information

Article

Cyber Security Tool Kit (CyberSecTK): A Python Library for Machine Learning and Cyber Security

Ricardo A. Calix *, Sumendra B. Singh, Tingyu Chen, Dingkai Zhang and Michael Tu

Purdue University Northwest, Hammond, IN 46323, USA; singh445@pnw.edu (S.B.S.); chen2694@pnw.edu (T.C.); zhang3385@pnw.ed (D.Z.); Michael.Tu@pnw.edu (M.T.)
* Correspondence: rcalix@pnw.edu; Tel.: +1-219-989-2013

Received: 21 December 2019; Accepted: 8 February 2020; Published: 11 February 2020

Abstract: The cyber security toolkit, CyberSecTK, is a simple Python library for preprocessing and feature extraction of cyber-security-related data. As the digital universe expands, more and more data need to be processed using automated approaches. In recent years, cyber security professionals have seen opportunities to use machine learning approaches to help process and analyze their data. The challenge is that cyber security experts do not have necessary trainings to apply machine learning to their problems. The goal of this library is to help bridge this gap. In particular, we propose the development of a toolkit in Python that can process the most common types of cyber security data. This will help cyber experts to implement a basic machine learning pipeline from beginning to end. This proposed research work is our first attempt to achieve this goal. The proposed toolkit is a suite of program modules, data sets, and tutorials supporting research and teaching in cyber security and defense. An example of use cases is presented and discussed. Survey results of students using some of the modules in the library are also presented.

Keywords: cyber security; machine learning; feature extraction; toolkit

1. Introduction

The cyber security toolkit (CyberSecTK) is a simple Python library for preprocessing and feature extraction of cyber-security-related data. As the digital universe expands, more and more data need to be processed using automated approaches. For instance, the statistics [1] show a rapid growth in internet of things (IoT) devices. Currently, there are over 12 billion devices that can connect to the internet, and it was predicted to reach 14.2 billion in 2019. According to the publication announcement on the Worldwide Semiannual Internet of Things Spending Guide forecast [2], a 13.6% compound annual growth rate (CAGR) is expected over the 2017–2022 period with an estimated investment of 1.2 trillion US dollars by 2022. The proliferation of the IoT will lead to new challenges in the near future. According to [1], the growth of IoT devices usage will lead to an important problem, which will make it difficult for smart environment operators to ascertain IoT devices within their networks and monitor their operations. The Mirai botnet [3] is an example of risks to IoT devices. This denial of service attack is a wake-up call to the IoT industry, which is possible if IoT devices are not secured. Security of IoT devices, in this example, will be key. GHOST [4], an EU Horizon 2020 Research and Innovation funded project, develops an architecture to safeguard home IoT environments with personalized real-time risk control measures to mitigate cyber security events. The system modules a network and data flow analysis (NDFA) to build a profile builder (PB) based on network activities, which in turn notifies a risk engine (RE) to act in order to secure the smart-home IoT ecosystem. Abnormal behavior detection is one of the crucial components in our daily life interaction with IoT environments and devices.

In recent years, cyber security professionals have seen opportunities to use machine learning (ML) approaches to help process and analyze their data. The challenge is that cyber security experts do not have necessary artificial intelligence (AI) trainings to apply ML techniques to their problems. One key aspect is that they do not know how to extract features from cyber-security-related data and to preprocess a data set for ML applications. According to He et al. [5], a deep learning approach is more efficient to analyze the IoT device network traffic within an edge computing environment. The edge computing helps to scale down the size of intermediate data, which is smaller in size compared to input data resulting from efficient feature extraction. An effective AI system trains to learn techniques based on its featured data set. Data preprocessing and feature selection are the steppingstones in ML techniques. Tohari et al. [6] developed a technique based on a combined correlation-based features selection (CFS) technique and particle swarm optimization (PSO) to perform feature selection. The model was designed to analyze the network traffic for intrusion detection systems (IDSs) and achieve high-accuracy prediction. The goal of this library is to help bridge this gap. In particular, we proposed the development of a toolkit in Python that can process the most common types of cyber security data. This will help cyber experts to implement a basic ML pipeline from beginning to end. The proposed research work is our first attempt to achieve this goal. The presented toolkit is a suite of program modules, data sets, and tutorials on YouTube [7], supporting research and teaching in cyber security and defense.

In this paper, we discussed a toolkit for cyber security and ML. In particular, our goal was to provide ways of extracting features from cyber-security-related data for ML applications. The library provides functions and data examples that show different extracted features and associated reasons. We have been developing a library for feature extraction of cyber-security-related data, so it is ready for ML with other libraries like Keras [8], TensorFlow [9], and sklearn [10]. The library in its current form has functions to extract features from Wi-Fi data, standard transmission control protocol/internet protocol (TPC/IP) data, malware related logs, etc.

Libraries are very valuable tools to help AI practitioners to implement their applications. There are many important libraries for data science with Python such as TensorFlow [9], sklearn [10], and pandas [11]. Domain-specific libraries have been used widely in the past and have proven valuable to improve efficiency in the development of ML applications. A good example of a domain-specific library of note is the natural language toolkit (NLTK) [12]. This library helps linguists, data scientists, and others to more efficiently process texts for ML applications. The aim of our proposed library is similar but for cyber security data.

Whereas for NLTK the main source of data was text and language, the main source of data for cyber-security-related problems must be identified. The main sources of data traditionally have been network data in the form of network packet capture protocols (PCAPs), log data, and system files. In [13], Calix and Sankaran looked at network data to address their cyber security problems. In these studies [14], the researchers looked at malware data. Other possibilities include phishing [15], biometrics [16], etc.

Feature extraction algorithms aim to extract or engineer meaningful features from their media. These features usually create vector spaces and need to have certain properties related to data distribution, normalization, and dimensionality [17]. Common problems with extracting features from network data are shown as following: differentiating the types of packets, dealing with encryption, and processing payloads. Most network-related studies focus on a specific type of packet (Internet Control Message Protocol (ICMP) or TCP) [13] and usually exclude a payload or treat it separately [18].

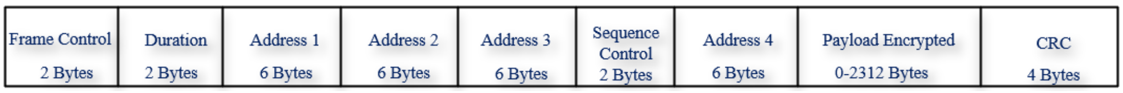
Malware detection is another problem, for which we want to apply ML approaches. There are several challenges, however, such as dealing with polymorphic viruses [14]. Malware usually consists of code that multiplies, makes copies of itself and can deliver a payload. Detecting it can be challenging. Traditionally, static malware analysis through the use of hashing of files has been an approach to dealing with it. Polymorphic malware can evade hashing techniques by changing every copy of a new malware instance. To address polymorphic malware detection, a set of techniques referred to as dynamic malware analysis have been used.

Dynamic approaches need malware to run in a sandbox environment and collect the dynamic behavior of the malware. This dynamic behavior consists, usually, of system calls, registry edits, network connections, access of dynamic link libraries (DLLs), etc. A basic pipeline for this was to take the dynamic malware, run it through an emulator, obtain log files of the behavior and extract features from the logs for ML purposes [14]. According to Hamed et al. [19], a components-based approach enables an easier comparison of methods. Where an intrusion detection system (IDS) can be analyzed in three major phases: preprocessing, detection, and empirical evaluation. For example, a network's physical, datalink, and network packet header's information can be useful to extract features to analyze network behavior. This can be useful to detect anomalies within a network. A malware image analysis technique in [20] was used to classify, detect and analyze particular malware based on image texture analysis. The visualization technique converts the sample malware's executable file into binary strings of 0's and 1's as a vector, which is then converted into two-dimensional images. The image is further compared in terms of factors such as creation, techniques, execution environment, propagation media, and negative impacts. A behavior-based malware-detection ML technique was also discussed in [21]. The combined malware and benign instance of the data set from Windows' portable executable (PE) binaries file is processed online to generate dynamic analysis reports. This is then further used for feature selection and analyzed based on available Weka classifiers. One of the proposed modules in our library was designed to process and extract features from log files (e.g., features from dynamic analysis). In its simplest form, the dynamic malware analysis module can treat each log file per malware (or goodware) sample as a text file (similar to how a natural language document would be treated). From there, a simple bag of words approach can be explored, and this is the most basic analysis that can be implemented. Once the features are extracted, to complete the pipeline, the data must be preprocessed and normalized, ML models are built, and results must be analyzed to evaluate performance. Our library interacts well with standard sklearn and TensorFlow functions to complete the ML pipeline. The point of this library is to bring together modules for feature extraction in the domain of cyber security. Whenever possible, we use existing libraries/dependencies to manipulate raw data. For example, we use Python Scapy functions to process network packets. Libraries like sklearn do not have specific modules to extract features from cyber data. Therefore, our proposed library can be used to extract features from cyber data and those extracted features can then be used for ML applications using other tools such as TensorFlow, and sklearn. This library is a work in progress, and more features can be added in the future by incorporating new modules.

The structure of this paper is as follows: first, an introduction of the library and its need was shown. This section was followed by methods and results. Finally, the discussion of results and future work were presented.

2. Materials and Methods

The library consists of simple Python scripts for feature extraction and some tutorial materials for research and teaching of ML for cyber security. The code for the library is available from GitHub via the link in [22]. The library focuses mainly on two types of data, which are network data and malware-related data. For the network data, we provide examples of how to extract data from Wi-Fi data and from regular network data. All inputs are assumed to be PCAP files. The library uses the Python Scapy [23] library to extract the network features. The outputs are comma-separated values (CSV) files that contain features and their values. Currently, we are focusing on TCP packets and the basic wireless local area network (LAN) link-layer headers without looking into the payloads. As we extend the library, we will explore modules for payload extraction. Payload data are more challenging, because they are more diverse. One common way of looking at payload data is to think of them as documents in language. Therefore, NLP types of approaches can be considered. To preserve confidentiality of data transferred, packets are usually encrypted. Analyzing encrypted packets for ML can be very challenging and is currently not addressed by the library directly. However, according to [24], an encrypted wireless network packet can still be analyzed using the link-layer header information. In this paper, we used the same approach to extract features from wireless network packets. Figure 1 shows an example of a wireless LAN frame.



Frame Control	Duration	Address 1	Address 2	Address 3	Sequence Control	Address 4	Payload Encrypted	CRC
2 Bytes	2 Bytes	6 Bytes	6 Bytes	6 Bytes	2 Bytes	6 Bytes	0-2312 Bytes	4 Bytes

Figure 1. Wireless local area network (WLAN) frame structure (IEEE 802.11 standard frame format).

The wireless network packets to test our modules were collected within a simple testbed lab environment, where multiple IoT devices were connected to an access point (AP). Figure 2 illustrates the lab setup environment. The research work focused on collecting wireless packets as an initial approach to feature extraction. The sample data set is available for the library. A commercial off-the-shelf (COTS) radio receiver was used in monitor mode, to passively eavesdrop the wireless network traffic within a controlled lab environment. We preferred the monitor mode over the promiscuous mode due to its capabilities to identify hidden APs, which can passively listen to the wireless network traffic without associating to the network. With this approach, features can be directly extracted based on nonencrypted link-layer header information.

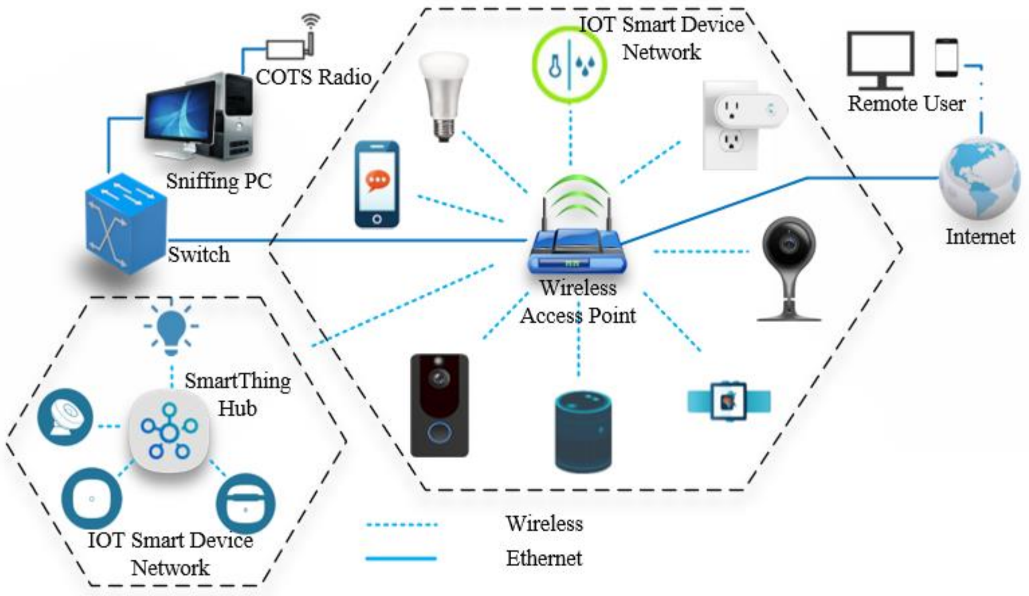
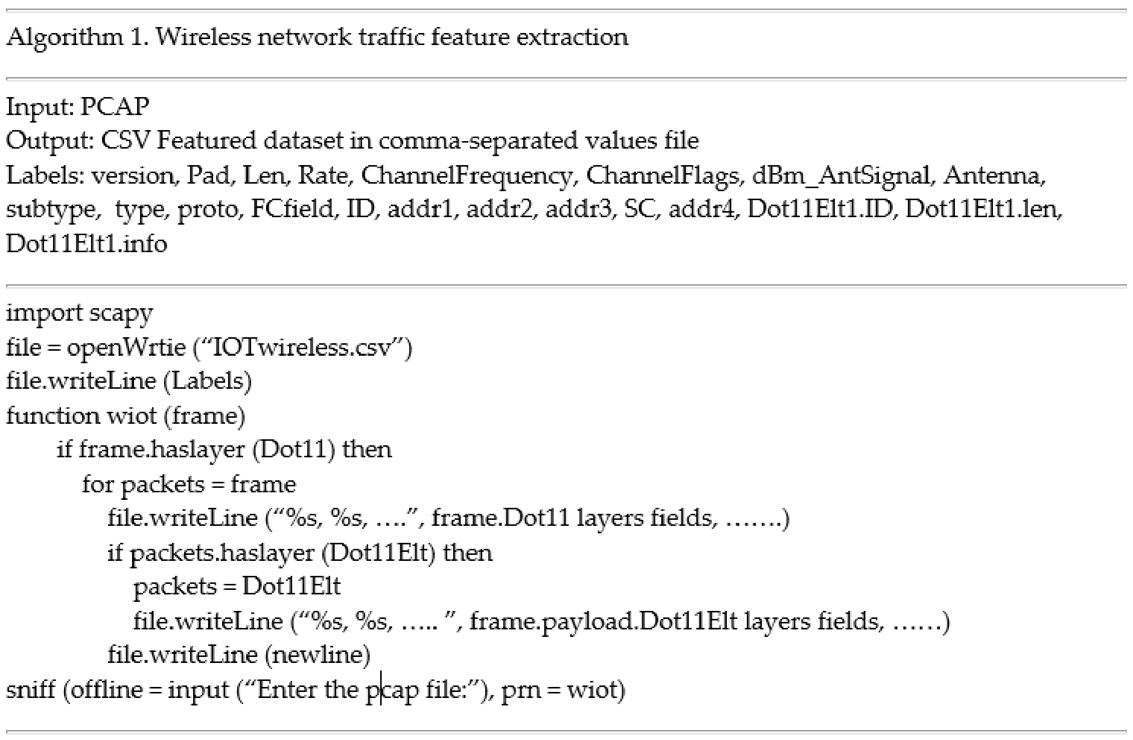


Figure 2. Internet of things (IoT) testbed environment setup.

For example, a sample IoT wireless LAN data set contains 21 features with 94 instances as a proof of concept to test the cyber security toolkit. We used Python Scapy [23], an open-source Python library, to collect the wireless network traffic and further extract features using the proposed library based on Scapy's built-in library support.

Scapy is handy to use in terms of its built-in functions and support community. The user can simply dump network packets and parse them through different layered information within each packet for further analysis. Similarly, Aircrack-ng [25], an open-source Wi-Fi network security toolset, was used to set up a wireless network adaptor in monitoring mode, before we started capturing the wireless network traffic within our testbed lab environment to create a data set.

For our Wi-Fi IoT data example, the library's module extracts features from wireless network traffic-based PCAP files. The module parses the wireless network layer information using Scapy functions and extracts features based on a predefined list into a final CSV file. In its simplest form, each packet becomes an instance or sample of the output CSV file. The procedure, for this example's feature extraction approach, is summarized in Algorithm 1 (Figure 3), where IOTwireless.csv is a labeled data set file. Sniff is a Scapy built-in function to control network packets, and it allows users to pass a function with an argument "prn" to perform custom actions. The variable Labels represents the feature labels list to be extracted from the input PCAP file. Dot11 and Dot11Elt are wireless frame layers, which hold information about the connected wireless network. Dot11 and Dot11Elt are layer fields and are the values associated within each packet with the labels defined in Labels.



```
Algorithm 1. Wireless network traffic feature extraction

Input: PCAP
Output: CSV Featured dataset in comma-separated values file
Labels: version, Pad, Len, Rate, ChannelFrequency, ChannelFlags, dBm_AntSignal, Antenna,
subtype,  type, proto, FCfield, ID, addr1, addr2, addr3, SC, addr4, Dot11Elt1.ID, Dot11Elt1.len,
Dot11Elt1.info

import scapy
file = openWrtie ("IOTwireless.csv")
file.writeLine (Labels)
function wiot (frame)
    if frame.haslayer (Dot11) then
      for packets = frame
        file.writeLine ("%s, %s, ….", frame.Dot11 layers fields, …….)
        if packets.haslayer (Dot11Elt) then
          packets = Dot11Elt
          file.writeLine ("%s, %s, ….. ", frame.payload.Dot11Elt layers fields, ……)
        file.writeLine (newline)
sniff (offline = input ("Enter the pcap file:"), prn = wiot)
```

Figure 3. Algorithm 1—feature extraction.

The malware module focuses on data collected via dynamic analysis (e.g., running a virus in a sandbox through an emulator). The emulator generates logs of the behavior of the malware that is processed for feature extraction. Each log is treated as a document, and its content is treated as words. With this view of the data, features can be extracted using a bag of words approach.

Evaluation

Evaluating a module statistically can be challenging given that we need to assess, in a way, how the code was written and if it is useful to the user. We settled on evaluating the use of some of the modules as part of a class. In particular, we used modules for malware feature extraction and for network data analysis. As part of the class, we also recorded videos and created a virtual machine (VM), data sets, lectures, and labs. We assessed the whole course together and did not focus on a specific module of the library. The survey was conducted to assess the content arrangement and design of the course and to obtain direct feedback on the learning experience. The data came from 19 students who attended classes during the summer of 2019. The open-source learning module was designed to be a week-long ML course for cyber security professional. The course was provided for professionals with a basic knowledge of network security and coding. Most of students in the course possessed very advanced programming skills. The learning module consists of 7 data sets distributed in 15 lectures and 10 labs. All the code modules are implemented in Python and tested in an Ubuntu virtual machine. The VM has many libraries already preinstalled and can be downloaded from [7].

In terms of survey methods and survey data, we followed standard approaches such as in [26]. The survey contained 17 questions, including questions of satisfaction with materials prepared in class, surveys of satisfaction with teaching content and materials, and analysis of teaching results. In addition, participants could write down their other feelings or concerns about the course.

All of the questions in the survey, except for the final comment, used a Likert scale, for a range of 1 to 5. Table 1 shows the scale meanings of the survey answers, and Table 2 demonstrates the survey questions.

Table 1. Likert scale.

Scale	Answer
1	Strongly disagree
2	Disagree
3	Neither agree nor disagree
4	Agree
5	Strongly agree

Table 2. Survey questions.

Number	Question
1	Was Weka a useful tool/topic to learn about?
2	Was sklearn a useful tool/topic to learn about?
3	Was TensorFlow a useful tool/topic to learn about?
4	Were the lectures helpful to better understand the topics?
5	Did you find the lecture and lab documents helpful to better understand the topics?
6	Was the use of a flash drive with a VM helpful for working on the lab problems?
7	Did you find the challenge data sets useful?
8	After this 1-week course, do you feel you have a better understanding of machine learning and how it can be applied to cyber security problems?
9	Did you find the use of AWS as a large data set useful to your learning?
10	Did you find the distribution of time between lecture and labs appropriate?
11	Were the video recording useful to your learning?
12	Overall, do you feel that you can convert raw data to the vector space model format for machine learning purpose?
13	Overall, do you feel you can now better analyze data sets with Weka?
14	Overall, do you feel you can now better analyze data sets with sklearn?
15	Overall, do you feel you can now better analyze data sets with TensorFlow?
16	Do you have a better understanding of deep neural networks and their advantages?
17	Comments (provide any additional comments)

Given that this is a library/coding module, it can be difficult to present statistical results of its usefulness and usability. Given that these modules were used in a class run in the summer of 2019, we used the results of the course surveys as an indirect assessment of these materials. In particular, we emphasized those questions that were more pertinent to the library itself. We presented a full list of questions for completion. For our purposes, we focused on question 12 (Q12), in particular, which we felt was the most relevant to assess our module/library.

Through statistical analysis of the survey data with the R language, we presented some issues of note. This paper tried to find out the relationship between course materials, course content, and course satisfaction, so as to help optimize the materials and meet the needs of target groups.

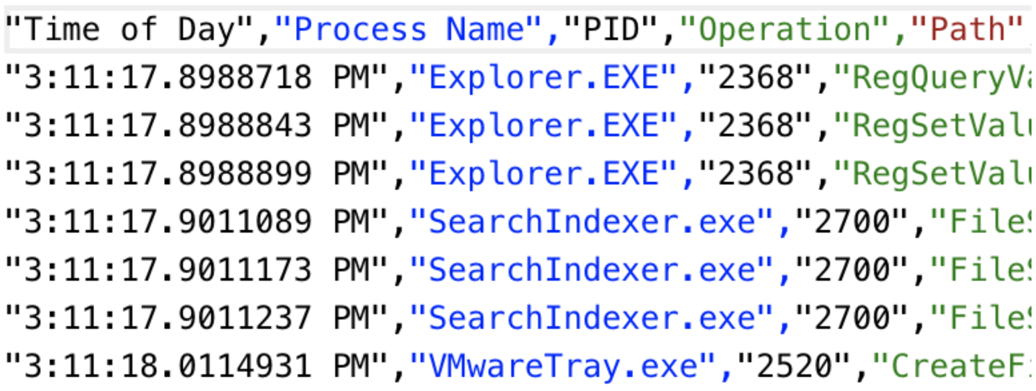
3. Results

Given that this is a library, it can be difficult to present statistical results. However, we ran some of these modules as labs in a course during the summer of 2019. At the end of the class, we collected course surveys. We used some questions in the survey as an indirect way of assessing the library for usability and usefulness. We also assessed the tutorial and videos with the survey to help gauge if the code written for the library was understood by students and if there were correlations.

Finally, we presented a simple use case to illustrate the use of the modules. This use case was taken directly from the lab materials in the course and was related to malware analysis.

3.1. Use Case

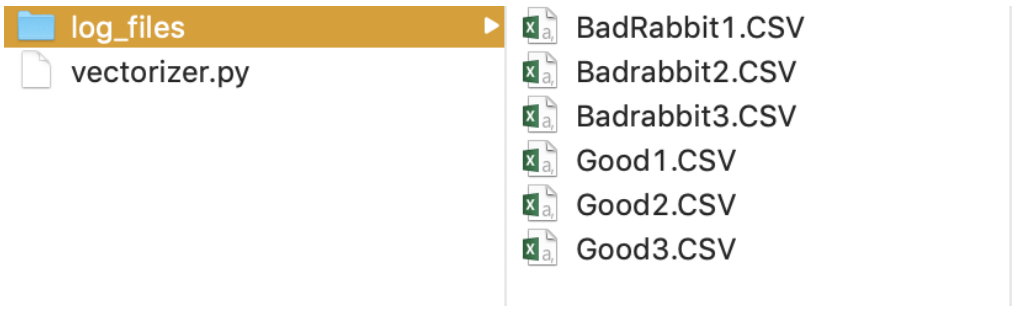
The malware module focuses on data collected via dynamic analysis (e.g., running a virus in a sandbox through an emulator). The emulator generates logs of the behavior of the malware that is processed for feature extraction. The contents of the log are shown in Figure 4.



```
"Time of Day","Process Name","PID","Operation","Path"
"3:11:17.8988718 PM","Explorer.EXE","2368","RegQueryV
"3:11:17.8988843 PM","Explorer.EXE","2368","RegSetVal
"3:11:17.8988899 PM","Explorer.EXE","2368","RegSetVal
"3:11:17.9011089 PM","SearchIndexer.exe","2700","File
"3:11:17.9011173 PM","SearchIndexer.exe","2700","File
"3:11:17.9011237 PM","SearchIndexer.exe","2700","File
"3:11:18.0114931 PM","VMwareTray.exe","2520","CreateF
```

Figure 4. Log of malware behavior.

Each log is treated as a document, and its content is treated as words. With this view of the data, features can be extracted using a bag of words approach. The log labels or classes can be included in the name itself for each goodware or malware sample, as can be seen in Figure 5.



```
log_files          BadRabbit1.CSV
vectorizer.py      Badrabbit2.CSV
                   Badrabbit3.CSV
                   Good1.CSV
                   Good2.CSV
                   Good3.CSV
```

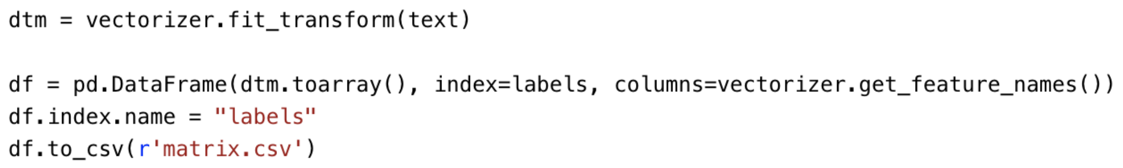
Figure 5. Log files for goodware and malware.

In Figure 6, it can be seen that the module uses CountVectorizer, a Python built-in function, to implement the bag of words approach on each log file. Therefore, each log file will become a sample or row in the output data set.

```
vectorizer = CountVectorizer(stop_words='english', max_features=1000)
```

Figure 6. Initialization of CountVectorizer.

Finally, in Figure 7, CountVectorizer uses pandas to create a data frame containing the output data for every single input log. The output is a CSV file containing all samples.

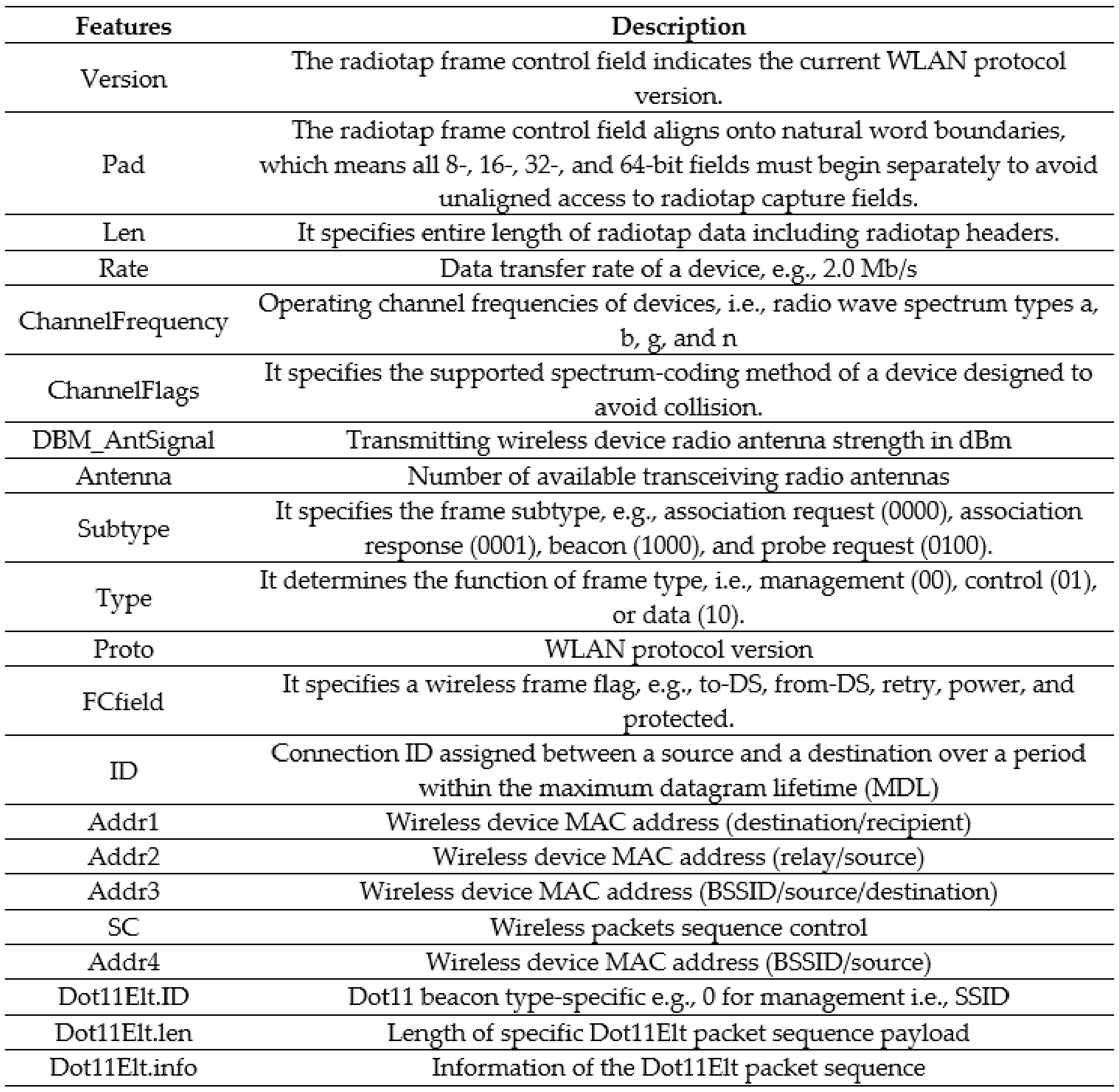


```
dtm = vectorizer.fit_transform(text)

df = pd.DataFrame(dtm.toarray(), index=labels, columns=vectorizer.get_feature_names())
df.index.name = "labels"
df.to_csv(r'matrix.csv')
```

Figure 7. Extraction of features.

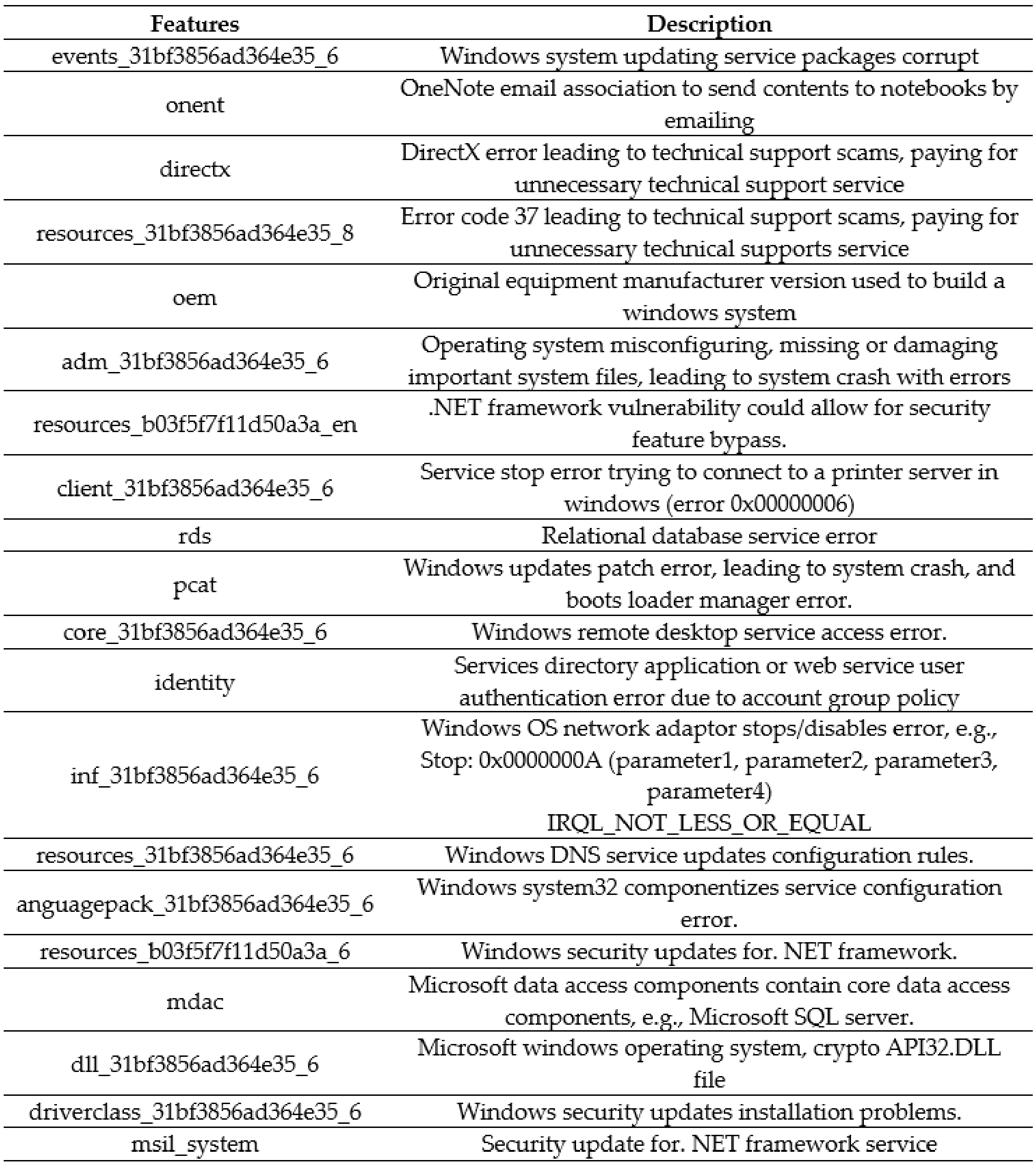
Figure 8 shows a list of some of the most important wireless IoT features and their respective descriptions. The focus is on protocol transmission and control features. Encrypted payloads were not considered.



Features	Description
Version	The radiotap frame control field indicates the current WLAN protocol version.
Pad	The radiotap frame control field aligns onto natural word boundaries, which means all 8-, 16-, 32-, and 64-bit fields must begin separately to avoid unaligned access to radiotap capture fields.
Len	It specifies entire length of radiotap data including radiotap headers.
Rate	Data transfer rate of a device, e.g., 2.0 Mb/s
ChannelFrequency	Operating channel frequencies of devices, i.e., radio wave spectrum types a, b, g, and n
ChannelFlags	It specifies the supported spectrum-coding method of a device designed to avoid collision.
DBM_AntSignal	Transmitting wireless device radio antenna strength in dBm
Antenna	Number of available transceiving radio antennas
Subtype	It specifies the frame subtype, e.g., association request (0000), association response (0001), beacon (1000), and probe request (0100).
Type	It determines the function of frame type, i.e., management (00), control (01), or data (10).
Proto	WLAN protocol version
FCfield	It specifies a wireless frame flag, e.g., to-DS, from-DS, retry, power, and protected.
ID	Connection ID assigned between a source and a destination over a period within the maximum datagram lifetime (MDL)
Addr1	Wireless device MAC address (destination/recipient)
Addr2	Wireless device MAC address (relay/source)
Addr3	Wireless device MAC address (BSSID/source/destination)
SC	Wireless packets sequence control
Addr4	Wireless device MAC address (BSSID/source)
Dot11Elt.ID	Dot11 beacon type-specific e.g., 0 for management i.e., SSID
Dot11Elt.len	Length of specific Dot11Elt packet sequence payload
Dot11Elt.info	Information of the Dot11Elt packet sequence

Figure 8. IoT wireless features.

Figure 9 includes a list of the top 20 dynamic malware analysis features. Because these features are based on the frequency of occurrence of the individual terms in the log files, the list can be extensive. Therefore, we selected only the top 20 most frequent features. The lists of features are also available on the GitHub link.



Features	Description
events_31bf3856ad364e35_6	Windows system updating service packages corrupt
onent	OneNote email association to send contents to notebooks by emailing
directx	DirectX error leading to technical support scams, paying for unnecessary technical support service
resources_31bf3856ad364e35_8	Error code 37 leading to technical support scams, paying for unnecessary technical supports service
oem	Original equipment manufacturer version used to build a windows system
adm_31bf3856ad364e35_6	Operating system misconfiguring, missing or damaging important system files, leading to system crash with errors
resources_b03f5f7f11d50a3a_en	.NET framework vulnerability could allow for security feature bypass.
client_31bf3856ad364e35_6	Service stop error trying to connect to a printer server in windows (error 0x00000006)
rds	Relational database service error
pcat	Windows updates patch error, leading to system crash, and boots loader manager error.
core_31bf3856ad364e35_6	Windows remote desktop service access error.
identity	Services directory application or web service user authentication error due to account group policy
inf_31bf3856ad364e35_6	Windows OS network adaptor stops/disables error, e.g., Stop: 0x0000000A (parameter1, parameter2, parameter3, parameter4) IRQL_NOT_LESS_OR_EQUAL
resources_31bf3856ad364e35_6	Windows DNS service updates configuration rules.
anguagepack_31bf3856ad364e35_6	Windows system32 componentizes service configuration error.
resources_b03f5f7f11d50a3a_6	Windows security updates for. NET framework.
mdac	Microsoft data access components contain core data access components, e.g., Microsoft SQL server.
dll_31bf3856ad364e35_6	Microsoft windows operating system, crypto API32.DLL file
driverclass_31bf3856ad364e35_6	Windows security updates installation problems.
msil_system	Security update for. NET framework service

Figure 9. Dynamic analysis of malware features.

As can be seen in Figure 9, the terms can relate to anything that can happen on a windows system. In general, binaries call DLLs, and they open files, make internet connections and perform registry edits. The goal of an ML algorithm would be based on lots of data to learn to associate DLL 37 to malware files and DLL 34 to benign binaries, for example, according to their co-occurrence.

3.2. *Survey Analysis*

The analysis in Table 3 showed that there was no correlation between Q12 and Q1, Q2, and Q3. Knowing how to extract features was not correlated to understanding Weka (Q1: p-value = 0.427), sklearn (Q2: p-value = 0.809), and TensorFlow (Q3: p-value = 0.303). Correlation was considered significant at p-value $\leq$ 0.05. This sounds reasonable, since feature extraction and the use of ML algorithms are things that can be learned separately.

Table 3. Significant correlations between machine learning tools and feature extraction (Q12).

		Q1	**Q2**	**Q3**
Q12	Correlation	–0.200	0.061	–0.257
	p-value	0.427	0.809	0.303
Q13	Correlation	0.033		
	p-value	0.897		
Q14	Correlation		0.238	
	p-value		0.341	
Q15	Correlation			–0.213
	p-value			0.397

In addition, a relevant analysis was also conducted to prove whether lecture and lab files helped students understand ML, and the results are shown in Table 4. It is obvious that the preparation of lectures and labs (Q4 and Q5) was helpful for understanding course logic and understanding ML (Q8: *p*-value = 0.029; Q10: *p*-value = 0.021).

Table 4. Significant correlations between course materials and course understanding.

		Q4	**Q5**
Q8	Correlation	0.537	0.502
	p-value	0.021	0.029
Q10	Correlation		0.524
	p-value		0.021

The survey results (Figure 10) indicated that after this one-week course, 77.36% students feel they have a better understanding of ML and how it can be applied to cyber security problems, 15.79% do not know if they understand how ML works, and 10.53% think they cannot understand ML from this course (Figure 10).

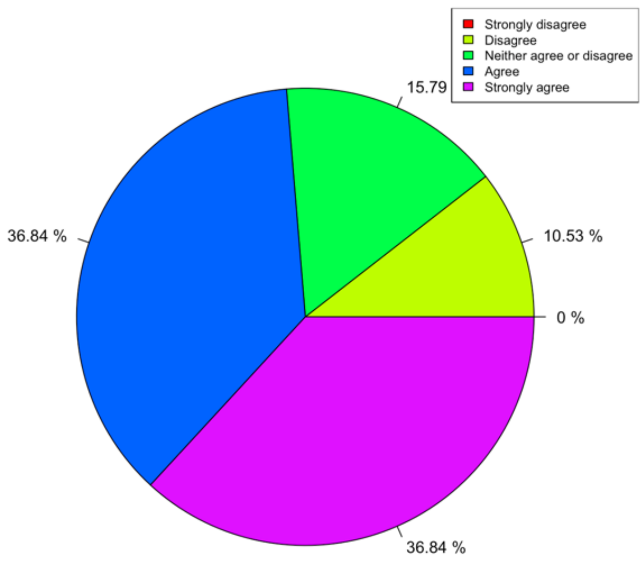


Figure 10. Survey results on students' answers to Q8.

In Figure 11, we focused again on Q12. The pie chart shown in Figure 11 shows that 88.88% students agree that they "can pre-process data into vector space model formats for use in machine learning applications", while 5.56% disagree with that.

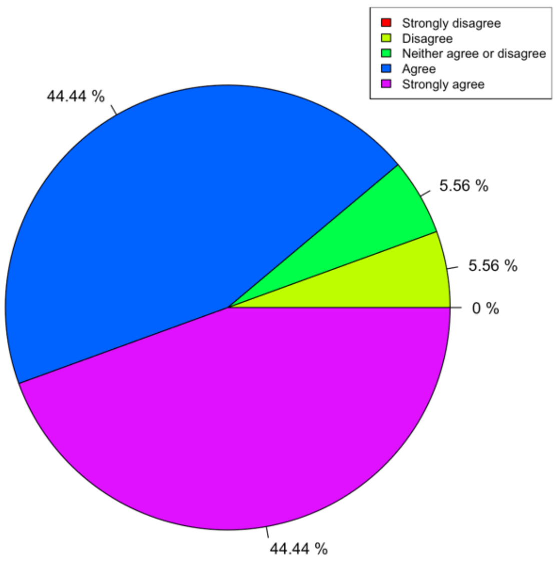


Figure 11. Survey results on students' answers to Q12.

From Figure 12, it can be seen that 88.89% have a better understanding of deep neural networks and the rest of them think they have the opposite opinions. This showed that there is strong interest in deep learning by students, and we will extend the library to reflect this aspect.

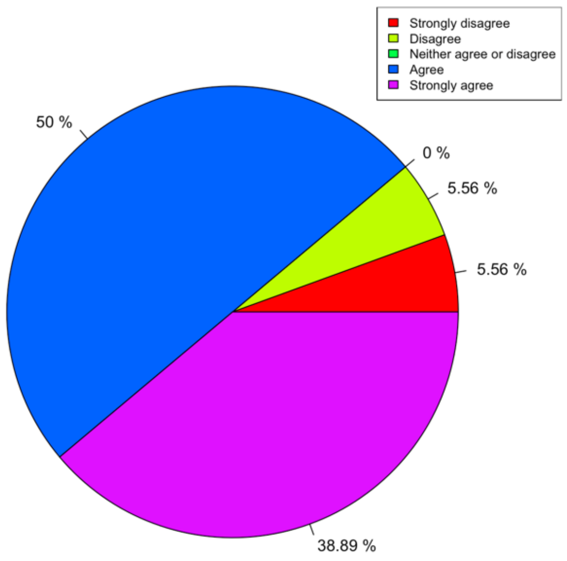


Figure 12. Survey results on students' answers to Q16.

Table 5 shows the summary statistics of each question. Only the mean of Q11 was under 4, which meant videos were not useful for helping students to get improvement from this course or they did not have time to watch them. It is important to note that the videos were not made available to most of students prior to taking the course.

Table 5. Summary statistics per question.

Question	Mean	SD	Minimum	Maximum
Q1	4.632	0.761	2	5
Q2	4.579	0.607	3	5
Q3	4.947	0.229	4	5
Q4	4.222	0.647	3	5
Q5	4.211	0.855	2	5
Q6	4.632	0.597	3	5
Q7	4.368	0.597	3	5
Q8	4	1	2	5
Q9	4.474	0.841	3	5
Q10	4.5	0.816	2	5
Q11	3.667	0.888	3	5
Q12	4.278	0.826	2	5
Q13	4.778	0.428	4	5
Q14	4.278	0.575	3	5
Q15	4.5	0.618	3	5
Q16	4.111	1.079	1	5

4. Discussion

In this paper, we presented and discussed a new library for ML and cyber security. To assess the usefulness of the materials (e.g., the modules for feature extraction), we considered the following question:

- "Is there any relation between learning materials and mastery of knowledge?"

Spearman correlations are often used to assess relationships related to sequential variables, which is well suited for the data studied in this paper, because a Likert scale was used to represent the results of the survey, from strong agreement to strong disagreement. According to Table 4, Q5 and Q8 are related, because the p-value obtained by the correlation test is less than 0.05. The result can be understood like this: "With a better understanding of lectures and labs, students feel they can solve cyber security problems using machine learning", which is a good example to show the course materials are helpful for students to use and learn. Finally, Figure 11 shows that 88.88% students agree that they "can pre-process data into vector space model formats for use in machine learning applications".

Future Work

Future work will involve improving the library and adding more modules. Possible future data sets and modules that can be added include web security data, phishing data, steganography data, etc. Additionally, we will provide more use cases of how to use the library in various other applications.

Author Contributions: R.A.C. wrote most of the paper and guided/supervised all work. S.B.S. collected data from IoT sources, created modules for Wi-Fi and integrated the full library. T.C. wrote code for network and malware feature extraction. D.Z. performed the statistical analysis. M.T. supervised the IoT network cyber security and cyber security feature extraction. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by DoD/NSA (grant number: H98230-18-1-0330).

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. IOT Statistics. Available online: https://ipropertymanagement.com/iot-statistics (accessed on 14 December 2019).
2. IDC Forecasts WorldWide Technology Spending on the Internet of Things to Reach $1.2 Trillion in 2022. Available online: https://www.idc.com/getdoc.jsp?containerId=prUS43994118 (accessed on 14 December 2019).
3. Kolias, C.; Kambourakis, G.; Stavrou, A.; Voas, J. DDoS in the IoT: Mirai and other botnets. *Computer* **2017**, *50*, 80–84. [CrossRef]
4. Spathoulas, G.; Evangelatos, S.; Anagnostopoulos, M.; Mema, G.; Katsikas, S. Detection of abnormal behavior in smart-home environments. In Proceedings of the 2019 4th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM), Piraeus, Greece, 20–22 September 2019.
5. Li, H.; Ota, K.; Dong, M. Learning IoT in edge: Deep learning for the Internet of Things with edge computing. *IEEE Network* **2018**, *32*, 96–101. [CrossRef]
6. Tohari, A.; Aziz, M.N. Data Preprocessing and Feature Selection for Machine Learning Intrusion Detection Systems. *ICIC Express Lett.* **2019**, *13*, 93–101.
7. Course. Available online: http://www.ricardocalix.com/teaching/teaching.htm (accessed on 10 February 2020).
8. Keras. Available online: www.keras.io (accessed on 10 February 2020).
9. Tensorflow. Available online: www.tensorflow.org (accessed on 10 February 2020).
10. Pedregosa. Scikit-learn: Machine Learning in Python. *JMLR* **2011**, *12*, 2825–2830.
11. Pandas. Available online: www.pandas.pydata.org (accessed on 10 February 2020).
12. Loper, E.; Bird, S. NLTK: the Natural Language Toolkit. In Proceedings of the ETMTNLP '02 Proceedings of the ACL-02, Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1, Stroudsburg, PA, USA, 17 May 2002; pp. 63–70.
13. Calix, R.A.; Sankaran, R. Feature ranking and support vector machines classification analysis of the NSL-KDD intrusion detection corpus. In Proceedings of the The Twenty-Sixth International FLAIRS Conference, St. Pete Beach, FL, USA, 22–24 May 2013.
14. Cabrera, A.; Calix, R.A. On the Anatomy of the Dynamic Behavior of Polymorphic Viruses. In Proceedings of the International Conference on Collaboration Technologies and Systems (CTS), Orlando, FL, USA, 31 October–4 November 2016; pp. 424–429. [CrossRef]
15. Basnet, R.; Mukkamala, S.; Sung, A.H. Detection of Phishing Attacks: A Machine Learning Approach. In *Soft Computing Applications in Industry. Studies in Fuzziness and Soft Computing*; Prasad, B., Ed.; Springer: Berlin/Heidelberg, Germany, 2008; Volume 226.
16. Amigud, A.; Arnedo-Moreno, J.; Daradoumis, T.; Guerrero-Roldan, A. A Behavioral Biometrics Based and Machine Learning Aided Framework for Academic Integrity in E-Assessment. In Proceedings of the 2016 International Conference on Intelligent Networking and Collaborative Systems (INCoS), Ostrawva, Czech Republic, 7–9 September 2016; pp. 255–262. [CrossRef]
17. Calix, R. *Getting Started with Deep Learning: Programming and Methodologies using Python*, 1st ed.; CreateSpace Independent Publishing Platform: Scotts Valley, CA, USA, 2017.
18. Iqbal, I.M.; Calix, R.A. Analysis of a Payload-based Network Intrusion Detection System Using Pattern Recognition Processors. In Proceedings of the 2016 International Conference on Collaboration Technologies and Systems (CTS), Orlando, FL, USA, 31 October–4 November 2016; pp. 398–403. [CrossRef]
19. Hamed, T.; Ernst, J.B.; Kremer, S.C. A survey and taxonomy on data and pre-processing techniques of intrusion detection systems. In *Computer and Network Security Essentials*; Springer: Cham, Switzerland, 2018; pp. 113–134.
20. Aziz, M.; Patrot, A. Overview of malware analysis and detection. *Int. J. Comput. Appl.* **2015**, *975*, 8887.
21. Firdausi, I.; Erwin, A.; Nugroho, A.S. Analysis of machine learning techniques used in behavior-based malware detection. In Proceedings of the 2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies, Jakarta, Indonesia, 2–3 December 2010.
22. CyberSecTK. Available online: https://github.com/sumendrabsingh/CyberSecTK-Library (accessed on 10 February 2020).
23. Scapy. Available online: https://scapy.net (accessed on 10 October 2019).

24. Rajib, R.M.; Sandra, S.; Ragv, S.; Nils, O.T. Link-Layer Device Type Classification on Encrypted Wireless Traffic with COTS Radios. In *European Symposium on Research in Computer Security*; Springer: Cham, Switzerland, 2017; Part II; pp. 247–264.
25. Aircrack-ng. Available online: https://www.aircrack-ng.org (accessed on 10 October 2019).
26. Calix, R.A.; Mallepudi, S.A.; Knapp, G.M.; Nahmens, I. Factors that Influence Usage of Knowledge Management by Information Technology Professionals at Institutions of Higher Education. *J. Manag. Eng. Integr.* **2010**, *3*, 73.