iGen: Toward Automatic Generation and Analysis of Indicators of Compromise (IOCs) using Convolutional Neural Network

by

Anupam Panwar

A Thesis Presented in Partial Fulfillment
of the Requirement for the Degree
Master of Science

Approved April 2017 by the
Graduate Supervisory Committee:

Gail-Joon Ahn, Chair
Adam Doupé
Ziming Zhao

ARIZONA STATE UNIVERSITY

May 2017

ABSTRACT

Field of cyber threats is evolving rapidly and every day multitude of new information about malwares and Advanced Persistent Threats (APTs) is generated by the security professionals in the form of malware reports, blog articles, forum posts etc. However, currently data received from different sources is examined and interpreted by an analyst manually, a procedure that extremely obscures the practical use of this data in an enterprises security infrastructure. Most of traditional approaches just consider either one or two data sources for the generation of Indicators of Compromise (IOCs) and misses some of the most valuable IOCs from other data sources. This leaves some scope for attackers to bypass the defense systems. Additionally, current state of art lacks support to diverse IOC formats (e.g. STIX, OpenIOC, snort etc.) which further reduce the effectiveness of these tools. Most importantly, lots of Threat Intelligence (TI) tools are generating IOCs directly using regex expression without understanding the contextual meaning of those IOCs from the data sources which allows the tools to include lot of false positives during the process of IOCs creation.

To overcome these limitations, we propose iGen, a novel approach to fully automate the process of IOC generation and analysis. iGen consists of different modules like data acquisition module, malware analysis module, intelligent IOC extraction module and security rule generator module. All these modules are arranged in a work-flow to fully automate the IOC generation process. Proposed approach is based on the idea that our model can understand English texts like human beings, and extract the IOCs from the different data sources intelligently. Identification of the IOCs is done on the basis of the syntax and semantics of the sentence as well as context words (e.g., attacked, suspicious) present in the sentence which helps the approach work on any kind of data source. Our proposed technique, first removes the words with no contextual meaning like stop words and punctuations etc. Then using the rest of the words in the sentence and output label (IOC or non-IOC sentence), our model intelligently learn to classify sentences into IOC and non-IOC sentences. Once IOC sentences are identified using this Convolutional Neural Network (CNN) based approach, next step is to identify the IOC tokens (like domains, IP, URL) in the sentences. This CNN based classification model helps in removing false positives (like IPs which are not malicious). Afterwards, IOCs extracted from different data sources are correlated to find the links between thousands of apparently unrelated attack instances, particularly infrastructures shared between them. Our approach fully automates the process of IOC generation from gathering data from different sources to creating rules (e.g. OpenIOC, snort rules, STIX rules) for deployment on the security infrastructure.

To my mother

ACKNOWLEDGMENTS

My journey through my Computer Science degree both B.S and M.S has been an exciting one which has played a major role in my career and life, especially in enhancing my knowledge and experience in this field of study. Having had the opportunity to work with Dr. Gail-Joon Ahn for two years has been an insightful experience, and I am utmost grateful to him for having given me the opportunity to work on cutting edge research projects at the laboratory for Security Engineering for Future Computing (SEFCOM). Dr. Ahn has been a constant source of motivation through my graduate career at Arizona State University, and has contributed significantly towards my abilities in reasoning, approaching and solving research problems impacting the society as a whole. I would like to extend my sincere gratitude to Dr. Partha Dasgupta and Dr. Adam Doupé for serving on my committee and providing their valuable feedback on my thesis.

My experience at the SEFCOM lab has provided me with great opportunities in my professional career, and has enabled me to connect with brilliant and innovative individuals who have always been there to provide valuable inputs during my research work. I would specially like to express my gratitude to Marthony Taguinod, who worked with me in the development of our test-bed for Cisco Inc. In addition to my committee and the SEFCOM lab members, I would also like to thank Cisco Inc. and more specifically Dr. Rodolfo Milito, for their support in the Fog Computing project and their valuable input during the development of our test-bed. Finally, and most importantly, I would like to extend my sincere love and regards to my parents for being a constant source of motivation and support throughout my academic career.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

According to Verizon's 2016 Data Breach Investigations Report ver (2017), over 100,000 security incidents were reported in 2016 across 82 countries, which is a 25% increase over the prior year. Because the number of security threats and breaches is rapidly increasing over time, every organization is attempting to protect their systems and their data. The threat landscape is always progressing, and the information security risk is increasing because of the organization's dependence on computing systems. This constantly shifting and constantly increasing number of threats results in tremendous pressure on organizations to manage threats. Though abundant information is available in the form of unstructured data, it is very difficult and time-consuming to mine meaningful information based on which preemptive measures can be established. This attracts more and more researchers towards Threat Intelligence (TI) as it helps to understand threats using the deluge of data and provides actionable insights.

Threat intelligence (TI) is proof-based knowledge, which includes reasoning, context, mechanism, indicators, implications, and actionable advice, about an existing or evolving cyber-attack that can be used to create preventive measures in advance McMillan (????). Attackers consistently exploit systems and networks to steal sensitive information, to take control of the target system, or for ransom (using ransomware) O'Gorman and McDonald (2012). TI allows an organization to expand its visibility into the fast-growing threat landscape, can allow early identification of an attack, and successfully prevent the attack.

Indicators of Compromise (IOCs) are forensic artifacts that are used as signs that a system has been compromised by an attacker or that a system has been infected with a particular piece of malware Catakoglu O. and Balzarotti (2016). The intent of accumulating IOCs related to malware or an attack is to be able to state, with a relatively high degree of confidence, whether or not such artifacts are present in a given environment. The goal of TI is to help security professional provide data-backed reasoning of why an artifact is identified as an IOC or not. Concretely, IOCs are composed of some combination of IP addresses, hostnames, filenames, processes, services, Windows registry entries, or hashes Andress (2015).

Figure **??** explains the typical life cycle for the use of IOCs in an organization. This life cycle consists of five steps: data collection, data analysis, IOC creation, deployment of IOCs on security infrastructure, and identification of affected systems using deployed IOCs. As a final step, log data of these compromised systems is collected and used for the creation new IOCs, which feeds back into the IOC life cycle in a cyclical way.

Several standards are commonly used to represent IOCs for expressing cyber-threat intelligence information such as: OpenIOC ope (2017b), Structured Threat Information eXpression (STIX) sti (2017), Cyber Observable eXpression (CybOX) cyb (2017), Trusted Automated eXchange of Indicator Information (TAXII) tax (2017), Snort sno (2017), Suricata sur (2017), YARA yar (2017), Malware Attribute Enumeration and Characterization (MAEC) mae (2017), and Common Attack Pattern Enumeration and Classification (CAPEC) cap (2017).

IOCs can help an organizations' security personnel to attain full automation: Given a set of IOCs for a particular security event, security tools scan through an environment or infrastructure to identify the existence of any IOC on the systems in question. IOCs complement and augment existing solutions, such as Intrusion Detection Systems (IDS) or Security Information and Event Managements (SIEMs), by providing an additional and important set of information that can be used to decide whether a particular artifact being examined is malicious or not. IOCs provide a rapid route to detect new or zero-day attacks for which virus signatures or detection rules have yet to be developed for existing security tools. Thus, timely generation of IOCs is important. Also, IOC formats document a threat in a consistent fashion, thus it becomes easier for organizations to share this threat information.

However, effective collection and sharing of threat information is a challenge. Current threat intelligence systems have following limitations:

- Threat data received from different sources, such as malware reports, APT white-papers, etc., is examined and interpreted by security analysts *manually*, a procedure that is timely and error-prone, which limits the practical usefulness of this data in an organizations' security infrastructure.
- Most of the traditional approaches consider either one or two data sources for the generation of Indicators of Compromises (IOCs) and miss valuable IOCs from alternative data sources, such as blog articles about recent attacks.
- Current state-of-the-art lacks support of diverse IOC formats (e.g., STIX, OpenIOC, or snort) which further reduces the effectiveness of these tools.
- Most importantly, many TI tools generate IOCs directly using regular expressions and white-listings from security literature or blog articles otx (2017) without understanding the contextual meaning of those IOCs from the data sources which allows the tools to include lot of false positives during the process of IOCs creation.

In this project we present iGen, which is a system that tackles all of these limitation by intelligently collecting threat information from publicly available security resources and sharing the threat information in the form of IOCs (e.g., virus signatures, IPs, domains, or MD5 hashes). iGen automates the entire process of collecting publicly available data (data collection) to IOC generation. iGen supports a flexible system to collect data from diverse data sources. Currently, iGen collects data from 20 security blogs, an APT reports database apt (2017), and a malware analysis system. The flexible nature of the system means that new data sources can be easily added. Note that the input to iGen is in unstructured English text, therefore iGen must understand the semantics of the sentences present in the input data to categorize IOC-alike strings into IOC or non-IOCs. Also, we added a practical improvement to iGen that allows transforming these IOCs into various threat information sharing standards.

First module in iGen is data acquisition module which collects key observations from the data sources like APT reports, malware analysis reports and blog articles. This module collects these reports in real-time and keep iGen updated with new IOCs from recent malware and APTs. In parallel, malware analysis system looks for new malware on web and generate malware analysis report for those samples. All the different kinds of reports accumulated from different data sources are then presented to machine learning based IOC extractor. IOC extractor module decides the parsing strategy based on the type of report. Since, malware analysis reports are based on real malware and available in machine readable format (e.g. JSON file). In this case, IOC extraction was done using a parser which accrue IOCs based on the JSON schema. But security reports or blog articles are relatively complex which tend to describe IOCs in intricate manner in English text using some context terms like “download”, “malware” etc. Google’s tensorflow Abadi *et al.* (2016) based Convolutional Neural Netowrk Krizhevsky *et al.* (2012) model was used to classification all the sentences in these reports into IOC or non-IOC sentences Bengio *et al.* (2003); Yih *et al.* (2011);

Mikolov *et al.* (2013); Collobert *et al.* (2011). Once sentence classification is done, next step is to identify IOCs intelligently. More explicitly, after classifying the sentence in the article into IOC and non-IOC sentences. iGen utilizes a set of regular expressions (regex) to locate IOC tokens (e.g., IP, hostname, domain, md5 etc.) within the sentence. Then, iGen extract these IOC tokens and convert them to machine-readable formats (IOC formats) which can be used directly by security tools.

Simple extraction of IOCs from publicly available security resources is straightforward: regular expressions can extract MD5s, URLs, filenames, filepaths, etc. However, this naïve approach will result in a significant amount of false positives: these security resources will include MD5s, URLs, and filenames that are *relevant to the security incident but are not malicious, and thus not an IOC.* Therefore, we need an approach that can automatically filter these potential IOCs *based on the context.* While prior work has shown that filtering potential IOCs can be done with manual creation of features Liao *et al.* (2016), iGen leverages a deep-learning Convolutional Neural Network, which selects features automatically. This enables iGen to be more effective and to use diverse data sets.

The main contributions of this paper are the following:

- We present the design of a novel system for intelligently generating IOCs, with high confidence, from different data sources using Convolutional Neural Networks (CNNs) (Section 3).
- A prototype implementation of our design in a tool called iGen (Section 4).
- An evaluation of iGen with other state-of-the-art methods. iGen identified around 400,000 IOCs with a precision and recall of 95% and 99% respectively (Section 5).

Chapter 2

BACKGROUND

In this section, we provide the background necessary for understanding the design of iGen. In Section 2.1, we describe Threat intelligence and how it is consumed by security tools. Then, in Section 2.2, we describe Convolutional Neural Network (CNN) and how it can be used in iGen.

2.1 Threat Intelligence

Given the increasing number and rapidly evolving nature of current threats, quick sharing and exchange of relevant threat information is the key to swiftly detecting, understanding, and responding to cyber-attacks. Threat intelligence is two things: (1) a process of collection of knowledge that defines security threats, which empowers an organization to determine its responses at the strategic, operational, and tactical levels, and (2) information that has been analyzed to discover actionable insights. Actionable threat intelligence is insight that an organization can act on–it enables informed decision-making that results in improved outcomes.

2.1.1 Consumption of Threat Intelligence

Indicators of Compromise (IOCs) are used for organizations to exchange threat intelligence. There are different standards for IOCs which includes OpenIOC, STIX, and snort rules. OpenIOC was introduced by Mandiant in 2011. It is primarily used in Mandiant products, but has also been released as an open standard. OpenIOC provides definitions for specific technical details including over 500 indicator terms. Adding new terms is easy because the terms are separated from the main schema.

Most of the terms are host-centric.

Similarly, STIX is another format for storing IOCs. STIX allows defining threat information, including threat details, as well as the *context* of the threat. STIX is developed by Mitre, and is designed to support four cyber threat use cases: investigating cyber threats, stating indicator patterns, response activities management, and sharing threat information. STIX uses XML to define threat-related constructs such as exploit target, campaign, indicator, threat actor, and TTP.

Snort is an open-source network intrusion prevention system (IPS) Sekar and Uppuluri (1999), which executes real-time traffic analysis and packet-logging on IP networks. It is also capable of performing protocol analysis, content searching, and matching, and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and more. Snort uses a rule language to describe traffic that it should collect or pass, as well as a detection engine that uses a modular plug-in architecture. Rules created using this language are called snort rules.

Suricata rules are the defacto method for sharing and matching threat intelligence against network traffic. A suricata rule has three components: The action, header and rule-options.

2.2 Convolutional Neural Network Architecture

Recently, different models based on deep learning have achieved significant results in computer vision and speech recognition. Convolutional Neural Networks (CNNs) were responsible for major breakthroughs in image classification Krizhevsky *et al.* (2012) and are the core of most computer vision systems today, from Facebook's automated photo tagging Stone *et al.* (2008) to self-driving cars Bojarski *et al.* (2016).

In the case of Natural Language Processing (NLP), much of the work that uses deep learning methods involves learning word vector representations through neural language models and performing composition over the learned word vectors for classification tasks. Word vectors, wherein words are projected from a sparse, 1-of-V encoding (here V is the vocabulary size) onto a lower dimensional vector space via a hidden layer, are essentially feature extractors that encode semantic features of words in their dimensions. In such compact representations, semantically close words are likewise close–in euclidean or cosine distance–in the lower dimensional vector space Mikolov *et al.* (2013).

Recently, researchers have also started applying CNNs with word embeddings to problems in Natural Language Processing (NLP) and got some interesting results. CNNs utilize layers with convolving filters that are applied to local features. CNNs are effective for NLP and have achieved excellent results in semantic parsing, search query retrieval, sentence modeling, and other traditional NLP tasks.

2.2.1 Convolution

Convolution Hu *et al.* (2014) is a sliding window function applied to a matrix. The sliding window is called a kernel, filter, or feature detector. Here we use a $N \times N$ filter, multiply its values element-wise with the original matrix, then sum them. To get the full convolution we do this for each element by sliding the filter over the whole matrix. Final matrix created after applying the filter is called convolved feature.

2.2.2 Convolution Neural Network

CNNs are fundamentally several layers of convolutions with nonlinear activation functions like rectified linear unit (ReLU) Nair and Hinton (2010) or hyperbolic tangent (tanh) Goodfellow *et al.* (2013) applied to the results. In a traditional feedforward neural network, each input neuron to each output neuron in the next layer. It is called as fully connected layer, or affine layer. But it's different in case of CNNs. CNNs use convolutions over the input layer to compute the output. This results in local connections, where each region of the input is connected to a neuron in the output. Each convolution layer applies different filters, characteristically hundreds or thousands, and combines their results. Then pooling is performed using the pooling (subsampling) layer. We will discuss more in Section **??**. CNN automatically learns the values of its filters based on the task during the training phase. For example, in image classification a CNN learns to detect edges from raw pixels in the first layer, then uses the edges to detect simple shapes in the second layer, and then uses these shapes to deter higher-level features, such as facial shapes in higher layers. The last layer is then a classifier that uses these high-level features.

In the case of iGen, input to a CNN is sentences extracted from blog articles and security reports instead of image pixels. Each sentence is represented as a matrix. Each row vector of the matrix corresponds to one token, typically a word. That is, each row is vector that represents a word. These vectors might be, e.g., outputs from trained word2vec or GloVe Pennington *et al.* (2014) models. We will discuss more about the iGen CNN in Section 3.2.

Chapter 3

SYSTEM DESIGN

Human are very intelligent in understanding natural language text and finding the information using reasoning. In our case, this information is IOCs from security reports, articles, and malware reports. However, it is very hard for the machines to do the information extraction with high accuracy and coverage.

iGen identifies sentences likely containing IOCs using a set of regular expressions. Then, IOCs are efficiently captured with high accuracy using CNN-based model. iGen tries to automatically identify the patterns or features which are frequent in IOC and non-IOC sentences. Architectural design of the iGen is shown in Figure 3.1.

3.0.1 Architecture Overview

iGen consists of 3 modules from left to right: 1) data acquisition module, 2) CNN based IOC extraction module, and 3) rule generation module. Data acquisition module consists of three sub-module: Blog Scrapper (BS), APT Report Collector (ARC), and Cuckoo Malware Analysis System (CMAS). BS is a collection of crawlers based on the HTML structure of the different blogs. For instance, there is a dedicated crawler which is continuously looking for new articles on Symantec public blog sym (2017). Similarly, ARC continuously looks for new whitepapers and reports about APT campaigns at “APTNotes” apt (2017) database. CMAS collects malware reports from Cuckoo Bayer *et al.* (2009), which is an open source malware analysis system.

Then, these reports are passed to an IOC extraction module based on the type of the report. Because blog articles and security reports are written in English and follow natural language semantics, these are given as input to the Relevant Content

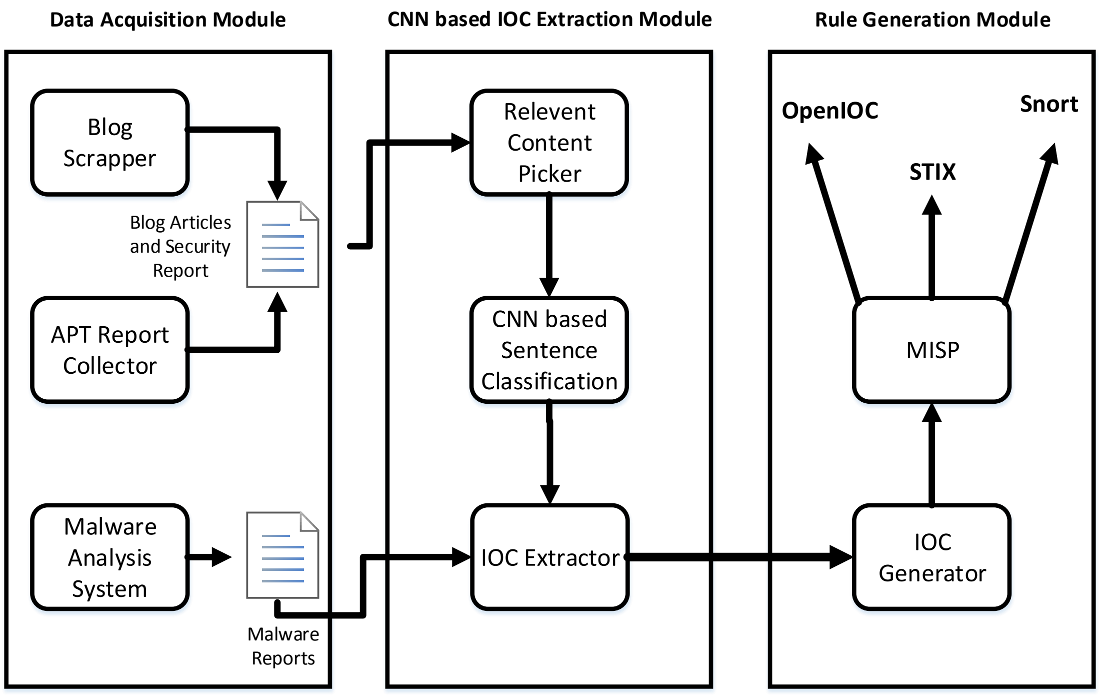

Figure 3.1: iGen Architecture Overview.

Picker (RCP) component for further processing. RCP parses those reports, cleans the text, and selects the sentences having likely IOCs using regular expressions. Then, these sentences are fed to the CNN based Sentence Classification (CSC) module for classification into two classes: IOC and non-IOC sentences. Whereas, malware reports follow JSON format with no natural language semantics involved. This makes it easier for IOC Extractor (IE) to extract IOCs based on the structure of the JSON report. Simultaneously, IE also performs regular expressions based extraction of IOCs from the sentences that are classified by CNN based Sentence Classifier (CSC) as IOC sentences.

Next step is to organize IOCs extracted from each report into one event (malware or APT campaign) is done by IOC Generator (IG). For example, Monkeys.exe and 200.125.133.28 are the IOCs related to the event CozyDuke APT campaign coz

(2017). Then, these events are stored and managed by Malware Information Sharing Platform (MISP) mis (2017). MISP also provide an additional functionality to iGen by converting these event into different output formats like STIX, OpenIOC, Snort etc.

3.0.2 Data Collection

There are two sources of TI data: internal, e.g., malware analysis reports or network traces, or external, such as technical blogs or security reports. Examples of external sources include the Kaspersky whitepapers kas (2017), Symantec blog sym (2017) etc. Recently, thousands of threats are reported every day, which has been broadly reported by the security companies in the form of APT/security reports Daly (2009) or blog articles and aggressively collected by different organizations. For the scope of our research, we collected 1000 security reports and 1500 blog articles from different security organizations. All these reports and articles are published from year 2011 to 2017. Our scrapper collected information from more than 20 prominent security organization like Verizon, Symantec etc.

To bootstrap our research, we also collected around 35,000 malware reports from our in-house Cuckoo Malware Analysis System (CMAS) Oktavianto and Muhardianto (2013). MAS provided some detailed results outlining what malware did when executed inside an isolated environment. While substantial volume of security reports, blog articles and malware reports are collected and analyzed, however it only constitutes a small part of the bigger IOC landscape. Summary of our dataset is in Table 1.

Table 3.1: Summary of Dataset

Dataset Type	Dataset Source	Number of articles/ reports
Security Reports	External	1000
Technical Blog Articles	External	1500
Malware Analysis Reports	Internal	35000

3.1 Data Acquisition Module

This module collects information from different sources. Because of the scalable nature of the module, new data sources can be added as well. This module scrap related web pages, APT report from different sites. Simultaneously, it continuously collects malware reports from our internal CMAS as well. All the sub-modules of data acquisition module are described in the following.

3.1.1 Blog Scrapper (BS)

iGen blog scraper is designed to continuously monitor a list of security blogs and collect their articles. BS first scraps all its current articles before it is set to monitoring mode to identify new ones for each blog site. Most of the blog articles on most of the blog sites have a particular page id associated with it. BS keep track of the range of page ids that are already scrapped by it. In monitor mode, it looks for new articles. If there are no new articles posted on the blog sites, then BS changes its mode to idle and want back to monitor mode after n days. For our research, we have set the n as 7 days. BS also collects the reports if there is any PDF report link within the blog article.

3.1.2 APT Report Collector (ARC)

APT report collector consists of autonomous data crawler, which continuously check for new reports and whitepapers at APTNotes database. It also keeps track of the already scrapped reports to avoid duplication. Around 1000 APT campaign report are collected till date.

3.1.3 Cuckoo Malware Analysis System (CMAS)

For our research, we have a Cuckoo malware analysis system (CMAS) Bayer *et al.* (2009) for automating analysis of suspicious files. CMAS monitors the behaviour of the malicious processes while running in an isolated environment. In other words, if we submit any suspicious file to it and in a matter of seconds CMAS will provide us back some detailed results outlining what such file did when executed inside an isolated environment. Features of CMAS are :

- Analyze different type of malicious files (executables, document expoits, malicious websites etc.).
- Dump and analyze network traffic, even when encrypted.
- Trace API calls and general behavior of the file.

For scope of this research, around 35000 malware were submitted and analyzed by CMAS. Later, these reports were further used for IOCs extraction.

3.2 IOC Extraction Module

After the collection of dataset, the next step is to clean the data, select relevant content, and find IOCs with high degree of confidence. In this module, Relevant Content Picker (RCP) cleans the data by removing the terms with no semantic meaning such as stop words, etc. RCP also finds sentences which are likely to contain IOCs. Afterwards, CNN based Sentence Classification (CSC) and IOC Extractor (IE) identify the IOC sentences and extract the IOCs from them respectively.

3.2.1 Relevant Content Picker (RCP)

Relevant Content Picker extracts text from PDF and HTML documents. In our implementation, the RCP uses the python library "pdfminer" pdf (2017) to extract the text from the PDF reports and "beautifulsoup" bea (2017) to obtain content from HTML pages as well to find links to security reports in blog articles. RCP scans for URLs with an extension ".pdf" in the blog articles to identify security reports linked with the blog articles. If RCP finds any PDF report linked with blog article, it sends it to PDFminer for further processing. After the collection of content, the next step is to remove the terms or token with no semantic meaning such as stop words. Stop words extremely common words which would appear to be of little value in terms of semantic significance like "a", "an", "the" etc. Removing stopwords is a part of the data cleaning process. Third and final step is to look for sentences which are likely to have IOCs. This part is done by finding the sentences which have patterns like IP, hostname, URL, filename, registry, email, filepath etc. These are patterns are identified using the set of regular expressions. Some sample regular expressions are shown in Table 3.2.

3.2.2 CNN based Sentence Classification (CSC)

The model architecture of CNN based Sentence Classification (CSC), shown in figure 3.2, is a minor variant of the CNN architecture suggested by Kim Kim (2014). First step of CSC involves converting all the words in the sentences into their respective word vector or embedding. Let $\mathbf{S}_i \in \mathbb{R}^d$ be the word vector of dimensionality d

corresponding to the i-th word or token in the sentence. A sentence of length l is characterized as

$$\mathbf{S}_{1:l} = \mathbf{S}_1 \oplus \mathbf{S}_2 \oplus \ldots \oplus \mathbf{S}_l, \tag{3.1}$$

where concatenation operator is defined as $\oplus$.

The first step is to calculate word embedding of S_i based on our dataset. We call this layer as embedding layer, which maps vocabulary word indices into low-dimensional vector representations. X is our embedding matrix that we learn during training. We initialize it using a random uniform distribution Claessen *et al.* (2014). $tf.nn.embedding\_lookup$ ten (2017) function in tensorflow creates the actual embedding operation. The result of the embedding operation is a 3-dimensional tensor of shape $[None, sequencelength, embeddingsize]$. TensorFlow's convolutional conv2d ten (2017) operation expects a 4-dimensional tensor with dimensions corresponding to batch, width, height, and channel. The results of our embedding does not contain the channel dimension, so we add it manually, leaving us with a layer of shape $[None, sequencelength, embeddingsize, 1]$.

A word embedding $\mathbf{X}\colon words \to \mathbf{R}^n$ is a paramaterized function mapping words in some language to high-dimensional vectors (perhaps 200 to 500 dimensions). For example

$$\mathbf{X}(\text{“}malware\text{”}) = (0.2, -0.4, 0.7, \ldots) \tag{3.2}$$

Characteristically, the function is a lookup table, parameterized by a matrix, θ, with a row for each word: $\mathbf{X}_\theta(w_n) = \mathbf{X}_n = \theta_n$.

$\mathbf{X}$ is initialized to have random vectors for each word. It learns to have meaningful vectors in order to classify the sentences into IOC and non-IOC sentences.

Let $\mathbf{S}_{i:j}$ refer to a sentence of length $j-i+1$ or the concatenation of words vectors $\mathbf{S}_i = \mathbf{X}_\theta(w_i), \mathbf{S}_{i+1} = \mathbf{X}_\theta(w_{i+1}), \ldots, \mathbf{S}_j = \mathbf{X}_\theta(w_j)$. Single convolution consist of a *filter*

$\mathbf{w} \in \mathbb{R}^{h \times d}$, which is applied to a window of h words to automatically generate a new feature. For example, a feature f_i is generated from a window of words $\mathbf{S}_{i:i+h-1}$ by

$$f_i = f(\mathbf{w} \cdot \mathbf{S}_{i:i+h-1} + c). \tag{3.3}$$

Here bais term $c \in \mathbb{R}$ is added and f is a non-linear function such as the rectified linear unit (ReLU) or hyperbolic tangent (tanh). Filter is applied to each possible window of words in the sentence $\{\mathbf{S}_{1:h}, \mathbf{S}_{2:h+1}, \ldots, \mathbf{S}_{l-h+1:l}\}$ to produce a *feature map*

$$\mathbf{f} = [f_1, f_2, \ldots, f_{l-h+1}], \tag{3.4}$$

where $\mathbf{f} \in \mathbb{R}^{l-h+1}$. We then apply a 1-max pooling operation Ranzato *et al.* (2007) over the feature map and take the maximum value $\dot{f} = \max\{\mathbf{f}\}$ as the feature corresponding to this particular filter. Intuition behind this is to capture the most important feature—one with the highest value—for each feature map. This kind of approach works naturally with variable sentence lengths.

Above described process extracts *one* feature from *one* filter. The model uses multiple filters with different region size to obtain multiple features. These features creates the penultimate layer and the final layer is a fully connected softmax layer whose output is the probability distribution over labels (in our case, IOC sentence or non-IOC sentence).

In our model, we have fine-tuned word vectors via backpropagation Rumelhart *et al.* (1986). For regularization Zou and Hastie (2005) we used dropout Srivastava *et al.* (2014) on the penultimate layer with a constraint on l_2-norms Ng (2004) of the weight vectors Hinton *et al.* (2012). Dropout helps in co-adaptation of hidden units by randomly dropping out—i.e., setting to zero—a proportion p of the hidden units during foward-backpropagation. That is, given the penultimate layer $\mathbf{y} = [\dot{f}_1, \ldots, \dot{f}_n]$ where n is total number of filters). Instead of using

$$z = \mathbf{w} \cdot \mathbf{y} + b \tag{3.5}$$

Table 3.2: Sample Regular Expressions used by Relevant Content Picker (RCP) and IOC Extractor (IE).

Type of Regular Expression	Regular Expression
Email	`\b([a-z][_a-z0-9-.]+@[a-z0-9-]+\.[a-z]+)\b`
IP	`\b(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})\b`
MD5	`\b([a-f0-9]{32}|[A-F0-9]{32})\b`
CVE	`\b(CVE\-[0-9]{4}\-[0-9]{4,6})\b`

during forward propagation for output unit y, dropout uses

$$z = \mathbf{w} \cdot (\mathbf{y} \circ \mathbf{r}) + b, \tag{3.6}$$

where $\circ$ is the element-wise multiplication operator and $\mathbf{r} \in \mathbb{R}^n$ is a 'masking' vector of Bernoulli random variables with probability p of being 1. During the training, gradientsFriedman (2002) are backpropagated only through the unmasked units. During the testing, all learned weight vectors are scaled by p such that $\dot{\mathbf{w}} = p\mathbf{w}$, and $\dot{\mathbf{w}}$ is used to score new sentences without dropout.

3.2.3 IOC Extractor (IE)

As shown in Figure 3.1, IOC extractor takes malware report as well the sentences which are classified as IOC sentences by CSC sub-system as input. Since, malware reports doesn′t have any natural language semantics, we skipped running RCP and CSC sub-systems on them. Additionally, MAS is run only on the malware files that implies all the indicator/tokens (e.g., file hashes, emails, etc.) are malicious.

In case of malware reports, structure of JSON file is used to extract the IOCs. Whereas, regex as shown in Table 3.2 were used to extract the IOC from the sentences.

These are different type of IOCs collected by the IE: URL, host, IP, email, MD5, SHA1, SHA256, CVE, registry, filename, and filepath. Next step is to bundle the IOCs related to one event/malware together.

3.3 Rule Generation System

This system works on the managing the related IOCs and convert them into different output formats so that these IOCs based formats can be directly deployed on the security infrastructure. Currently, iGen supports STIX, OpenIOC, snort, suricata, and BRO formats. Rule generation system has two sub-systems: IOC Generator (IG) and Malware Information Sharing Platform (MISP) mis (2017).

3.3.1 IOC Generator (IG)

IOC generator takes IOCs from IE and bind them together into one IOC event if they are coming from the same report. This sub-system creates event into MISP acceptable JSON format. Then, this JSON is submitted to MISP.

3.3.2 Malware Information Sharing Platform (MISP)

MISP is an open source software for sharing, storing and correlating IOCs of targeted attacks. MISP benefits security community by showing collaborative knowledge about existing malware or threats. The aim of this trusted platform is to help improving the counter-measures used against targeted attacks and set-up preventive actions and detection. It stores data in a structured format and allows automated use of the database to feed detection systems or forensic tools. For example, it generates rules from IOCs (e.g. IP addresses, domain names, hashes of malicious files, filenames, filepath etc.) for different Network Intrusion Detection Systems (NIDS) like snort, suricata etc. MISP adds an extra feature of generating diverse rules from the IOCs collected from different data sources.

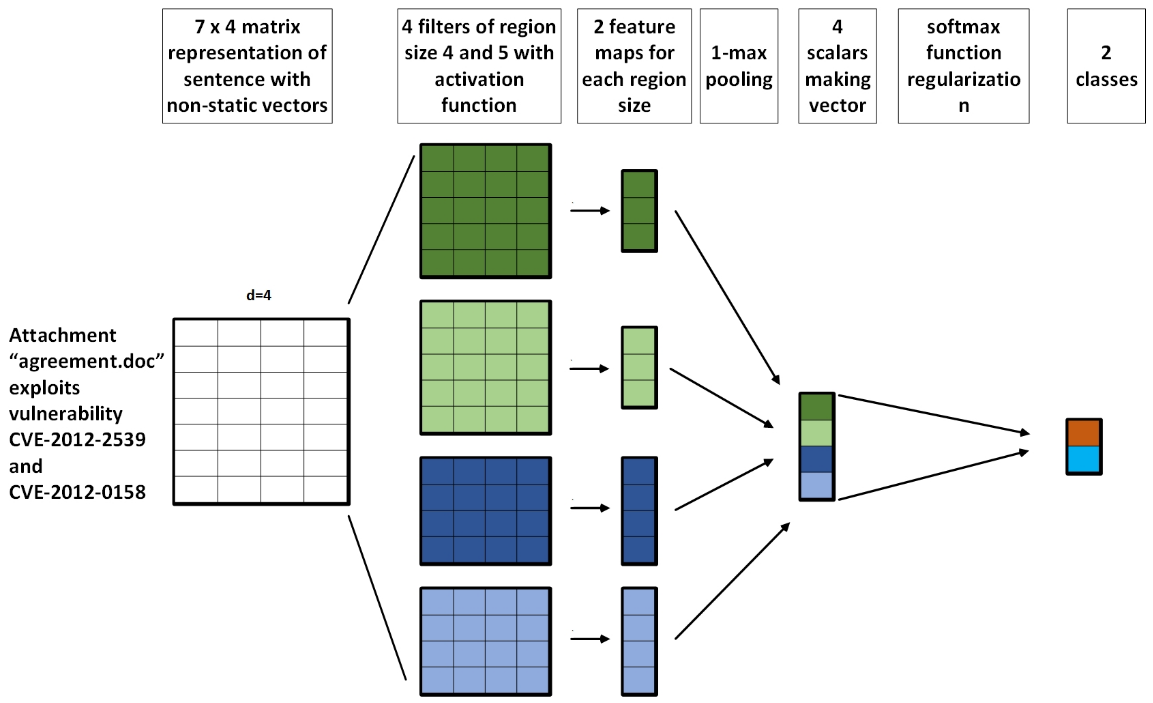
7 x 4 matrix representation of sentence with non-static vectors
4 filters of region size 4 and 5 with activation function
2 feature maps for each region size
1-max pooling
4 scalars making vector
softmax function regularizatio n
2 classes
d=4
Attachment "agreement.doc" exploits vulnerability CVE-2012-2539 and CVE-2012-0158

Figure 3.2: Architecture of CNN based sentence classification model. We show two filter region sizes: 4 and 5, each of which has 2 filters. Filters or kernels accomplish convolutions and apply activation functions on the sentence matrix and generate (variable-length) feature maps; 1-max pooling is performed over each map, i.e., the largest number from each feature map is recorded. Thus a univariate feature vector is generated from all 4 maps, and these 4 features are concatenated to form a feature vector for the penultimate layer. The final softmax layer then receives this feature vector as input and uses it to classify the sentence; here we are performing binary classification and hence depict two possible output states.

Chapter 4

IMPLEMENTATION

We implemented iGen that collects IOCs from various external and internal IOC data sources. The automatic crawling and analysis systems were implemented in Python. For example, Blog Scrapper (BS) and APT report Collector (ARC) were built using Python based libraries like pdfminer, beautifulsoup and textblob. Additionally, we are generating the malware reports using our in-house Cuckoo malware analysis system. Relevant Content Picker (RCP), IOC Extractor (IE) and IOC Generator (IG) uses re and nltk for information extraction. CNN based Sentence Classification (CSC) uses numpy and Google's tensorflow library. Different output formats are generated using the open-source tool Malware Information Sharing Platform (MISP).

We deployed our system iGen modules on our Openstack ope (2017a). Data acquisition module and IOC extraction module on a dedicated Openstack instance with 8GB RAM. Whereas, rule generation module runs on other instance with 4GB RAM.

Chapter 5

EVALUATION

5.1 Experiments

Datasets used in the experiments are shown in Table 3.1. For sentence classification, only blog articles and security reports were used. From these articles, we further extracted two kinds of sentences, those with IOCs (IOC sentences) and those without but involving non-malicious IOC-like strings (non-IOC sentences). More specifically, the 5,000 IOC sentences and 2,500 non-IOC sentences were used in the experiments. These sentences are manually tagged by security experts to maintain the quality of data.

In the experiments, the parameters of our iGen (most of them related to CNN sub-module) system were set as follow:

- *Sequence length.* The length of our sentences before inputting them for classification. Remember that we padded all our sentences to have the same length of 70. We selected 70 as sequence length, since longest sentence in our dataset has 60 word length.

Table 5.1: Change in accuracy with embedding size.

Embedding Size(d)	Accuracy
32	89.3
64	93.8
128	95.7

Table 5.2: Change in accuracy with number of filters.

Number of filters	Accuracy
32	88.05
64	95.7
128	95.7

- *Embedding Size.* The dimensionality of our word embeddings or word vectors is kept as 128. Word embedding dimensionality was chosen as 128 based on the experiments where we compared accuracy with dimensionality of word embeddings as shown in Table 5.1. We did not increased the dimensionality beyond 128 as it has bad effect on the performance.

- *Filter Sizes.* Filter size is the number of words we want our convolutional filters to cover. We have used three type of filter 3, 4 and 5 that slide over 3, 4 and 5 words respectively.

- *Number of Filter.* It is number of filters per filter size. Based on the experiments shown in Table 5.2, it is chosen as 64. Total $3*64$ filters are used with 64 filters each of size 3, 4, and 5.

- *Dropout Keep Probability.* Dropout is the most popular method to regularize convolutional neural networks.A dropout layer stochastically disables a fraction of its neurons. This prevent neurons from co-adapting and forces them to learn individually useful features. The fraction of neurons we keep enabled is defined by the *dropout_keep_prob* input to our network. We set this to something like 0.5 during training, and to 1 (disable dropout) during evaluation.

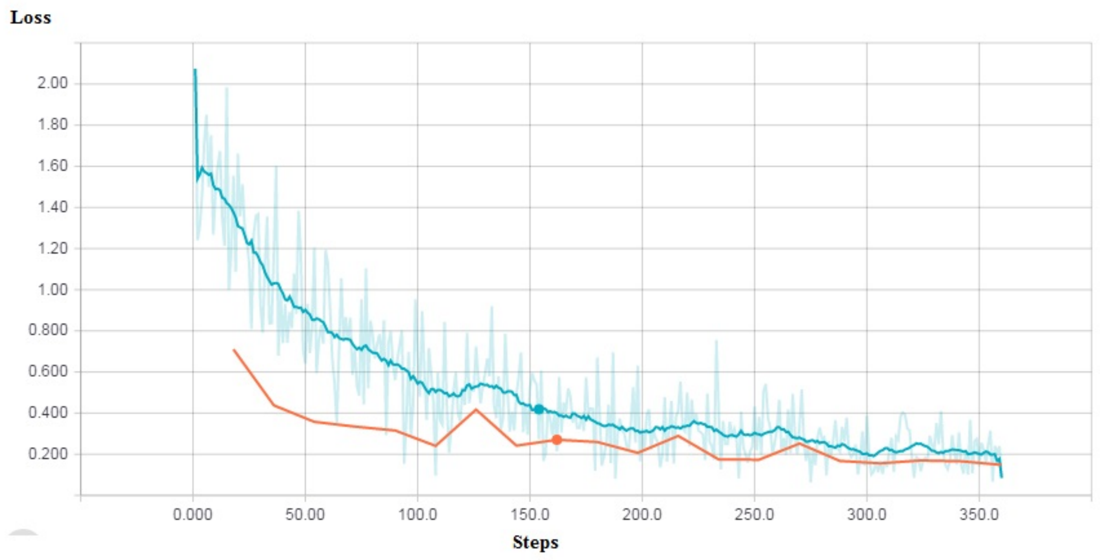

Figure 5.1: Change in loss over steps (blue is training data, red is 10% dev data).

5.2 Results

5.2.1 Loss Function and Accuracy

The loss is a measurement of the error our network makes, and our goal is to minimize it. The loss function for categorization problems is the cross-entropy loss. $tf \cdot nn \cdot softmax \cdot cross\_entropy\_with\_logits$ is a function that calculates the cross-entropy loss for each class, given our scores and the correct input labels. We have then taken the mean of the losses. We could also use the sum, but that makes it harder to compare the loss across different batch sizes and train/dev data. Figure 5.1 shows the change cross-entropy loss over steps.

We also used an expression for the accuracy, which is a useful quantity to calculate the performance during training and testing. Change in accuracy over steps is shown in Figure 5.2.

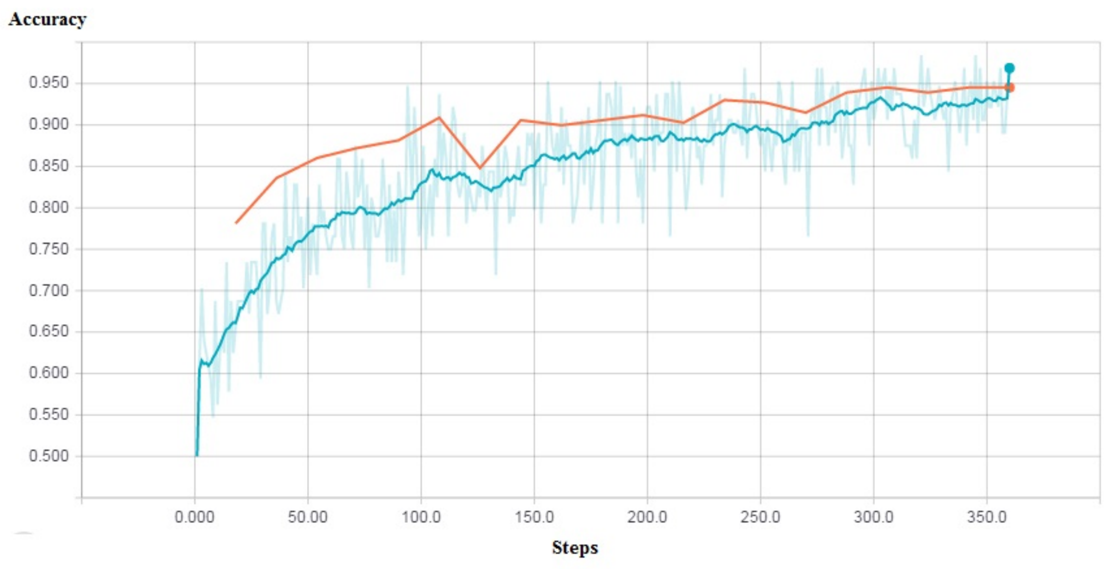

Figure 5.2: Change in accuracy over steps (blue is training data, red is 10% dev data).

5.2.2 Method Comparison

We compared iGen with other state-of-the-art methods like Support Vector Machine (SVM) Cortes *et al.* (1995), Nave Bayes Classifier Friedman *et al.* (1997), and KNN (K-Nearest Neighbour) Liao and Vemuri (2002) classifier used for IOC sentence categorization. Three type of feature vector are used for each of the classifier described above. 10-fold cross validation technique is used to compare precision, recall and F-measure. Our comparative analysis is shown in Figure 5.3.

iGen outperformed other methods by generating IOCs with a precision of 95% and a recall of 99%, while among other SVM with tfidf as feature vector performed well with a precision and recall of 90%.

Similarly, another approach iACE Liao *et al.* (2016) proposed by Liao gave promising result in extracting IOC with high accuracy. But iACE used around 5,283 terms

Table 5.3: Results of our iGen models against other methods. **SVM** (Support Vector Machine), **NB** (Naive Bayes), and **5 − NN** (5-Nearest Neighbours) classifiers were compared with feature vectors as bag of words (BOW), term frequency (tf), and term frequency - inverse document frequency (tfidf).

Methods	Precision	Recall	F-measure
iGen	**95**%	**99**%	**97**%
SVM (Bag of words)	89%	88%	88%
SVM (tf)	90%	89%	89%
SVM (tfidf)	90%	90%	90%
Naive Bayes (Bag of words)	81%	78%	79%
Naive Bayes (tf)	83%	79%	81%
Naive Bayes (tfidf)	85%	83%	83%
5-NN (Bag of words)	80%	80%	80%
5-NN (tf)	81%	81%	80%
5-NN (tfidf)	83%	82%	82%

as features which were manually gathered from the dataset. This implies that it will work very accurately on one dataset but may fail on sentences where different terminologies were used while writing the article. In case of iGen, features were not selected manually but features were selected by CNN automatically. This property makes iGen more adaptive towards any kind of data.

Chapter 6

DISCUSSION AND FUTURE WORK

Our experiments show that iGen not only fully automated the process of cyber threat intelligence collection but also generates deployable security rules from the data. With a large amount of IOCs automatically recovered from the wild and converted into a machine-readable form, these can be quickly and effectively utilized to counter emerging threats. For example, knowing the IP of C&C server from APT report and then finding the email associated with IP address from malware report can help us find the actor behind the attack. This will enable the defender to disable or block the servers associated with that email to stop future attacks. On the other hand, our current design is still preliminary. Here, we discuss the limitations of iGen and potential follow-up research.

Limitations

Although iGen extracts IOC with high accuracy, precision and recall, but it's necessary to recognize our system limitations. First, iGen assumes that all the IOCs mentioned in the article are part of some sentence. But there is a possibility that some of them are present in images (e.g., command prompt screenshots) or tables inserted into the article. Second, even though iGen detects IOCs with high accuracy. iGen still introduces some false discoveries and misses some IOCs. These problems mostly arises due to abnormal ways of writing the report. Typos in articles or reports effects the performance of iGen. For example, if one forgets to place a space after the period, a sentence becomes held with the follow-up one, which could cause an error in IOC sentence classification. Further efforts are desirable to better address these issues.

Chapter 7

CONCLUSION

In this paper, we introduced — CNN based iGen, a novel system for automatic extraction of IOCs from different sources of unstructured data. Our approach starts from collection of security data from diverse sources. Then, it cleans the data using advanced NLP techniques. CNN based IOC identification technique is found to be highly effective, immensely outperforming the top industry IOC analyzer tool in terms of accuracy, precision and recall. iGen generates ready-to-deploy rules from these IOCs that can be directly deployed on security infrastructure like NIDS etc.

iGen uses different sources for data to make fully automated cyber threat intelligence gathering more collaborative. iGen has collected around 1 million IOCs till now with a precision of 95%, better than any state-of-art method.

REFERENCES

"Alien Vault Open Threat Exchange", URL `https://otx.alienvault.com/` (2017).

"APTNotes database", URL `https://github.com/aptnotes/data` (2017).

"Beautiful Soup", URL `https://www.crummy.com/software/BeautifulSoup/` (2017).

"Common Attack Pattern Enumeration and Classification", URL `https://capec.mitre.org/` (2017).

"CozyDuke Apt Report", URL `https://www.f-secure.com/documents/996508/1030745/CozyDuke` (2017).

"Cyber Observable eXpression", URL `https://cyboxproject.github.io/` (2017).

"Kaspersky White Papers", URL `http://usa.kaspersky.com/enterprise-security/resources/white-papers` (2017).

Malware Attribute Enumeration Characterization (2017), URL `https://maec.mitre.org/`.

"Malware Information Sharing Platform (MISP)", URL `http://www.misp-project.org/` (2017).

"Openstack", URL `https://www.openstack.org/` (2017a).

"PDFMiner", URL `http://www.unixuser.org/{~}euske/python/pdfminer/` (2017).

"Snort rules", URL `https://www.snort.org/` (2017).

"Structured Threat Information eXpression", URL `https://stixproject.github.io/` (2017).

"Suricata rules", URL `https://suricata-ids.org/` (2017).

"Symantec blog", URL `https://www.symantec.com/connect/blogs` (2017).

"Tensorflow Api docs", URL `https://www.tensorflow.org/api{\_}guides/python/nn` (2017).

"The OpenIOC Framework.", URL `http://www.openioc.org/` (2017b).

"Trusted Automated eXchange of Indicator Information", URL `https://taxiiproject.github.io/` (2017).

"Verizon's 2016 Data Breach Investigations Report", Tech. rep., URL `http://www.verizonenterprise.com/verizon-insights-lab/dbir/2016/` (2017).

"Yara rules", URL `https://virustotal.github.io/yara/` (2017).

Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu and X. Zheng, "TensorFlow: A system for large-scale machine learning", Google Brain p. 18, URL `http://arxiv.org/abs/1605.08695` (2016).

Andress, J., "Working with Indicators of Compromise", in "Journal Information Systems Security Association (ISSA) 5", (2015).

Bayer, U., I. Habibi, D. Balzarotti, E. Kirda and C. Kruegel, "A view on current malware behaviors", Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more p. 8, URL `http://portal.acm.org/citation.cfm?id=1855676.1855684` (2009).

Bengio, Y., R. Ducharme, P. Vincent and C. Janvin, "A Neural Probabilistic Language Model", The Journal of Machine Learning Research **3**, 1137–1155 (2003).

Bojarski, M., D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. Jackel, M. Monfort, M. Zhang, X. Zhang, J. Zhao and Z. K.:, "End to end learning for self-driving cars", in "arXiv preprint arXiv:1604.07316", (2016).

Catakoglu O., B. M. and D. Balzarotti, *Automatic extraction of indicators of compromise for web applications* (In WWW, 2016).

Claessen, K., J. Duregard and M. Palka, "Generating Constrained Random Data with Uniform Distribution", Functional and Logic Programming, Lecture Notes in Computer Science **8475**, pp. 18–34 (2014).

Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa, "Natural Language Processing (Almost) from Scratch", Journal of Machine Learning Research **12**, 2493–2537 (2011).

Cortes, C., Vapnik and V., "Support-vector networks.", Machine Learning pp. pp. 273–297 (1995).

Daly, M. K., "Advanced persistent threat", Usenix, Nov **4**, 4, 2013–2016 (2009).

Friedman, J. H., "Stochastic gradient boosting", Computational Statistics and Data Analysis **38**, 4, 367–378 (2002).

Friedman, N., D. Geiger and M. Goldszmidt, "Bayesian network classifiers.", Machine Learning pp. pp. 131–163 (1997).

Goodfellow, I. J., D. Warde-Farley, M. Mirza, A. Courville and Y. Bengio, "Maxout Networks", in "Proceedings of the 30th International Conference on Machine Learning (ICML)", vol. 28, pp. 1319–1327 (2013).

Hinton, G. E., N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors", ArXiv e-prints pp. 1–18, URL `http://arxiv.org/abs/1207.0580` (2012).

Hu, B., Z. Lu, H. Li and Q. Chen, "Convolutional neural network architectures for matching natural language sentences", NIPS pp. pp. 2042–2050 (2014).

Kim, Y., "Convolutional neural networks for sentence classification", in "Proceedings of the Empiricial Methods in Natural Language Processing", (2014).

Krizhevsky, A., I. Sutskever and G. Hinton, "ImageNet classification with deep convolutional neural networks", NIPS (2012).

Liao, X., K. Yuan, X. Wang, Z. Li, L. Xing and R. Beyah, "Acing the IOC Game : Toward Automatic Discovery and Analysis of Open-Source Cyber Threat Intelligence", in "CCS '16 Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security", pp. 755–766 (2016).

Liao, Y. and V. Vemuri, "Use of K-Nearest Neighbor classifier for intrusion detection.", Computers and Security. **21**, 5, pp. 439–448 (2002).

Lunt, T. F., "A survey of intrusion detection techniques", In: Computers & Security **12**, 4, pp. 405–418 (1993).

McMillan, R., "Open Threat Intelligence", **2013**, URL `https://www.gartner.com/doc/2487216/definition-threat-intelligence` (????).

Mikolov, T., I. Sutskever, K. Chen, G. Corrado and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality", in "Proceedings of Neural Information Processing Systems", (2013).

Nair, V. and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines", Proceedings of the 27th International Conference on Machine Learning , 3, 807–814 (2010).

Ng, A., "Feature selection, \$l_1\$ vs. \$l_2\$ regularization, and rotational invariance.", in "Proceedings of the Twenty-First International Conference on Machine Learning", (2004).

Obrst, L., P. Chase and R. Markeloff, "Developing an ontology of the cyber security domain", in "CEUR Workshop Proceedings", vol. 966, pp. 49–56 (2014).

O'Gorman, G. and G. McDonald, *Ransomware: A growing menace (Technical report)* (Symantec Corporation, 2012).

Oktavianto, D. and I. Muhardianto, *Cuckoo Malware Analysis* (Packt Pbl. Ltd, 2013).

Pennington, J., R. Socher and C. D. Manning, "Glove: Global vectors for word representation", in "Proceedings of the Empiricial Methods in Natural Language Processing", (2014).

Ranzato, M., F. J. Huang, Y. L. Boureau and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition", in "Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition", (2007).

Rumelhart, D., G. Hinton and R. Williams, "Learning Representations by Back-Propagating Errors", Nature **323**, pp. 533–536 (1986).

Sekar, R. and P. Uppuluri, "Synthesizing Fast Intrusion Prevention/Detection Systems from High-Level Specifications", in "Proceedings of the USENIX Security Symposium", (1999).

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", JMLR **15**, pp. 1929–1958 (2014).

Stone, Z., T. Zickler and T. Darrell, "Autotagging Facebook: Social network context improves photo annotation", in "2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops", (2008).

Yih, W., K. Toutanova, J. Platt and C. Meek, "Learning discriminative projections for text similarity measures", Proceedings of the Fifteenth Conference on Computational Natural Language Learning pp. 247–256, URL `http://dl.acm.org/citation.cfm?id=2018965` (2011).

Zou, H. and T. Hastie, "Regularization and variable selection via the elastic net", Journal of the Royal Statistical Society. Series B: Statistical Methodology **67**, 2, 301–320 (2005).

APPENDIX A

RAW DATA