Automating Static File Analysis and Metadata Collection Using Laika BOSS

GIAC (GREM) Gold Certification

Author: Chuck DiRaimondi, charles.diraimondi@gmail.com
Advisor: Christopher Walker
Accepted: February 14, 2018

Abstract

Laika BOSS is a file-centric recursive object scanning framework developed by Lockheed Martin that provides automation of common analysis tasks, generation of rich file object metadata and the ability to easily apply file-based signature detections to identify malicious files through static analysis. While performing triage and analysis of malware, analysts typically perform repeatable tasks using a variety of standalone utilities and use these tools to gather information that will be useful in understanding adversary tools and in developing future detections. This paper will provide guidance to analysts by reviewing concepts core to the Laika BOSS framework, integrating custom Yara rules for file-based detections, searching and filtering scan object metadata, and describing how to develop, test and implement new Laika BOSS modules to extend and automate new functionality and capabilities into the framework. As part of performing this research, new modules and tools will be released to the security community that will enhance the capabilities and value obtained by using the Laika BOSS framework to perform static malware analysis and metadata collection.

1. Introduction

Laika BOSS is a "file-centric intrusion detection system" (Hutchins, Cloppert and Amin, 2009), open-sourced by Lockheed Martin that recursively analyzes file objects by using a modular framework to automate common analysis processes used by analysts to determine if a file is malicious. There are capabilities within Laika BOSS that allow analysts to drive the detection and metadata collection capabilities of objects during various stages of the Cyber Kill Chain such as the identification of weaponization techniques and delivery of malware. Analysts can utilize the Laika BOSS framework functionality and metadata stored during the analysis of objects to better understand the tools that threat actors are using to carry out their mission.

The successful use of Laika BOSS at different stages of the Cyber Kill Chain can provide immediate value to an organization as it helps automate common analyst tasks associated with file and object analysis. During analysis and documentation, it is common for analysts to review various characteristics and properties of a file, both statically and dynamically. While analysis using both methodologies is a common standard and practice, automating the static analysis of files and collection of analysis metadata can greatly help increase the efficiency and accuracy of analysis.

Throughout this research, new modules and tools have been developed that will showcase how analysts can gain additional value out of Laika BOSS by integrating it into their analysis workflow. Case studies will be reviewed to help solidify various discussion points presented throughout this research paper. The intention of this research paper is to understand the benefits of using the framework, how to implement, configure and enhance the framework and how to best utilize the metadata collected throughout the analysis process as a means to analyze and hunt through historical scan metadata.

1.1. Benefits of Using Laika BOSS

Analysts commonly perform various tasks as part of analyzing files or objects. Specific to malware, these tasks are typically broken up into phases, commonly known as static and dynamic analysis. Each of these phases can be broken down further to include both basic and advanced analysis that requires varying skill sets in order to complete analysis successfully (Sikorski, 2012). Many of these tasks are similar in nature and may

involve substituting different tools based on the type of analysis being conducted. Most of the time, the acquisition of data and knowledge as part of analysis is fairly consistent.

One of the benefits of utilizing Laika BOSS is that it enables analysts to perform consistent and accurate static file analysis against objects, whereby a common set of data is collected based on the file type being analyzed. As will be discussed in detail later, Laika BOSS can be configured to analyze objects of a similar type the same way and to collect the same type of metadata. This static analysis capability can be used as a signature detection platform to help identify and classify specific threat actor tools. The type of intelligence gathering specific to tools, as depicted on the Pyramid of Pain in Figure 1, may be more complex depending on the skillsets that exist within an organization but can greatly aid in the understanding of an attacker (Bianco, 2014).

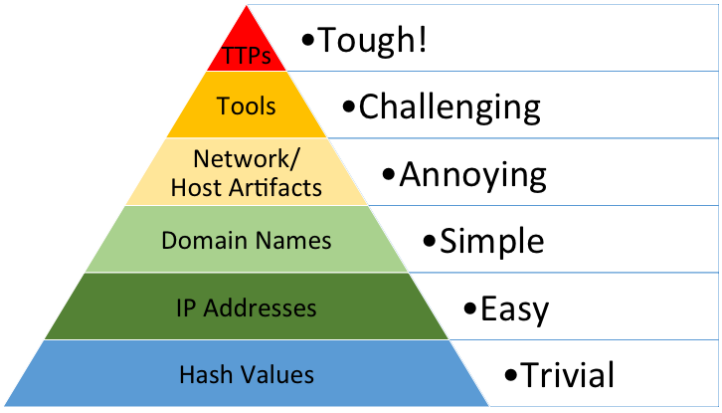

Figure 1 – Pyramid of Pain

Another benefit is the ability to extract and gather file object metadata. This metadata can then be stored for future analysis and searching. By storing the object metadata, this enables an analyst to answer questions such as, “Have I ever seen a file with the same MD5 hash before?”, “Have I ever seen or analyzed a Word document with the same Creator metadata value before?” or “Did I ever see another email analyzed that contained the same X-Mailer or upstream IP address value coming from a different email sender?”. Having access to this type of metadata to answer those types of questions can greatly enhance the ability for the analyst to successfully analyze a particular intrusion or identify a potential campaign based on adversary tactics, techniques and procedures (TTPs).

In Figure 2 below as referenced by Hutchins, Cloppert and Amin (2009), the reader can see the ability to identify common indicators across different activity as a method of potentially identifying related intrusion activity. As part of incident response and analysis, an analyst can search Laika BOSS scan metadata gathered by scanning objects such as emails and files from previous incidents to find indicator overlaps that can help identify potential campaigns. As depicted in the weaponization section in Figure 2, an analyst could have developed a Yara rule to identify an obfuscation technique unique to a particular piece of malware and could have also identified unique atomic indicators such as an X-Mailer or sending email address used by an adversary during the Delivery phase of their campaign. All of these can be placed in Yara rules and utilized by the Laika BOSS framework for detection purposes. The scan metadata can also be used to find overlaps in activity and malware that may not have previously been known due to a lack in detection capability at the time the events occurred. An example could be a weaponized Word document with a particular Creator name and code page value. Based on open source intelligence (OSINT) released in a recent vendor whitepaper, you develop a Yara signature to detect a particular type of malicious Word document. While developing the rule, the analyst performs historical searching within Laika BOSS scan metadata and identifies other documents previously sent in a few months ago that had the same metadata values. During the analysis it was also observed that the email senders were different than the ones being used in the vendor whitepaper. An analyst could then start to develop hypothesis to determine if this activity is part of a larger campaign and can use the newly found data as a source of intelligence in identifying a potential adversary.

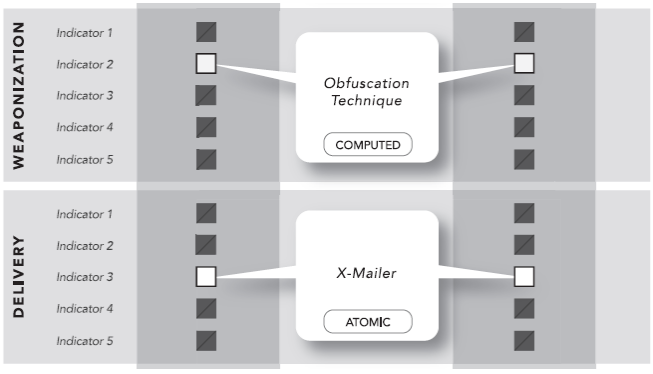

Figure 2 – Common Indicators Across Multiple Intrusions

Laika BOSS also provides a modular framework with the ability for an analyst to enhance the functionality of the framework by adding new object analysis and detection capabilities. These new capabilities can very easily be added to the framework and provide immediate value for analysts in attempting to analyze new attacker tactics and techniques. The customizable nature of the framework allows the analyst to change the analysis workflow based on file types and to specify different types of functionality they want run against those object types. A common technique for attackers during the delivery phase of the Cyber Kill Chain is to weaponize a Microsoft Office document with malicious macros. A common analysis task is to utilize a tool such as olevba (or similar) to extract out the Visual Basic for Application (VBA) macros and to review them in order to identify patterns of maliciousness. An example of enhancing the framework could be adding the capability to automatically extract and decompress VBA macros from a Microsoft Office document by using an EXPLODE module (to be discussed later) to process OLE objects from Microsoft Office documents. An analyst can easily develop a new module utilizing existing python libraries to extract and decompress the VBA objects. These are now new objects that can be scanned by the Laika BOSS framework using Yara to detect malicious VBA code. Figure 3 shows object analysis flows before and after implementing a new EXPLODE_VBA and META_OLEVBA module, including the detection of malicious VBA macros due to SCAN_YARA executing.

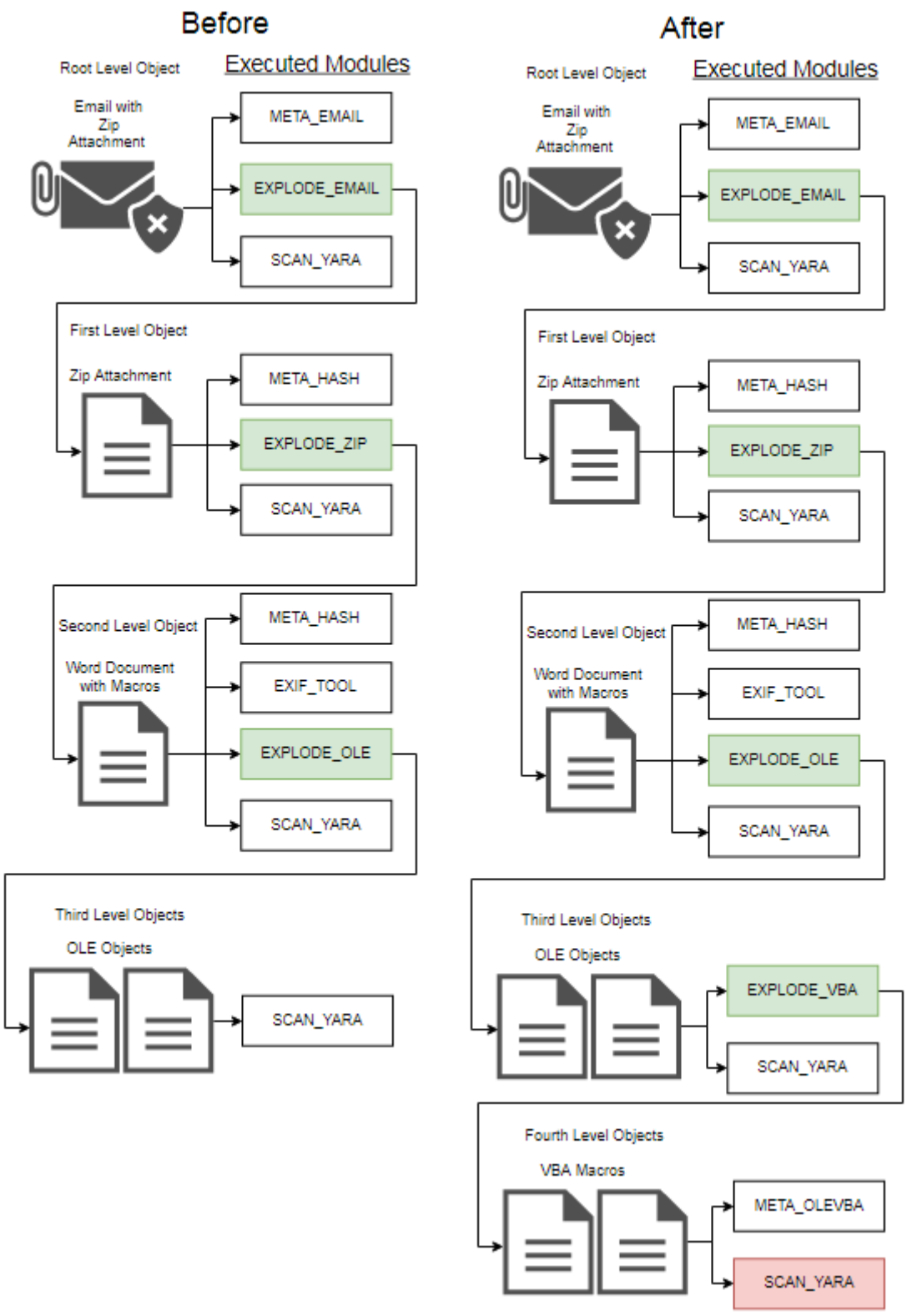

Figure 3 – Object Analysis Workflow

1.2. Operationalizing Laika BOSS As An Analysis Tool

Laika BOSS can be utilized in different ways depending on the requirements of the security organization and the resources available to implement and support such an environment. For the purposes of this paper, Laika BOSS will be described in the context of an analyst tool that can accept files passed to it on the command line by using one of the delivered utilities such as `cloudscan.py` or `laika.py`. This would typically be done when an analyst obtains an email, file or other object that they would like scanned for initial triage and analysis. Laika BOSS would be running as a daemon process by having executed `laikad.py` on the server hosting Laika BOSS. Figure 4 shows a typical configuration.

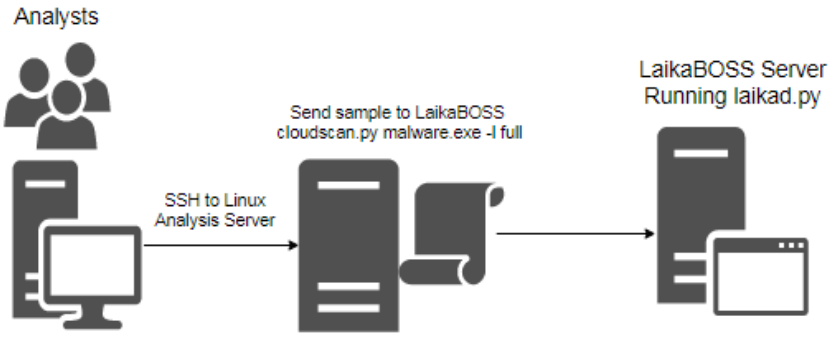

Figure 4 – Analyst Submitting Sample Using cloudscan.py to Laika BOSS Server

In addition to using the standalone utilities to send objects to the scanner on an ad hoc basis, Laika BOSS can be used as a scanner against other network streams that contain SMTP or HTTP traffic in order to extract and scan objects. Configuring and integrating Laika BOSS in this way is beyond the scope of this research paper. The value of integrating the scanner into email and web traffic is its ability to extract objects from emails or web traffic and inspect them using various modules and Yara rules in order to identify malicious objects. One such example would be its ability to identify an email stream, extract out all attachments and recursively analyze them. An example of sending an email into the scanner will be discussed later in this paper. For additional information on configuring Laika BOSS for email and web traffic analysis please reference the `laika_redis_client.py` script on LM Laika BOSS GitHub and the work done by Josh Liburdi who integrated Laika BOSS and Bro (Liburdi, 2017).

1.3. Prerequisites for Utilizing Laika BOSS

In order to obtain the most value from Laika BOSS, it is recommended that analysts are familiar with Python, Yara, jq and Linux. While you don't have to be an expert in using any of these technologies, having a familiarity with all of them will help increase the immediate value obtained by using Laika BOSS. Lastly, a drive to develop new modules and write Yara rules during your malware analysis is all you need!

For this research paper, Laika BOSS was installed on Ubuntu 14.04 in a virtualized environment using 1024 MB of memory and 1 CPU. In order to store the scan result output, MongoDB 3.6.1 was installed on the Laika BOSS virtual machine. The `lbq.py` tool was developed to access the scan result data stored in MongoDB. There were various indexes created against different scan result fields in the MongoDB. The commands used to perform the index creation are listed in Section 10.2 Appendix D – MongoDB Indexes for lbq.py. The default MongoDB database created by LOG_MONGO, which will be discussed later, is named laikaboss.

2. Framework Overview and Configuration

2.1. Laika BOSS Framework

There are multiple files associated with the framework that control its configuration and functionality. The framework itself uses Yara as its configuration language and Python to provide the module functionality. Yara is also used as a signature detection mechanism across any object during analysis. The default installation of Laika BOSS as described in the Laika BOSS documentation results in the main installation of configuration files being located in `/etc/laikaboss`.

```
[/etc/laikaboss]
analyst-> ls
      cloudscan.conf  conditional-dispatch.yara  dispatch.yara  laikaboss.conf
laikad.conf  modules
```

The three .conf files listed above are responsible for setting configuration options for Laika BOSS, cloudscan and the Laika daemon process. The modules folder contains the `disposition/disposition.yara` file and a `scan-yara/signatures.yara` file. These files are responsible for configuring the disposition and Yara rules.

All of the module code itself is stored in the python installation directory `/usr/local/lib/python2.7/dist-packages/laikaboss-2.0-py2.7.egg/laikaboss/modules`. Any new modules developed will get placed into this directory. More details regarding this process will be discussed later when reviewing module development and implementation.

There are different module categories depending on the type of functionality required. These module types are META, EXPLODE, SCAN, EXTRACT, DECODE and TACTICAL. The META modules capture metadata for an object such as file hashes, file metadata, object properties and much more. Examples include META_HASH and META_PE. EXPLODE modules will take an object and extract out any identifiable sub-objects that exist within. These extracted objects are then typically submitted into the scanner for analysis. Examples include EXPLODE_ZIP and EXPLODE_RAR. SCAN modules are used to detect something about an object, typically for identifying malicious characteristics. Examples include SCAN_YARA and SCAN_CLAMAV. EXTRACT modules will extract a specific piece of data from an object. An example module would be one that extracts links from emails. DECODE modules are responsible for transcoding data from one format to another. One example is the DECODE_BASE64 module that decodes base64 character encoded data. DECODE modules can also be utilized for identifying encoded data within malware backdoors. TACTICAL modules are typically used to take advantage of existing scripts or tools that are present on the scanner server that contains functionality already required.

2.2. Configuration Settings

Laika BOSS includes multiple configuration files that are responsible for controlling the functionality of delivered framework tools as well as altering the ways in which the framework identifies and detects malicious objects. There are six main configuration files in Laika BOSS that help control the core framework functionality and delivered tools: `laikaboss.conf`, `laikad.conf`, `cloudscan.conf`, `dispatch.yara`, `conditional-dispatch.yara`, and `disposition.yara`.

2.2.1. laikaboss.conf

The `laikaboss.conf` file is the main configuration file that controls core framework settings. The first section includes general settings responsible for specifying the location of dispatch configuration files, global module timeout settings and the file system location for storing temporary files. The second section specifies the location of Yara rules used by the SCAN_YARA and DISPOSITIONER modules. The last section allows you to configure various logging options. The section that you may want to configure, depending on the type of modules you are executing, are the global timeout settings. Some modules may take longer to run than others and you don't want scans to timeout. Finding the right global timeout setting can help you avoid a timeout issue.

2.2.2. cloudscan.conf

The `cloudscan.conf` file allows you to include configuration information of the remote Laika BOSS server if running the `laikad.py` daemon process. These options can also be set on the command line when executing `cloudscan.py` using the options "-s" and "-a".

2.2.3. laikad.conf

The `laikad.conf` contains two sections of configuration settings labelled Network and General. The network section contains settings for client and worker listening addresses and ports. The general section includes the location of the `laikaboss.conf` file and various framework timeout settings. One of the more important settings related to timeouts are those for the number of maximum objects to process before a worker shuts down and the number of minutes to accept new objects before shutting down. These settings are important to keep in mind because new Yara rules introduced into the framework will not be active until the time to live (ttl) settings, 10000 and 30, are met. If the timing of loading a new Yara rule is of importance, adjusting these settings may help meet your needs. Otherwise you can scan an object using `laika.py` which will cause all rules and configuration options to reload.

2.2.4. Dispatch

The dispatch settings are controlled through the `dispatch.yara` file located in `/etc/laikaboss`. The dispatcher is considered the “brains” of Laika BOSS. Contained in

the dispatch file are multiple Yara rules that help define and identify file types and what modules should be executed against those object types. Examples of object types defined by default are email, Microsoft Office 2003 and 2007+, PDF, RTF, ZIP and much more. For each of these type definitions are configurations specifying the modules and order of modules that should be executed should Laika BOSS see that object type during analysis. Each of these modules produces metadata or objects that can be used by the framework for detection purposes. Within dispatch are global variables that can be used for identifying when modules should be run. The variables are clearly defined in `dispatch.yara`. Some example use cases include only executing modules based on a specific file name or as a result of a parent module having been run. Examples of customizing dispatch settings will be reviewed in section 3 of this paper.

2.2.5. Conditional Dispatch

The conditional dispatch settings are defined in `conditional-dispatch.yara` and are located in `/etc/laikaboss`. Conditional dispatch is considered a second pass at analyzing the object and it gives the opportunity to log metadata or perform additional actions against objects that have flagged in previous analysis. LOG modules are typically used at this stage of analysis as all object analysis and metadata collection has been completed so all results can be logged. Dispositioning of objects also occurs during conditional dispatch and it is responsible for determining the outcome of an object scan. The DISPOSITIONER module uses a configuration file to determine how certain flag hits should be dispositioned. Lastly, DECODE and EXPLODE modules are also often typically run during conditional dispatch as they usually require the outcome of previous modules to complete in order to determine if they should execute. An example previously given was that of a backdoor that may include embedded configuration data. You need to first flag on the backdoor sample and then during conditional dispatch or the "2nd pass", run a DECODE module against the sample to pull out the configuration data. In the conditional dispatch configuration for this DECODE module, you would need to specify that it will only execute if it has a string match for the backdoor Yara rule.

2.2.6. Disposition

The disposition settings are defined in `disposition.yara` located in `/etc/laikaboss/modules/dispositioner/`. Dispositioning defines the outcome of a scan. It is here where you can classify certain flag hits as being the determining factor for if something is seen as malicious or even to identify different priority levels should flags hit. As an example, you may have written some backdoor Yara rules related to an advanced persistent threat attack at your company. You may want to classify those as CRITICAL severity hits should you scan an object that flags on those same backdoor rules. It is within the `disposition.yara` file that you can specify those settings. Example disposition buckets may be INFO, LOW, MEDIUM, HIGH, CRITICAL. Other buckets such as Opportunistic and APT may be viable options as well. If the scanner was a passive sensor on your network, extracting objects and automatically scanning them, these types of disposition settings could help your team prioritize alerts originating from Laika BOSS.

6. Conclusion

Analysts use a variety of tools while performing research and analysis on all different types of malware. It is often the case that a common set of analysis steps and data is gathered during this process. The customization options afforded to analysts by using a framework such as Laika BOSS allows them to quickly develop signatures for new variants of malware, create new modules that expand analysis and detection capability and extract and store meaningful metadata that can be searched for historical analysis purposes. Threat actors will change the way in which they develop, package and deliver their malware and it is necessary for analysts to also change the ways in which they perform analysis. Automating much of this work helps the analyst focus on the data and the actual analysis work that is needed to understand the threat.

References

Bianco, David (2014, Jan 17). The Pyramid of Pain. Retrieved from http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html

Hanel, Alexander (2013, Jan). Pe-carv.py Python PE Carver. Retrieved from http://hooked-on-mnemonics.blogspot.com/2013/01/pe-carvpy.html

Hutchins, Eric M., Cloppert, Michael J., Amin, Rohan M. (2009). Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. Retrieved from https://www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf

Kessler, Gary (2017, Dec 27). File Signatures Table. Retrieved from https://www.garykessler.net/library/file_sigs.html

Liburdi, Joshn (2017, February 18). Laika BOSS + Bro = LaikaBro(?!). Retrieved from https://medium.com/@jshlbrd/laika-boss-bro-laikabro-d324d99fddae

Sikorski, M., & Honig, A. (2012). *Practical Malware Analysis*. San Francisco, CA: No Starch Press Inc

Stevens, Didier (2017, July 17). Quickpost: Analyzing .ISO Files Containing Malware. Retrieved from https://blog.didierstevens.com/2017/07/17/quickpost-analyzing-iso-files-containing-malware/

Wallace, Brian (2015, June 25). Using .NET Guids to help hunt for malware. Retrieved from https://www.virusbulletin.com/virusbulletin/2015/06/using-net-guids-help-hunt-malware

Zorz, Zeljka (2012, May 31). Tiny but deadly banking Trojan discovered. Retrieved from https://www.helpnetsecurity.com/2012/05/31/tiny-but-deadly-banking-trojan-discovered/