UNIVERSITY OF
GLOUCESTERSHIRE

This is a peer-reviewed, post-print (final draft post-refereeing) version of the following published document, ©2020 IEEE. and is licensed under All Rights Reserved license:

**Robic–Butez, Pierrick and Win, Thu Yein ORCID logoORCID: https://orcid.org/0000-0002-4977-0511 (2020) Detection of Phishing Websites using Generative Adversarial Network. In: 2019 IEEE International Conference on Big Data (Big Data), 9th-12th December 2019, Los Angeles. ISBN 9781728108575**

Official URL: https://doi.org/10.1109/BigData47090.2019.9006352

EPrint URI: https://eprints.glos.ac.uk/id/eprint/8528

**Disclaimer**

The University of Gloucestershire has obtained warranties from all depositors as to their title in the material deposited and as to their right to deposit such material.

The University of Gloucestershire makes no representation or warranties of commercial utility, title, or fitness for a particular purpose or any other warranty, express or implied in respect of any material deposited.

The University of Gloucestershire makes no representation that the use of the materials will not infringe any patent, copyright, trademark or other property or proprietary rights.

The University of Gloucestershire accepts no liability for any infringement of intellectual property rights in any material deposited but will remove such material from public view pending investigation in the event of an allegation of any such infringement.

PLEASE SCROLL DOWN FOR TEXT.

# Detection of Phishing websites using Generative Adversarial Network

Pierrick Robic–Butez, Thu Yein Win.

**Abstract:** Phishing is typically deployed as an attack vector in the initial stages of a hacking endeavour. Due to it low-risk rightreward nature it has seen a widespread adoption, and detecting it has become a challenge in recent times. This paper proposes a novel means of detecting phishing websites using a Generative Adversarial Network. Taking into account the internal structure and external metadata of a website, the proposed approach uses a generator network which generates both legitimate as well as synthetic phishing features to train a discriminator network. The latter then determines if the features are either normal or phishing websites, before improving its detection accuracy based on the classification error. The proposed approach is evaluated using two different phishing datasets and is found to achieve a detection accuracy of up to 94%.

**Index Terms:** Security analytics, phishing detection, generative adversarial networks, cloud security, big data analytics

## I. Introduction

Phishing exploits the natural curiosity as well as the emotional states of human users and is used during the initial stages of a hacking endeavour. This involves sending a crafted email to unassuming users who are tricked into clicking onto the crafted web link which is similar to a legitimate looking website service. The consequences of a successful phishing attack ranges from identity and credit card thief to the compromise of the entire organisational network through a drop-by malware attack. Due to its low-risk high-reward, it has been used extensively as an attack vector and detecting it has been a challenge due to its dynamic nature.

Detection of phishing attacks typically features the uses of a database which stores all the known phishing URLs (Uniform Resource Locators) and IP addresses, before using it to look up any incoming web link to determine if it is a phishing website. While the approach is adequate against well-known phishing websites, it is limited in detecting phishing sites which do not exist in the database. In addition it does not take into account both the internal structure as well as external metadata of a website in detecting phishing websites.

In this paper, we present a novel solution to detect phishing websites using Generative Adversarial Networks [1]. Taking both the internal structure and external metadata of a website, it consists of a generator network and a discriminator network. The former generates both synthetic and legitimate phishing features, while the latter detects them and improves its detection accuracy by iteratively minimising its classification error rate.

The remainder of the paper is arranged as follows. Section II presents our GAN based phishing detection solution. Section III presents a detailed implementation of the solution, with the test evaluations discussed in Section IV. Section V discusses the related works, before discussing the future work and conclusion in Section VI.

## II. Phishing Website Detection using Generative Adversarial Networks

### A. System architecture

The aim of the proposed approach is to detect in realtime potential phishing attempts through traversing the web application history of the guest VMs and analysing the features of the recorded web links. To that end, the following design principles are adopted in its implementation:

- $r1$ - Holistic: The proposed phishing detection approach should be able to detect potential phishing attempts by taking into account the different feature characteristics of potential phishing links.
- $r2$ - Real-time: The proposed approach should be to detect phishing web links in real-time.
- $r3$ - Adaptability: The proposed approach should be able to adapt to changes in phishing feature characteristics and be adaptable in detecting them.

The proposed approach first extracts all web links by traversing the web history logs of the guest virtual machines (VMs) in real-time. It then extracts both the internal structural as well as the semantic characteristics of the web links, before combining them

with their external metadata in order to generate feature vectors (*r*1). The feature vectors are then input into a GAN-based detection framework to train a model which detects potential phishing websites.

In order to adapt to the dynamic nature of phishing attacks and their features, the proposed approach features the use of a model which is trained using a GAN network. It consists of a generator network which creates synthetic phishing features for training the discriminator together with real phishing features, and a discriminator network which determines if the features are indicative of a legitimate phishing attack.

The model is then trained based on the feedback obtained from the discriminator network, and this allows it to learn the different variations in the phishing features and be adaptable in detecting them.

The proposed approach features the use of a GAN network consisting of a generator network and a discriminator network. Given a phishing website dataset the generator network generates both legitimate as well as phishing data and provides them as input to the discriminator network. The latter then determines whether it is indicative of either a synthetic or a legitimate phishing feature, before determining if it is either a phishing or a legitimate website. The detection error obtained is then used to further improve the accuracy of both the generator and the discriminator networks. This form of model training allows for variations in different feature characteristics, and enable to final model to be adaptable in detecting future phishing websites (*r*3).

The proposed approach consists of two phases which are namely:

- Feature extraction
- Model training

### Features extraction

Prior to training the GAN-based phishing detection model, the features which are present in both phishing and legitimate websites are first extracted and encoded. Given a web *URL* (Uniform Resource Locator), these features can be classified into three categories namely:

- Structural features, which are extracted by analyzing the structure of the URL and which are listed in Table I.
- Features based on source code, which are extracted by analyzing the website itself and which are listed in Table II.
- Metadata from third party sources, which are extracted by analyzing information given by online services.

All the features together with their descriptions are provided in Tables I, II and III.

For each URL a list of 46 features is extracted and encoded in a form of a feature vector based on the encoding technique proposed in Rami M. Mohammad *et.al* [8] and Schppen *et.al* [9]. This involves encoding each feature as either fixed values (0, 1 and for some features 0.5) or real values between 0 and 1 based on pre-defined thresholds.

These thresholds have been chosen by calculating the means of each feature for both phishing and clean websites and then finding the best value which maximizes the separation between the two groups. This results in obtaining 22 features with fixed values and 24 with ranged values. Once obtained, the values of the features are normalised between 0 and 1 before being used to train the GAN model.

## III. Implementation

### A. GAN architecture

The GAN is implemented using *Keras* [10]. The size of the input data is 46 which is the number of features that are extracted from a URL and the output size is 1. Figure 1 provides a graphical description of the GAN architecture for the proposed approach.

The generator takes as input a feature vector from the training set and introduces noise to it based on a normal distribution, and consists of eleven layers. The discriminator, on the other hand, determines if the features are either legitimate or synthetic and consists of eight layers. Both use the Adam optimization algorithm to update weights, with the same learning rate which can be chosen at the beginning of the training.

The implementation of the proposed approach consists of two phases, namely

- Training the GAN
- Predicting using GAN

TABLE I
STRUCTURE FEATURES

| Feature | Description | Thresholds for discrete type |
|---|---|---|
| IP address | Ip address in the hostname | |
| Hostname length | Length of the hostname | Safe: $<15$<br>Unsafe: $\geq 19$ |
| Use a shortening service | Shortening service used | |
| @ symbol | @ symbol in the URL | |
| Double slash | '//' in the URL | |
| Dash symbol | Number of dash in the URL | Safe: $=0$ |
| Subdomains count | Number of subdomains | Safe: $\leq 1$<br>Unsafe: $>2$ |
| Http token | Is 'http' in the URL except at the begining | |
| Mean subdomains length | Average of the length of subdomains | Safe: $\leq 9$<br>Unsafe: $>15$ |
| www token | 'www' at the beginning of the URL | |
| Valid TLD | The URL have a valid TLD* | |
| Single character subdomain | A subdomain is composed by only one character? | |
| Exclusive Prefix Repetition | The URL is composed by a repetition of a characters sequence? | |
| TLD as subdomain | Is there a subdomain that is a valid TLD? | |
| Ratio of exclusive digit subdomains | Ratio of subdomains composed only by digit | Safe: $=0$ |
| Ratio of exclusive hexadecimal subdomain | Ratio of subdomains composed only by hexadecimal | Safe: $=0$ |
| Ratio of underscore | Ratio of underscore in the hostname | Safe: $=0$ |
| Hostname contains digit | Does the hostname contain digit? | |
| Vowel ratio | Ratio of vowel in the hostname | Safe: $<0.27$ |
| Digit ratio | Ratio of digit in the hostname | Safe: $=0$ |
| Alphabet cardinality | Number of alpha-characters in the hostname | Safe: $\leq 11$<br>Unsafe: $>14$ |
| Ratio of repeated characters | Ratio of the characters that are repeated in the hostname | Safe: $\leq 0.17$ |
| Ratio of consecutive consonant | Ratio of consonant that precedes or succeeds another consonant | Safe: $\leq 0.05$ |
| Ratio of consecutive digit | Ratio of digit that precedes or succeeds another digit | Safe: $\leq 0.01$ |

* Using the Mozilla TLD list [2].

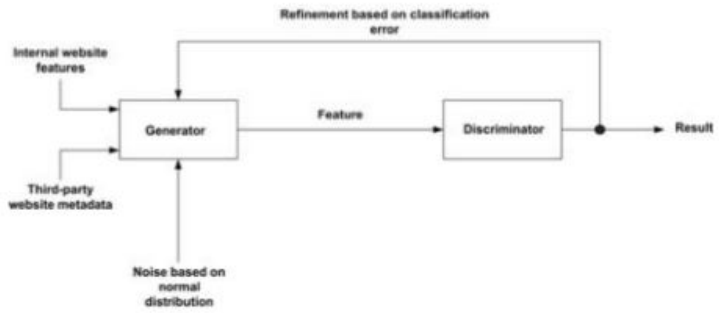


Fig. 1. System architecture of the proposed approach

TABLE II
FEATURES BASED ON WEBSITE SOURCE CODE

| Feature | Description | Thresholds for discrete type |
|---|---|---|
| Favicon link | Favicon is a local link | |
| Ports opened | Are there ports in abnormal state * | |
| Links of resources requested | Ratio of image/sound/video links that are external | Safe: $<0.22$<br>Unsafe: $\geq 0.61$ |
| Anchor links | Ration of anchor links that are external | Safe: $<0.31$<br>Unsafe: $\geq 0.67$ |
| Tag links | Ratio of tag links that are external | Safe: $<0.05$<br>Unsafe: $\geq 0.81$ |
| Server Form Handler | Is there form that point nowhere? | |
| Email submitting | Is there any information submitted by email | |
| Redirect | Number of redirections | Safe: $\leq 1$<br>Unsafe: $>4$ |
| Custom toolbar | Is the browser toolbar abnormally modified? | |
| Disable right-click | Is the right-click disabled? | |
| Popup Count | Number of popup | Safe: $\leq 1$<br>Unsafe: $>3$ |
| iFrame links | Is there iFrame external link? | |

* Ports tested : 21, 22, 23, 80, 443, 445, 1433, 1521, 3306, 3389.

TABLE III
FEATURES FROM THIRD PARTY SOURCES

| Feature | Description | Thresholds for discrete type |
|---|---|---|
| Age of TLS/SSL certificate | Validity duration of the TLS/SSL certificate | Safe: $\geq 365$d |
| Time until the domain expires | Time until the end of the domain registration ‡ | Safe: $>330$d |
| Abnormal URL | Does the domain registrant contain the hostname? | |
| Age of domain | Time since the first registration of the domain ‡ | Safe: $>1095$d<br>Unsafe: $<365$d |
| DNS record | Website known by a DNS | |
| Amazon traffic rank | Rank of the domain given by Amazon AWIS * | Safe: $<100000$ |
| PageRank | PageRank of the domain † | Safe: $\geq 4$<br>Unsafe: $\leq 2$ |
| Google indexation | Domain indexed by Google | |
| Links pointing to the page | How many links are pointing to the domain? * | Safe: $\geq 30$<br>Unsafe: $>5$ |
| Know as phishing | Is the website already known as a phishing website? § | |

* Extracted from the Amazon AWIS database [3].
† Extracted from OpenPageRank [4].
‡ Extracted from WHOIS record [5].
§ Extracted from services like StopBadWare [6] (PhishTank [7] can also be used, but it was not used during the study due to the fact all phishing websites are from this database).

### B. Training the GAN

After choosing a learning rate for the Adam optimization algorithm, a batch size (corresponding to how many sets of 46 features are provided to the GAN at each epoch) and a dataset (containing either phishing webistes or clean websites), the training set is split again into two parts. The first part for training. The *training set* consists of 90% of the original training set. The second portion which is the validation set however contains the remaining 10%.

Then, for each epoch, a sequence of actions is made: . A randomly set of data is extracted from the training set. The size of this set is the batch size chosen at the beginning of the simulation. The same thing is done with the validation set.

- The discriminator is trained. First with the real data extracted from the training set and telling it these data are valid, then with the data generated by the generator from noise, telling it these data are fake.
- The generator is trained with random data.
- A step of validation is done for the discriminator and the generator as for the training step with the validation dataset. . The discriminator is tested with the test dataset and a report is generated containing statistics.

*C. Predicting using GAN*

After the training phase, the discriminator of the GAN is used to make predictions on websites by giving to the discriminator the features extracted from websites it has never seen before. But to maximize the efficiency of the predictions, it is necessary to find the best way to extract features. Indeed, as explained previously in section II-A, some features can be used with fixed or ranged value. So the graphs generation described previously has been used to determine the most efficient shape for each of these features.

With this in mind, a number of graphs were created and simulations are created. The proposed GAN model is trained for a batch size ranging between 30 and 125 with a step of 5. For each batch size, the GAN is trained at a learning rate ranging between 0.004 and 0.013 with a step of 0.0005. All experiments are conducted over 1000 epochs. This involves 500 simulations for each generator as well as discriminator, with each simulation taking approximately 15 minutes on a CPU core.

After all these simulations, the results were compared to determine, for each feature, which way gives the best results.

## IV. Test Evaluation

### A. Experimental setup

The tests during the study are conducted on two different types of machines. The first one is a Microsoft Azure [11] virtual machine type of F4S, with 8GB of RAM and 4 virtual CPU cores on Intel Xeon E5-2673 v3 and was used for all work about features extraction and analysis. The second one is a set of 25 workstations with the same configuration but with 16GB of RAM, an Intel core i7 6700 and a Nvidia GTX
1050Ti.

Furthermore, in order to optimize the time required to analyze websites more than one time, all information (HTML content, whois record, pageRank, SSL/TLS certificate) about each URL of each dataset is stored in a database[1] to avoid having to request these information each time.

*B. Datasets used*

By following the method previously explained, 3 datasets are used:

- A dataset containing features extracted from the Top 25000 most visited websites [12].
- A dataset containing features extracted from the Phishtank database [7].
- A dataset which groups the first and features extracted from the web browser history of a basic workstation. After grouping, the dataset had been shuffled.

Each dataset is split into two parts, namely a training set and a test set. The former consists of 90% of the dataset while the latter consists of the remaining 10% which is used to test the accuracy of the trained GAN model.

### C. Features extraction

To maximize the efficiency of the GAN, the first step involves optimizing the feature extraction. As described in the part II-A, 24 features can be extracted in two ways: as ranged value or as fixed value. To determine the most efficient means of feature extraction, a number of datasets are evaluated with the GAN training configuration as shown in Table IV. All the datasets used in this step are from the Amazon top 25000 database and web browser history of a basic workstation, only the way to extract features is different.

[1] https://github.com/Khuzdz/PhishGan/blob/master/DB/How to _get database.txt

TABLE IV
TEST CONFIGURATION

| Parameter | Minimum | Maximum | Step |
|---|---|---|---|
| Batch size | 30 | 130 | 5 |
| Learning rate | 0.001 | 0.0130 | 0.0005 |

First, a dataset with all features used with fixed values was tested and the best accuracy of phishing detection is saved. For each of the 24 features, a dataset with only one feature used with ranged value was tested and, as previously, the best accuracy has been saved.To determine which extraction method is the most efficient, each ranged value feature is evaluated against all other fixed feature values in the dataset with the ranged feature contributing to higher accuracies selected for use in the dataset. This results in the use of eight ranged features for the simulations.

The graphs of the accuracies and losses during the training and validation steps are shown on Figures 2 and 3. Indeed it can be seen on the graphs that the best accuracy is obtained at 23 epochs, with a learning rate of 0.004 and a batch size of 100. It can also be seen on graphs that, at the epoch number

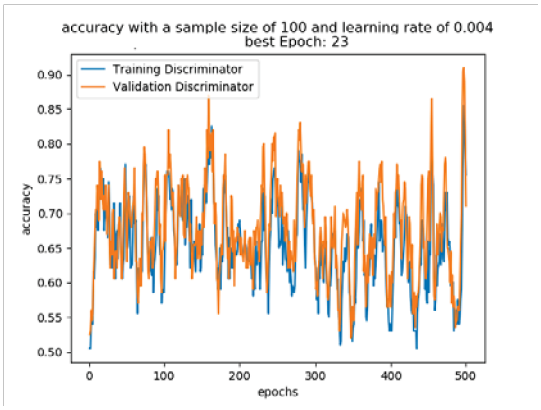


Fig. 2. Accuracies for the best results with GAN trained with Phishing data during training and validation

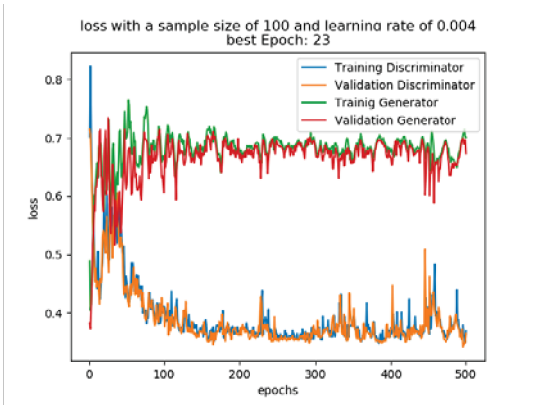


Fig. 3. Losses for the best results with GAN trained with Phishing data during training and validation

23, there are peaks for the accuracies of the discriminator and of the generator, and the losses of the discriminator, and there are troughs for the losses of the generator.

Then, the clean dataset is extracted from the top 25000 of the most visited websites and the web browser history of a standard workstation. This dataset contains 24084 websites (before any split). In addition the phishing dataset which is extracted from the PhishTank database [7] is used which contains 11267 websites (before any split).

The graphs of the accuracies and losses of the training and validation steps are shown on Figures 4 and 5. Moreover the results show that the best accuracy is obtained at 758 epochs, with a learning rate of 0.0045 and a batch size of 105. As for the results with the other dataset, it can be seen that at the epoch number 758 peaks for the accuracies of the discriminator and of the generator, and the losses of the discriminator, and there are troughs for the losses of the generator.

All the results obtained with different datasets are shown in the confusion matrices, in Tables V and IV-C.

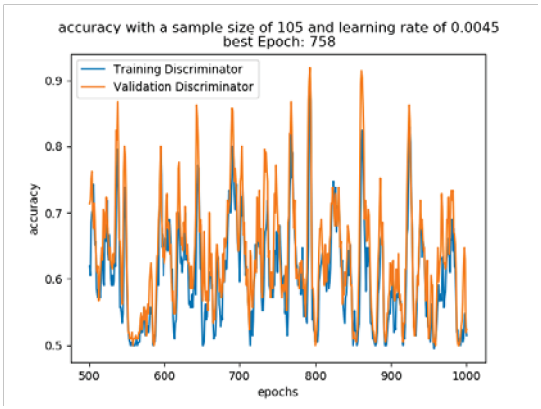


Fig. 4. Accuracies for the best results with GAN trained with Clean data during training and validation

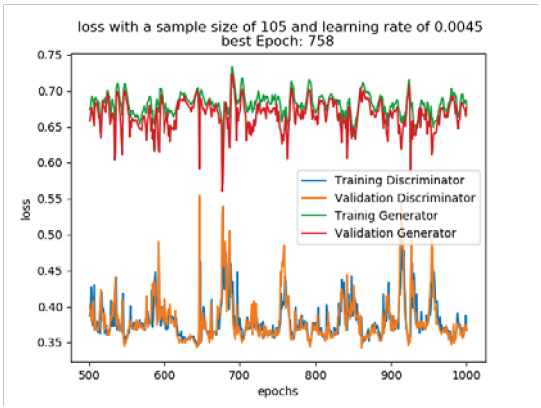


Fig. 5. Losses for the best results with GAN trained with Clean data during training and validation

TABLE V
CONFUSION MATRIX FOR GAN TRAINED WITH CLEAN DATA

| Real class \ Predicted class | Phishing | Clean |
|---|---|---|
| Phishing | 1106 | 107 |
| Clean | 110 | 2297 |

TABLE VI
CONFUSION MATRIX FOR GAN TRAINED WITH PHISHING DATA

| Real class \ Predicted class | Phishing | Clean |
|---|---|---|
| Phishing | 1150 | 63 |
| Clean | 228 | 2179 |

## D. GAN training

After extracting features from a lot of phishing and clean websites and storing it, the GAN could be trained in two ways: either giving it clean data and trained it to detect clean websites and, when data from a phishing website are given to it, it will detect that are not clean data, or doing exactly the opposite by giving it phishing data.

As explained in section III-B, to produce the best results, the classification threshold is calculated again after the training of these GAN, using the training set. To that end, the accuracy of the GAN is calculated by varying the threshold value between the average of the probabilities obtained with clean data and the average of the probabilities obtained with phishing data, with a step of 0.0001. The thresholds obtained are shown in the Table VII.

TABLE VII
CLASSIFICATION THRESHOLDS

| Dataset | Value |
|---|---|
| Phishtank | 0.7914 |
| Amazon top 25000 + History | 0.6953 |

*Amazon top 25000 dataset:* This dataset is extracted from the top 25000 of the most visited websites [12] and contains 17761 websites (before any split).

The results obtained are positive overall as it was possible to reach a global accuracy of 97.12% with a learning rate of 0.009 and a batch size of 16, without having to search for the best way to extract features. Moreover, these results are obtained with only 30 features, and not with 46 (the decision to add new features was taken after).

But after further investigation, it was discovered that this dataset doesn't accurately represent all clean website. Indeed, if other clean data extracted from web browser history of a basic workstation were given to the GAN after training, the global accuracy has dropped to around 80%.

This difference can be explained by the fact that the URLs contained in the top 25000 are the homepage of websites and not specific web-pages, with a longer URL that can contain GET requests. So, most of the output of features are be different.

The use of this dataset alone therefore was stopped.

*E. Performance evaluation*

The accuracy and precision analysis of the proposed GAN approach is evaluated using a confusion matrix consisting of four metrics, which are namely,

- True positives (*TP*) that are the phishing websites detected as phishing.
- False positives (*FP*) that are the clean websites detected as phishing.
- True Negatives (*TN*) that are the clean websites detected as clean.
- False Negatives (*FN*) that are the phishing websites detected as clean.

The evaluation results of the proposed approach are shown in Table VIII.

First of all, in terms of accuracy and precision, the GAN trained with the clean dataset is significantly more efficient than the GAN trained with the phishing dataset. In the opposite, the recall of the second GAN is quite greater than the first one.

TABLE VIII
TABLE OF THE RESULTS OF PREDICTION

| Statistic \ Dataset | PhishTank | Amazon top 25000 + History |
|---|---|---|
| Accuracy | 91.96% | 94.00% |
| Precision | 83.45% | 90.95% |
| Recall | 94.80% | 91.17% |
| F1-score | 88.76% | 91.06% |
| False positive rate | 9.47% | 4.57% |
| False negative rate | 5.19% | 8.82% |

After a detailed analysis, it appears that the second GAN detects a greater proportion of the phishing websites, and also generates a lower amount of false positives.

Based on the results obtained, it can be seen that the second GAN model is able to detect phishing websites at a greater accuracy than the first model. This is due to the use as training set a dataset consisting of a relatively balanced distribution of phishing and clean features. While its false negative rate is a bit higher than that of the first model, its lower false positive rate makes it suitable to be deployed in real-world environments for phishing website detection.

## V. Related Works

The use of GAN to generate additional features is used to generate synthetic network traffic flows which are used in NIDS (Network-based Intrusion Detection Systems) in [13]. Using the *CIDDS-001* [14] dataset, it first uses three different approaches (namely vectorisation, scaling, and embeddings) to represent the original network flow features. Each of the encoded features is then passed into a GAN network to generate three separate datasets to be used in NIDS. While it is found that the use of embedding allows for the learning of interrelationships between the features, it is limited in generating previously unseen features which in turn affects the accuracy of the trained model. The proposed approach overcomes this problem by first training the model using features which are represented in terms of both categorical and continuous values, as well as incorporating new phishing features as part of its training model during it execution which results in obtaining an accuracy of up to 94%.

A similar approach is also used in the implementation of *Bot-GAN*, which is designed to improve network flow-based detection of botnets. It uses a LSTM network as the generator to generate synthetic network flows, and a 4-layer neural network is used as a discriminator to determine if the flows are either synthetic or legitimate attack features. Evaluated using the ISCX dataset [15], it achieved an overall precision of 74.04%. While *Bot-GAN* features the use of two different deep learning networks for the generator and the discriminator components of a GAN, the overall precision is relatively low due to its assumption that the features are time-dependent which is not the case. The phishing website detection solution, however, takes a holistic account of the different features in training the GAN, resulting in a higher precision value.

## VI. Conclusion

The framework proposed in this paper shows encouraging results and, due to that two GANs have been trained, this framework can be used in different situations. The results are quite relevant, with an accuracy between 93% and 94%. To obtain these results, number of simulations were made to determine the best parameters for the GAN and for the features extraction.

The future work of the proposed approach will involve improving the results by classifying the website into three classes rather than two classes. In this way, by adding a class *suspicious*, the false positive rate could decrease while allowing an effective detection. Moreover, to further increase the efficiency of the phishing detection, the framework could be used combined with other detection mechanisms. In this way, most of the phishing websites would be detected.

Finally, this framework could be improved by modifying the structure of the neural networks of the GAN. Indeed, no research has been done on this point and that could be a way to increase again the efficiency of the framework. All of these will be addressed in the future development of the proposed approach.

## References


[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[2] M. Foundation, "Public Suffix List," https://publicsuffix.org/list/, last accessed on 26/09/19.

[3] Amazon, "Alexa Web Information Service," https://www.alexa.com/ siteinfo/, last accessed on 26/09/19.

[4] domcop, "Open Page Rank," https://www.domcop.com/openpagerank/, last accessed on 26/09/19.

[5] "Whois tool," https://www.whois.com/whois/, last accessed on 26/09/19.

[6] "StopBadWare," https://www.stopbadware.org/top-50, last accessed on 26/09/19.

[7] OpenDNS, "PhishTank," http://www.phishtank.com, last accessed on 26/09/19.

[8] R. M. Mohammad, F. Thabtah, and L. McCluskey, "An assessment of features related to phishing websites using an automated technique," in *2012 International Conference for Internet Technology and Secured Transactions*, Dec 2012, pp. 492–497.

[9] S. Schuppen, D. Teubert, P. Herrmann, and U. Meyer, "FANCI : Feature-¨ based automated nxdomain classification and intelligence," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, 2018, pp. 1165–1181. [Online]. Available: https: //www.usenix.org/conference/usenixsecurity18/presentation/schuppen [10] "Keras," https://keras.io/, last accessed on 26/09/19.

[11] Microsoft, "Azure," https://azure.microsoft.com/, last accessed on 26/09/19.

[12] Amazon, "Top 1Million," http://s3-us-west-1.amazonaws.com/ umbrella-static/top-1m.csv.zip, last accessed on 26/09/19.

[13] M. Ring, D. Schlor, D. Landes, and A. Hotho, "Flow-based network¨ traffic generation using generative adversarial networks," *Computers & Security*, vol. 82, pp. 156–172, 2019.

[14] M. Ring, S. Wunderlich, D. Grdl, D. Landes, and A. Hotho, "Flow-based benchmark data sets for intrusion detection," in *Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS)*. ACPI, 2017, pp. 361–369.

[15] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *2014 IEEE Conference on Communications and Network Security*. IEEE, 2014, pp. 247–255.