesets. As these pitfalls are avoided, Operations staff will be able to focus exclusively on the data pertinent to detecting and alerting on malicious code execution.

With this exercise trying to prove the above-discussed concepts, twenty-five alerts were made within the testing environment. These detections focused on the concepts that were addressed in Table 1 of this paper and will be used as a baseline for detecting the activities of ransomware on the test endpoint labeled “Sheepone.” For further review of these detections and the logic that they express, please see the Appendix.

4.1.4. Care and feeding + False positives

It is also of note that just like with any security product, false positives are a fact. The daily care and feeding of one’s data, as well as Splunk in this case, needs to be monitored and tuned. As tuning and detection writing is beyond this paper's scope, these two topics will not be discussed. However, they also play pivotal roles in how Splunk continues to produce important and worthwhile alerts for detection teams everywhere. Data is ever-changing, and creating something within one's environment to soon have it undermined by a shift in log types or log formats would result in a noisy and costly exercise. When data changes or false positives are found, it is imperatively important that proper attention is given to tune and care for one's environment.

5. Decomposition of Results

As a means of relying only on the Linux auditing capabilities, the test environment provided a perfect occasion within which the tests of this thesis could be effectively measured. Two virtual machines that communicated with each other and relayed logging to a sample Splunk Search head were the only dedicated tools needed to accomplish the goal of signatureless and device-less detection of ransomware.

Unanticipatedly, the phase in which the ransomware samples were executed proved to be painfully tricky within the lab environment. The ransomware that was collected for this experiment proved difficult during the phase of infection due to its desire to only send communications to command servers that would respond. These responses unfortunately, were needed prior to launching any encryption activities by the multiple stains of ransomware that were attempted. Because said control servers were either shut down or were outdated and refused to respond, a proper ransomware execution was unable to be achieved during testing. Samples such as ERUBUS, Lilock, and Linux Encoder were tried multiple times and with multiple different base code variations. While each execution was different, these samples did each successfully run and established persistence within the Oracle Linux 7 image if but to only start to beacon home.

Despite various setbacks, the Linux detection set of twenty-five detections created for this thesis proved useful in catching and detecting activity regardless of the issues encountered with the ransomware samples. Twenty-five total detections were created, and with each iteration of ransomware or change in malware, an apparent infection and consequential alert was visibility correlated within Splunk for the prey virtual machine “Sheepone." As observable activates such as command and control, running from /tmp and, consistent calls to other directories and applications became evident from within Splunk’s indexed data, “Notable of Notable” events successfully bubbled to the top of the alerting stack.

As a particular note of success, the auditd and sysstat configurations proved useful in catching and tracing the process trees initiated and used by all users who were fictionally created on the endpoint. Sysstat’s ability to provide frequent CPU information was used to create the detection labeled as “Encryption-004 Use of Crypto” in the hopes that it would trigger during the infection stage of ransomware. This was undermined by the issues experienced while trying to establish the second stage of the ransomware application.

To ensure that the detections and alerts generated were not that of fluke happenstance due to the setbacks in establishing C2 communications, other miscellaneous Linux-based viruses were used as well. This generic infection attempt was to ensure that individual events would trigger and be alerted on by Splunk regardless of ransomwares presence or not. Samples attempted included but were not limited to; botnets, coin miners, and rootkits. Each was used as a means to ensure that the detection logic used in the twenty-five unique detections would trigger an alert when suspicious activity was taking place. Due to the rich nature of the content created for this research, all screenshots created Splunk queries, and information pertaining to testing activates can be found in the Appendix of this paper for further review and download.

As was stressed in the above presentation of ideas, triggering events can be easy and allows for single events to be investigated as a "moment-in-time" event. However, most importantly, as correlation happens, these events become more than just single data points; they start to form a story of what an attacker or piece of malicious code is doing on an endpoint. This unifying fact is what truly helps to provide the confidence that the results of this exercise are worthwhile and can be built upon in the future by security teams everywhere. As further time permits, the ideas discovered and proved herein can be used as a launching pad into other detection sets or masteries. Detection of ransomware was the primary goal, but having had such success can have a far-reaching effect on how signatureless activities should become one's first line of defense within an enterprise environment.

6. Conclusion

Moving from a signature centric model for malware or, importantly, ransomware detection to that of relying on the Linux operating system to provide meaningful results might prove to be a fear-inciting concept for seasoned security practitioners. From what can be determined or viewed of the current threat landscape, the metaphorical Band-Aids that we have placed over the requirement for detection signatures needs to be ripped off. Vendors have, for too long, received ample compensation for their extremely poor efforts to provide secure, future proof coverage for those who need security most. Moving past this crutch of a third-party solution relies heavily on our personal ability to be in control of our own security. As a security community, and as seasoned professionals, taking back what is rightfully ours will be difficult. The concept of being our own security vendor should not though, be viewed with the eyes of impossibility, but rather confidence, possibility, and freedom. As ransoms have been issued and Antivirus and Anti-malware companies have been the only means of finding and identifying malice, enterprise security teams have done themselves a disservice by allowing someone else to be responsible for what should be theirs and only theirs. Having proved the possibility of creating a focalized solution for any Linux environment, security groups may now detach from the financial noose that holds them hostage and secure their own environments in ways that only they should be able to do.

Proving that being independently in control of ones alerting and detection set has required only a small set of opensource tools. With such accessible tools, success can be found when the right ideas and capabilities are brought together, and the responsibility for attacks is brought back to its origin. As experts building upon the standards of Linux and its open-source mantra, developing tools and practices will prove to be the security standard now and into the future. Cutting the strings that vendors and devices have used to control our detection and containment capabilities needs to end, and can so long as focus remains within the confines of one's own network.

References

Balaban, D. (2020, Jun 24). *Linux Ransomware - Notorious Cases and Ways to Protect*. Retrieved from Hacked: https://hacked.com/linux-ransomware-notorious-cases-and-ways-to-protect/

Chuang, T. (2020, Feb 03). *How SamSam ransomware took down CDOT and how the state fought back - twice.* Retrieved from Colorado Sun: https://coloradosun.com/2020/02/03/how-samsam-ransomware-took-down-cdot-and-how-the-state-fought-back-twice/

ESET. (2017, JAN 06). *Destructive KillDisk malware encrypts Linux machines, ESET researchers discover*. Retrieved from ESET:

https://www.eset.com/us/about/newsroom/press-releases/destructive-killdisk-malware-encrypts-linux-machines-eset-reseachers-discover/

Exabeam. (2016, JUL 01). *The Anatomy of a Ransomware Attack*. Retrieved from Exabeam Threat Report: https://www.exabeam.com/wp-content/uploads/2017/07/Exabeam_Ransomware_Threat_Report_Final.pdf

FBI. (2019, Oct 02). *Alert Number I-100219-PSA*. Retrieved from HIGH-IMPACT RANSOMWARE ATTACKS THREATEN US BUSINESSES AND ORGANIZATIONS: https://www.ic3.gov/media/2019/191002.aspx

Lockheed Martin. (2011, JUL 01). *LM White Paper Intel Derive Defense*. Retrieved from Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains: https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf

Morgan, S. (2019, Jun 10). *Global Cybersecurity Spending Predicted To Exceed $1 Trillion From 2017-2021*. Retrieved from Cybercrime Magazine: https://cybersecurityventures.com/cybersecurity-market-report/

Red Hat. (2020, JUL 16). *CHAPTER 7. SYSTEM AUDITING*. Retrieved from Red Hat Documentation: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security_guide/chap-system_auditing

The Linux Foundation. (2020, JUL 16). *Linux is the most successful open source project in history*. Retrieved from The Linux Foundation: https://www.linuxfoundation.org/projects/linux/

Trend Micro. (2017, Jun 15). *Erebus Linux Ransomware: Impact to Servers and Countermeasures*. Retrieved from Trend Micro: https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/erebus-linux-ransomware-impact-to-servers-and-countermeasures

Willis, M. (2018, Feb 18). *CDOT Cyber Incident.* Retrieved July 2020, from Colorado.gov: https://www.colorado.gov/pacific/dhsem/atom/129636

Appendix

All configurations and other documents associated with this paper can be referenced and viewed at: https://sites.google.com/view/richardhorne/home