National and Kapodistrian University of Athens

School of Science
Department of Informatics and Telecommunications

BSc Thesis

LoRa: Combining different schemes to enhance threat hunting and incident analysis

Christos G. Dimitriou

Supervisors: **Alexis Delis**, N.K.U.A Professor
Dimitris Mitropoulos, Adjunct Faculty
Spyros Papageorgiou, Cyber Defence Directorate Director

ATHENS
JUNE 2018

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Σχολή Θετικών Επιστημών
Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Πτυχιακή Εργασία

LoRa: Ενισχύοντας τη διαχείριση απειλών και συμβάντων συνδυάζοντας διαφορετικές προσεγγίσεις

Χρήστος Γ. Δημητρίου

Επιβλέποντες: Αλέξης Δελής, Καθηγητής ΕΚΠΑ
Δημήτρης Μητρόπουλος, Επισκέπτης Καθηγητής
Σπυρίδων Παπαγεωργίου, Διευθυντής Κυβερνοάμυνας ΓΕΕΘΑ

ΑΘΗΝΑ
ΙΟΥΝΙΟΣ 2018

Bsc THESIS

LoRa: Combining different schemes to enhance threat hunting and incident analysis

Christos Dimitriou
S.N.: 1115201000066

Supervisors: Alexis Delis, N.K.U.A Professor
Dimitrios Mitropoulos, Adjunct Faculty
Spyros Papageorgiou, Captain Cyber Defence Directorate Director

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

LoRa: Ενισχύοντας την διαχείριση απειλών και συμβάντων συνδυάζοντας διαφορετικές προσεγγίσεις

Χρήστος Δημητρίου
Α.Μ.: 1115201000066

Επιβλέποντες: Αλέξης Δελής, Καθηγητής ΕΚΠΑ
Δημήτρης Μητρόπουλος, Επισκέπτης Καθηγητής
Σπυρίδων Παπαγεωργίου, Διευθυντής Κυβερνοάμυνας ΓΕΕΘΑ

ABSTRACT

Cyber threat hunting is the process of proactively and iteratively searching through networks and systems to detect and isolate advanced threats that evade existing security solutions. In this thesis, we seek to meet the growing need for methods that can help us with threats, which are automated but also configurable to help with the project. The tools Loki, Rastrea2r with the help of Yara Rules are combined and expanded in an effort to create a tool capable of helping security engineers of this modern era. The best elements of the first two are taken into creating the LoRa and meet the needs of even large commercial environments.

SUBJECT AREA: Threat Hunting and Incident Response

KEYWORDS: Threat Hunting, Security, Indicators of Compromise, Yara Rules

ΠΕΡΙΛΗΨΗ

Το κυνήγι απειλής του κυβερνοχώρου είναι η διαδικασία της προορατικής και επαναληπτικής αναζήτησης μέσω δικτύων και συστημάτων για τον εντοπισμό και την απομόνωση προηγμένων απειλών που αποφεύγουν τις υπάρχουσες λύσεις ασφάλειας. Σε αυτή την πτυχιακή εργασία, επιδιώκουμε να καλύψουμε την αυξανόμενη ανάγκη για μεθόδους που μπορούν να μας βοηθήσουν στο κυνήγι απειλών, που να είναι μεν αυτοματοποιημένες αλλά και διαμορφώσιμες ώστε να βοηθήσουν στο έργο. Τα εργαλεία Loki, Rastrea2r με τη βοήθεια των Yara Rules συνδυάζονται και επεκτείνονται σε μια προσπάθεια να δημιουργηθεί ένα εργαλείο ικανό να βοηθήσει τους ερευνητές της σύγχρονης εποχής. Τα καλύτερα στοιχεία αυτών των δύο πρώτων έχουν ληφθεί για τη δημιουργία του LoRa και την κάλυψη των αναγκών ακόμη και μεγάλων εμπορικών περιβαλλόντων.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Κυνήγι απειλών και απόκριση συμβάντων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Κυνήγι Απειλών, Ασφάλεια, Δείκτες Κινδύνου, Κανόνες Yara

AKNOWLEDGMENTS

I would like to thank Prof. Alexis Delis for giving me the opportunity to work on this project.

Also, I thank my supervisor Dr. Dimitrios Mitropoulos for his assistance in completing this thesis.

Finally, I am very grateful for the guidance and help offered by Captain Mr. Spyridon Papageorgiou, as well as, many thanks to all members of the Cyber Defence Directorate for their support and knowledge.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον καθηγητή Αλέξη Δελή ότι μου έδωσε την ευκαιρία να εργαστώ σε αυτό το έργο.

Επίσης, ευχαριστώ τον επιβλέποντα καθηγητή Δημήτρη Μητρόπουλο για τη βοήθειά του στην ολοκλήρωση αυτής της διατριβής.

Τέλος, είμαι πολύ ευγνώμων για την καθοδήγηση και βοήθεια που προσέφερε ο Πλοίαρχος κ. Σπυρίδων Παπαγεωργίου, καθώς επίσης και ευχαριστίες προς όλα τα μέλη της Διεύθυνσης Κυβερνοάμυνας για την υποστήριξη και τη γνώση τους.

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

The chances are exceptionally high that covered up threats are as of now in any organization's networks. Organizations can't bear to accept that their security measures are imperfect and vulnerable, no matter how intensive their security safety measures may be. Having a border and guarding it is not sufficient since the border has blurred absent as new advances and interconnected gadgets and gear have risen. Prevention systems alone are inadequately to counter concentrated human enemies who know how to urge around most security and monitoring instruments by, for instance, making their assaults look like normal activity. Prevention frameworks and devices offer assistance that decreases openings for adversaries and enable analysts to function more successfully. The key, however, is to continually hunt for attacks that get past security frameworks and to capture interruptions in advance instead of after attackers have completed their tasks and done more harm to the business. This act is alluded to as "cyber threat hunting". Numerous organizations nowadays do some type of formal or casual hunting. For example, instead of waiting for any notice indicating a breach, they are discontinuously or continually looking through their own networks for proof of dangerous activity. To obtain a deeper understanding of the subject, we will further analyze topics and answer questions related to its nature, the reason and time we perform the task, along with its place inside a chain of diverse cyber security action groups [1].

1.1 What is Threat Hunting?

It is about setting a fitting, dedicated focus on the exertion by examiners who intentionally set out to recognize and counteract adversaries which will as of now be within our environment. Threat Hunting requires some specific expository aptitudes, such as recognition with the enterprise and the capacity to generate and examine speculations. Hunting benefits from examiners utilizing automation to make these hunts faster, simpler, more periodic and more precise [2]. Therefore, Yara [3] is a tool aimed at (but not limited to) helping malware researchers to identify and classify malware samples. With it we can create descriptions of malware families based on textual or binary patterns. Each description, known as rule, consists of a set of strings and a boolean expression which determine its logic. Yara can also leverage more complex and powerful rules can be created by using wild-cards, case-insensitive strings, regular expressions, special operators and many other features that you'll find explained in its online documentation.

1.2 Motivation

Behind every threat there is a human. It is the adversaries, not just their tools, such as malware, that interest threat hunters. They are persistent and adaptable and frequently evade network defenses. The threats are regularly recognized as advanced persistent threats (APTs), not just because of the capabilities that the enemies use, but also of their ability to start and keep up long-term operations against targets. Focused and supported foes will not be countered by security boxes on the network alone. And threat hunters are not essentially holding up to respond to alarms or indicators of compromise (IOCs). They are effectively looking for dangers to anticipate or minimize harm [4].

1.3 When do we hunt?

Bringing threat hunting into development requires a security stance that incorporates the tools, people, forms and buy-in from decision makers that empower defenders to hunt. The organization ought to set ground rules with respect to roles, duties and the way in which risk chasing will be utilized. For example, the hunt group ought to not be seen as a one-stop shop for taking care of each issue on the organize. For that reason, the organization must make TH portion of its general security methodology and order it and understand it from the top down. Simply put, TH is available to all, but an organization must be mature enough to induce a legitimate return on investment from it and make it a process that is repeatable and consistent [5].

1.4 Where Threat Hunting fits in

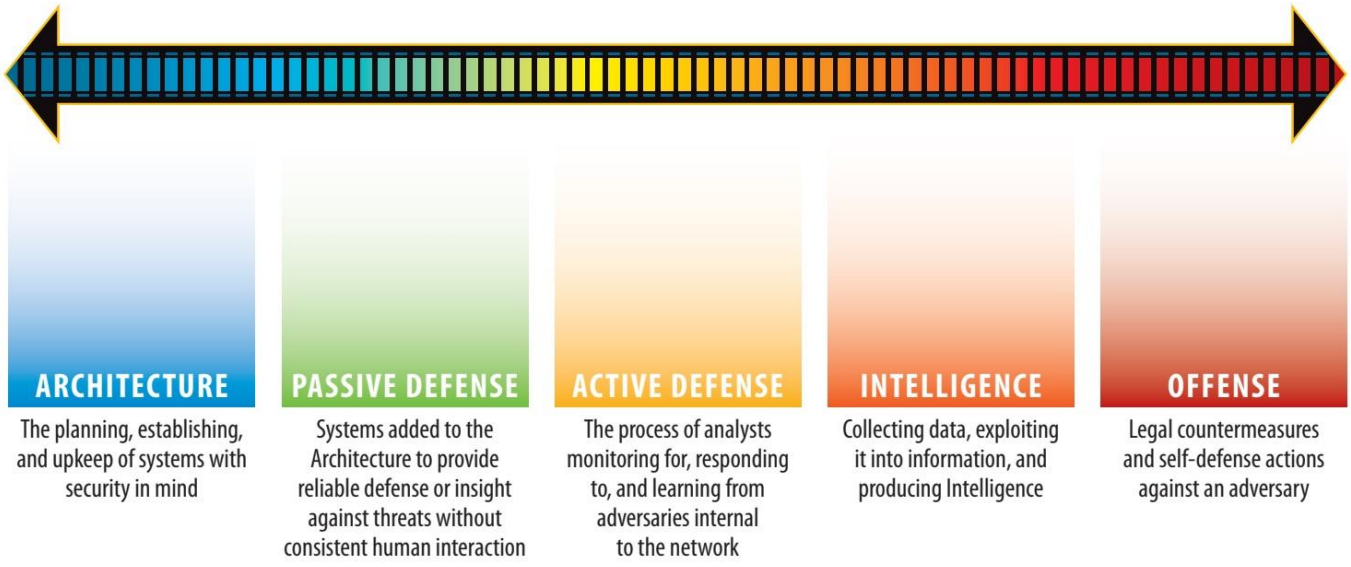


Figure 1: The Sliding Scale of Cyber Security

The Sliding Scale of Cyber Security [6], in Figure 1, is a model for providing a nuanced discussion to the categories of actions and investments that contribute to cyber security. An organization that has not until now endowed at the left part of the scale, for example supporting a security engineered architecture, will never see high return on investment into fields to the right such as active defense. Personnel occupied on active defense are looking to respond to, monitor for, and learn from adversaries internal to the network cannot exert at this effectively without proper architecture and passive defenses that enable the environment to be visible while concurrently dismissing the noise of trivial assets that are not worth spending time on. On the other hand, it is not necessary for organizations to fully invest in one category before advancing to the next, but it is crucial to be understood that investments in the left side of the scale has a direct impact on the capabilities of the right side. Passing few steps and going straight to intelligence too soon will not solve anything. There is certain order and filtration in which the data collected are actually made into intelligence as shown in Figure 2 below. In any case, each organization must consider the investment value of each component as presented in figure above in its own environment. TH is part of the active defense category and integrates a few of the leading items of intelligence. As suitable and different investments are made along the scale - such as monitoring infrastructure to enable active defense activities like TH – the group produces significantly more yield. In that way, it is a movement that continually provides expanding value to organizations as they develop in their maturity and is a field that has become prominent in the recent years. Unfortunately, most organizations are still reacting to alerts and incidents instead of proactively seeking out the threats and this is one aspect this project tries to solve. Nonetheless, TH itself cannot be fully automated; the act begins where automation ends, although it heavily leverages it. As this first stage of reconnaissance completes, the second phase of response commences [4, 7].

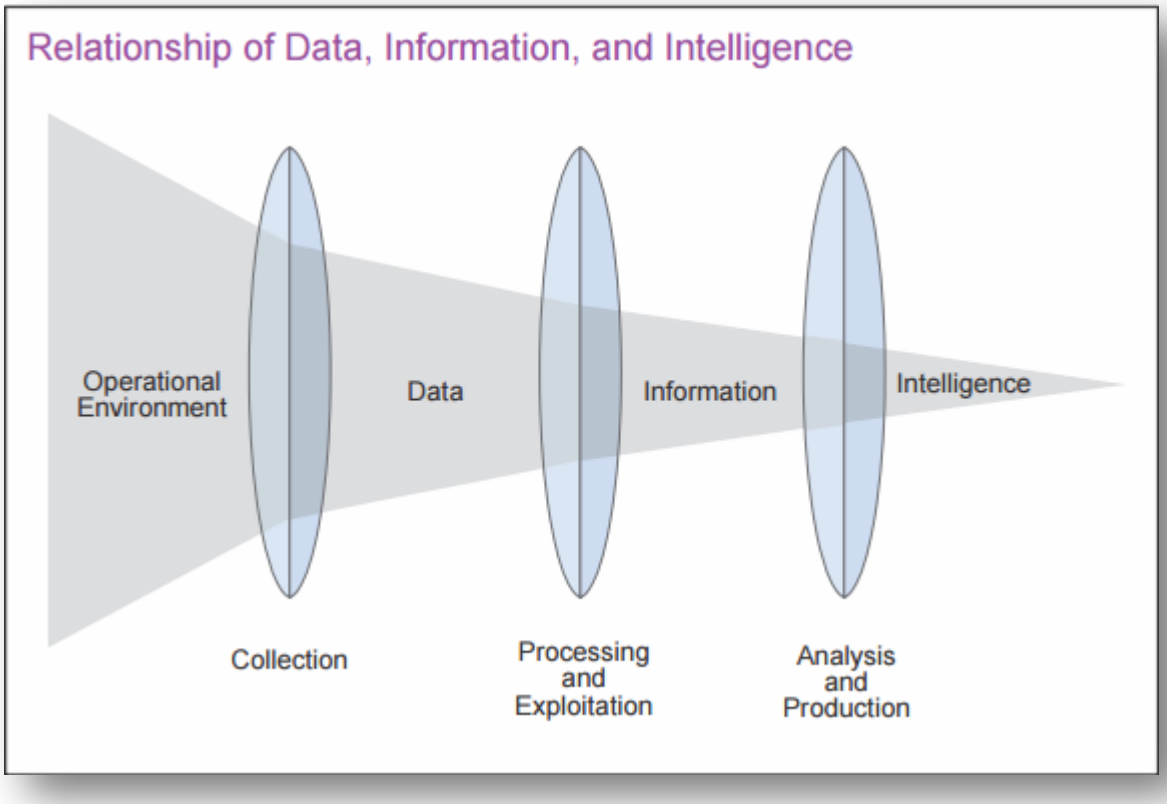


Figure 2: The difference between Data, Information, Intelligence and their stages of obtainment

1.5 Loki Summary

Loki [8] is a scanner for *Simple Indicators of Compromise*. It follows four different methods:
1) File Name IOC Regex: match on full file path/name,
2) Yara Rule Check: Yara signature match on file data and process memory,
3) Hash check: Compares known malicious hashes (MD5, SHA1 and SHA256) with scanned files,
4) C2 Back Connect Check: Compares process connection endpoints with C2 IOCs.

Additional Checks:
- Regin file system check
- Process anomaly check
- SWF file decompressed scan
- SAM dump check
- DoublePulsar check, tries to detect DoublePulsar backdoor on port 445/tcp and 3389/tcp
- PE-Sieve process check

1.6 Rastrea2r Summary

Rastrea2r [9] (pronounced "rastreador" - hunter- in Spanish) is a multi-platform open source tool that allows incident responders and SOC analysts to triage suspect systems and hunt for Indicators of Compromise (IOCs) across thousands of endpoints. To parse and collect artifacts of interest from remote systems (including memory dumps), rastrea2r can execute sysinternal, system commands and other 3rd party tools, collecting triage information, across multiples endpoints for automated or manual analysis. By using a client/server RESTful API, rastrea2r can also hunt for IOCs on disk and memory across multiple systems using YARA rules.

1.7 LoRa

LoRa is an acronym standing for Loki – Rastreador, the tools used for making a new better and more complete program used in IR. The best characteristics of the first two have been taken into consideration. Starting from the common functionalities, scanning with yara was present in both projects. As a result, some basis had to be chosen and after examination I adjusted the methods of Loki as I found they were more thorough. The client – server communication was a feature only Rastreador had which was added as well as the baseline functionality, memory dump, prefetch and web history acquisition. The operations not existing and therefore added will be mentioned and described below.

The main resource from which we draw information about threats we want to search in our endpoints is Yara Rules, which are JSON like formatted structures and hold the indicators of malware or any threat presence. Through analysis and forensics the scientific community has managed to produce a large database of indicators of compromise. Vendors such as MISP, Threatexpert [10], Github repositories, Mandiant and Alien Vault host such databases with plethora and up-to-date entries.

In this thesis we are going to focus on samples gathered from different Github repos by different vendors. The file formats vary from yara files, csv, pdf, txt, ioc and even excel. Some of the indicators that help us in our hunt are MD5s, SHA1, SHA256, IPs, domains, filenames, file size, registry values, and mutexes.

In the Evaluation section we will introduce the results of our experiments regarding the efficacy an efficiency of our methods. We are going to show that we can catch those indicators and alert the user accordingly.

2. ANALYSIS OF SYSTEM REQUIREMENTS

The architecture of our system involves five basic components, namely a Server, a Client, an Upgrader and two parsers [11], one for extracting IOCs from ClamAV [12] files and one for obtaining them from the ThreatExpert site. This decomposition is not only for the sake of functional analysis, but also serves to identify common mechanisms and design elements across the various parts of the system. We will start by presenting different types of Views from the 4+1 view model of architecture [13] in order for the architectural structure to become comprehensible. The physical view is not something of a constraint and thus won't be reported. As for the use cases, the section of evaluation will provide plenty.

2.1 Architecture

2.1.1 Logical View

The logical view primarily supports the functional requirements – the services the system should provide to its end users. Designers decompose the system into a set of key abstractions, taken mainly from the problem domain. These abstractions are objects or object classes that exploit the principles of abstraction, encapsulation and inheritance. In addition to aiding functional analysis, decomposition identifies mechanisms and design elements that are common across the system. The actor represents the end-users and as a result the considerations taken into presenting this view are what the system should provide in terms of services to its users. On the Figure 3 below we present the classes or filenames which can be used by the users autonomously, for the purpose of updating the signature database and examine closer the information or rules obtained before the actual operation, or by the workflow of the server or the client. We use a simple approach representing each individual class as each template focuses on each individual class; they emphasize the main class operations, and identify key characteristics. The first field of the template is the name of the file or class we are referring to, the second consists of the class fields and the third contains the class/file functions that may be user by the user may or the workflow of an operation we choose.

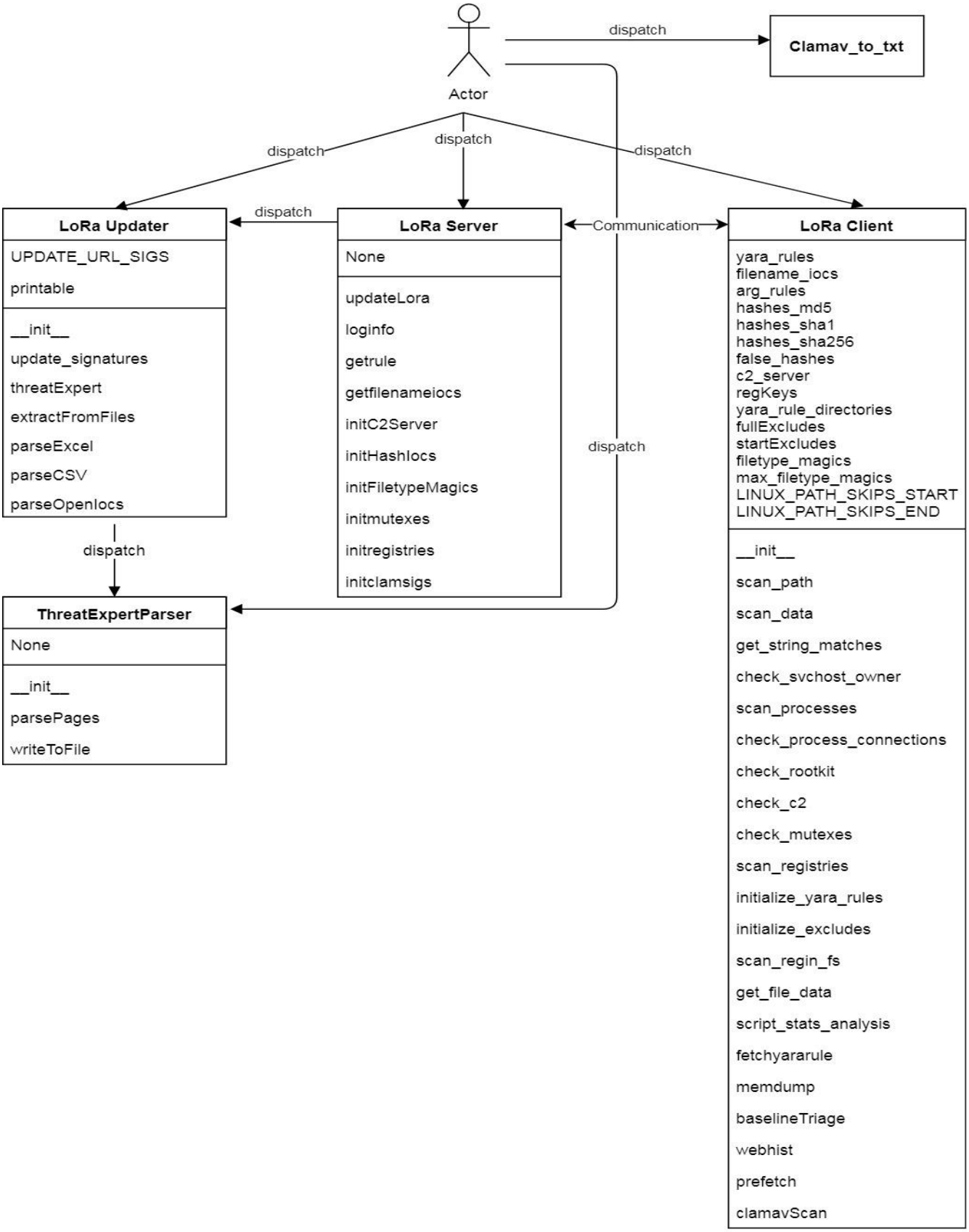


Figure 3: Logical View of LoRa

2.1.2 Process View

The process view takes into account some non-functional requirements, such as performance and availability. It addresses issues of concurrency and distribution, of fault tolerance, and how the main abstractions from the logical view fit within the process architecture—on which thread of control is an operation for an object actually executed.

By default, CherryPy's [14] internal web server is going to create a thread pool with ten worker threads each carrying out a request received by the clients, and it won't grow beyond that automatically but instead through manual configuration. It also has a default listen queue of five, so essentially can backlog up to 50 requests before they start being rejected. Communication between the different components was pictured on the logical view and beside the further description of each one, due to the simple nature of this commune there is no need for a detailed examination. Every error or exception occurred during the runtime is captured and depending on the occasion, the execution is halted, showing an appropriate message to the user, or it continues adopting a default value informing the user about the incident.

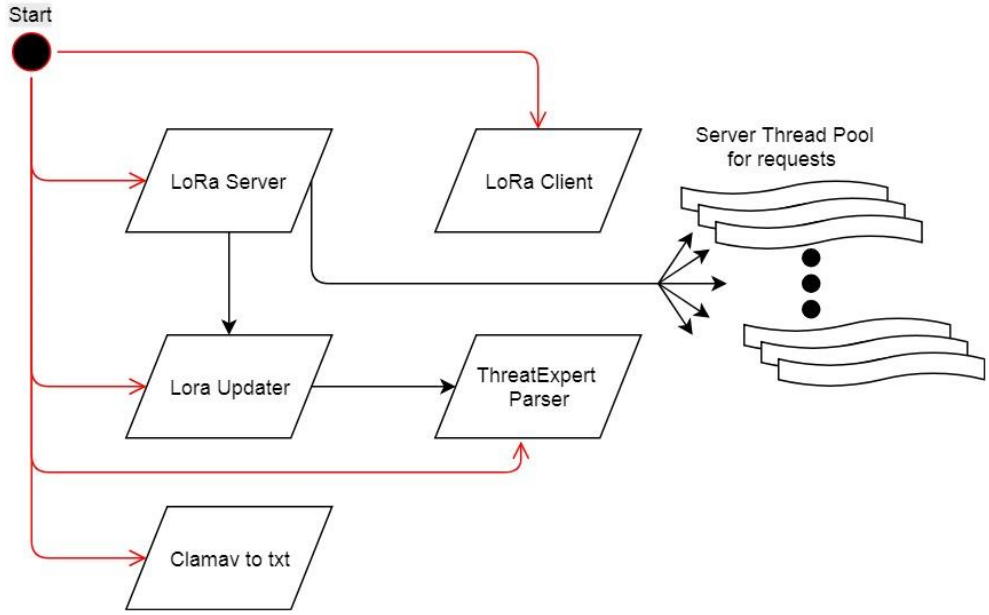


Figure 4: Process View of LoRa

Server process is responsible for serving the signatures to the clients and updating the data base though the lora-upgrader.py.

Client process chooses the type of operation he wants to execute and exchanges messages with the Server.

Upgrader process is responsible for downloading signatures and parsing those needed.

ThreatExpert Parser process collects and extracts signatures from the requested number of pages from the site and updates the database

2.1.3 Development View

The development view focuses on the organization of the actual software modules in the software-development environment. It takes into account internal requirements related to ease of development, software management, reuse of commonality and constraints imposed by the toolset or the programming language. Also, it supports the allocation of requirements and work to teams and supports cost evalutation, planning monitoring of projects progress and reasoning about software reuse, portability and security. It is the basis for establishing a line of product. The development view illustrates a system from a programmer's perspective and is concerned with software management. This view is also known as the implementation view. Below, in Figure 4 we can assemlbe the main development entities and a brief deploy course ending in an X which points out the termination of the running processes. The dotted black lines represent in a timeline how to normal deployment of each element of the project shall commence, while the red ones are optional for the user depending on which signatures he or she wishes to update. The green ones show the association between the project files meaning that one may call procedures of the other one, while the glod ones state the update of signatures in our database. On the right corner of the lora_win32.py, the agent loop signalizes the ability to restart the scanning process after a given time period performing all or a collection of the available procedures.

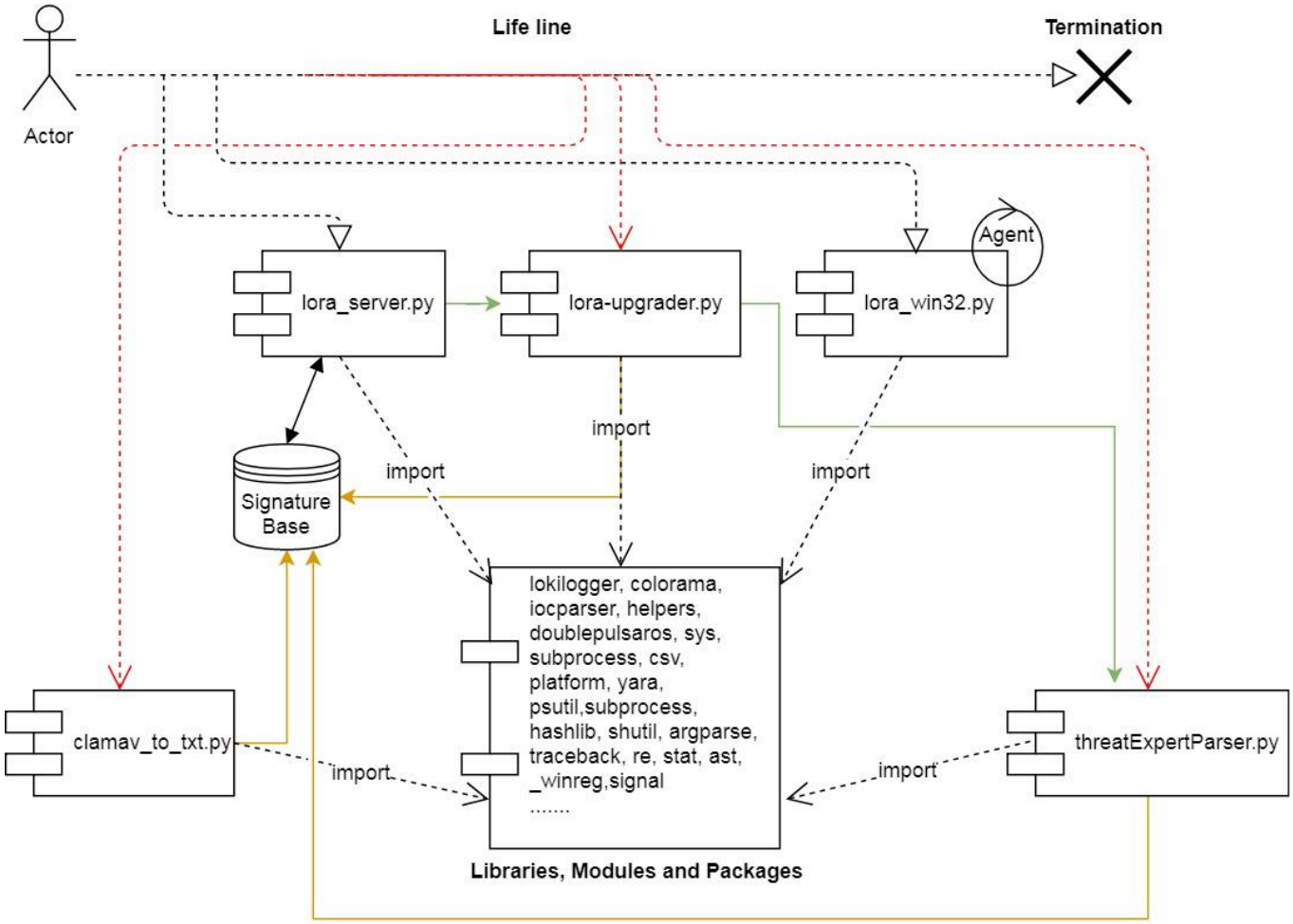


Figure 5: Development and Deployment View

Packages and Modules used

Client :lokilogger, helpers, doublepulsar, colorama, tests
Server: iocparser, iocp, lokilogger, colorama, win32com.shell

Class libraries

For the client side, the imported libraries are the following:
os, sys, subprocess, csv, platform, yara, psutil, subprocess, hashlib, zipfile, shutil, glob, argparse, traceback, re, stat, ast, _winreg, signal, time, binascii, from collections import Counter, from requests import post, from sets import Set, from mimetypes import MimeTypes

For the threat expert parser, the imported libraries are the below.
os, sys, tempfile, urllib2, re, datetime, from bs4 import [BeautifulSoup, SoupStrainer], from sets import Set

For the clamav_to_txt, the imported libraries are the following:
sys, os, re, from optparse import OptionParser

For the server side, the imported libraries are the following:
time, bottle, codecs, socket, sys, os, re, traceback, yara, stat, psutil, argparse, subprocess, collections, wmi, win32api

Execution Environments

The whole project was run under Windows 7 and Windows 10 as well as Ubuntu 14.04 LTS and Ubuntu 16.05 LTS.

2.2 Functionality Description

2.2.1 Server:

lora_server.py file

Usage: lora_server.py [-h] [-s server-address] [-p] [--update] [--threxp]

Optional arguments:

-h, --help show this help message and exit
-s Server address
-p Port
--update Update the signatures from the "signature-base" sub repository
--threxp Search and parse the threat expert website for signatures

The purpose of server is to send the rules asked by the client inside the signature-base/ folder. It loads .yar, .yara and .txt files and returns them as JSON. The client then has to compile some of them in order to start scanning. Filenames, hashes, host names, ip addresses, mutexes and registries are loaded from the signature-base and passed to the clients. The cherrypy server was selected for this job as it was the simplest and efficient choice for python and parallelism of requests from clients.

Except from the repositories hosting the necessary files composing our database for the scan, another source is the ClamAV official signatures as well as unofficial signatures. The clamav-unofficial-sigs [15] are third-party signature databases provided by Sanesecurity, FOXHOLE, OITC, Scamnailer, BOFHLAND, CRDF, Porcupine, Securiteinfo, MalwarePatrol, Yara-Rules Project, etc. This was achieved by downloading and then following the instructions for the initialization of its database. Then, with the help of existing tool “sigtool” [16] we were able to extract the information from the .cvd files. From the extraction we get many other files with different extensions. Those that interest us are the .ndb and .hdb. The first are hexadecimal signatures used in hunting files and the latter one is MD5 signatures. Finally, our script clamav_to_txt.py is called and parses the files extracting either MD5s or hex part-values of malware and exports them into txt files that can be used later for scanning. This method must be done manually if a user wants to integrate the ClamAV signatures to the scan process. As a note the name of the cvd when passed as argument to sigtool had to be without extension and just the name without any full or relative path.

2.2.2 Client:

lora_win32.py file

usage: lora_win32.py [-h] [-s kilobyte] [-l log-file] [-r remote-loghost]
[-t remote-syslog-port] [-a alert-level] [-w warning-level] [-n notice-level]
[-y yara-rules] [--agent] [--clamav] [-timeint time-interval] [--printAll]
[--allreasons] [--noprocscan] [--nofilescan] [--scriptanalysis] [--rootkit]
[--noindicator] [--reginfs] [--regs] [--mtxes] [--dontwait] [--intense]
[--csv] [--onlyrelevant] [--nolog] [--debug]
{all,mem-scan,mem-dump,baseline,web-hist,prefetch,disk-scan}

LoRa - Simple IOC Scanner

Positional arguments:
{all, mem-scan, mem-dump, baseline, web-hist, prefetch, disk-scan} modes of operation

all	Start all the scans
mem-scan	Scan the memory
mem-dump	Make dump file of current memory
baseline	Creates a baseline to compare states at any time
web-hist	User account to generate history for
prefetch	The Windows prefetch folder is a specific location within the Windows operating system (OS) that contains a series of small files detailing the startup activities and frequently-used application programs.
disk-scan	Disk scanning operation

Optional arguments:

-h, --help	Show this help message and exit
-s kilobyte	Maximum file size to check in KB (default 5000 KB)
-l log-file	Log file
-r remote-loghost	Remote syslog system
-t remote-syslog-port	Remote syslog port
-a alert-level	Alert score
-w warning-level	Warning score
-n notice-level	Notice score
-y yara-rules	Yara rule files to be checked
--agent	Start the LoRa agent
Use the clamav official and unofficial sigs to scan files matching	hex signatures
-timeint time-interval	Time interval for LoRa agent to begin checking, default is 60 mins
--printAll	Print all files that are scanned
--allreasons	Print all reasons that caused the score
--noprocscan	Skip the process scan
--nofilescan	Skip the file scan
--scriptanalysis	Activate script analysis (beta)
--rootkit	Skip the rootkit check

--noindicator	Do not show a progress indicator
--reginfs	Do check for Regin virtual file system
--regs	Do check for registries in the system malware creates
--mtxes	Do check for mutexes in the system malware creates
--dontwait	Do not wait on exit

Intense scan mode (also scan unknown file types and all extensions)

--csv	Write CSV log format to STDOUT (machine processing)
--onlyrelevant	Only print warnings or alerts
--nolog	Don't write a local log file
--debug	Debug output

The client starts with some optional arguments and one mandatory, the last is the type of scan or action that will be executed. Help for each operation can be displayed for example with the command *python lora_win32.py disk-scan -h.* Summarizing the core functions of the client, we have:

disk-scan: Traverses from the path given as argument and down and checks for filenames, hashes and then does a yara check matching the content with the compiles rules that were collected.

mem-scan: For all the processes in the system performs a file name check matching malicious process names as well as checks for open connections matching ips, domains and hosts. For certain processes does a special check manually (e.g. lsm.exe, winlogon.exe, svchost.exe)

mem-dump: Using the tool winpmem a .img is created for further investigation

baseline: Using a set of tools (Sysinternal and others) located in w1000/tools directory we manage to create files describing a baseline for future comparison of the state of the endpoint. That is important if we want to check for any alterations in our system between two different moments.

web-hist: for all or a specific user, it collects using the browsinghistoryview.exe from a variety of browsers the web history and stores them in csv files with column data (URL, Title, Visit Time, Visit Count, Visited From, Visit Type, Web Browser, User Profile, Browser Profile, URL Length, Typed Count).

prefetch: The Windows prefetch folder is a specific location within the Windows operating system that contains a series of small files detailing the startup activities and frequently-used application programs. Using the winprefetchview.exe we get the display of the list of files stored inside the selected Prefetch file, which represents the files that were loaded by the application in the previous times that you used it.

2.2.3 Upgrader:

Usage: lora-upgrader.py [-h] [-l log-file] [--nolog] [--debug] [--threxp] [-f F] [-t T] [-v V]

Optional arguments:

-h, --help show this help message and exit
-l log-file Log file
--nolog Don't write a local log file
--debug Debug output
--threxp Search and parse the threat expert website for signatures
The number of the first page signatures will be downloaded, default is 1
The number of the last page signatures will be downloaded, default is 10
The minimum threat level of the signatures will be downloaded from a scale 0-5, default is 3, e.g. with threat level 3 signatures with level >=3 will be downloaded

Given links from where we can get signatures the upgrader downloads and extracts them to the signature-base folder. The formats of files vary from .yar, .yara, .txt, .pdf, .csv, .xls, .xlsx and according to their type are handled accordingly to extract the information into .yar or .txt files that we use in our hunt. For text and pdf files we have the IocParser class [11] while for the other types I use custom parsers defined inside the upgrader.

2.2.4 IocParser:

usage: iocp.py [-h] [-p INI] [-i INPUT_FORMAT] [-o OUTPUT_FORMAT] [-d] [-l LIB] PATH

positional arguments:

PATH File/directory/URL to report(s)

optional arguments:
-h, --help show this help message and exit
-p INI Pattern file
-i INPUT_FORMAT Input format (pdf/txt/html)
-o OUTPUT_FORMAT Output format (csv/tsv/json/yara/netflow)
-d Deduplicate matches
-l LIB PDF parsing library (pypdf2/pdfminer)

Parsing tool for extracting indicators from pdf and txt files with the help of pdfminer library and exporting them as yara rules. With the help of patterns.ini file inside data folder it can understand different indicators and export them.

2.2.5 ThreatExpertParser:

Threatexpert.com is a threat analysis system designed to analyze and report the behavior of computer viruses, worms, trojans, adware, spyware, and other security-related risks. By creating a parser which with regular expressions manages to collect indicators including mutexes created from processes and registries, other than hashes and file names. As arguments it accepts a fromPage and toPage integer argument indicating from which page until where it shall parse. A threat level integer meaning what type or dangers and above shall look at.

3. EVALUATION

In this section I will report the results of my work with reference to the methodology I discussed in the previous sections. I seek to bolster my arguments concerning the validity of my statements and claims I made in the first section of this thesis and can now be substantiated as well. I will try to addresses a single claim at a time by applying my methodology, reporting a result and discussing the result in the context of my thesis statement.

In Figure 3 we show the result returned after starting a scan to one of our systems directory, in this case a directory with two files within. The operation we chose after launching lora_win32.py was the disk-scan using the simple yara rule:

```
rule myrule
{
strings:
        $String1 = "myvirus" condition:
    $String1
}
```

We were able to match the contents of the testing file virus.txt. It is important to note that even though we have put an executable named “myvirus” inside the directory we have no match. That is because the yara scan searches the contents of a file. For the purpose of picking suspicious files based on their names, we have different files in our signature-base containing this info accordingly as we will see in the next test cases.

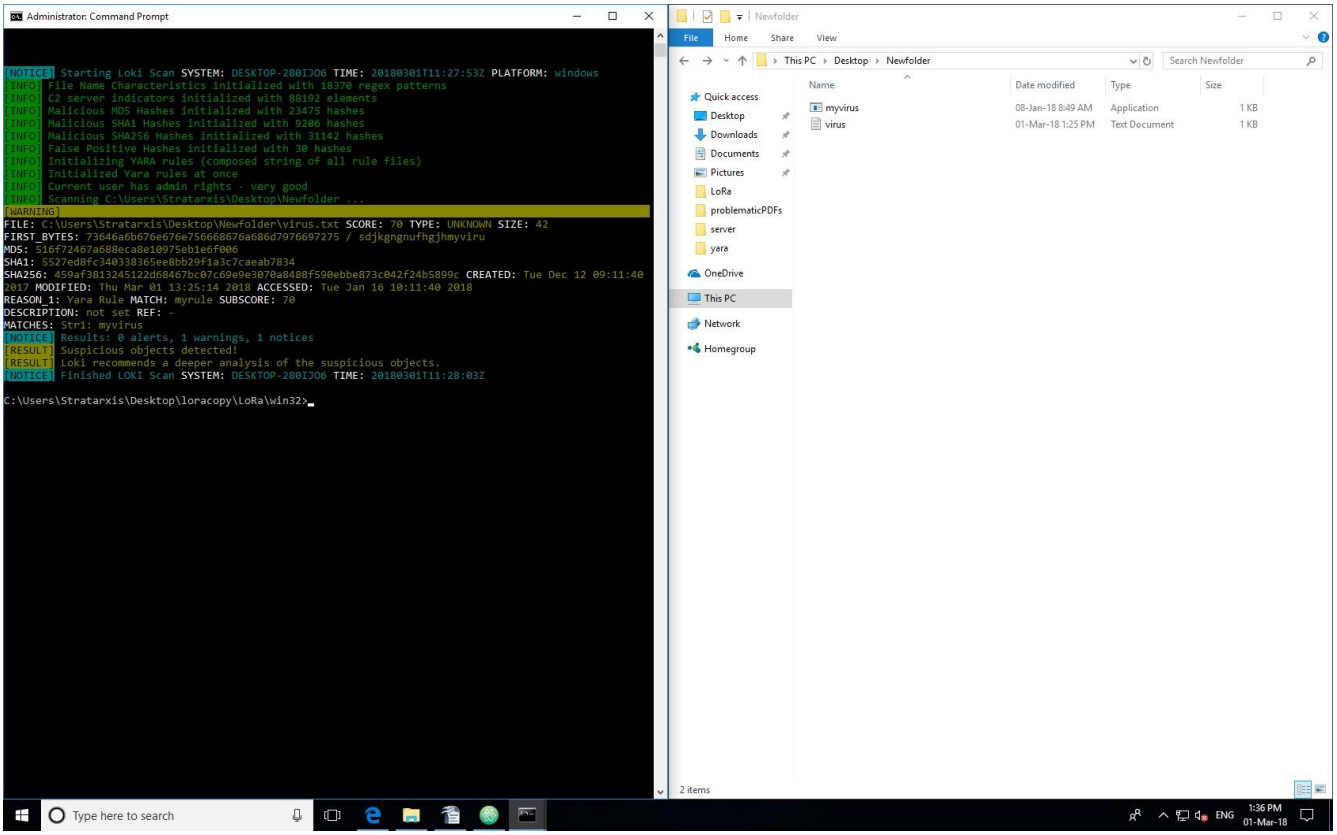


Figure 6: Matching the contents of a file with certain text string

On the right side of Figure 3 we see the folder we are about to scan and on the left side the command prompt executing the client, passing the initialization phase with the INFO tags representing each step. The yellow label indicates a warning, a finding worth investigating further as it may be a potential threat. The level of severity in our data base is usually given along with the name or path of frequent malicious objects, but it can also change fitting our needs. The message we collected from the warning has the full path of the object, its score (level of severity) which is the sum of all the reasons this object is suspicious, type of file and size, the sequence of first bytes and its ASCII representation separated with a forward slash, its MD5, SHA1 and SHA256 digests, the creation, modification and accessed times. Reason is from which side of our data base the warning is coming from, in our case we had a yara rule match. In other test cases we can have for example, MD5 hash match or file name match. The match tag declares the name of the rule, if matched by a yara rule. The subscore is the contribution of this certain match to the total score summed by all the reasons an artifact is considered suspicious. Along with every yara rule, most of the times, there is a description along with comments from the various authors giving details about the threat the rules are about to confront and explaining in few words what trails they are trying to detect. The matches tag states which exact data, consisting of hexadecimal, text or regular expression strings from the yara rule were matched.

In our next test case we launched the scanning procedure but this time with no yara rule given as input and only the database files containing dangerous file names with or without common paths they often take residence. When searching for the filename of the myvirus.exe file I got a match again using the regex patterns as shown in Figure 7. The difference between the colors we first encounter here is designed to separate the warning level findings and the alert, which is more severe, level of hazard. Of course, some of the prints on the left side of Figure 4, inside the command prompt depicting the file paths read from the database and of the files being scanned are solely for the purpose of showing the deviation between the simple strings and the interpretation of regular expressions. The opened file on the right is the actual database file containing the regex patterns of suspicious objects. Most of our files incorporating information such as this also have a comment section explaining its structure. In general, each line is a new entry carrying unique data. Most of the times separated at the end with a semicolon from its score. If not given, then LoRa assigns the entry a standard scoring value of 70, placing as a warning but also borderline between notice and warning level, according to the standard scoring levels which can be changed by the user.

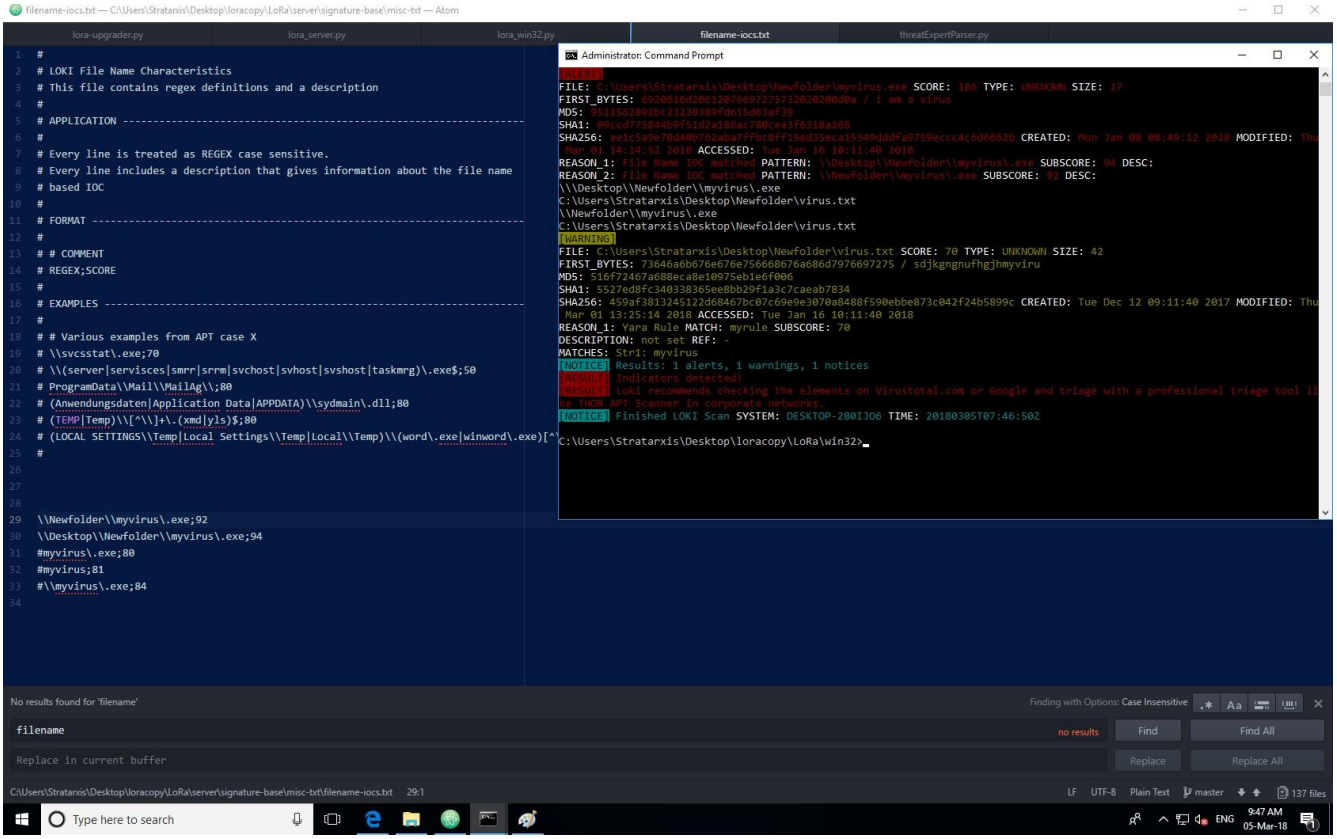


Figure 7: Matching the path – location of certain file using regex patterns

In our database there are also files consisting of MD5, SHA1 and SHA256 hashes of dangerous files. The format followed by these is similar to the previously discussed case of file name holding files. There is a comment section which exists in some files as it is optional describing the composition of the data inside. Below that, there are the different entries optionally separated by a semicolon from a comment giving further details about each hash, for example the APT it belongs to and its original file name. In order to test the scenario of discovering malicious hashes, I added the three hash values (MD5, SHA1, SHA256) of a test file I created inside the data base and launched the scanning process giving the path accommodating the test file. Despite the fact we have all hashing values, only one showed up as an alert message inside the command prompt. That is the value of the MD5 hash which is the first value computed in our program and since we examine a certain file, one match is enough to consider it evil.

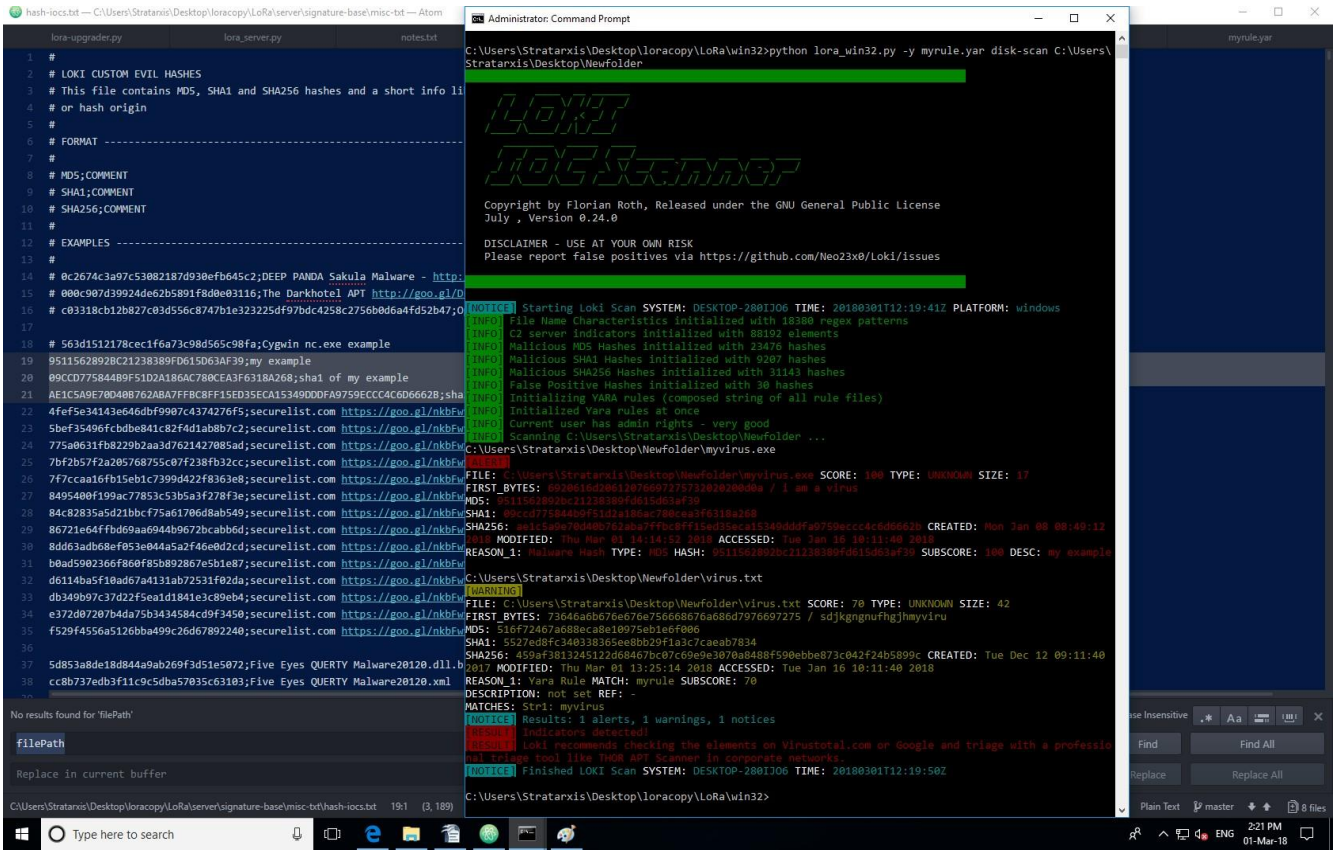


Figure 8: Matching the MD5, SHA1 and SHA256 of certain file

For the purpose of scanning processes, we utilize yara rules, check for possible process connections as well as potential malicious names of them. In Figure 6, the command prompt on the right side of the screen shows infos and alerts. The first simply inform us about the current process being examined along with some information such as the process id and the command line arguments, while the latter warn us about probable threats with the same additional information. In this instance, we used the regex expression atom\.exe and the result we got back was the detection of our text editor process Atom. On the left side of Figure 6 we present some elements that are taken into consideration when triaging a given process, for example the process id, its name, command line, parent process id, priority, working set size, execution path and owner.

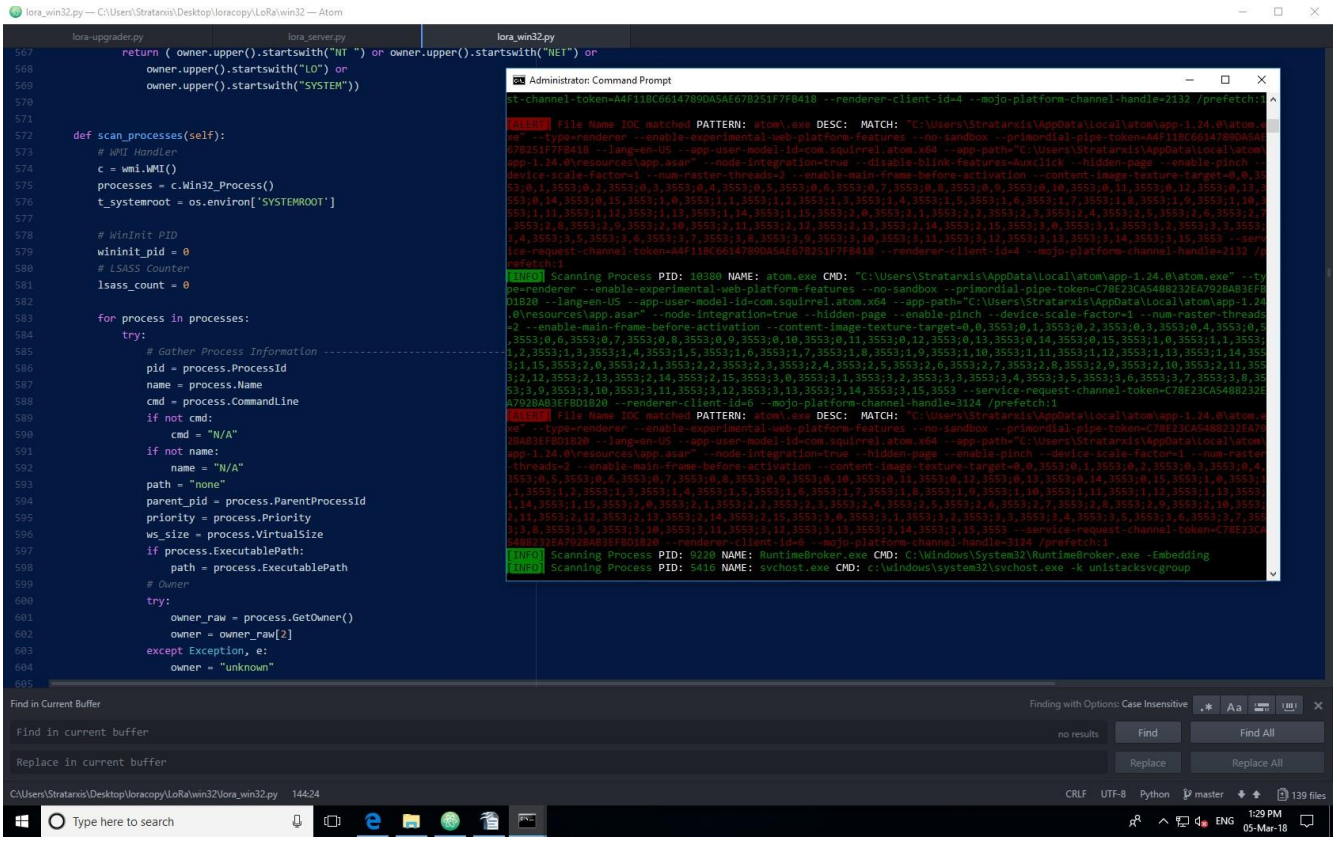


Figure 9: Matching process names using regex or txt files

In addition, many known malicious programs when executed they create mutex objects in the compromised systems. With the help of threatExpert.com and its online database which we employ, we can obtain more than a few indicators of mutex values we must pay heed to. To accomplish this we use the windows sysinternal handle.exe executable. After its execution we manage to get all mutex objects in our system. Comparing the entries in our data base and the results of handle.exe we can conclude if there is anything suspicious. In Figure 7, on the left side of the image we can see our mutex scanning method and on the right we observe the alert raised because of a testing entry found in our environment.

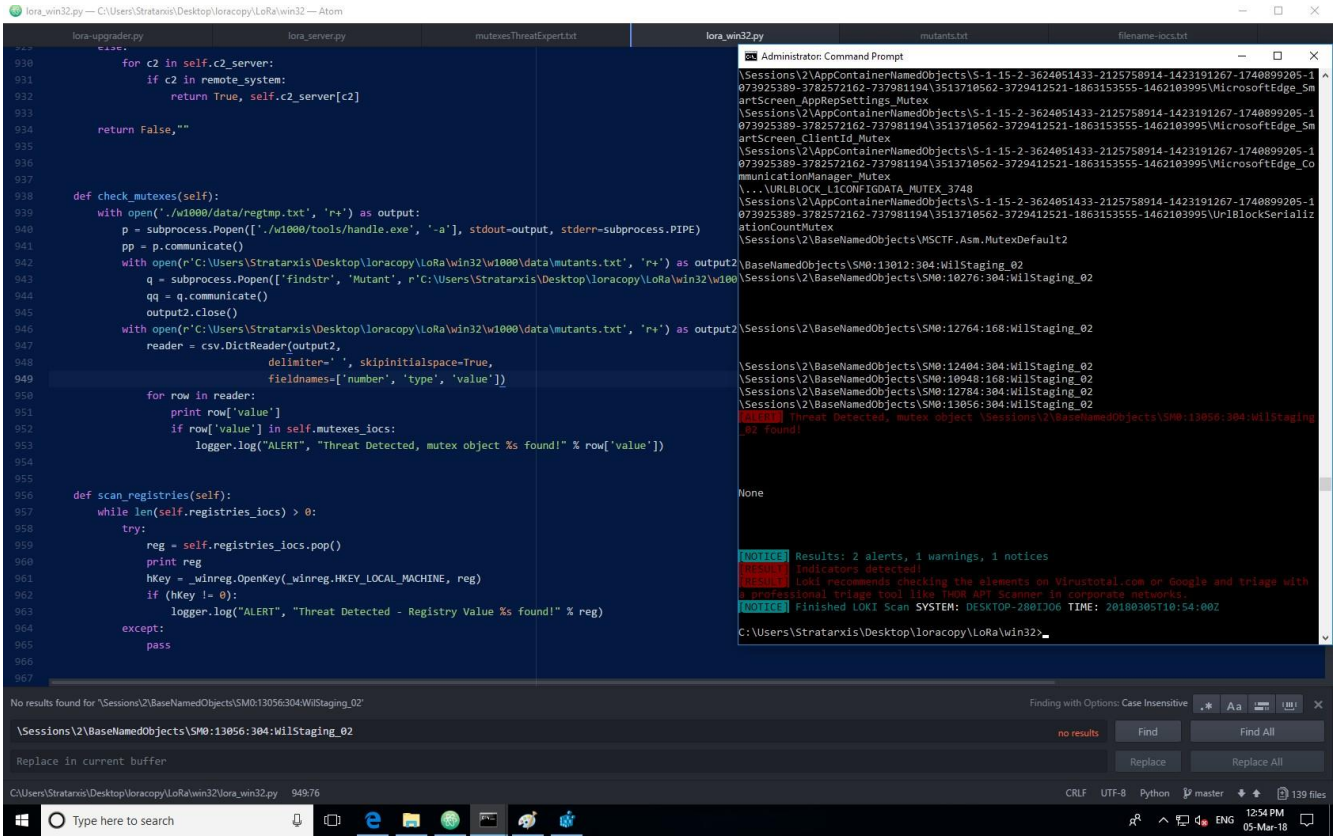


Figure 10: Matching certain mutex value and raising alert

In parallel with the previous case, again with the help of threatExpert.com we obtain values of registry keys created by malicious programs. In Figure 8, we can see the method scan_registries, used in the scanning process on the left side of the picture and the results of the execution inside the command prompt on the right side. Taking an existing value from the registries and putting it in the signature base to load it; we managed to get a match.

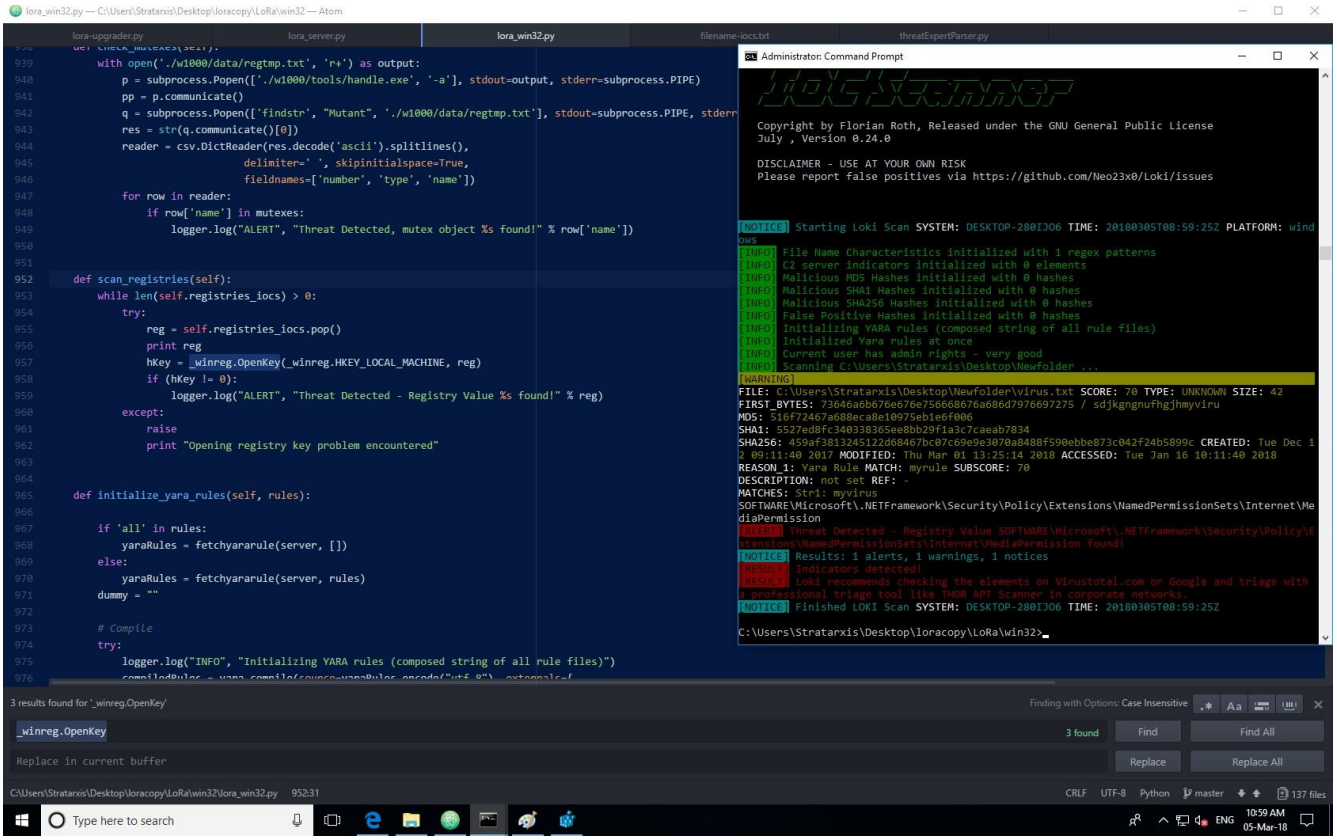
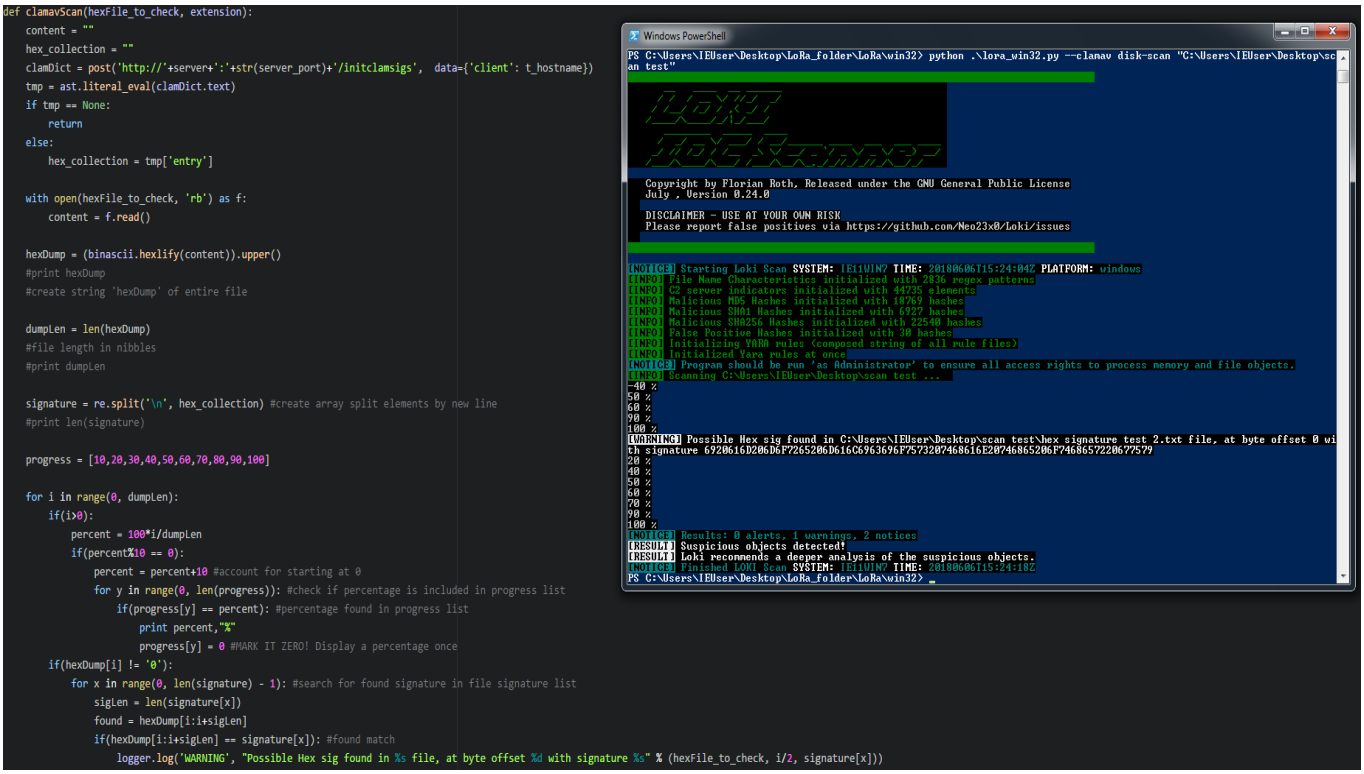


Figure 11: Matching certain registry value and raising alert

To automate finding file signatures within a hex file, we utilize the clamavScan method that can be seen on the right side of Figure 9. The test hex file may also represent a hard drive image and so it could too be used for an actual image file. The method handles two files, first being a CSV file that includes all the file signature values we are looking for. This file is in the following format: File Signature, File Extension. The second file is a hex file that we will be searching for said files. I created a test hex file using the online tool HxD – Freeware Hex Editor and Disk Editor by inserting bytes of random data and then inserting a few file signatures in a location throughout the file. When this scan is performed, it opens the hex file to be read as a binary and creates a single string converted to uppercase. It also opens the signatures file and creates an array of all the signatures and their extensions, splitting the elements by commas. The progress bar I incorporated gives an idea of where the found signatures reside in the file. When a signature if found, it displays on screen what percentage and Byte offset it is from the beginning of the file.



```
def clamavScan(hexFile_to_check, extension):
    content = ""
    hex_collection = ""
    clamDict = post('http://'+server+':'+str(server_port)+'/initclamsigs',  data={'client': t_hostname})
    tmp = ast.literal_eval(clamDict.text)
    if tmp == None:
        return
    else:
        hex_collection = tmp['entry']

    with open(hexFile_to_check, 'rb') as f:
        content = f.read()

    hexDump = (binascii.hexlify(content)).upper()
    #print hexDump
    #create string 'hexDump' of entire file

    dumpLen = len(hexDump)
    #file length in nibbles
    #print dumpLen

    signature = re.split('\n', hex_collection) #create array split elements by new line
    #print len(signature)

    progress = [10,20,30,40,50,60,70,80,90,100]

    for i in range(0, dumpLen):
        if(i>0):
            percent = 100*i/dumpLen
            if(percent%10 == 0):
                percent = percent+10 #account for starting at 0
                for y in range(0, len(progress)): #check if percentage is included in progress list
                    if(progress[y] == percent): #percentage found in progress list
                        print percent,"%"
                        progress[y] = 0 #MARK IT ZERO! Display a percentage once
        if(hexDump[i] != '0'):
            for x in range(0, len(signature) - 1): #search for found signature in file signature list
                sigLen = len(signature[x])
                found = hexDump[i:i+sigLen]
                if(hexDump[i:i+sigLen] == signature[x]): #found match
                    logger.log('WARNING', "Possible Hex sig found in %s file, at byte offset %d with signature %s" % (hexFile_to_check, i/2, signature[x]))
```

```
Windows PowerShell
PS C:\Users\IEUser\Desktop\LoRa_folder\LoRa\win32> python .\lora_win32.py --clamav disk-scan "C:\Users\IEUser\Desktop\scan test"

  Copyright by Florian Roth, Released under the GNU General Public License
  July , Version 0.24.0

  DISCLAIMER - USE AT YOUR OWN RISK
  Please report false positives via https://github.com/Neo23x0/Loki/issues

[NOTICE] Starting Loki Scan SYSTEM: IEIVIN7 TIME: 20180606T15:24:04Z PLATFORM: windows
[INFO] File Name Characteristics initialized with 2836 regex patterns
[INFO] C2 server indicators initialized with 44735 elements
[INFO] Malicious MD5 Hashes initialized with 18769 hashes
[INFO] Malicious SHA1 Hashes initialized with 6927 hashes
[INFO] Malicious SHA256 Hashes initialized with 22540 hashes
[INFO] False Positive Hashes initialized with 30 hashes
[INFO] Initializing YARA rules (Composed string of all rule files)
[INFO] Initialized Yara rules at once
[NOTICE] Program should be run 'as Administrator' to ensure all access rights to process memory and file objects.
[INFO] Scanning C:\Users\IEUser\Desktop\scan test ...
-40 %
50 %
60 %
90 %
100 %
[WARNING] Possible Hex sig found in C:\Users\IEUser\Desktop\scan test\hex signature test 2.txt file, at byte offset 0 with signature 6920616D206D6F7265206D616C6963696F7573207468616E207468652046696572206675
20 %
40 %
50 %
60 %
70 %
90 %
100 %
[NOTICE] Results: 0 alerts, 1 warnings, 2 notices
[RESULT] Suspicious objects detected!
[RESULT] Loki recommends a deeper analysis of the suspicious objects.
[NOTICE] Finished LOKI Scan SYSTEM: IEIVIN7 TIME: 20180606T15:24:18Z
PS C:\Users\IEUser\Desktop\LoRa_folder\LoRa\win32>
```

Figure 12: Clamav Scanning method and matched clamav hex signature for a file being processed

4. CONCLUSION

No matter how well protected an organization is, there's no such issue as zero risk, even with rained personnel, correct technology, and tested procedures. It's not conceivable to accurately and methodically, anticipate the kind, recurrence, or seriousness of attacks. Vulnerabilities are published at an ever-increasing rate and because the complexities of technology will increase, thus the chances of vulnerabilities arising are increasing in turn. The character of computers and networking is escalating the initial threat base and is introducing new motivations and capabilities that some years back did not exist. The result is that computer security incidents can and will occur.

IR is clearly the selection for organizations that need to avoid wasting cash and have better shielded data. Whereas reaching a state of total IR is so a journey, it starts by shortening the time to detection. When detecting a compromise, it ensures containment and remediation are to be solely quicker than traditional approaches and more complete. As shown during this whitepaper, any investments in building an IR program means to have the ability to quickly and accurately determine the scope of a compromise, one thing that may lead to the distinction between assuming the worst and knowing what truly happened.

The flexibility to run automated analysis over most company assets at the same time lowers the inquiring effort needed per machine extending the investigative reach. Using these skills, investigators can raise complex questions about the state of their setting and acquire the answers within minutes rather than days. LoRa is an open source tool supporting the major client platforms. It is scalable and allows analysis on all assets at the same time. This will increase forensic readiness by lowering the cost of response and increasing the standard of proof obtained. It's clear that already similar tools such as Google's GRR are beneficial for managing massive scale investigations within the enterprise. The sector of live forensic analysis presents new challenges, like information snapshots and non-interactive analysis, requiring the event of new data models and user interface designs.

ABBREVIATIONS - ACRONYMS

IR	Incident Response
APT	Advanced Persistent Threat
IOC	Indicator Of Compromise
TH	Threat Hunting

REFERENCES

[1] Brownlee, N., and Guttman, E. RFC 2350: Expectations for Computer Security Incident Response. (June 1998). Internet Engineering Task Force, Network Working Group. Available: https://tools.ietf.org/html/rfc2350
[2] SANS Whitepaper Written by Robert M. Lee and David Bianco. August 2016. “Generating Hypotheses for Successful Threat Hunting."
[3] Yara Rules Available: https://resources.infosecinstitute.com/yara-simple-effective-way-dissecting-malware/
[4] Tom Campbell, CISSP, ABCP. An Introduction to the Computer Security Incident Response Team (CSIRT) Set-Up and Operational Considerations 2003.
[5] Mandia, K., Prosise, C., and Pepe, M. 2003. Incident Response and Computer Forensics. McGrawHill.
[6] “The Sliding Scale of Cyber Security,” SANS Reading Room, August 2015, www.sans.org/reading-room/whitepapers/analyst/sliding-scale-cyber-security-36240
[7] SANS. 2013. "2013 Report on Digital Forensics and Incident Response Survey."
[8] Loki Available: https://github.com/Neo23x0/Loki
[9] Rastreador Available: https://github.com/aboutsecurity/rastrea2r
[10] Threat Expert Available: http://www.threatexpert.com/
[11] Ioc Parser Available: https://github.com/armbues/ioc_parser
[12] Clam AntiVirus Available : https://www.clamav.net/
[13] 4+1 view model of architecture available: https://en.wikipedia.org/wiki/4%2B1_architectural_view_model
[14] Cherrypy Web Framework Available: https://cherrypy.org
[15] Clam AntiVirus unofficial signatures Available: https://github.com/extremeshok/clamav-unofficial-sigs
[16] Sigtool man page Available: https://www.mankier.com/1/sigtool