International Journal of Recent Advances in Information Technology & Management
http://eurekajournals.com/ITM.html
ISSN: 2581-3609

Exploring Security Advancements in Distributed Systems through Artificial Intelligence and Machine Learning Approaches: A Comprehensive Review

Narinder Pal Singh[1], Rajiv Anand[1]

[1]Jawaharlal Nehru Technological University, Kakinada.

Abstract

The evaluation of intrusion and malware detection systems in network security encounters challenges due to the scarcity of publicly available and current datasets. This paper introduces the HIKARI-2021 dataset, encompassing encrypted simulated attacks and benign traffic, addressing content and process requirements for dataset development. The outlined requirements aim to facilitate future dataset creation, and both the HIKARI-2021 dataset and its creation methodology are made publicly accessible.

Focusing on a crucial aspect of intrusion and malware activities—specifically, the distribution and installation of programs on a large number of victim computers—this study delves into drive-by download attacks. These attacks entice victims to websites initiating exploits against their web browsers, leading to the automatic download and execution of malicious programs through injected shellcode. While prior research primarily concentrated on identifying the drive-by exploit stage and subsequent network traffic, the intermediary phase of malware download has received limited attention.

Our system demonstrates notable capabilities in identifying malicious applications, achieving a high accuracy rate (97.69 percent true positive) and a low false positive rate (0.43 percent). Importantly, this detection proficiency is observed weeks or even months before the appearance of identified threats on public blacklists.

Introduction

In the field of Intrusion and Malware Detection Systems (IMDS), assessing advancements in malicious detection technologies poses challenges. Machine learning-based IMDS relies on training datasets, but obtaining a reliable dataset for comparative analysis proves challenging. The difficulty lies in the absence of comprehensive documentation of approaches, a lack of standardized comparison methodology, and limited critical aspects like surface labels and publicly accessible 3D vision traffic [1][2]. Moreover, the prevalence of encrypted network

traffic for communication security and privacy adds complexity, as only a few databases remain unencrypted.

A significant challenge for attackers is the widespread installation of malware applications on target PCs. Social engineering, such as sending enticing email messages to victims, is one method; however, it requires victim participation and is often ineffective. An alternative, more potent strategy involves enticing individuals to visit web pages initiating attacks against browser vulnerabilities, known as drive-by downloads. This approach requires no human involvement, allowing malware to be loaded and activated on the victim's PC without their awareness. Drive-by downloads have emerged as a primary method for attackers to propagate malware due to their effectiveness and stealthiness, forming the focus of this research.

Current datasets lack the necessary encrypted trails and practicality to construct a comprehensive model for identifying new threats. While some research addresses encrypted traffic, with a focus on areas like traffic classification and analysis [3], available datasets are limited and not publicly accessible due to data sensitivity [4].

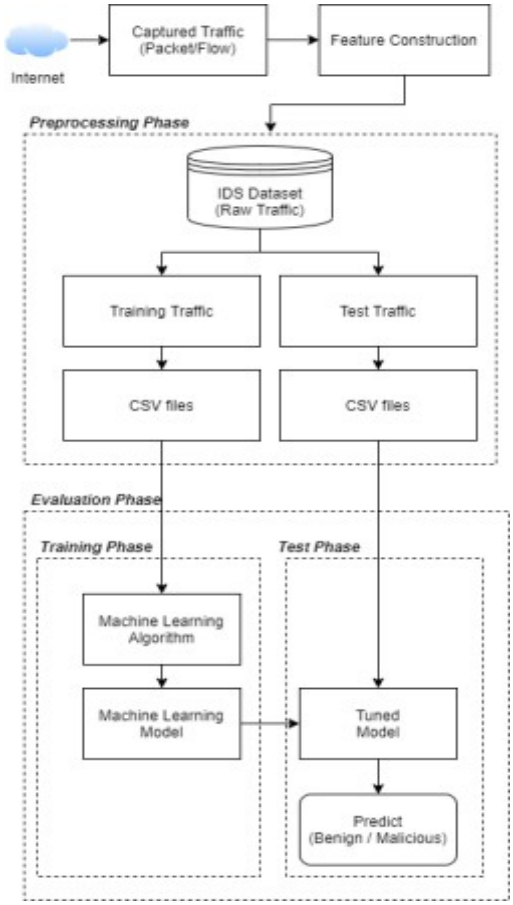

Figure 1: A generic IMDS pipeline basedon Machine Learning.

Benchmark datasets play a crucial role in the evaluation and comparison of various Intrusion Detection Systems (IDS). IDS are categorized into three types based on detection methods: signature-based, anomaly-based, and a hybrid of the two. The widely used KDD99 dataset is employed by all three types to assess their performance. In the signature-based approach [5], the emphasis is on automating signature generation, utilizing a pattern-matching method to

detect and match with signatures from a database. Although this method is highly accurate with minimal false alarms, it cannot identify unknown attacks. On the other hand, the anomaly-based approach [6] focuses on detecting deviations from the typical profile, potentially identifying unknown threats by comparing abnormal traffic to normal patterns. However, this approach tends to have a higher false alarm rate.

Contemporary malware security solutions predominantly concentrate on the initial and later stages of infection. Much effort is dedicated to detecting pages with vulnerabilities for drive-by downloads during the early phases of exploitation, preventing browsers from accessing malicious websites. Honeyclients, for instance, actively scan the internet to identify pages containing attack code, subsequently adding them to domain and URL blacklists. In response, attackers have adopted strategies such as rapidly cycling through malicious domains, rendering blacklists ineffective. Additionally, to evade detection, attackers have begun fingerprinting honey clients and employing code obfuscation techniques.

The following are the paper's main contributions:

- We describe a new method for detecting malware downloads and installations via web requests. Our technique works on huge networks and detects the characteristics of malware distribution networks that are organised. Nazca can detect previously undetected malware and is unaffected by content obfuscationbecause our technique does not analyze the downloaded programmes.
- We describe a three-step technique for detecting malware downloads. The first step is to get a summary of HTTP request metadata. Then we seek for suspicious candidate connections with features that are out of the ordinary for benign downloads. These capabilities identify malware creators' evasive attempts to escape detection by traditional security systems. Finally, we combine candidates to identify potentially dangerous activity clusters.
- We compare seven days of data from a commercial ISP against prominent blacklists to see how good our techniqueis at detecting malware downloads.

Related Work

Because the current study focuses on defence against adversarial malware instances, we review analogous former studies in four categories: ensemble learning, input prepossessing, adversarialtraining/regularization, and DAE basedrepresentation learning.

By diversifying the building-block models, ensemble learning can minimise generalization error. Bagging and random subspace approaches, as demonstrated by Biggio et al. [18], [19], can improve the resilience of linear models against evasion attempts. To identify adversarial malware, Smutz and Stavrou [20] suggest utilising the confidence score provided by random forest classifiers. The durability of ensemble DNNs against evasion attempts is investigated by Stokes et al. [21]. We use the random subspace approach and randomly initialised parameters to diversify the building- block models in this study.

To reduce the amount of perturbations applied to the original data, input prepossessing modifies the input's representation. For example, in both the training and test phases, Random Feature Nullification (RFN) cancels features at random [12], HashTran [13] employs municipality hashing to reduce tiny perturbations, and DroidEye [14] quantizes binary representationvia count featurization. Binarization, which isinspired by the concept of feature pinching, is used to reduce disturbance in our system.

Foster learning adds adversarial instances to the training data. [5], [12], [13], [26], [27] are only a few examples of heuristic training procedures that have been developed. These solutions, on the other hand, are usually targeted at certain evasion techniques and are ineffective againstothers. In addition, researchers suggest using adversarial training with the ideal attack, which is similar to the worst-case situation and might lead to classifiers that are resistant to non- optimal assaults [7], [28]. In our system, we usea gradient descent approach to find the best attack while using a closest neighbour search to fulfil the need of discrete inputs.

Adversarial regularisation is an adversarial training strategy that seeks to train a model using minimally disrupted data while keeping functionality. Small perturbations, intuitively, help DNN models generalise [13], [27], [29],[30], [31]. Because the defender may not know the attacker's manipulation set in the context of malware detection, this technique may be beneficial.

DAE aids in the learning of strong representations [32], [33]. DAE is proposed by Li et al. [23] for identifying adversarial malware instances. DAE is used in our system to learn a resilient representation that is unaffected by disturbances.

Requirement Datasets

Despite the fact that the authors of ISCX [8,UGR'16 [10], and CICIDS-2017 [13] all supply a new dataset and state major requirements for it, their research aims and scope differ. Unlike their prior dataset, our effort is a supplement to fill in the gaps left by the previous requirement.

IDS Evaluation Datasets Requirement

Distinct datasets, on the whole, have different assets and requirements. By using a systematic profile to produce the dataset, Shiravi et al. [8] focused on correct labelling in the dataset. They claimed that network traffic should be as realistic as possible, which necessitated a comprehensive capture in a genuine network. It will have an influence on anonymity and may result in privacy concerns. Fernandez et al. [10] only supplied flow data and concentrated on the capturing time. Furthermore, if the malicious communication is provided via an encrypted protocol, such as HTTPS, a flow format with just 5 tuples is insufficient and requires extra characteristics. We discovered that Sharafaldinet al. [52] had a long list of prerequisites for creating an IDS dataset. Unfortunately, the traffic they create is generated on an emulated network that lacks a genuine environment.Furthermore, their research lacked information on ground-truth and how labelling works, and so has the potential to be incorrect and

untrustworthy for analysis. Cordero et al. [59] developed an ID2T tool, and we found that their criteria are more realistic. They divided the needs into two categories: functional and non- functional. Non-functional requirements describe a collection of criteria that must be met in order for a dataset to be useful. Functional requirements focus on what is required to create datasets, but non-functional requirements specify a set of criteria that must be met in order for a dataset to be useful.

Comparison of Existing Datasets to the Above-mentioned Requirements

We couldn't find any information regarding the UMass dataset's anonymity, thus no indicator was supplied. In terms of the IMPACT dataset, this platform offers a range of datasets, some anonymised and others not. In one area of the CICIDS-2017 dataset, there are examples of encrypted communication with benign and attack characteristics. Four conclusions may be drawn from the above comparisons. To get started, you'll need encrypted samples of both benign and malicious transmission. We noticed that [15] has information in their dataset on whether traffic is anomalous or suspicious, but it is reliant on anomaly detectors. The study did not include the payload from the packet traces. The possibilities of IDS are limited since many assaults cannot be detected by network traffic with just 5-tuple characteristics. In addition, traffic from benign and SSH attack profiles is included in [13]'s data.

While this is beneficial, the attack's diversity should be expanded to encompass applications such as browser attacks and different protocols, such as HTTPS, which we did not find in their dataset. Second, we found that the great majority of datasets aren't anonymized. This is most likely due to the fact that their testing beds are in a controlled environment or that they have authorisation to operate. The ideal option is the first, which means that the traffic will have more phoney traffic and less actual traffic. The latter is preferred if they can protect their privacy. Furthermore, anonymizing traffic can protect privacy; yet, anonymizing traffic substantially may diminish the analysis' findings [8].Third, we noticed that most datasets lacked ground-truth data and background traffic, confining the inquiry to their model. Fourth, there must be a way to generate a new dataset.

This is owing to the fact that the network environment is always changing. It is critical to understand how to develop new datasets with practical implementation so that researchers may create their own datasets and analyse them in their own environments. This technique may be used as a guideline for IDS researchers to create a useful dataset.

Configuration of the Network for Dataset Generation

Figure 2 depicts our network design, which places the attackers on a different network than the victims. The data we collected was in pcap format. The following is the most significant aspect of this configuration:

- CentOS 7 and CentOS 8 are used to run the attacker network, which consists of two computers. The attackers' machines do not need to meet any precise criteria as long as they can execute Bash and Python programmes. From Miniconda 3, the Python version is 3.8.8.

- One Debian 8 system running Joomla3.4.3 and two Debian 9 workstations running Drupal 8.0 and WordPress 5.0 are deployed in the victim network. There are no special requirements for the OS version of the victim network, and default themes and plugins are used by the three distinct Content Management Systems (CMS), such as Drupal, WordPress, and Joomla.

Because of their popularity, these three open-source CMSs were chosen. Background traffic is collected by these computers.

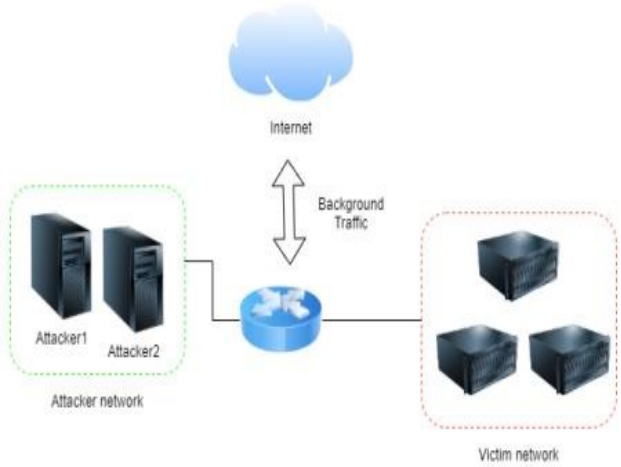

Figure No. 2: Configuration of the Network forDataset Generation

It's critical to collect data that is realistic. We recorded all background traffic in the victim network without using any filters or firewalls. As a result, there's a chance that background traffic contains malicious traffic or assaults. We anonymised various pieces of data, including theIP address and the payload, to protect privacy without compromising the analysis' results.

To develop the benign profile, we looked into using a benign profile that was similar to human behaviour. To accomplish so, we used Selenium [12], a headless browser that runs Google Chrome and Mozilla Firefox. These two browsers resemble individuals by browsing random websites, registering as a user, logging in, sending an article to the intended victim's server, and then logging out. We used a variety of factors, such as user-agent and random delay, to make each set of activities behave like a person and prevent being identified as a bot or web spider. Alexa's top 1 million visitors [13] provided the website URLs. The benign profile, which duplicates innocuous traffic, was created using a Python script.

The attack traffic is created in two ways: first, by targeting a specific page for CMS user login, and second, by scanning for vulnerabilities in the CMSs. The HTTPS protocol is used to deliver both assaults. The assaults are carried out on several days and in various settings (details inSection 4.5). The following are the different sorts of attacks:

- The brute force assault is the most well- known method for breaking passwords. The attacker frequently attempts all possible combinations using a dictionary of probable common passwords [64] over and over again to get access to the target. We created a script that simulates a brute force assault and delivers it through a browser. We added a

user with the role of admin and a password that we chose at random from [64] to each of the three CMSs. The goal is to ensure that the brute force attack goes off without a hitch.

- Brute force attack using two alternative attack vectors: the first utilises the browser as the attack vector, while the second employs XMLRPC as the attack vector. We created a script that uses XMLRPC to acquire access to valid credentials.
- Vulnerability probing is another term for probing. This software examines web applications for vulnerabilities, such as Joomla, WordPress, and Drupal. Vulnerability scanning software is freely accessible. The programmes employed the probing scripts droopescan [65] for WordPress and Drupal, and joomscan [66] for Joomla, for this dataset.

Researchers can utilise the template script to tweak the attack profile and create bespoke assaults utilising other vectors. The source IP address, source port, destination IP address, destination port, protocol, and the day both of the profiles were formed are used to distinguish between an attack profile and a benign profile. Furthermore, any destination addresses in the Alexa list are considered benign for determining benign traffic.

Processing of Dataset

Tcpdump with complete packet capture was used to capture the traces. In terms of background traffic, we fully collected everything but anonymized it to protect our clients' privacy. We employed a Crypto-PAn-based method to maintain privacy [67]. There are multiple files in the whole dataset: pcap files from background traffic and synthetic assaults. Downloads of the flowmeter files in pkl and CSV are available [68]. Figure 3 depicts the preprocessing route from pcap files to CSV files.

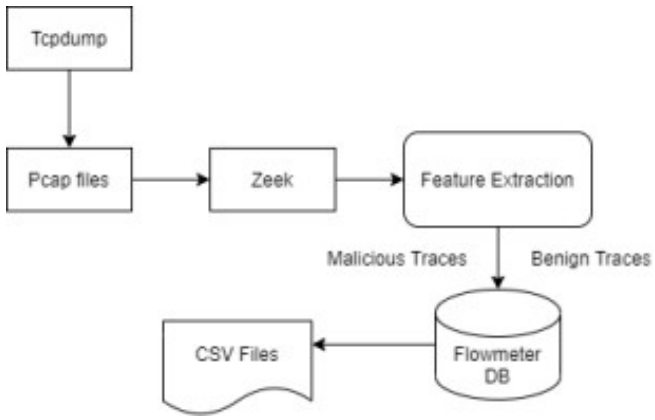

Figure 3: The HIKARI-2021 dataset'spreparation flow.

Selection of Candidate

Nazca operates by combining data from a variety of sources. Nazca evaluates the set of all metadata records acquired during a certain time period T in the candidate selection stage. The candidate selection process can be repeated on a regular basis, evaluating the connections in T chunks. Alternatively, a sliding window of length T could be kept. This

step's purpose is to generate a candidate set with questionable connections. These are connections that behave in ways that are usually linked with malicious download or installation activities.

To find prospects, we employ four distinct methods. Repackaged malware, distributed malware hosting, ad hoc harmful domains, and the usage of malware droppers are all covered by these four strategies. Our algorithms are intended to be quick so that they can handle enormous datasets. Of course, we can't expect a single approach to be able to manage all types of malware distribution. Our experiments, on the other hand, show that the combination of our strategies provides enough coverage. Furthermore, other procedures can be quicklyadded if necessary.

We'll go over the filtering strategies in more depth now. Keep Figure 1 handy as a guide to deciphering the numerous symbols on the graphsthat go with them.

File Mutation Detection

Our initial detection system catches malware developers trying to get around antivirus signatures. Malware writers regularly alter their files to prevent signature-based detection at the end host. This is usually accomplished by packaging the basic malware software with each repetition using a new (encryption) key. Server- side polymorphism refers to the process of developing and providing a new variation of a malware programme (or family). Malware developers can also create a series of harmful executables and deliver each request with a new one.

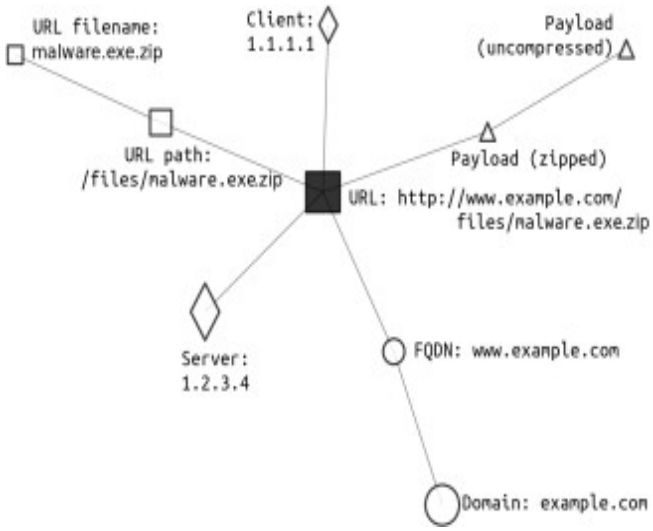

Figure 4: Throughout the publication, thereis a legend to help you read the graphs.

Our method for detecting file modifications (also known as server-side polymorphism) searches for download records that are I connected with a single URI and (ii) download more than n distinct files (determined by the hashes of the files). Later, we'll talk about howto choose an appropriate n threshold. The wonderful thing about this method is that as cyber criminals try harder to avoid being caught by antivirus signatures, our technique becomes more capable of detecting such behaviour.

Cacaoweb.org is an example of a malicious website that employs this method, since it contains URIs that serve an executable that changes every few hours (and is still operating at the time of writing). Other examples are the domains www.9530.ca, www.wgcqsf.com, and 585872.com. All three sites use the /dlq.rar URL route, which carries malware that targets InternetExplorer.

This example is particularly interesting becauseit demonstrates how malware distributors are attempting to evade antivirus signatures not onlyby repackaging their malware (thus exposing themselves to detection by our technique), but also by using multiple domains to achieve robustness against blacklisting and take-downs, a behaviour that our **subsequent techniques handle.**

Malware distributors often have a small number of URLs (sometimes only one) providing executables from each of their domains. Genuine CDNs, on the other hand, have massive directory structures. This discrepancy is interpreted as a sign of hostile intent. Malware distributors may, with some effort, imitate this component of a legal CDN. However, we discovered that this was a rare occurrence. We count all URIs across all domains in the cluster and divide by the number of domains for this characteristic.

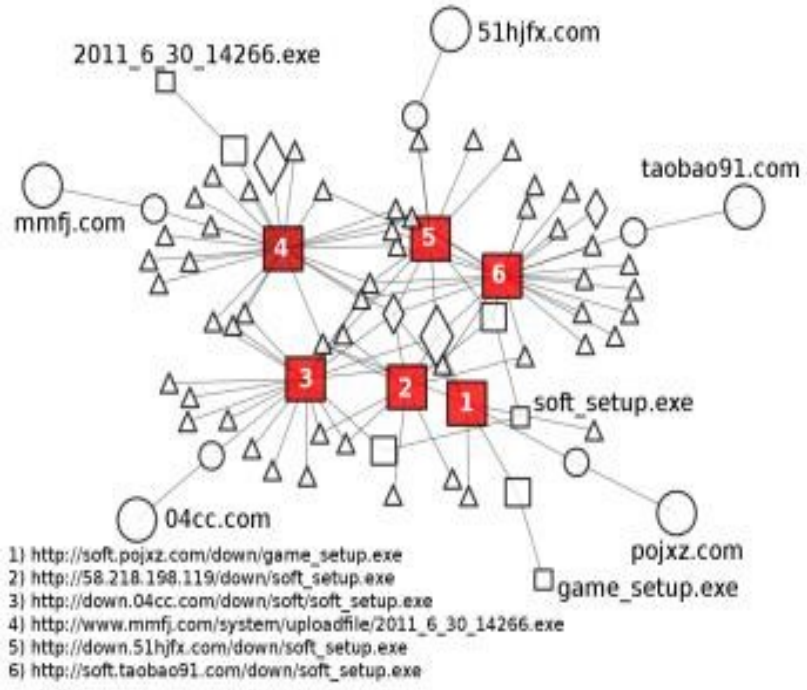

Figure 5: Candidates are chosen using theDistributed Hosting method

Detection Methodology

The ultimate result of Nazca's candidate selection phase is a list of URIs that have been picked by one (or more) of our candidate selection algorithms because they have displayed some suspicious behaviour. These algorithms are designed to efficiently filter the bulk of benign traffic; nevertheless, due to a lack of contextual information, they occasionally misclassify benign traffic as suspicious.

We use the size of our dataset, in particular the fact that it frequently contains multiple clients' interactions with various components of a malware distribution infrastructure (which may span multiple hosts and domains), to eliminate false positives and direct security administrators' attention to large malware campaigns. If Nazca is successful in gathering a sufficient number of these exchanges, the whole structure of these viral dissemination services, as well as their connections, will be disclosed. When one (or more) malicious distribution services are bundled together, they form a "malicious neighbourhood," which is a collection of domains, hosts, executable files, and URLs that define and comprise one (or more) maliciousdistribution services.

Figure 5 is an example of a fully malevolent neighbourhood. We have multiple customers here that download a variety of malicious executables from a number of sources, which are housed on a variety of servers. They are being turned into zombies by the virus they are installing. These clients connect to a number of IP addresses and download their instructions to accept commands from the botnet's C&C server. When we operate on a single connection or single host level, the full structure of themalicious infrastructure is not evident.

Finally, the malware economic environment is complicated, and cybercriminals prefer to specialise in providing a certain service [6]. Some, for example, concentrate on infiltrating hosts and selling their bots. Others buy bots and utilise them as a platform for sending spam. Others create exploits that are marketed to individuals looking to infect computers. Clients that connect domains associated to attackers who infect machines are expected to appear in the harmful neighborhood graph. Following that, we find infected devices connecting to a new set of domains, ones that are tied to the malware's C&C infrastructure.

We create harmful neighborhood graphs for each candidate obtained by the candidate selection stage in this detection step. If a candidate is malicious, there are likely to be additional malicious candidates in the same graph, belonging to the same harmful neighbourhood (as we have empirically proven). We next compute a confidence score for each candidate based on its chance of being malicious using this graph. This score is determined by the number of other candidates on the same graph and their proximity to the input candidate.

Applying Principles into Practice

We offer a methodology for adversarial malware categorization and detection based on the ideas outlined above, which is emphasised in Figure 1 and detailed below. First, we look to see if the assaults contain any helpful information that may be integrated via suitable preprocessing (according to Principle 1).

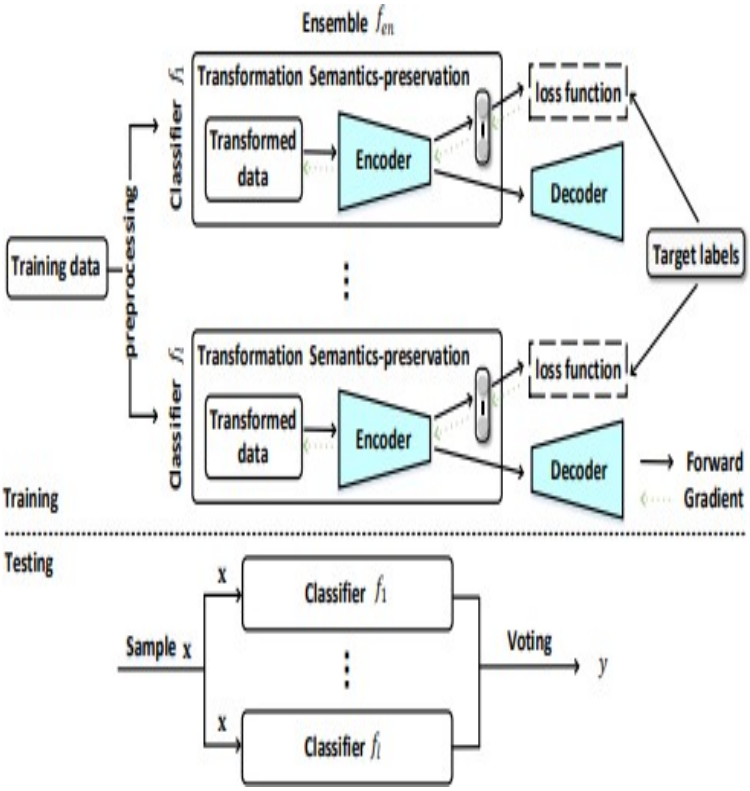

Figure 6: The adversarial malwareprotection architecture in action.

Algorithm 2: Training classifier f_i

Input: The training set (X, Y), number of repeat times K, epoch N_{epoch} and mini-batch size N.

Select a ratio Λ of sub-features from the feature set;
Transform input X to $\overline{X}$ via binarization;
for $epoch = 1$ to N_{epoch} **do**
  Sample a mini-batch $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$ from the $(\overline{X}, Y)$;
  for $k = 1$ to K **do**
    Apply slight salt-and-pepper noises to $\{\mathbf{x}_i\}_{i=1}^{N}$;
    Derive the perturbed representation $\{\mathbf{x'}_i^k\}_{i=1}^{N}$ via Algorithm 1 using Adam method;
  end
  Select $\mathbf{x}'_i$ from $\{\mathbf{x'}_i^k\}_{k=1}^{K}$ for $\mathbf{x}_i$ $(i = 1, \cdots, N)$ to maximize the cross-entropy loss;
  Calculate the reconstruction loss via Eq.(9);
  Backpropagate the loss and update the denoising autoencoder parameters;
  Calculate the adversarial training loss via Eq.(6);
  Backpropagate the loss and update classifier parameters;
end

An ensemble of classifiers fili=1 trained from a random subspaceof the original feature space is suggested (according to Principle 3). To harden each classifier fi, three countermeasures are used: input transformation via binarization (as per Principle 4); adversarial training/regularization models on attacks using Adam optimizer (dot arrows in Figure 1, as per Principles 2 and 5); and semantics preservation via encoder and decoder (as per

Principles 2 and 5). (according to Principles 2 and 5). (as stated in Principle 6) We use block coordinate descent to learn classifier fi and use it to optimise numerous components of the model in order to accomplish adversarial training while maintaining semantics.

Experiments on Attacks and Classification Results

We propose threat models based on whether theattacker uses grey-box or white-box assaults, aswell as limits on the manipulation set of the attacker.

White-box vs. grey-box Attacks. We look at two different assault scenarios. I Grey-box attacks: In this scenario, we recreate the AICS'2019 Challenge organisers' attack scenario. That is,the attacker knows the dataset and feature set,but not the learning technique used by the defence. A surrogate classifier is used by the attacker to produce adversarial instances. On the training set, we use a surrogate model with two fully-connected hidden layers (200 neurons each layer) to learn the model. (ii) White-box attacks: In this scenario, the attacker has complete control over the malware detection system. As a result, the adversarial instances are created straight from the malware detector.

Constraints on Manipulation We consider both incremental and decremental operations with an APK. To prevent detection, the attacker can use incremental modification to inject some items (for example, activity) into an APK example. The attacker can use decremental manipulation to hide particular items (for example, activity) and evade detection. In any scenario, the dangerous functionality of the virus from which the adversarial example is built should bepreserved.

The attacker can inject some manifest features (e.g., seek extra rights and hardware, state additional services, Intent-filter, etc.) while using incremental manipulations. However, other components, such as content-provider, are difficult to put since an APK example would be corrupted if the URI is missing. A dead function or class (that is never used) containing specified system APIs can be injected into the.dex file without ruining the APK example. String injection (e.g., IP address) can also be done using similar methods.

The metadata in the xml files in the APK can be modified if the attacker performs decrementaloperations (e.g., package name). However, because an activity may represent a class implemented in the.dex code, it is hard to completely delete it. However, we can rename an activity and alter its necessary information (e.g., activity label) while keeping in mind that the accompanying components in the.dex must be updated as well. Other components (e.g., service, provider, and receiver) can be updated in the same way, as can resource files (e.g., pictures, icons). Developer-defined method and class names in dexcode might be changed as well. It's important to note that the relevant statement, instantiation, reference, and announcements should all be updated. Furthermore, encryption may be used to obfuscate user-specified strings, with the cipher-text being decoded at runtime. Furthermore, the attacker can use Java reflection and encryption to mask public and static system APIs. The example in List 1 demonstrates this. All of the aforementioned tweaks just obscure an APK without altering its functionality.

Mapping Manipulation to future space, The aforementioned alterations change static Android features like API calls and manifest file components. The Drebin feature space can be perturbed in two different ways. I The adding of a feature. The attacker can change the feature values of applicable objects, such as components (e.g., activity), system APIs, and IP address, by flipping '0' to '1'. (ii) The loss of a feature. The attacker can change the value of '1' to '0' by removing or concealing things (e.g., activity, APIs.) Table 2 outlines our Drebin feature space alterations. We see that neither feature addition nor removal may be done to S6 since these features are dependent on S2 and S5, which means that changes to S2 or S5 may result in changes to S6 features.

Experimental Results

When there are no hostile attacks, the defense framework's effectiveness.

Table No. 1: Classification Results

Defense	FNR (Per)	FPR(Per)	ccuracy(Per)
DNN Basic	3.721	0.410	99.30
Adverbial Training	3.331	1.881	98.51
Regularization	4.612	0.193	99.30
DAE	3.314	0.461	99.16
AT & DAE	3.712	1.791	98.23
Ensemble AT & DAE	2.552	0.532	98.10

Adversarial Attacks in the Absence of Adversarial Attacks. Table 3 highlights the test set's classification outcomes, which are quantified using the conventional metrics of False Negative Rate (FNR), False Positive Rate (FPR), and classification Accuracy (i.e., the proportion of properly categorized test cases) [62]. We find that Adversarial Training has a lower FNR (0.438 percent lower) but a greater FPR when compared to the Basic DNN (1.457 percent higher). DAE, AT+DAE, and Ensemble AT+DAE all have a similar trend.

This can be explained as follows: by injecting adversarial malware examples into the training set, the learning process causes the model to search for malware examples over a larger area, resulting in a decrease in FNR and an increase in FPR, and thus a slight drop in classification accuracy (1.74%). Adversarial Regularization has a classification accuracy that is equivalent to Basic DNN, but it has the greatest FNR among the classifiers we looked at. DNN is regularised using modest perturbations applied to both benign and malicious data, which causes this.

Conclusion

To review and compare IDS, it's vital to have publicly available, up-to-date information, especially since network traffic varies over time. This research provides two significant contributions. To begin, we defined a new criterion for developing new datasets that aren't included in existing datasets, such as anonymization, payload, ground-truth, encryption, and a plausible method for doing it. Anonymizing specific data will alleviate privacy concerns, but

collecting data with the payload will improve the amount of data we can collect for detecting harmful activities in encrypted communication. It's vital to supply ground-truth data so that the dataset doesn't contain any unlabeled assaults.

Despite the fact that most current traffic employs encrypted communication to deliver assaults, we have been concerned about the paucity of existing datasets including encrypted traffic. Second, we produced the HIKARI-2021 IDS dataset, which contains encrypted network activity traces. The datasets were constructed utilising a mix of ground-truth data that wasn't available in the IDS databases at the time.

The datasets are accessible for free [68]. As a starting point, we employed over 80 features from CICIDS-2017, including a source IP address (origin), source port (originp), destination IP address, and destination port. Each flow was categorised as benign or malicious, with benign flows being divided into two categories (Benign or Background) and malicious flows being divided into four (Bruteforce, Bruteforce-XML, Probing, and XMRIGCC CryptoMiner).

We want to emphasise what differentiates our dataset apart from others in the IDS. This is based on the ideal specifications we've laid down. The first stems from content requirements, such as comprehensive capture, for which we provide all traces (e.g., background traffic, benign, and attack) as pcap files; the payload is provided with the exception that we anonymize the background traffic, whichis required to protect privacy.

The ground-truth and labels are manually evaluated based on the source IP address, source port, destination IP address, destination port, and protocol. This technique assures that unlabeled assaults do not affect the ground truth. The final need is encryption. This is one of the most pressing requirements, since we know thatunknown hostile traffic employs a variety of attack vectors in its attacks.

The process is the second requirement. Its goalis to ensure that researchers can build their datasets according to the criteria. It should be feasible to learn how to develop synthetic attacks as well as network configurations. Scripts were provided for collecting and producing synthetic attacks from the attack profile. There are technology that can mimic human interaction, such as browsing and clicking random links.These two profiles, the attack profile and the benign profile, are crucial for providing new data if researchers want to add more attack routes and update the trafficwith their own needs. The script for the ground-truth data labelling technique is also given. Researchers can generate fresh datasets based on their network configuration by following the steps in a controlled setting. In terms of Accuracy, Balanced Accuracy, Precision, Recall, and F1, we employed four machine learning approaches to examine the performanceof the HIKARI-2021 dataset.

Reference

L. Chen, Y. Ye, and T. Bourlai, “Adversarial machine learning in malware detection: Arms race betweenevasion attack and defense,” in EISIC’2017, 2017, pp. 99–106.

Jonker, M.; King, A.; Krupp, J.; Rossow, C.; Sperotto, A.; Dainotti, A. Millions of targets

under attack: A macroscopic characterization of the DoS ecosystem. InProceedings of the 2017 Internet Measurement Conference, London, UK, 1–3 November 2017; pp. 100–113.

K. Sato, K. Ishibashi, T. Toyono, and N. Miyake. Extending black domain name list by using co-occurrence relation between dns queries. In Third USENIX LEET Workshop, 2010.

T. Holz, C. Gorecki, K.Rieck, and F. Freiling. Measuring and detecting fast- flux service networks. In Proceedings of NDSS, 2008.

T. Holz, C. Gorecki, K.Rieck, and F. Freiling. Measuring and detecting fast- flux service networks. In Proceedings of NDSS, 2008.

Al-Dujaili, A. Huang, E. Hemberg, and U.-M. O'Reilly, "Adversarial deep learning for robust detection of binary encoded malware," in 2018 IEEE Security and Privacy Workshops (SPW). IEEE, 2018, pp. 76–82.

Barbosa, R.R.R.; Sadre, R.; Pras, A.; van de Meent, R. Simpleweb/University of Twente Traffic Traces Data Repository; Centre for Telematics and Information Technology, University of Twente: Enschede, The Netherlands, 2010.

S. Hou, Y. Ye, Y. Song, and M. Abdulhayoglu, "Make evasion harder: An intelligent android malware detection system," in Proceedings of the Twenty- Seventh IJCAI, 2018, pp. 5279–5283.

P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," Journal of machine learning research, vol. 11, no. Dec, pp. 3371–3408, 2010.

T. Hothorn and B. Lausen. Doublebagging: Combining classifiers by bootstrap aggregation. Pattern Recognition, 36(6):1303–1309, 20

Ding, Y.; Zhai, Y. Intrusion detection system for NSL-KDD dataset using convolutional neural networks. In Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence, Shenzhen, China, 8–10 December 2018; pp. 81–85. [CrossRef]

Nasr, M.; Bahramali, A.; Houmansadr, A. Deepcorr: Strong flow correlation attacks on tor using deep learning. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1962– 1976. [CrossRef]

Demontis, M. Melis, B. Biggio, D. Maiorca, D. Arp, K. Rieck, I. Corona, G. Giacinto, and F. Roli, "Yes, machine learning can be more secure! a case study on android malware detection," IEEE Transactions on Dependable and Secure Computing, vol. 16, no. 4, pp. 711–724, July 2019

Z. Qian, Z. Mao, Y. Xie and F. Yu. On networklevel clusters for spam detection. In Proceedings of the USENIX NDSS Symposium, 2010.