DeepPhish: Simulating Malicious AI

Alejandro Correa Bahnsen, Ivan Torroledo, Luis David Camacho and Sergio Villegas
Cyber Threat Analytics, Cyxtera Technologies
Email: {alejandro.correa, ivan.torroledo, luis.camacho, sergio.villegas}@cyxtera.com

***Abstract*—In this work we describe how threat actors may use AI algorithms to bypass AI phishing detection systems. We analyzed more than a million phishing URLs to understand the different strategies that threat actors use to create phishing URLs. Assuming the role of an attacker, we simulate how different threat actors may leverage Deep Neural Networks to enhance their effectiveness rate. Using Long Short-Term Memory Networks, we created DeepPhish, an algorithm that learns to create better phishing attacks. By training the DeepPhish algorithm for two different threat actors, they were able to increase their effectiveness from 0.69% to 20.9%, and 4.91% to 36.28%, respectively.**

***Keywords*—Malicious AI; phishing detection; cybercrime; recurrent neural networks; long-short term memory networks; deep adversarial learning.**

I. Introduction

Machine Learning (ML) and Artificial Intelligence (AI) have become essential to any effective cybersecurity and defense strategy against unknown fraud attacks [1]. AI enhanced detection systems have improved detection compared to traditional manual classification, reaching a 98% detection rate in some cases. However, there is little work on the weaponization of Machine Learning as a threat actor tool.

Phishing URL generation is traditionally a manual process, but during the past few years this process has become automated using randomly generated URLs [2]. The defensive technologies for phishing detection that include ML show significant improvement on detection, reducing the effectiveness and success rates of the attacks [3]. Threat actors are constantly seeking new ways to bypass detections systems. As threat actors improve their attacks, is AI the new technology they will use?

In this work, we explore a database of more than one million phishing URLs collected from Phishtank[1]. We identify different 3 actual threat actors by following URLs with similar patterns and hosted on the same compromised domains, then cluster them to better understand the strategies used by these actors. Using an existent AI phishing detection algorithm [1], we measure the threat actors' effectiveness rate as the percentage of URLs that bypass the detection system. We found that Threat Actors 1 & 2, outperform the other attackers measured by their effectiveness.

Using the effective URLs from Threat Actors 1 & 2, we create the DeepPhish algorithm using a Long Short-Term Memory Network [4] that learns the intrinsic patterns that allow those URLs to bypass the AI phishing detection algorithm. DeepPhish is used to generate new synthetic phishing URLs with the objective of maximizing the effectiveness of the attacks. The results show that Threat Actors 1 and 2 were able to increase their effectiveness from 0.69% to 20.9%, and from 4.91% to 36.28%, respectively.

The remainder of this paper is organized as follows: In Section II, we provide a background on phishing detection and AI phishing detection systems. In Section III, we show our analysis of known phishing attacks and describe the strategies used by different threat actors. Subsequently, in Section IV, we will describe our DeepPhish algorithm. Section V presents the experimental results. Finally, we will provide conclusions in Section VI and then highlight the future work in Section VII.

II. Related Work

In this section we first give a background on phishing detection systems. Then we explain the AI phishing detection system presented in the research paper 'Classifying Phishing URLs Using Recurrent Neural Networks' [1]. Lastly, we highlight the known cases of the malicious use of ML.

A. Phishing Detection

Phishing URL detection can be done via proactive or reactive means. On the reactive end, we find services such as Google Safe Browsing API[2]. These types of services expose a blacklist of malicious URLs to be queried. The blacklists are constructed using different techniques, including manual reporting, honeypots, or by crawling the web in search of known phishing characteristics [5], [6]. For example, browsers make use of blacklists to block access upon reaching the URLs contained within them. One drawback of such a reactive method is that in order for a phishing URL to be blocked, it must be previously included in the blacklist. This implies that web users remain at risk until the URL is submitted and the blacklist is updated. What is more, since the majority of phishing sites are active for less than a day [6], [7], their mission is complete by the time they are discovered and added to the blacklist.

Proactive methods mitigate this problem by analyzing the characteristics of a webpage in real time in order to assess the potential risk of a webpage. Risk assessment is done through a classification model [8]. Some of the Machine Learning methods that have been used to detect phishing include: support vector machines [9], streaming analytics [10], gradient boosting [11], [12], random forests [13], latent Dirichlet allocation [14], online incremental learning [15], and neural networks [16]. Several of these methods employ an array of

[1] PhishTank (https://www.phishtank.com/)

[2] https://safebrowsing.google.com/

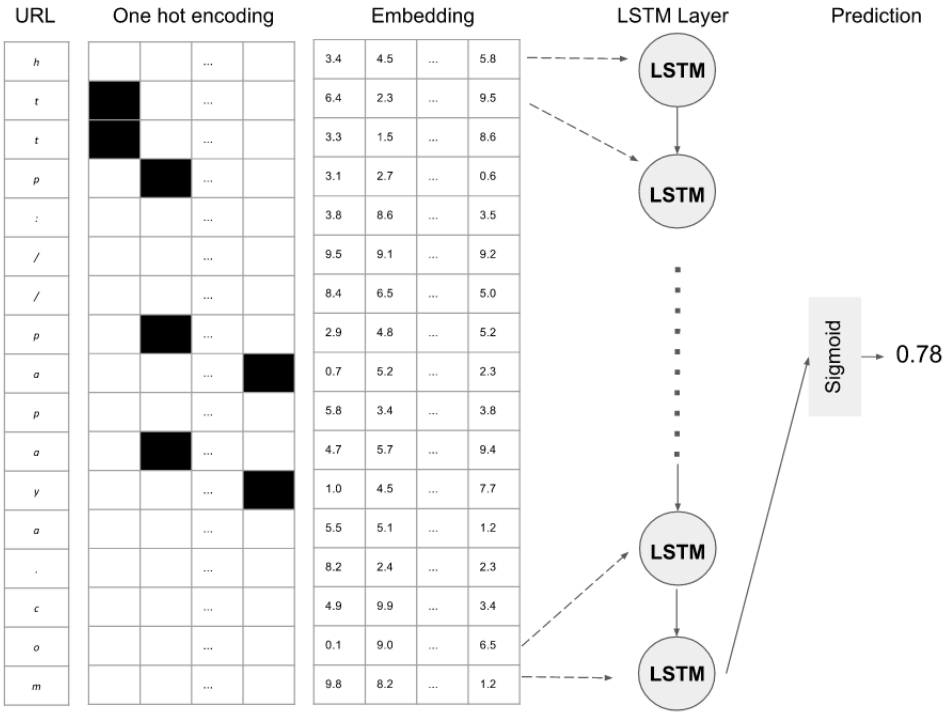


Fig. 1. Recurrent neural network for classifying phishing URLs based on LSTM units. Each input character is translated by an 128-dimension embedding. The translated URL is fed into a LSTM layer as a 150-step sequence. Finally, the classification is performed using an output sigmoid neuron.

website characteristics, which mean that in order to evaluate a site, they first have to be rendered before the algorithm can be used. This adds a significant amount of time to the evaluation process [17], [18]. Using URLs, instead of content analysis, reduces the evaluation time because only a limited portion of text is analyzed.

Lately, the application of machine learning techniques for URL classification has been gaining attention. Several studies proposing the use of classification algorithms to detect phishing URLs have come to the light in recent years [12], [13], [19]. These studies are mainly focused on creating features through expert knowledge and lexical analysis of the URL. Then, the phishing site's characteristics are used as quantitative input for the model. The model in turn learns to recognize patterns and associations that the input sequences must follow in order to label a site as a possible legitimate or malicious.

B. Phishing URL classification using Deep Recurrent Neural Networks

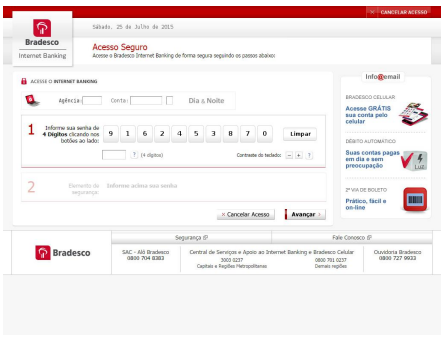
In a recent work, we proposed a method to detect phishing URLs using Deep Recurrent Neural Networks [1]. A Neural Network is a bio-inspired machine learning model that consists of a set of artificial neurons with connections between them. Recurrent Neural Networks (RNN) are a type of neural network that is able to model sequential patterns. The distinctive characteristic of RNNs is that they introduce the notion of time to the model, which in turn allows them to process sequential data one element at a time and learn their sequential dependencies [20].

One limitation of general RNNs is that they are unable to learn the correlation between elements more than 5 to 10 time steps apart [4]. A model that overcomes this problem is Long Short-Term Memory (LSTM). This model can bridge elements separated by more than 1,000 time steps without loss of short time lag capabilities [21].

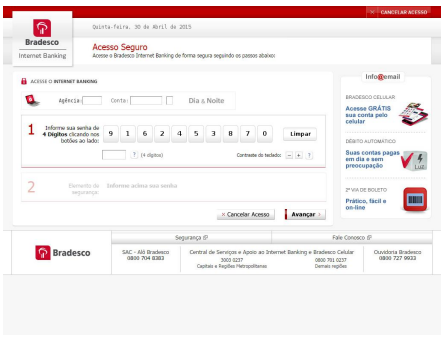
LSTM is an adaptation of RNN. Here, each neuron is replaced by a memory cell that, in addition to a conventional neuron representing an internal state, uses multiplicative units as gates to control the flow of information. A typical LSTM cell has an input gate that controls the input of information from the outside, a 'forget cell' that controls whether to keep or forget the information in the internal state, and an output gate that allows or prevents the internal state to be seen from the outside.

This approach is different as instead of manually extracting the features, we directly learn a representation from the URL's character sequence. As each character in the URL sequence exhibits correlations, that is, nearby characters in a URL are likely to be related to each other. These sequential patterns are important because they can be exploited to improve the performance of the predictors [22].

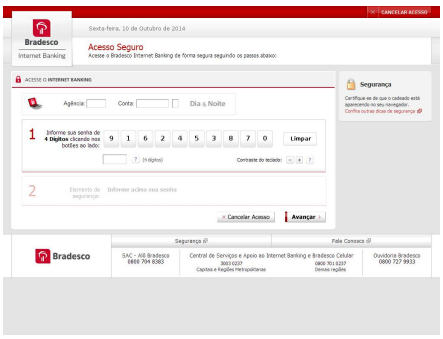
Using LSTM units, we built a model that receives as input a URL as character sequence and predicts whether or not the URL corresponds to a case of a possible phishing. The architecture is illustrated in Fig. 6. Each input character is translated by a 128-dimension embedding. The translated URL is fed into a LSTM layer as a 150-step sequence. Finally, the classification is performed using an output sigmoid neuron. The network is trained by back-propagation using a cross-entropy loss function and dropout in the last layer.



(a) http://www.naylorantiques.com/JavaScript/charset=iso-8859-1/httpequiv/marginbottom



(b) http://www.debbiebright.co.za/modules/mod_weblinks/tmpl/acessodeseguranca/H347HR3883H8.html



(c) http://waldronfamilygppractice.co.uk/xmlrpc/includes/ibk2/5D4FG98DF74FD65H.html

Fig. 2. Visual analysis of the phishing attacks made by Threat Actor 1. We can confirm that these attacks are targeting the same brand, therefore, it is safe to assume they are being made by the same threat actor.

This model showed to outperform traditional machine learning approaches such as the Random Forest (RF) algorithm. Using a database comprised of one million legitimate URLs from the Common Crawl database, and one million phishing URLs from Phishtank, both models showed great statistical results. On one hand the RF had an F_1-Score of 0.93 and an accuracy of 93.5%, while the LSTM had F_1-Score of 0.98 and an accuracy of 98.7%.

C. ML as a weapon

Defensive AI has been widely used in cybersecurity. Examples vary from malware detection [23], intrusion detection [24] to phishing detection [1]. However, there is little work regarding the use of ML as a malicious tool by threat actors [25]. Recent approaches to use Machine Learning as a weapon include: Honey-Phish [26], SNAP_R [27] and Deep DGA [28].

Honey-Phish: The Honey-Phish project uses Markov Chains for natural language processing to create spear phishing for actual phishers. The idea is to automate responses to phishing emails in order to establish an email communication with the attacker, these responses to scamming emails contain a link that traces the geographical location. Even if this project was not exactly successful, and had no intention of harmful application, it demonstrates is possible to harness AI to create targeted spear-phishing.

SNAP_R: SNAP_R project uses the basis of Honey-Phish for spear-phishing creation using not emails but Twitter as target communication channel. This approach finds profiles and personalizes Twitter phishing posts by scoring the target's probability of responding and clicking on generated phishing links. In this case they take advantage of shortened links in Twitter posts to conceal the URL.

Deep DGA: The Deep DGA approach uses Generative Adversarial Networks (GAN) to create artificial malware domains that are hard to detect, even by a deep learning based detector. Using multiple rounds of generator and detector, the Deep DGA algorithm showed that the generator increased the rate of undetected domains in each round, and that the detector improved its performance in detecting domains after each round.

TABLE I. MOST WIDELY USED DOMAINS BY THE THREAT ACTORS

Domain	No of Phishing URLs
toughbook.cl	8132
corpzim.com	2112
bancosemesas.com.br	1801
11qnt.info	1739
dniassociates.com	1686
securusair.com	1653
esy.es	1631
hyd.me	1491
netspbsstore.com	1459
bjcurio.com	1416
kzstudent.com	1399
roofmont-fm.cz	1373
central-process-payment.eu	1325
creeksideshowstable.com	1204
addr.com	1177
robweb.com.br	1136
amelaca.com	1100
californiaimport.de	1050

III. UNCOVERING THREAT ACTORS

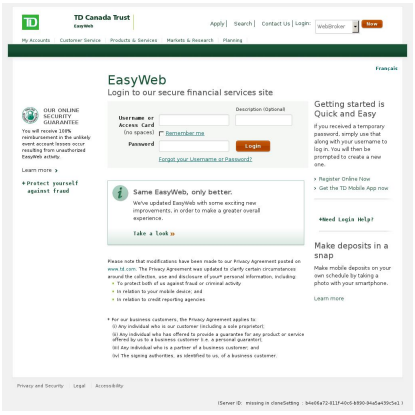
In attempting to uncover threat actors, we explored a database of over 1,146,441 phishing URLs collected during 2017 from Phishtank, a website used as a phishing attacks repository. In TABLE I, we show the most common non-hosting domains we have in our database.

In the remainder of this section, we show how we identified different threat actors by looking at their strategies to creating phishing URLs, the domains used, the patterns in the URL path, and screen shots of the phishing site.

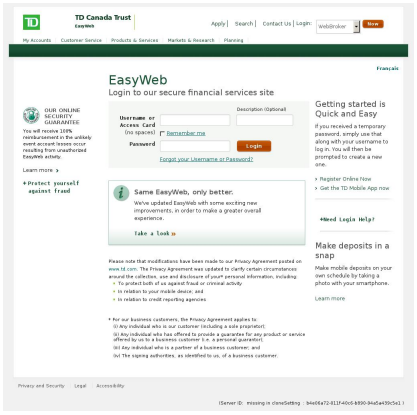
A. Threat Actor 1

During our exploration of phishing URLs we identified naylorantiques.com as a widely used compromised domain. Afterward, we analyzed the patterns used in this attack and extracted the most common words in the URLs paths. These are the top most common found words: *atendimento, jsf, identificacao, ponents, views, TV, mail, SHOW, COMPLETO, VILLA, MIX, ufi, pnref, story, tryy2ilr, Autentico*.

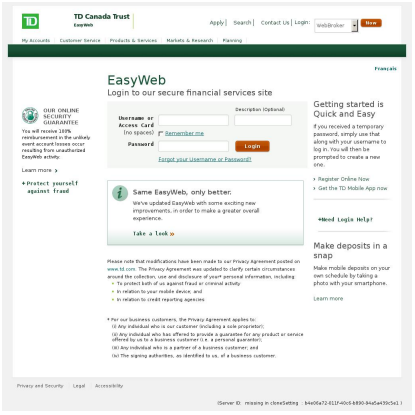
Using the keywords, we look for similar patterns in our entire database. We identified a total of 106 domains. Threat Actor 1 used a total of 1,007 attack URLs. A sample of these URLs are shown in TABLE II.



(a) http://www.vopus.org/es/images/cursos/thumbs/tdcanadatrust/index.html



(b) http://kramerelementary.org/media/tdcanadatrust/index.html



(c) http://www.backfire.se/components/com_media/easyweb.tdcanadatrust.com/tdcanadatrust/index.html

Fig. 3. Visual analysis of the phishing attacks made by Threat Actor 2. We can confirm that these attacks are targeting the same brand, therefore, it is safe to assume they are being made by the same threat actor.

TABLE II. SAMPLE OF URLS USED BY THREAT ACTOR 1

http://bbw.com.br/ibpflogin-identificacao.jsf/
http://excelavansat.ro/includes/js/calendar/lang/identificacao-jsf/index.php
http://www.netshelldemos.com/gamesite/upload/cic/51a3735bd1f8238ae08add4970ca70f6/lb.php?id=13698&default=019e92359d488ca8c1ebd5283a5a3d3b
http://www.debbiebright.co.za/modules/mod_weblinks/tmpl/acessodeseguranca/?1PKXZLKHCIQAKTSS3FB75FTJC12L64L7T64SRO047ZEQT8IV
http://naylorantiques.com/mail/GPC16FGT/
http://www.netshelldemos.com/gamesite/upload/cic/cddb8d6dc22d83cc9c6661068819aad4
http://waldronfamilygppractice.co.uk/xmlrpc/includes/ibk2/5D4FG98DF74FD65H.html
http://www.debbiebright.co.za/modules/mod_weblinks/tmpl/acessodeseguranca/H347HR3883H8.html
http://naylorantiques.com/j1s8f9gbzgqyll0x5t3y8jj2ksr2pgxwcuy/
http://www.naylorantiques.com/JavaScript/charset=iso-8859-1/http-equiv/margin-bottom

TABLE III. SAMPLE OF URLS USED BY THREAT ACTOR 2

http://www.friooptimo.com/images/tdcanadatrust/index.html
http://www.kalblue.com/language/overrides/tdcanadatrust/index.html
http://www.kalblue.com/language/en-GB/tdcanadatrust/index.html
http://www.vopus.org/es/images/cursos/thumbs/tdcanadatrust/index.html
http://www.vopus.org/ru/media/tdcanadatrust/index.html
http://vopus.org/descargas/otros/tdcanadatrust/index.html
http://www.vopus.org/descargas/otros/tdcanadatrust/index.html
http://kramerelementary.org/media/tdcanadatrust/index.html
http://kramerelementary.org/cli/tdcanadatrust/index.html
http://www.vopus.org/es/images/escargas/otros/tdcanadatrust/index.html
http://www.artwood.co.kr/gumia_bbs/data/notice/1159318175/tdcanadatrust/index.html

TABLE IV. SAMPLE OF URLS USED BY THREAT ACTOR 3

http://v057261.home.net.pl/live.html
http://www.m242fleetone.org/Emailsettings/index.html
http://www.m242fleetone.org/Emailsettings/
http://v032395.home.net.pl/en/xmlrpc2/Sincronismo
http://justpeaceint.org/Album%20Abbotabad
http://dskumara.asia.lk/www/login.php
http://web80.dnchosting.com/%7Eppersup1/.cgi
http://dskumara.asia.lk/www/
http://www.eru.com.pt/
http://2myideas.com/

Finally, as a sanity check, we manually compare a sample of the identified phishing sites' screenshots. These are shown in Fig. 2. We can confirm that these attacks are targeting the same brand, therefore, it is safe to assume they are being made by the same threat actor.

B. Threat Actor 2

Repeating the exploration exercise looking for a different attack pattern and actor, we found vopus.org as a new commonly used domain. From the new domain we extracted the pattern: *tdcanadatrustindex.html* widely used among URLs. Looking for this pattern in our entire database, we realized this attacker used 19 domains. Threat Actor 2 used a total of 102 attack URLs. A sample of these URLs are shown in TABLE III.

Once again, as a sanity check, we manually compared a sample of the identified phishing sites' screenshots. These are shown in Fig. 3 to confirm that these attacks are targeting the same brand.

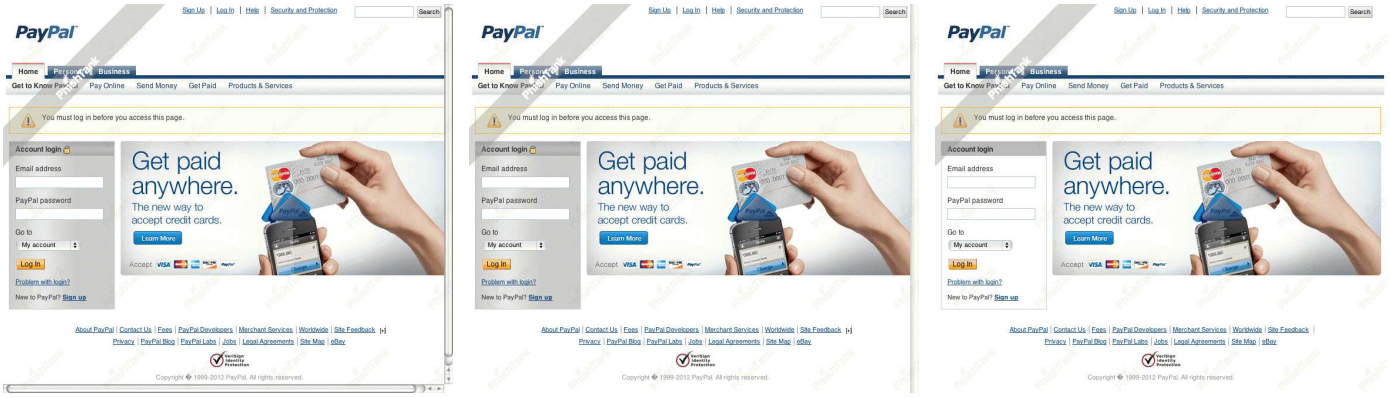
C. Threat Actor 3

During the database exploration we found creeksideshowstable.com domain and started looking for a pattern in the URL. The most common pattern found was *Paypal_Virefication* and we noticed the URLs have always a random segment and also there is the word *"virefication"* which is misspelled. Threat Actor 3 used a total of 309 domains and 7,927 phishing URLs. sample of attack URLs used are shown in TABLE IV.

We repeated the sanity check exercise by comparing some of the attack URLs as shown in Fig. 4.

D. Other Actors

During the phishing database exploration we found other patterns, yet, we were not able to match enough attack URLs from other actors to give us valuable information. TABLE V shows several compromised domains from other threat actors. In TABLE VI is shown a sample of URLs, associated with previous domains, characterized by having greater randomness than the URLs from Attackers 1 & 2.



(a) http://canaanproyectosmetalicos.net/404.php (b) http://creeksideshowstable.com (c) http://shoalhavendeliveries.com.au/

Fig. 4. Visual analysis of the phishing attacks made by Threat Actor 3. We can confirm that these attacks are targeting the same brand, therefore, it is safe to assume they are being made by the same threat actor.

TABLE V. OTHER ACTORS DOMAINS WITH MORE THAN 10 URLS

Domain	No of Phishing URLs
162.144.71.74.	1739
gamedayenterprise.com	101
netspbsstore.com	1459
11qnt.info	1739
indiandeliveryboy.com	900
toughbook.cl	8132
netspbsstore.com	1459
omarfaruquemosque.org.uk	5
kuchijewelleryonlinestore.com	109
wigscwd.com.au	21

TABLE VI. SAMPLE OF URLS USED BY OTHER THREAT ACTORS

http://gamedayenterprise.com/Account/c33576ba2ff545f03751f04114f1895b/ http://gamedayenterprise.com/Account/83b083517297cd36198bf97bbfdc843d/ http://gamedayenterprise.com/Account/7aa7d986e6e1a9a070a548bad210c094/ http://gamedayenterprise.com/Account/01e94b0978d22d647a970e6dbdac3918/
http://wigscwd.com.au/themes/redcharm/css/customers.online http://wigscwd.com.au/little.redirecting/logdgdfgdf/wqrweterye/plmnju/efsdgsdhdfy/omnsgdsd http://wigscwd.com.au/little.redirecting/pidgsdgsd/andsdgsddgs/adedsgsdsd/dejdsgsdg/meldsdsgs http://wigscwd.com.au/templates/working/data http://wigscwd.com.au/templates/working/data/webscr.php?cmd=_login-run&dispatch=5885d80a13c0db1f1ff80d546411d7f8a8350c132b
http://kuchijewelleryonlinestore.com/fonts/yahoo/m.i.php?n= http://kuchijewelleryonlinestore.com/gmh/indexx.php?Fmail.google.com%2Fmail%2F;action=billing_login=true;amp;disp;dispht http://kuchijewelleryonlinestore.com/gmh/indexx.php?Fmail.google.com%2Fmail%2F=&action=bi http://kuchijewelleryonlinestore.com/fonts/yahoo/m.i.php?amp;fid=4&n http://kuchijewelleryonlinestore.com/gmh/indexx.php?Fmail.google.com%2Fmail%2F;ltmpl;scc=1;ss=1 http://kuchijewelleryonlinestore.com/gmh/indexx.php?770e57f36e6524248383a3f26adc http://kuchijewelleryonlinestore.com/gmh/indexx.php?770e57f36e6524248383a3f26adc=&Fmail.google.com/mail
http://www.omarfaruquemosque.org.uk/about/mbt-womens-casual-fora-blue-shoes.html http://www.omarfaruquemosque.org.uk/about/tiffany-co-love-and-love-bracelet-p-282.html http://www.omarfaruquemosque.org.uk/about/tiffany-co-collection-numerical-cuff-bnagle-p-171.html

IV. GENERATING PHISHING URLS

In this section we present an effective AI approach to produce a URL Phishing generator. In the first section, we establish the main objectives of a threat actor. In the second section, we design an algorithm called DeepPhish based on the defined goals.

A. Understanding threat actor motivations

By taking the role of a threat actor and analyzing the vast amount of data, we were able to uncover the underlying structure of a phishing attack. Such attacks follow several values summarized in the following objectives:

1) To maximize the *effectiveness rate* defined as

$$\epsilon = \frac{n_\epsilon}{n_T}, \tag{1}$$

where n_ϵ is the number of URLs that bypass a Proactive Phishing Detection System from Section II-B and n_T is the total generated URLs with the same technique.

2) To maximize the *success rate* defined as,

$$s = \frac{n_s}{n_T}, \tag{2}$$

where n_s is the number of URLs that actually steal user credentials.

3) To maximize the *operational efficiency* defined as

$$e = \frac{n_T}{t}, \tag{3}$$

where t is the wasted time creating a certain amount n_T of URLs.

Above objectives trace the general road for a phishing URL generator, however each one reaches different aspects of an attack. Objective 1 is focused on promoting URL generators able to defeat detection systems. By contrast, objective 2 is concerned with tricking the end user in order to steal their credentials. In general, both objectives are not necessarily accomplished simultaneously, and there is a trade-off between between the two, such that increasing the first one will decrease the other, and vice-versa. Given the purposes of this paper and the available data, objective 1 will be used to define the metric performance of the AI phishing URL generator deployed.

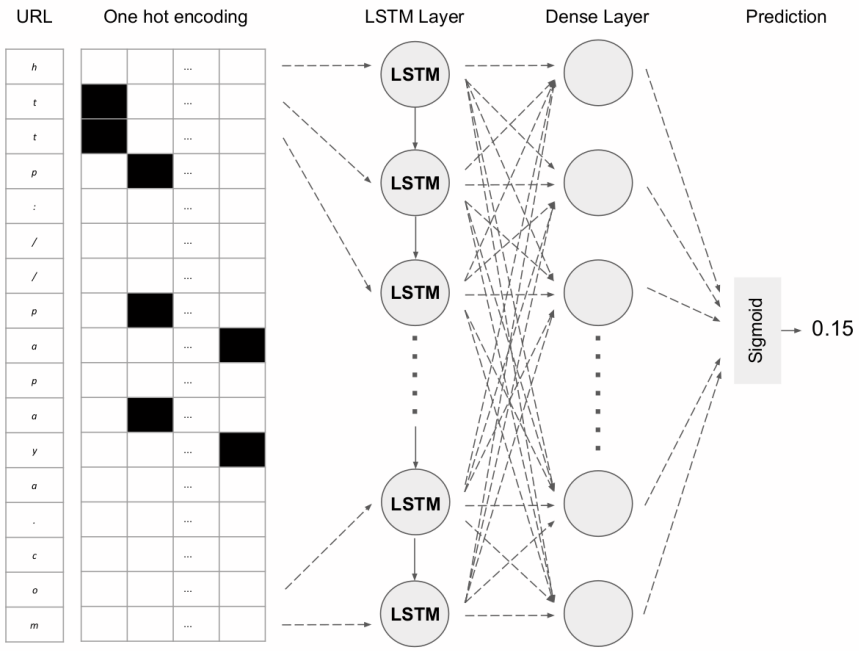


Fig. 5. LSTM architecture for the implementation of DeepPhish algorithm. Each input character is translated into a one-hot encoding. The encoded data is fed into a LSTM layer. The LSTM layer is fully connected to a dense layer. Finally, the classification is performed using an output sigmoid neuron.

B. DeepPhish Algorithm

DeepPhish is an AI algorithm that enhances the threat actor work by learning the effective patterns of their previous attacks. Roughly, DeepPhish uses effective URLs as an input to learn intrinsic structure, such that it allows the generation of new synthetic URLs preserving those characteristics.

First, we collected and concatenated in a full prose text, the effective URLs from historical attacks. Then, from full text, taking steps of size S, we created sentences with lengths L that the model will use to learn which is the next character. With this setup, we created a one-hot encoding representation of the data based on previously defined vocabulary, such that X and Y take the form:

$$X \text{ features with shape } N \times L \times V,$$

$$Y \text{ label with shape } N \times V,$$

where N is the number of sentences and V the number of different characters in the vocabulary. To summarize, for each row in X representing a sentence is predicted a row in Y representing a probability distribution of the next character.

Using a Recurrent Neural Network, in particular a Long Short-Term Memory Network [21], we built a model that receives as input the one-hot encoding and fed them into a LSTM layer. Then, the LSTM layer is connected to a dense layer with V neurons and the classification is performed using an output sigmoid neuron. A visual representation of the selected architecture is illustrated in Fig. 5. The network is trained by back-propagation using a categorical cross-entropy loss function and a *RMSprop* optimizer to prevent oscillations in the optimization process.

Algorithm 1 DeepPhish Algorithm

Input: effectiveURLs
Output: syntheticURLs
Initialization :
vocabulary = generateVocabulary(effectiveURLs)
sentences = generateSentences(effectiveURLs)
Enconding :
oneHot = oneHotEncoding(vocabulary,sentences)
Training :
model = modelImplementation(oneHot)
text = seed()
for sentence ∈ sentences **do**
  text ← model.predictCharacter(sentence)
end for
paths = getPaths(text)
syntheticURLs = completeURLs(domains,paths)
return syntheticURLs

Finally, to generate synthetic URLs, we defined a seed sentence[3] and predicted the next character iteratively. To get variability in the prediction, we tuned a degeneration parameter λ to shift predicted probability distribution. Once the model generates a full prose text, we split it by *http* structure to produce a list of *pseudo* URLs. To clean the data, we removed repeated *pseudo* URLs, dropping meaningless and forbidden characters and taking only the generated synthetic paths of the *pseudo* URLs. For each synthetic path we assigned a compromised domain, such that the synthetic URLs take the form: **http://** + tld + $path$. Algorithm 1, shows core steps of DeepPhish.

[3]The algorithm uses as seed a random segment from initial text

TABLE VII. THREAT ACTORS' PERFORMANCE VS PHISHING DETECTION SYSTEM

Threat Actor	Number of URLs (n_T)	Number Effective (n_ϵ)	Effectiveness Rate (ϵ)	Average Score
All Attacks	1,146,441	2,729	0.24%	0.996
Threat Actor 1	1,007	7	0.69%	0.989
Threat Actor 2	102	5	4.91%	0.946
Threat Actor 3	8,978	7	0.08%	0.999

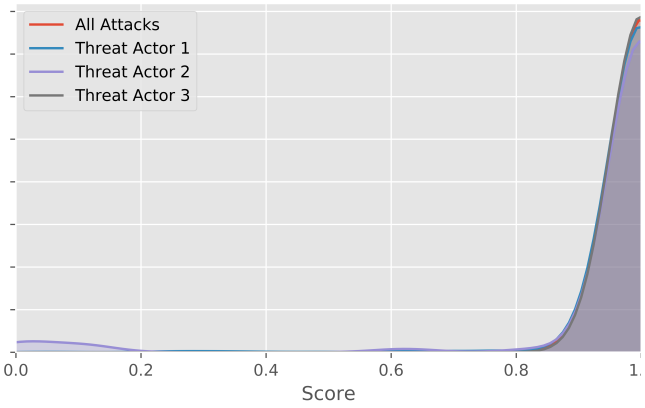


Fig. 6. AI phishing detection score by threat actor. The higher the score, the higher the probability of it being a phishing attack, according to the detection system.

V. RESULTS

In this section we describe our experiment results. We used a database of more than a million phishing URLs to evaluate the effectiveness ϵ (1) of defeating an AI phishing detection system. In particular, we implemented a phishing URL classifier using Deep Recurrent Neural Network as described in Section II-B. To classify URLs this model produces a score from 0 to 1, where 1 means a 100% probability of the URL being used for phishing purposes. By taking a threshold equal to 0.5, we classify each URL as phishing or not, according to its score.

A. Current Attacker Strategies

From the exploration of the phishing database, in Section III we uncovered different threat actors and measured their effectiveness against the AI detection system implemented. We found that the URLs with less randomly generated segments tend to be more effective. As shown in TABLE VII, Threat Actor 2 has the highest effectiveness rate with 4.91%, and Threat Actor 1 the second best effectiveness rate with 0.69%. This is significantly better than the average effectiveness rate of 0.24%. Threat Actor 3, on the other hand, used too many random characters, so they were easily spotted by the detection system. In Fig. 6, we show the AI phishing detection system score distribution of phishing URLs by actor. It was observed that the threat actors' URL score from the AI phishing detection system are skewed towards high scores, and that only Threat Actor 2 has a slight number of effective URLs with low scores.

TABLE VIII. DATA PROCESSING PARAMETERS BY THREAT ACTOR

Threat Actor	Length (L)	Step Size (S)	Degeneration (λ)
Threat Actor 1	200	3	1.40
Threat Actor 2	80	5	1.10

TABLE IX. THREAT ACTORS' PERFORMANCE USING DEEPPHISH VS PHISHING DETECTION SYSTEM

Threat Actor	Number of URLs (n_T)	Number Effective (n_ϵ)	Effectiveness Rate (ϵ)	Average Score
Threat Actor 1	1,007	7	0.69%	0.989
DeepPhish 1	1,007	210	20.90%	0.778
Threat Actor 2	102	5	4.91%	0.946
DeepPhish 2	102	37	36.28%	0.617

B. Attackers Using DeepPhish to Generate Phishing URLs

For the experiment, we selected Threat Actors 1 & 2 to apply our DeepPhish algorithm and enhance its performance. First, we selected a *Categorical Cross Entropy* as loss function and a *RMSprop* optimizer to prevent oscillations in the optimization process. Second, in the training process, we defined number of epochs equal to 30 and a batch size of 256, such that the learning curve would converge. Finally, we found that by tuning the length L, step size S and degeneration parameter λ for each threat actor, the model can learn enough to produce similar patterns in the synthetic URLs. TABLE VIII shows the selected values for each threat actor.

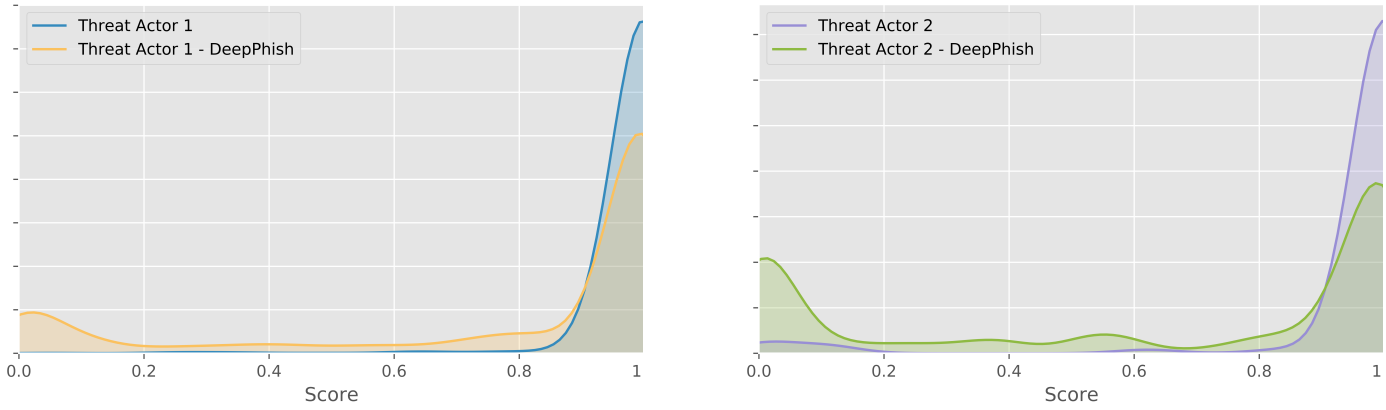
In the end, the implemented model showed improvements in the effectiveness rate for each threat actor. In TABLE IX we show that by training the DeepPhish algorithm with the selected parameters described above by threat actor, we were able to increase Threat Actor 1's effectiveness by a factor of nearly 30, rising from 0.69% to 20.9%. For threat Actor 2 we achieved an increase of nearly 8 times their effectiveness, from 4.91% to 36.28%. This effectiveness increase was accomplished by a decrease in average score, from 0.98 to 0.77 for Threat Actor 1, and 0.946 and 0.61 for Threat Actor 2.

Finally, Fig. 7 shows a comparison of initial score distribution versus final score distribution after DeepPhish implementation. For each threat actor, DeepPhish algorithm generates a subset of effective (low score) synthetic URLs alongside (high score) non-effective URLs. This multi-modal distribution with a peak in high values and a small peak on low values is the same type of result we uncovered in the initial strategy of Threat Actor 2 as Fig. 6 shows. Thus, the DeepPhish algorithm shows that it is able to replicate what this initial strategy does, but with higher effectiveness.

VI. CONCLUSIONS

This work demonstrates how threat actors can enhance the effectiveness of their phishing attacks by using AI as a malicious tool. We analyzed more than one million phishing attacks to understand the URL creation strategy of different threat actors. This analysis showed different strategies used by threat actors and their effectiveness against a AI phishing detection system.

We then created DeepPhish, a tool that demonstrates the potential impact when threat actors use AI in their processes. The DeepPhish algorithm learns from the previous effective attacks to then generate new synthetic URLs with a higher chance of bypassing fraud defense mechanisms. This learning



(a) Threat Actor 1 Score Distribution

(b) Threat Actor 2 Score Distribution

Fig. 7. Comparison of the Threat Actors 1 & 2 AI phishing detection score distribution when using DeepPhish.

is done by using Recurrent Neural Networks, in particular Long Short-Term Memory Networks, to understand the patterns in the sequence of characters of the effective URLs, and then create the synthetic URLs. Using the algorithm with the data of two threat actors, they were able to improve their effectiveness rate, measured as the percentage of attacks not blocked by a proactive phishing detection system, from 0.69% to 20.9%, and from 4.91% to 36.28%, respectively.

VII. Future Work

As previously discussed, our work focused on maximizing the effectiveness rates against an AI phishing detection system, however, because of data limitations, we were not able to model the success defined as the percentage of attacks in which the attacker actually carries out their objective of acquiring credentials. We foresee the importance of collecting more data that allows us to optimize at the same time the effectiveness and success rates, for a more robust experiment that takes into account all of the threat actors' objectives.

Lastly, it is important to automate the process of retraining the AI phishing detection system by incorporating the new syntetic URLs that each threat actor may create. There are several lines of research that may be taken, with the use of Generative Adversarial Networks being the most straightforward [29].

References

[1] A. Correa Bahnsen, E. C. Bohorquez, S. Villegas, J. Vargas, and F. A. González, "Classifying phishing urls using recurrent neural networks," in *Electronic Crime Research (eCrime), 2017 APWG Symposium on*. IEEE, 2017, pp. 1–8.

[2] J. Vargas, A. Correa Bahnsen, S. Villegas, and D. Ingevaldson, "Knowing your enemies: Leveraging data analysis to expose phishing patterns against a major US financial institution," in *2016 APWG Symposium on Electronic Crime Research (eCrime)*, 2016, pp. 52–61.

[3] J. Saxe and K. Berlin, "expose: A character-level convolutional neural network with embeddings for detecting malicious urls, file paths and registry keys," *arXiv preprint arXiv:1702.08568*, 2017.

[4] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: continual prediction with LSTM." *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[5] J. Zhang, P. Porras, and J. Ullrich, "Highly Predictive Blacklisting," in *17th USENIX Security Symposium*, 2008, pp. 107–122.

[6] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond Blacklists : Learning to Detect Malicious Web Sites from Suspicious URLs," *World Wide Web Internet And Web Information Systems*, pp. 1245–1253, 2009.

[7] C. Whittaker, B. Ryner, and M. Nazif, "Large-Scale Automatic Classification of Phishing Pages," *NDSS '10*, 2010.

[8] S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, "A comparison of machine learning techniques for phishing detection," in *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit on - eCrime '07*, 2007, pp. 60–69.

[9] G. L'Huillier, A. Hevia, R. Weber, and S. Rios, "Latent semantic analysis and keyword extraction for phishing classification," in *International Conference on Intelligence and Security Informatics*, 2010, pp. 129–131.

[10] S. Marchal, J. Francois, R. State, and T. Engel, "PhishStorm: Detecting Phishing With Streaming Analytics," *IEEE Transactions on Network and Service Management*, vol. 11, no. 4, pp. 458–471, 2014.

[11] S. Marchal, K. Saari, N. Singh, and N. Asokan, "Know Your Phish: Novel Techniques for Detecting Phishing Sites and their Targets," oct 2015.

[12] ——, "Know Your Phish: Novel Techniques for Detecting Phishing Sites and Their Targets," in *International Conference on Distributed Computing Systems*, 2016, pp. 323–333.

[13] R. Verma and K. Dyer, "On the Character of Phishing URLs: Accurate and Robust Statistical Learning Classifiers," in *ACM Conference on Data and Application Security and Privacy*, 2015, pp. 111–121.

[14] V. Ramanathan and H. Wechsler, "Phishing detection and impersonated entity discovery using Conditional Random Field and Latent Dirichlet Allocation," *Computers & Security*, vol. 34, pp. 123–139, 2013.

[15] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying Suspicious URLs : An Application of Large-Scale Online Learning," in *International Conference on Machine Learning*, Montreal, Canada, 2009, pp. 681–688.

[16] R. M. Mohammad, F. Thabatah, and L. McCluskey, "Predicting phishing websites based on self-structuring neural network," *Neural Computing and Applications*, vol. 25, no. 2, pp. 443–458, 2014.

[17] C. Ardi and J. Heidemann, "Poster: Lightweight content-based phishing detection," USC/Information Sciences Institute, Tech. Rep. ISI-TR-2015-698, May 2015.

[18] G. Wang, H. Liu, S. Becerra, K. Wang, S. Belongie, H. Shacham, and S. Savage, "Verilogo: Proactive phishing detection via logo recognition," UC San Diego, Tech. Rep. CS2011-0969, Aug. 2011.

[19] A. Le, A. Markopoulou, and M. Faloutsos, "PhishDef: URL Names Say It All," in *INFOCOM, 2011 Proceedings IEEE*, 2011.

[20] Z. C. Lipton, "A Critical Review of Recurrent Neural Networks for Sequence Learning," *CoRR*, vol. abs/1506.0, pp. 1–38, 2015.

[21] S. Hochreiter and J. J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1–32, 1997.

[22] T. Dietterich, "Machine learning for sequential data: A review," *Structural, syntactic, and statistical pattern recognition*, pp. 1–15, 2002.

[23] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," *Journal of Computer Security*, vol. 19, no. 4, pp. 639–668, 2011.

[24] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *Expert Systems with Applications*, vol. 36, no. 10, pp. 11 994–12 000, 2009.

[25] M. Brundage, S. Avin, J. Clark, H. Toner, P. Eckersley, B. Garfinkel, A. Dafoe, P. Scharre, T. Zeitzoff, B. Filar *et al.*, "The malicious use of artificial intelligence: Forecasting, prevention, and mitigation," *arXiv preprint arXiv:1802.07228*, 2018.

[26] R. Gallagher, "Where do the phishers live? collecting phishers geographic locations from automated honeypots," in *ShmooCon*, 2016.

[27] J. Seymour and P. Tully, "Weaponizing data science for social engineering: Automated e2e spear phishing on twitter," in *Black Hat USA*, 2016.

[28] H. S. Anderson, J. Woodbridge, and B. Filar, "Deepdga: Adversarially-tuned domain generation and detection," in *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, ser. AISec '16. New York, NY, USA: ACM, 2016, pp. 13–21.

[29] 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*, 2014, pp. 2672–2680.