2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA) | 978-1-6654-2529-2/20/$31.00 ©2021 IEEE | DOI: 10.1109/CyberSA52016.2021.9478220

An Empirical Study: Automated Subdomain Takeover Threat Detection

Yunjia Wang
School of Computer Science
University of St Andrews
United Kingdom
yw43@st-andrews.ac.uk

Ziling Li
Fuying Lab
NSFocus IT Co.,Ltd
China
liziling@nsfocus.com

Tiejun Wu
Fuying Lab
NSFocus IT Co.,Ltds
China
wutiejun@nsfocus.com

Ishbel Duncan
School of Computer Science
University of St Andrews
United Kingdom
immd@st-andrews.ac.uk

Qixin Lyu
The Country Day School
Canada
lyuqixin0627@hotmail.com

***Abstract*—Due to the development of E-commerce, phishing attacks are one of the major current cyber threats. Phishing attacks have become increasingly sophisticated and have exploited both individuals and organizations. For the enterprise, a successful duplicate phishing website may affect an organizations reputation or be the basis of a subdomain takeover attack. This latter attack can completely escape the detection of an SSL certificate and have a direct impact on the enterprise. A successful subdomain takeover attack has a higher threat level as a controllable subdomain owns the same SSL certificate with its parent website, and yet it does not require an advanced technical skill to exploit. In this paper, two techniques have been presented as potential solutions. One is a query approach based on machine learning for querying existing subdomains and the second is an auto-detection system to identify the potentially risky subdomains.**

***Index Terms*—Phishing, Subdomain Takeover Attack, Subdomain Enumeration Tools, Subdomain Takeover Detection.**

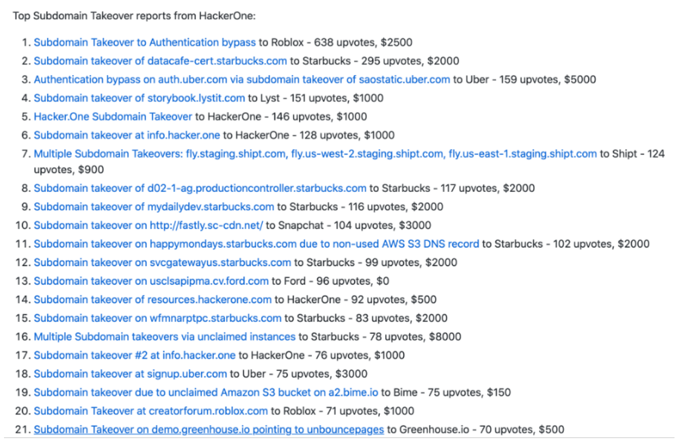
I. Introduction

Phishing threats have been in existence for many years, since the establishment of the Internet. Also, phishing has continuously evolved and increased in application. In particular, due to the development of E-commerce, phishing attacks are one of the major current cyber threats [1] [2]. This attack has become increasingly sophisticated and has exploited both individuals and organizations [3]. The victims of phishing threats can be classified into two parts: the general user (individuals), and the enterprise (organizations).

For the general user, phishing is a technique used by hackers or attackers to trick users into trusting an illegitimate website, to steal their data or sensitive credentials such as usernames, passwords, and credit cards details etc [2]. In our previous works, we presented the relevant solutions to mitigate various phishing attacks. For example, in [4], an OCR approach has been presented to classify malicious (phishing) websites. This approach not only has a high accuracy, but also overcomes the current limitations of existing solutions (Blacklist and Machine Learning). Subsequently, the usability and applicability of this approach has been examined to defend against DNS hijacking attacks. Also, a prototype implementation of this approach has been developed and conducted on mobile platforms in [5]. Finally, the ISP hijacking attack has been mitigated by computing the consistency of the DOM tree in [6].

For the enterprise, a successful duplicate phishing website may affect an organizations reputation. The SSL certificate is used to distinguish between the phishing website and the original. Although the certificate warning can be minimized by a HTTP downgrade attack as mentioned in [5], there is another situation (attack) that is named as the Subdomain Takeover. This attack can escape the detection of an SSL certificate check completely and cause a direct impact to the enterprise.

Recent research elsewhere revealed that the subdomain takeover issue is a serious and widespread security risk [7] [8]. The large organization that owns a vast number of subdomains may suffer the potential problem from their DNS servers with an incorrect configuration, which can cause corresponding users to be redirected to a malicious website due to the resolution of a subdomain CNAME record [9]. A successful subdomain takeover attack has a higher threat level as this controllable subdomain shares the same SSL certificate with its parent website. From HackerOne report [10], the threat of a subdomain takeover is not a rare case, and also has a higher bonus for the security researcher, as shown in the Figure 1 below. For example, Patrik Hudal found two subdomain takeover issues on Starbucks on June 26 2018 and May 23 2019 respectively [11] [12]. The subdomain *mydailydev.starbucks.com* and *svcgatewayus.starbucks.com* were available to register by anyone, and he was awarded $4000 by Starbucks. Subsequently, Parzel Zenker submitted another report to Starbucks and was awarded a $2000 bonus on August 29 2019, as the subdomain *datacafe-cert.starbucks.com* had an CNAME record pointing to an unclaimed Azure webservice [13].



Top Subdomain Takeover reports from HackerOne:

1. Subdomain Takeover to Authentication bypass to Roblox - 638 upvotes, $2500
2. Subdomain takeover of datacafe-cert.starbucks.com to Starbucks - 295 upvotes, $2000
3. Authentication bypass on auth.uber.com via subdomain takeover of saostatic.uber.com to Uber - 159 upvotes, $5000
4. Subdomain takeover of storybook.lystit.com to Lyst - 151 upvotes, $1000
5. Hacker.One Subdomain Takeover to HackerOne - 146 upvotes, $1000
6. Subdomain takeover at info.hacker.one to HackerOne - 128 upvotes, $1000
7. Multiple Subdomain Takeovers: fly.staging.shipt.com, fly.us-west-2.staging.shipt.com, fly.us-east-1.staging.shipt.com to Shipt - 124 upvotes, $900
8. Subdomain takeover of d02-1-ag.productioncontroller.starbucks.com to Starbucks - 117 upvotes, $2000
9. Subdomain takeover of mydailydev.starbucks.com to Starbucks - 116 upvotes, $2000
10. Subdomain takeover on http://fastly.sc-cdn.net/ to Snapchat - 104 upvotes, $3000
11. Subdomain takeover on happymondays.starbucks.com due to non-used AWS S3 DNS record to Starbucks - 102 upvotes, $2000
12. Subdomain takeover on svcgatewayus.starbucks.com to Starbucks - 99 upvotes, $2000
13. Subdomain takeover on usclsapipma.cv.ford.com to Ford - 96 upvotes, $0
14. Subdomain takeover of resources.hackerone.com to HackerOne - 92 upvotes, $500
15. Subdomain takeover on wfmnarptpc.starbucks.com to Starbucks - 83 upvotes, $2000
16. Multiple Subdomain takeovers via unclaimed instances to Starbucks - 78 upvotes, $8000
17. Subdomain takeover #2 at info.hacker.one to HackerOne - 76 upvotes, $1000
18. Subdomain takeover at signup.uber.com to Uber - 75 upvotes, $3000
19. Subdomain takeover due to unclaimed Amazon S3 bucket on a2.bime.io to Bime - 75 upvotes, $150
20. Subdomain Takeover at creatorforum.roblox.com to Roblox - 71 upvotes, $1000
21. Subdomain Takeover on demo.greenhouse.io pointing to unbouncepages to Greenhouse.io - 70 upvotes, $500

Fig. 1. The bonus summary about Subdomain Takeover Events in HackerOne [10].

The subdomain takeover attack often happens on CNAME, NS and MX records. The CNAME record attack has the highest proportion and the NS and MX are relatively rare. The current approach to detect this potential vulnerability is divided into three experimental steps:

1) Querying subdomains.
2) Resolve and discover the related record (such as CNAME, NS and MX).
3) Detect the usability of destination domain.

However, there is not an automatic approach or tool (with a higher accuracy or a faster speed) to execute the above steps to detect this potential vulnerability, so far. Therefore, in this research, we investigate how to detect a potential subdomain takeover risk by an automatic approach (in this study, we focus on the CNAME threat), and we also present a method to generate a valuable dictionary for querying subdomains by using machine learning. The following research questions will be considered in the next sections:

1) What is the best tool for discovering subdomains?
2) What is the limitation of the current querying subdomain approach?
3) How to generate a valuable dictionary (using in Brute Force to discover subdomains) based on machine learning?
4) How to perform an automatic approach to discover subdomain takeover risks?

II. Related Work

Prior to presenting the methodology of this research, the related works will be discussed in this section. Three aspects will be introduced; the identification of the subdomain takeover attack, the approach of querying subdomains and the comparison of present subdomain enumeration tools.

What is the subdomain takeover attack, and how does it happen?

A subdomain takeover is the process of registering a non-existing domain name to gain control over another domain [14]. Before uncovering this attack, we need to explain how DNS resolution works with a CNAME record. The specific procedure is shown as follows in Figure 2:

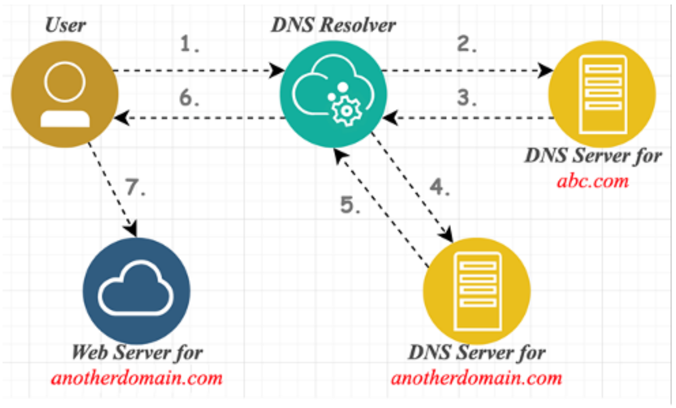

User
DNS Resolver
1.
2.
6.
3.
7.
4.
5.
DNS Server for abc.com
Web Server for anotherdomain.com
DNS Server for anotherdomain.com

Fig. 2. An illustration of accessing *subdomain.abc.com* with IP address 1.1.1.1.

Step 1. In order to access the targeted website *subdomain.abc.com*, a user will send a DNS query to a DNS resolver requesting the corresponding IP address of *subdomain.abc.com*.

Step 2. The DNS resolver requests the IP address of *subdomain.abc.com* from the DNS server of *abc.com*.

Step 3. In the DNS server of *abc.com*, a CNAME record has preliminary configured, the targeted website, *subdomain.abc.com* is pointed to *anotherdomain.com*. Thus, this DNS server will return a message that says the targeted website has a CNAME record to *anotherdomain.com*.

Step 4. Subsequently, the DNS resolver will send another queried package to the DNS server of *anotherdomain.com* for asking the IP address of *anotherdomain.com*.

Step 5. In the DNS server of *anotherdomain.com*, an A record (refers to the IP address) can be found regarding the required website *anotherdomain.com*, for example, *anotherdomain.com* IN *A* 1.1.1.1. This IP address will be then returned to the DNS resolver.

Step 6. The DNS resolver will respond this IP address, 1.1.1.1, to the previous user.

Step 7. Finally, the user will establish the connection to *subdomain.abc.com* according to this gained IP address 1.1.1.1. During this process, the user requests *subdomain.abc.com* rather than *anotherdoamin.com* as the browser is not aware that *anotherdomain.com* even exists. Even though the CNAME record is used, the URL bar in the browser still displays *subdomain.abc.com*.

However, in this procedure, a potential risk may exist if the domain *anotherdomain.com* is available for re-registration by anyone for some reasons, such as it is expired or shutdown. An attacker could obtain the full control of the website *subdomain.abc.com* through purchasing the available domain *anotherdomain.com* if the previous CNAME setup has not been deleted in the DNS server of *abc.com*. For example, see Figure 3 below.

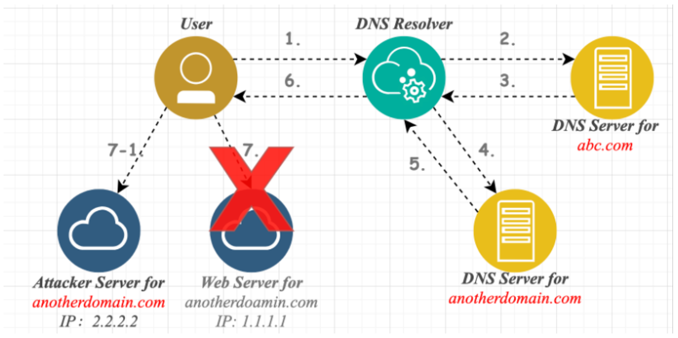


Fig. 3. An illustration of the takeover of the subdomain *subdomain.abc.com* with IP address 2.2.2.2.

In the DNS server of *anotherdomain.com*, the A record regarding *anotherdomain.com* will be changed if the attacker purchases this domain and configures it to point to his server (in this example, the IP address 2.2.2.2) as it is available. Subsequently, the DNS server of *anotherdomain.com* will return this IP address 2.2.2.2 to the DNS resolver.

Finally, the DNS resolver will deliver this IP address to the user and they will establish the connection to the attacker's server (IP address 2.2.2.2) rather than the previous server (IP address 1.1.1.1). Which means this subdomain (*subdomain.abc.com*) will be fully controlled by the attacker who purchases *anotherdomain.com* as the CNAME record has not been updated from the DNS server of *abc.com*, until this record is deleted.

For this type of CNAME subdomain takeover attack, it does not require an advanced technical skill, but it will cause a serious impact on any enterprise. Most organizations prefer to configure **.example.com* in their issued SSL certificate for establishing an encrypted connection. This is a significant convenience through this kind of configuration in their SSL certificate as all subdomains of *example.com* will be automatically given this certificate. However, a potential risk has been introduced at the same time under this configuration; the attacker will use this SSL certificate and obtain the same permissions once any subdomain is hijacked through a successful CNAME takeover attack.

The querying subdomains approach

From the specific explanation above, in order to detect the vulnerable subdomains, the critical part is discovering subdomains as a potential risk could be ignored if the queried subdomain is not comprehensive. In [15], the methodology of each subdomain enumeration tool is summarised by Esteban.

- *Querying subdomain through search engines*

Most subdomains can be gathered by using specialised hacking queries. For example, Google hacking queries are often used to gather subdomains from the results of the latest Googlebot crawl, using the command of *"site: example.com -www"*. The response is useful for discovering subdomains that are not protected by robots.txt.

Some subdomain discovery tools rely on this type of built-in hacking query language to find subdomains from various search engines, such as Google, Bing, Yahoo. However, a challenge exists in this approach; the subdomain that has not been queried by search engines in previous searchers would not be found by this method.

- *Discover subdomain through brute force*

In some discovery tools, the recursive brute force approach has been implemented with an extra dictionary (or plugin wordlist) to generate the relevant subdomain names. The applicability of subdomains will be detected through traversing the dictionary. So far, the gathered result has a relatively high accuracy, though the time consumption of this process is longest in all of the subdomain discovery approaches. However, the generated results are not absolutely accurate, it is still subject to the integrity of the so-called dictionary. Also, the full procedure has a higher time consumption if the dictionary is large.

- *Running DNS zone transfers*

The remote DNS zone content can be fully duplicated through the method of DNS zone transfer, and all the configured subdomain information can be revealed. This approach only works when the DNS zone is not protected or limited by the system administrators. Moreover, this approach requires an interface or permission to connect the remote (primary) DNS.

- *Discover SSL public information*

An SSL certificate is used to encrypt the data during the network connection between the users and servers, it is also useful for penetration information research. Within the SSL certificate, further information can be explored. For example, the Subject Alternate Name (SAN) in SSL certificates can be used to extract domains and subdomain names. However, a potential limitation may exist while discovering subdomains from the SSL certificate. As mentioned above, all subdomains that are under this targeted website (domain) will achieve the same certificate automatically if the organization configures **.example.com* in their issued SSL certificate. This means that the attacker can use this SSL certificate to obtain the same permission once any subdomain is hijacked through a successful CNAME takeover attack.

A comparison of subdomain enumeration tools

The current subdomain enumeration tools associate one or more functions above to query the existing subdomain names, some of which were introduced in [15] [16], such as, amass, SubBrute, Knock and Sublist3r. Due to their various functions, a security researcher can always pick one or more tools to discover the targeted domain.

Jaspher [17] provided a comparative study to analyze five subdomain enumeration tools (Amass, Subfinder, Knock, Sublist3r and Altdns), with the comparison including usage, methodology and time consumption. Their reconnaissance is classified into five aspects; active information gathering, passive information gathering, brute force, DNS enumeration and the use of search engines. The architectural diagram of each enumeration tool is illustrated in Figure 4:

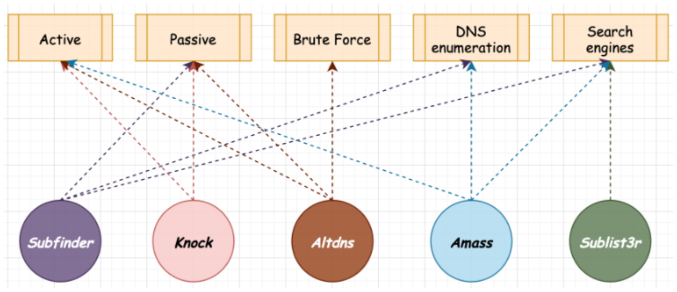


Fig. 4. The methodology in each enumeration tool used.

Subsequently, the time consumption of each enumeration tool was compared, and they indicated that Subfinder is the fastest amongst them with the final speed ranking as: Subfinder <Sublist3r <Knock <Altdns <Amass. Although the accuracy of the generated result in these tools was illustrated in the later sections of this paper, the description is not clear and there is not a specific demonstration or explanation to prove their result by duplication.

Overall, research on subdomain enumeration tools or subdomain takeover detection systems is limited, most of the published documentation is focused on the usage and function of the tool. Therefore, both perspectives are worth researching in order to improve the effectiveness and accuracy of the enumeration tool as well as overcome the risks from subdomain takeover attacks.

III. Methodology

In order to overcome the threat from a subdomain takeover attack, an automatic detection tool is worth researching, in particular, for the security maintenance team in any organization. The current approaches to detect this potential vulnerability involve three steps:

- **Querying Subdomains**: Via utilizing subdomain enumeration tools to collect existing subdomain names. However, in this step, in order to improve result integrity, the final subdomain result is often gathered by the combination of two or more enumeration tools.
- **Discovering the related information**: Via picking the relevant subdomains which involve the CNAME record. (In this experiment, we only focus on CNAME issues, otherwise, NS and MX records need to be considered as well).
- **Examining the usability of the destination domain**: Via confirming the usability of the destination domain from collected CNAME records. Any destination domains which are without an IP address would be labelled as risky. Risky subdomains may be attempted to be registered in a Name Vendor, such as Namecheap[1]. Vulnerable subdomains can be identified if its destination domain is available (to be registered) in any Name Vendor.

[1]Available at:https://www.namecheap.com/.

However, there is not an available resource to fully effect these manipulations for auto-detecting this potential vulnerability. For either novices, security teams or bonus hunters, the detection effectiveness will increase significantly by implementing an auto-system, not only to reduce the manipulation complexity, but also to improve the accuracy of the result. In our research, we present a scheme that combines the steps of querying, discovering and examining to auto-examine potential risks in subdomains. Moreover, in order to break the inherent pattern in the present dictionary, we attempt to overcome its limitation (the composition of dictionary is based on known words), another scheme is presented to generate existing subdomains through Char-RNN, which is a well-designed machine learning model.

The description of Char-RNN model

RNN is a Recurrent Neural Network that is based on time cycles, constructed as sequences for recognition. For example, a text or a speech signal has internal cycles that indicate the presence of short term memory in the network. RNN often works on handling sequence issues and it involves three layers; the input layer, a hidden layer and an output layer [18]. The cell of RNN is circulative in a neural network, as shown in Figure 5 below:

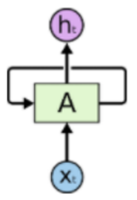


Fig. 5. The architecture of an RNN cell [19].

In the RNN model, the input sequence $\mathrm{x_t}$ is encoded to a fixed-length hidden state at time *t*. A is a hidden state that is updated with the passage of time. In this cyclic part, the data is transferred from one step to the next one within the neural network. A accepts the input data from $\mathrm{x_t}$ and uses cyclic data from the previous *t-1* state to then generate an output $\mathrm{h_t}$. In this way, the RNN cell remains the text sequence message. Also, time passage is considered within the RNN model, an earlier input sequence will have the less weight after an update.

The construction of N vs N RNN model is generated after executing the cycles above, as shown in Figure 6 below:

The following equation is used within the process of Layer 1:

$$\mathrm{h_t} = f\left(Ux_t + Wh_{t-1} + b\right) \tag{1}$$

Where;

- *U* is the matrix of n*m, which represents the weight from the input layer to the hidden layer.

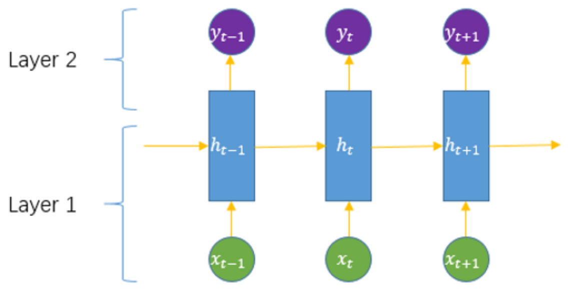


Fig. 6. The architecture of N vs N in RNN model [19].

- x_t is the matrix of m*1, which represents the current input.
- W is the matrix of n*n, which represents the weight between hidden layer to itself.
- h_t is the matrix of n*1, which is the hidden state at the time of t.
- h_{t-1} is the matrix of n*1, which is the hidden state at the time of t-1.
- b is the vector matrix, which is a bias parameter.
- f is the activation function.

The following equation is used within the process of Layer 2:

$$\mathrm{y_t} = \mathrm{Softmax}\left(Vh_t + c\right) \tag{2}$$

Where;

- V is the matrix of n*n, which represents the weight between hidden layer to output layer.
- c is the vector matrix, which is a bias parameter.
- y_t is the current output.

Both Char-RNN and Word-RNN belong to the RNN neural network. Char-RNN has longer time steps that are needed to consider the dependencies amongst more text tokens. Thus, a larger hidden layer is required in Char-RNN. The advantage of Char-RNN is that the character level model does not need to deal with much vocabulary as the count of unique character is much less than the count of unique words. Therefore, Char-RNN is an appropriate way for the subdomain generation with limited string length as the components of subdomain names can usually be set to characters [20]. Here, the classic N vs N model was utilized in Char-RNN, with the input a sequence of length N, and the output is a sequence of the same length. For Char-RNN models, the input sequence is the letter in a sentence, the output is the next letter, which means this model is used to predict the possibility of the next letter through the input letter.

For example, for a simple word Hello, the input sequence is {H,e,l,l,o}, and the output sequence is {e,l,l,o,!}. These two sequences have the same length, thus RNN N vs N model can be used to model this case.

During the process of sequence generation, a character (x_1) is selected (in this case, it is ”h”) and set up to read the initial letter. The probability of the next corresponding character (x_2) is obtained by using the trained model. In order to output the relevant possibility, the Softmax activation function is used in the output layer of the model. A character will be outputted (in this case, the most probable value refers to letter ”e”) based on this function. Subsequently, this letter ”e” will be the next input for x_2 and used to execute the next prediction that generates the next letter. In the end, a string is generated by repeating this procedure.

The Description of an Auto-Detection System

Prior to narrating the scheme of subdomain auto-detection system, a comparative experimentation is conducted to examine five popluar subdomain enumeration tools, these are Amass, Massdns, Knock, Sublist3r and SubDomainBrute. The methodology of each enumeration tool is explained as follow Figure 7:

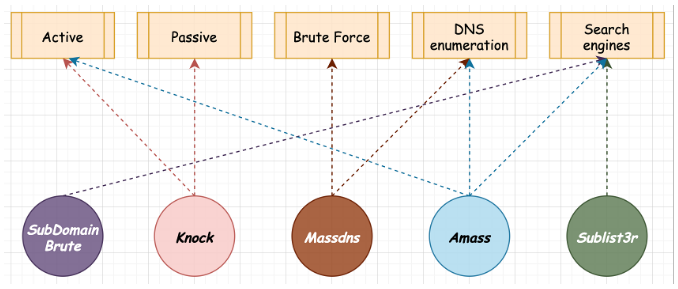


Fig. 7. The methodology in each enumeration tool used.

From this comparison, we will identify the enumeration tool which has the best performance in either efficient computing speed or accurate subdomain results. Subsequently, this enumeration tool will be selected and re-developed for the next experiment. The queried subdomains will be examined for a destination domain according to their specific DNS information.

In order to examine this kind of CNAME takeover attack, we only focus on the usability of final destination domains as the risk often comes from the last CNAME record / address. We assumed two scenarios to explain this threat. For example, the following DNS information is from the result of *”dig edu.abc.com”*:

Scenario 1 (S1):

edu.abc.com **IN *CNAME*** edu.yw43.com
edu.yw43.com **IN *CNAME*** edu.xz32.com
edu.xz32.com **IN *CNAME*** stafocus.com
stafocus.com **IN *A*** 10.10.10.10

The CNAME may be configured from a subdomain to another subdomain (or domain), such as the DNS configured information from scenario 1. From the original subdomain *eud.abc.com* CNAME points to another subdomain *edu.yw43.com*, and the next CNAME to another subdomain *edu.xz32.com*, which the CNAME associates with another domain *stafocus.com*. Finally, the domain *stafocus.com* is assigned to an IP address 10.10.10.10 from the A record. This is a normal DNS configuration of subdomain *edu.abc.com* and the process does not show any potential CNAME takeover risks.

Scenario 2 (S2):

edu.abc.com **IN *CNAME*** edu.yw43.com
edu.yw43.com **IN *CNAME*** edu.xz32.com
edu.xz32.com **IN *CNAME*** stafocus.com

There is a potential takeover risk under the scenario 2, as the final CNAME destination domain *stafocus.com* is not assigned with an A record. Three reasons may cause this phenomenon:

- R1: The Network administrator forgot to assign an IP address to the domain of *stafocus.com*.

- R2: The IP address of domain *stafocus.com* is expired and recycled by a cloud vendor.

- R3: The domain *stafocus.com* is expired and recycled by a Name vendor. It causes the previous DNS configuration of this domain to be emptied.

The takeover attack appears in R2 and R3. In R2, an IP address will be randomly assigned after purchase from a cloud vendor. However, this IP address will be available and assigned to another purchaser if the previous purchaser's rights had expired. The assignment of an IP address is random, so this takeover risk exists if the new purchaser is assigned to a previous IP address, although the probability is rare. Different from the IP address, an available domain can be freely registered from a Name vendor, and the specific name is selectable. Compared to R1 and R2, R3 has the highest success rate. Anyone can take over this domain if the domain is expired from the previous purchase.

However, in both scenarios, what happened if the domain of *xz32.com* is shut down (either expired or discarded)? The configuration under this domain will be emptied, any DNS information is removed, which means the next CNAME record *edu.xz32.com* CNAME *stafocus.com* does not exist. Thus, the result of *"dig edu.abc.com"* in both scenarios are:

edu.abc.com **IN *CNAME*** edu.yw43.com
edu.yw43.com **IN *CNAME*** edu.xz32.com

In this case, we use command of *"dig xz32.com"* to verify the existence of the domain *xz32.com* and then attempt to register this domain with any Name vendor to identify whether it is available. Therefore, the risk is always confirmed from the CNAME record, and the usability of the final destination domain is the key to evaluating a subdomain takeover threat. The specific implementation will be demonstrated in the next section.

IV. Experiment Implementation

In this section, the experiment implementation is divided into two parts: the implementation of querying subdomain using the Char-RNN model and the implementation of an auto-detection system.

A. The Implementation of Querying Subdomains using the Char-RNN Model

In this experiment, the training data of the Char-RNN text generation model are found from an open resource in GitHub [21].

Data Pre-processing

In the structure of a domain name, there are various implications for each level of label, and the character distribution is quite different. For example, the common top-level domain includes IP suffixes such as .com, .net, .org etc. Second-level domain involves companies such as Baidu, Facebook, Google, etc. And the third-level domain contains lower level directories: news, game, sports. The dataset of domain characters is summarised in Table I below:

TABLE I
The character sets used in domain names

Classification	Character Set
Number	0-9
Letter	a-z, A-Z
Specific Character (Symbol)	-, _

From this table, three classifications are involved: Number, letter and specific character (symbol). Thus, since the name of domains are not case sensitive, the domain character dataset can be considered as a total of 38 characters with the initial character only allowed to be a number or a letter, according to domain name generation regulations.

Text Vectorization

The dataset of the gathered subdomain is composed of various strings and the input of the Char-RNN model must be a vector since text data processing cannot be used directly in the machine learning model. Thus, it is necessary to encode each character. Here, the vector of one-hot encoding is utilized to represent the character in the dataset.

One-hot encoding is a simple and widely used encoding method. For example, if there are three variables {male, female, other}, which represents different categories. This can be coded with 3-dimensional feature vectors: {[1,0,0], [0,1,0], [0,0,1]}. In the resulting vector space, each category is orthogonal and equidistant from each other [22].

After encoding the characters, the vector representation results of all unique characters are obtained and stored in the form of a dictionary. In this way, before a subdomain is entered into the model, it needs to check the dictionary to get its related vector representation, and then the vector would be entered into the model for training.

Model training and implementing

During the procedure of model training, firstly the weight matrix of each layer is randomly initialized. Subsequently, the main prefix part of a domain name is set to the input text data. For example, in the case of *app.baidu.com*, app is the main part of this domain. The input of dataset comes from the main part of the subdomain, rather than the entire prefix domain name. This model considers the relationships between each character as well as predicting the next character for each character input.

Based on the original text data, we will add a start symbol <s>and a terminator symbol <e>to the string. For a subdomain string sequence of length 10, there will be 10 labels. The labels correspond to the same sequence as the input string, but with an offset of 1 (the label of each character in a specific position would be the character that is in the next position, as shown in Figure 8 below. During the training process, the green block will be removed, and the blue block will remain.).

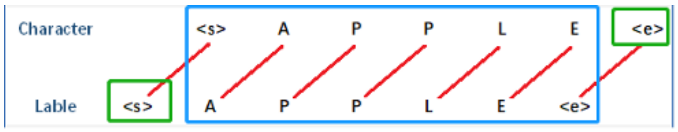


Fig. 8. The explanation of offset, red line represents offset.

Subsequently, further inference is conducted by using the trained RNN model. The initial domain letter can be set to any specific letter or be random. The length of the generated label is controllable as well. For example, if this length is set to 1000, the final count of generated result would be 1000. This length is an option parameter, the generated result is unlimited and random if this setup is empty.

B. The Implementation of an Auto-System

Prior to starting the implementation, two available domains (*stafocus.com* and *ly1026.me*) were registered from *Namecheap.com*, and the specific configurations in Advanced DNS are shown in Tables II and III as below:

TABLE II
ADVANCED DNS CONFIGURATION IN DOMAIN: *stafocus.com*

stafocus.com			
A	@	stafocus.com	68.183.32.17
CNAME	blog	blog.stafocus.com	liziling98.com
CNAME	cs	cs.stafocus.com	hubs.sta.social
CNAME	edu	edu.stafocus.com	allways-cam.com
CNAME	education	education.stafocus.com	allways-cam.com
CNAME	pay	pay.stafocus.com	ly1026.me
CNAME	payment	payment.stafocus.com	ly1026.me
CNAME	search	search.stafocus.com	baidu.com
CNAME	vpn	vpn.stafocus.com	allways-cam.com
CNAME	vpn1	vpn1.stafocus.com	ly1026.me

(**NB**: The URL of *liziling98.com* is a personal blog, and a traffic redirection (301) has been configurated in this server; the URL of *hubs.sta.social* is a social website for HCI group in the School of Computer Science at the University of St Andrews; The URL of *baidu.com* is one of most famous search engine in China; The URL of *allways-cam.com* is website for an education organization; The URL of *ly1026.me* is a domain we registered for the further experimentation.)

Firstly, we used five subdomain enumeration tools to discover subdomain information from the targeted domain. A remote server was purchased to implement this experiment, the setup of this server is: *CPU with 1 vCore, RAM with 1024 MB and Storage with 25 GB SSD.*

TABLE III
ADVANCED DNS CONFIGURATION IN DOMAIN: *ly1026.me*

ly1026.me			
A	@	ly1026.me	45.32.49.95
CNAME	blog	blog.ly1026.me	blog.staofcus.com
CNAME	edu	edu.ly1026.me	edu.stafocus.com

Subsequently, the gathered content is recorded for further comparison, and the time consumption in each subdomain enumeration tools are as shown in Table IV below:

TABLE IV
TIME CONSUMPTION FOR VARIOUS TARGETED DOMAINS IN EACH TOOL.

	Amass	Massdns	Knockpy	Sublist3r	subDomainsBrute
StA	>30m	ct: 7.4s;	7m32s	22m7s	11m24s
Cam	>30m	ct: 9.2s;	1m29s	17m26s	15m18s
Ox	>30m	ct: 7.1s;	2m49s	18m28s	28m33s
IC	>30m	ct: 10.6s	3m24s	11m38s	5m56s
UCL	>30m	ct: 13.2s;	3m18s	13m33s	5m27s

Where;

- StA: *st-andrews.ac.uk*
- Cam: *cam.ac.uk*
- Ox: *ox.ac.uk*
- IC: *imperial.ac.uk*
- UCL: *ucl.ac.uk*

According to the results of using a subdomain querying approach, the most efficient approach is based on the querying of the SSL certificate, although most results may not be alive in this method. The most accurate method is from Brute Force, though the accuracy of the generated result is subject to the integrity of the dictionary.

Therefore, the first conclusion we reached, regarding these subdomain enumeration tools, is that SubDomainBrute has the highest performance in either efficient computing speed or in subdomain result accuracy. Amass also provided a comprehensive subdomain result, though it will consume a lot of time. The transparency of the SSL certificate is utilized in Massdns, but most generated subdomain names are not alive. In further experiments, we refer to the framework and methodology of the SubDomainBrute enumeration tool to redesign the workflow of discovery process.

According to the DNS configuration of the test registered domains noted above, we attempted to execute the terminal command *"dig"* to reveal the map relationship for each subdomain. The result is summarized as follows:

The *stafocus.com* domain:

stafocus.com **IN *A*** 68.183.32.17
blog.stafocus.com **IN *CNAME*** liziling98.com **IN *A*** 149.248.36.25
cs.stafocus.com **IN *CNAME*** hus.sta.social
edu.stafocus.com **IN *CNAME*** allways-cam.com
education.stafocus.com **IN *CNAME*** allways-cam.com
pay.stafocus.com **IN *CNAME*** ly1026.me **IN *A*** 45.32.49.95
payment.stafocus.com **IN *CNAME*** ly1026.me **IN *A*** 45.32.49.95

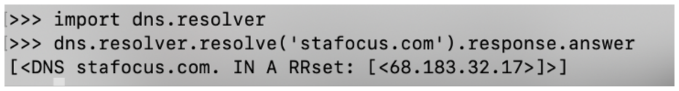
search.stafocus.com **IN** ***CNAME*** baidu.com **IN** ***A*** 39.156.69.79 (220.181.38.148)
vpn.stafocus.com **IN** ***CNAME*** allways-cam.com
vpn1.stafocus.com **IN** ***CNAME*** ly1026.me **IN** ***A*** 45.32.49.95

The *ly1026.me* domain:

ly1026.me **IN** ***A*** 45.32.49.95
blog.ly1026.me **IN** ***CNAME*** blog.stafocus.com **IN** ***CNAME*** liziling98.com **IN** ***A*** 149.248.36.25
edu.ly1026.me **IN** ***CNAME*** edu.stafocus.com **IN** ***CNAME*** allways-cam.com

From the map relationships above, it appears that both domains have potential subdomain takeover risks. In the *stafocus.com* domain, the risks may exist in the subdomain of *cs.stafocus.com*, *edu.stafocus.com*, *education.stafocus.com* and *vpn.stafocus.com* where the CNAME is incorrect. In the domain of *ly1026.me*, risks may exist in the subdomain of *edu.ly1026.me*. Thus the purpose of our system is to identify these potentially risky subdomains from the target domain.

In most subdomain enumeration tools, a python library, named dnspython, has been used to resolve targeted domains to reveal the relevant DNS information. For example, as shown in Figure 9 Below, the domain *stafocus.com* is resolved through calling the function *dns.resolver.resolve()* and using the *.response.answer* function to show the result.



```
>>> import dns.resolver
>>> dns.resolver.resolve('stafocus.com').response.answer
[<DNS stafocus.com. IN A RRset: [<68.183.32.17>]>]
```

Fig. 9. Using the dnspython library.

However, an exception will be notified if an A record does not exist in the targeted domain. For example, under the following scenarios:

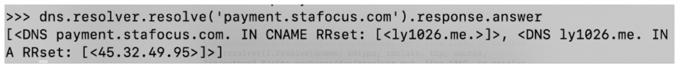
Scenario 1:

payment.stafocus.com **IN** ***CNAME*** ly1026.me **IN** ***A*** 45.32.49.95

Scenario 2:

edu.stafocus.com **IN** ***CNAME*** allways-cam.com

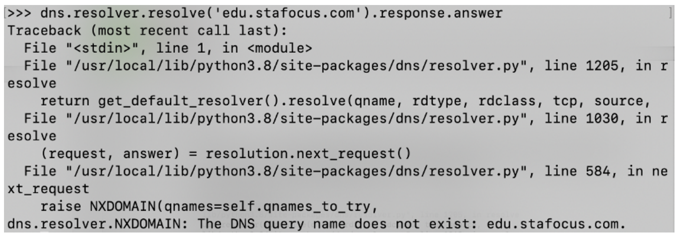
A correct and complete result is resolved from the targeted website *payment.stafocus.com* by using this library in Scenario 1, as shown in Figure 10 below:



```
>>> dns.resolver.resolve('payment.stafocus.com').response.answer
[<DNS payment.stafocus.com. IN CNAME RRset: [<ly1026.me.>]>, <DNS ly1026.me. IN
A RRset: [<45.32.49.95>]>]
```

Fig. 10. The DNS resolution of *payment.stafocus.com* in the dnspython library.

In Scenario 2, an exception is shown if we resolve the targeted website *edu.stafocus.com*. Because an A record does not exist in the destination address *allways-cam.com*, as shown in Figure 11 below:



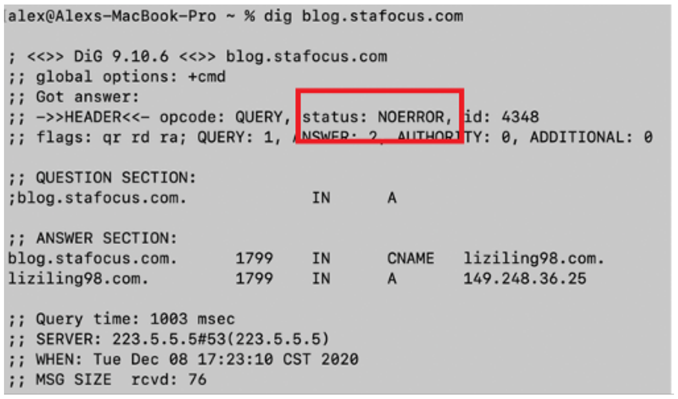
```
>>> dns.resolver.resolve('edu.stafocus.com').response.answer
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/local/lib/python3.8/site-packages/dns/resolver.py", line 1205, in r
esolve
    return get_default_resolver().resolve(qname, rdtype, rdclass, tcp, source,
  File "/usr/local/lib/python3.8/site-packages/dns/resolver.py", line 1030, in r
esolve
    (request, answer) = resolution.next_request()
  File "/usr/local/lib/python3.8/site-packages/dns/resolver.py", line 584, in ne
xt_request
    raise NXDOMAIN(qnames=self.qnames_to_try,
dns.resolver.NXDOMAIN: The DNS query name does not exist: edu.stafocus.com.
```

Fig. 11. The exception in the dnspython library.

Therefore, for the current subdomain enumeration tools, there is an appropriate option to query the relevant subdomain, rather than for discovering the risky subdomain. Any subdomain that does not have an A record will be discarded during the collection procedure. In addition, these discarded addresses may involve non-existent addresses as well, so it is hard to identify risky addresses.

For example, if we have *[abc.aa.bb.pay.edu]* in the dictionary, and the targeted domain is *sta.com*. Using brute force, each string in the dictionary, will combine with the targeted website and is resolved subsequently. However, only the address that returns a correct result will be stored. Which means, if *edu* is an existing (with no risk) subdomain, *pay* is a risky subdomain which has the takeover issue, and *abc*, *aa* and *bb* do not exist, after querying by the enumeration tools, only *edu.sta.com* will be stored. The non-existent addresses, *abc*, *aa* and *bb* are discarded, and if the risky subdomain *pay* cannot be resolved to an A record, this will be discarded as well. Thus, in these discarded addresses, it is hard to identify which has the takeover risk as there may be too many addresses if the length of original dictionary is large.

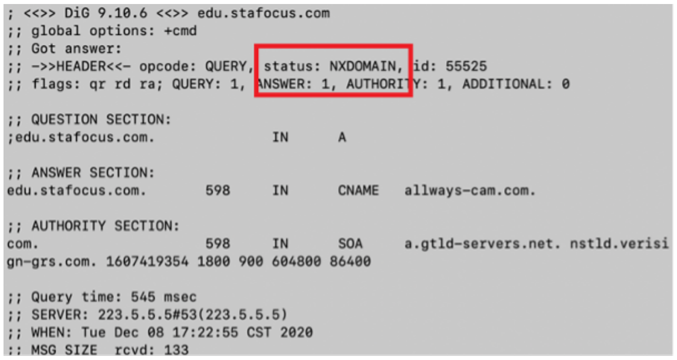
In the first step of our auto detection system, in order to overcome the discarding issues, we utilize the command *"dig"* in the operating system to query the subdomains. Then, key words such as **status: NOERROR** and **CNAME** are used to confirm whether the queried address exists, and we gather the related subdomain information. For example, for a correct address, the status will display **NOERROR**, as shown in Figure 12 below:



```
alex@Alexs-MacBook-Pro ~ % dig blog.stafocus.com

; <<>> DiG 9.10.6 <<>> blog.stafocus.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4348
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;blog.stafocus.com.             IN      A

;; ANSWER SECTION:
blog.stafocus.com.      1799    IN      CNAME   liziling98.com.
liziling98.com.         1799    IN      A       149.248.36.25

;; Query time: 1003 msec
;; SERVER: 223.5.5.5#53(223.5.5.5)
;; WHEN: Tue Dec 08 17:23:10 CST 2020
;; MSG SIZE  rcvd: 76
```

Fig. 12. Example of *dig blog.stafocus.com*.

For either a non-existent or risky address, the status will show **NXDOMAIN**, as shown in Figure 13 below. But in our system, in order to collect the risky addresses, we use the key word "CNAME" in the ANSWER SECTION as a filter.



```
; <<>> DiG 9.10.6 <<>> edu.stafocus.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 55525
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;edu.stafocus.com.              IN      A

;; ANSWER SECTION:
edu.stafocus.com.       598     IN      CNAME   allways-cam.com.

;; AUTHORITY SECTION:
com.                    598     IN      SOA     a.gtld-servers.net. nstld.verisi
gn-grs.com. 1607419354 1800 900 604800 86400

;; Query time: 545 msec
;; SERVER: 223.5.5.5#53(223.5.5.5)
;; WHEN: Tue Dec 08 17:22:55 CST 2020
;; MSG SIZE  rcvd: 133
```

Fig. 13. Example of *dig edu.stafocus.com*.

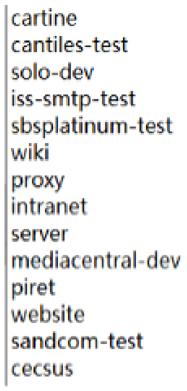
In the second step, we still use the key word "CNAME" to filter the relevant subdomains which involve the CNAME record from the step above.

In the last step, we utilize the python library dnspython to resolve the gathered subdomains that match the key word condition in the second step. For each examined subdomain, the risky subdomain can be identified and evaluated if an exception has been notified. Therefore, all risky subdomains will be listed, and the relevant information will be displayed to the user.

V. Evaluation

The Evaluation of Querying Subdomains in Char-RNN Model

In our experiment, we set random character as the initial letter, rather than a specific one. The final generated result is either a known word or a strange letter combination according to the probability distribution. The form of predicted subdomains is shown in Figure 14 below.



cartine
cantiles-test
solo-dev
iss-smtp-test
sbsplatinum-test
wiki
proxy
intranet
server
mediacentral-dev
piret
website
sandcom-test
cecsus

Fig. 14. The form of predicted subdomains.

In order to evaluate the accuracy of subdomain predictions, we set various lengths during the implementation, which are 1000, 2000, 3000 and 10000 strings respectively. Subsequently, these subdomains will be combined with the targeted website (in our experiment, we use five university official domains as targeted websites), and verify the alive rate by using the command *"dig"*. The ratio of successful alive subdomains has significantly improved with the increasing length of the final generated subdomain list.

In our approach, most of the generated data can be used as subdomain names. Compared to other tools that build their dictionary by using the Brute Force approach, the subdomain generation algorithm that is based on Char-RNN does not depend on the limitation of word composition in the existing dictionary. It supports a comprehensive index list, which not only covers the content of existing dictionaries, but also provides a prediction method to generate the extra content that does not currently exist in the dictionary by learning the existing subdomain names. This means, a generated word (string) that is a strange composition of various letters (an unknown word) from this model is not valueless, it remedies the limitations of the present dictionary, and this newly generated word may actually exist as a subdomain.

The Evaluation of Auto-Detection System

In our Auto-Detection system, the dictionary we used is from the subdomain enumeration tool SubDomainsBrute[2]. The subdomains are combined with targeted domains through traversing this dictionary. All subdomains that involve the potential takeover risk will be shown to the user after executing this system. So far there is not an available interface on any Domain Name vendor to allow a user to register the relevant domain. Therefore, the user has to determine for themselves whether this risky domain is available online.

For example, in the *stafocus.com* domain, the risky subdomains are identified to be *edu.stafocus.com*, *education.stafocus.com* and *vpn.stafocus.com*. In the domain of *ly1026.me*, the risky subdomain is identified to be *edu.ly1026.me*. All of their final CNAME destination domains in this experiment are *allways-cam.com* which does not have an A record. However, whether or not this domain is registered on a Name vendor, it demonstrates a weakness. If this domain is available, as shown in Figure 15 below, then we can confirm the corresponding (initial) subdomain has the takeover risk.

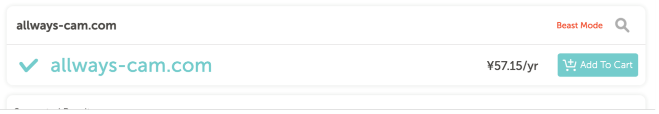


Fig. 15. The search for the domain *allways-cam.com* in Namecheap.

Compared to other subdomain enumeration tools, most tools provide a full discovery when gathering existing subdomains, but it is hard to identify which subdomains have potential risks. In our auto-system, we provide a comprehensive procedure to find the subdomains that may contain a potential takeover risk, although the final step in confirming the availability of

[2]Available at:https://github.com/lijiejie/subDomainsBrute

the domain requires us to search corresponding domain name buying websites by ourselves. Initially, the time consumption is large although it can be improved through techniques such as parallel computing. Also, the accuracy of our results is still subject to the integrity of the dictionary and we attempted to improve this by using the Char-RNN model in the querying step. This will be enhanced in future versions.

VI. Conclusion

Phishing attacks are one of the major cyber threats, either to an individual user or organization. For the individual, sensitive credentials are always of interest to hackers, in particular, due to the development of E-commerce. For an enterprise, a successful phishing attack, such as a subdomain takeover attack, may affect their organization's reputation as well as cause financial loss.

A successful subdomain takeover attack has a higher threat level as a controllable subdomain owns the same SSL certificate with its parent website, though it does not require an advanced technical skill to exploit this weakness. In this paper, two techniques have been presented as potential solutions. One is a query approach based on machine learning for querying existing subdomains and the second is an auto-detection system to identify the potentially risky subdomains. Although both techniques have limitations, they are still viable for real world applications, and the corresponding limitations can be improved upon.

The presented subdomain querying approach is based on a Char-RNN neural net model. The final generated result includes a comprehensive subdomain name content list, which not only almost covers the content of an existing dictionary, but also provides a prediction method to generate more extraneous content. In a future version, in order to improve the accuracy of the associated dictionary, the classification of industry naming conventions regarding potential targeted domains may need to be considered, because there are a variety of subdomain names under different industries.

In an auto-detection system, parallel or cloud-based computing could be implemented to reduce time consumption in the entire procedure. Also, an accurate dictionary that is generated from the Char-RNN model could be used to replace existing dictionaries, so that the system could provide a more sophisticated and accurate detection.

VII. Acknowledgments

This research is supported by China Scholarship Council (CSC), China; School of Computer Science, University of St Andrews, United Kingdom; and Fuying Lab, NSFOCUS Technologies Group Co., Ltd, China.

References

[1] K. Nirmal, B. Janet, and R. Kumar, "Phishing - The threat that still exists," in Proceedings of the International Conference on Computing and Communications Technologies, ICCCT 2015, 2015, pp. 139143.

[2] M. Thaker, M. Parikh, P. Shetty, V. Neogi, and S. Jaswal, "Detecting Phishing Websites using Data Mining," in Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology, ICECA 2018, 2018, pp. 18761879.

[3] M. Rosenthal, "Phishing Statistics (Updated 2021) — 50+ Important Phishing Stats — Tessian," 2021. [Online]. Available:https://www.tessian.com/blog/phishing-statistics-2020/. [Accessed: 03-Mar-2021].

[4] Y. Wang and I. Duncan, "A novel method to prevent phishing by using OCR technology," in 2019 International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2019, 2019.

[5] Y. Wang, Y. Liu, T. Wu, and I. Duncan, "A Cost-Effective OCR Implementation to Prevent Phishing on Mobile Platforms," in International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2020, 2020.

[6] Y. Wang, Y. Liu, T. Wu, and I. Duncan, "ISP Hijacking protection via DOM tree comparison," 2020.

[7] K. Borgolte, T. Fiebig, S. Hao, C. Kruegel, and G. Vigna, "Cloud Strife: Mitigating the Security Risks of Domain-Validated Certificates," in Proceedings of the Applied Networking Research Workshop, 2018, vol. 18.

[8] D. Liu, S. Hao, and H. Wang, "All your DNS records point to us: Understanding the security threats of dangling DNS records," in Proceedings of the ACM Conference on Computer and Communications Security, 2016, vol. 24-28-October-2016, pp. 14141425.

[9] M. Squarcina et al., "Can I Take Your Subdomain? Exploring Related-Domain Attacks in the Modern Web," 2020.

[10] I. Modin, "hackerone-reports/ GitHub," 2021. [Online]. Available: https://github.com/reddelexc/hackerone-reports/blob/master/tops_by_bug_type/TOPSUBDOMAINTAKEOVER.md. [Accessed: 03-Mar-2021].

[11] P. Hudak, "#325336 Subdomain takeover on svcgate-wayus.starbucks.com," 2018. [Online]. Available: https://hackerone.com/reports/325336. [Accessed: 03-Mar-2021].

[12] P. Hudak, "#570651 Subdomain takeover of mydailydev.starbucks.com," 2019. [Online]. Available: https://hackerone.com/reports/570651. [Accessed: 03-Mar-2021].

[13] P. Zenker, "#665398 Subdomain takeover of datacafe-cert.starbucks.com," 2019. [Online]. Available:https://hackerone.com/reports/665398. [Accessed: 03-Mar-2021].

[14] P. Hudak, "Subdomain Takeover: Basics." [Online]. Available: https://0xpatrik.com/subdomain-takeover-basics/. [Accessed: 03-Mar-2021].

[15] E. Borges, "Top 7 Subdomain Scanner tools to find subdomains," 2019. [Online]. Available: https://securitytrails.com/blog/subdomain-scanner-find-subdomains. [Accessed: 03-Mar-2021].

[16] S. M. Z. U. Rashid, M. I. Kamrul, and A. Islam, "Understanding the Security Threats of Esoteric Subdomain Takeover and Prevention Scheme," in 2nd International Conference on Electrical, Computer and Communication Engineering, ECCE 2019, 2019.

[17] G. Jaspher et al., "COMPARATIVE ANALYSIS OF SUBDOMAIN ENUMERATION TOOLS AND STATIC CODE ANALYSIS," J. Mech. Cont.& Math. Sci, vol. 15, no. 6, pp. 158173, 2020.

[18] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. E. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," Heliyon, vol. 4, no. 11. Elsevier Ltd, p. e00938, 01-Nov-2018.

[19] C. Olah, "Understanding LSTM Networks," 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed: 03-Mar-2021].

[20] N. Leonard, "Language modeling a billion words," 2016. [Online]. Available: http://torch.ch/blog/2016/07/25/nce.html. [Accessed: 03-Mar-2021].

[21] J. Ling, "OneForAll/ GitHub," 2021. [Online]. Available:https://github.com/shmilylty/OneForAll/blob/master/docs/en-us/README.md. [Accessed: 03-Mar-2021].

[22] P. Cerda, G. Varoquaux, and B. Kgl, "Similarity encoding for learning with dirty categorical variables," 2018.