applied sciences

MDPI

Article

Providing Email Privacy by Preventing Webmail from Loading Malicious XSS Payloads

Yong Fang, Yijia Xu, Peng Jia and Cheng Huang *

College of Cybersecurity, Sichuan University, Chengdu 610065, China; yfang@scu.edu.cn (Y.F.); whiterabbitxyj@gmail.com (Y.X.); pengjia@scu.edu.cn (P.J.)
* Correspondence: codesec@scu.edu.cn

Received: 29 May 2020; Accepted: 24 June 2020; Published: 27 June 2020

Abstract: With the development of internet technology, email has become the formal communication method in modern society. Email often contains a large amount of personal privacy information, possible business agreements, and sensitive attachments, which make emails a good target for hackers. One of the most common attack method used by hackers is email XSS (Cross-site scripting). Through exploiting XSS vulnerabilities, hackers can steal identities, logging into the victim's mailbox and stealing content directly. Therefore, this paper proposes an email XSS detection model based on deep learning technology, which can identify whether the XSS payload is carried in the email or not. Firstly, the model could extract the Sender, Receiver, Subject, Content, Attachment field information from the original email. Secondly, the email XSS corpus is formed after data processing. The Word2Vec algorithm is introduced to train the corpus and extract features for each email sample. Finally, the model uses the Bidirectional-RNN algorithm and Attention mechanism to train the email XSS detection model. In the experiment, the AUC (area under curve) value of the Bidirectional-RNN model reached 0.9979. When the Attention mechanism was added, the accuracy upper limit of the Bidirectional-RNN model was raised to 0.9936, and the loss value was reduced to 0.03.

Keywords: Webmail; XSS; Word2Vec; deep learning; Attention mechanism

1. Introduction

The development of Internet has brought people closer together. Using email to deliver information has become one of the most common methods for communicating with others. Even large-scale enterprises commonly use Webmail to communicate between their employees. Due to the enormous daily workload and communication demands, a company will generate thousands of work-related emails each day. To meet the business demands, Webmail architecture has been optimized for enterprises since as early as 2007 [1].

After collecting information from enterprises, we found a phenomenon that is at risk. Many companies put too much trust in their internal framework and employees, so their network systems are often very simple and lack security [2]. The isolation of the intranet and the loyalty of employees help businesses believe that it is not worthwhile to invest in network security construction. But the reality demonstrates that networks security is needed. Actually, enterprises have experienced negative retaliation from a departing employee. There are many ways for hackers to enter the intranet [3,4], such as a USB-Stick with viruses, social engineering, VPN, and so on. What is even worse, when hackers enter the company's internal network, it is easy for them to find network vulnerabilities.

In fact, many enterprises have already experienced attacks on their Webmail systems. Due to the differences in business needs, the internal systems of a variety of enterprises face disparities. But the email system is an essential part of any enterprises framework. Business emails contain a wealth of information about work plans, personal privacy, corporate business, and confidential

attachments. Which makes the Webmail system extremely attractive for information theft [5]. The most common methods for email information stealing are phishing and XSS (Cross-Site Scripting) attacks. When a hacker successfully infects a host or connects to the intranet, he can start attacking from one node and send attack-emails to other users. The phishing email exploits the trust among the employees. As long as the employee's security awareness is improved, the email phishing attack can be effectively avoided. In fact, the phishing email detection is already an advanced technology, and many researchers have published the latest research results [6–8]. Therefore, for Webmail, the XSS email attack is still the most difficult way to be defended.

Although XSS vulnerabilities are easy to exploit, defenders are constantly improving their detection methods. In the early days, simple regular expression matching was used to identify XSS payloads. However, defenders found it is difficult to describe complex XSS payloads just by regular expression. Therefore, they began to build the XSS attack "sample library", which repairs the defect of regular expression matching. These two methods can resist most known XSS attacks, but still cannot predict for new XSS attack methods. To identify unknown XSS attacks, researchers began to attempt with machine learning, such as Support Vector Machine (SVM), Random Forest, Long Short Term Memory (LSTM), and so forth. After analyzing many researchers' experiment, we found XSS detection is more inclined to natural language processing, so deep learning often performs better than traditional machine learning methods.

Most mailbox security detections are focused on filtering junk emails and preventing attachment virus transmission. But they do not realize that the email system has become a severely affected area of the XSS vulnerability. Therefore, this paper applies XSS detection technology to email security detection, and proposes a method of email XSS detection based on deep learning.

With the development of Webmail and the growing demand, many Webmails have been open to the Internet, which can be accessed by all users. These Webmails tend to be more secure and stable within the framework. In this paper, we used these Webmails as experimental objects. Using their emails as training samples, we tried to construct a deep learning model with the Attention mechanism. The model could identify enterprise emails that contain malicious XSS payloads. This paper made the following contributions:

- Collected and organized the email XSS dataset. Researchers often used general-purpose XSS datasets for machine learning training. The detection of these models were comprehensive, but the accuracy in the specific scenario were not high. For Webmail application scenario, there is not any public email XSS dataset at present. Therefore, we decided to collect the real research data for the experimental dataset. We spent two months collecting email XSS vulnerabilities datasets that were publicly available on the Wooyun community (a vulnerability submission platform). The collected dataset covers more than 20 well-known Webmail service institutes and will be published for other research.
- Built an effective email XSS detection framework. The paper constructed the entire email XSS detection framework which included data processing, feature extraction, and deep learning detection. The framework is a hierarchical structure that ensures the email XSS detection is running efficiently. The experiment showed an accuracy value could rise to 0.9936 with loss value 0.03.
- Added Attention mechanism to the model. We introduce Bidirectional-RNN (Bidirectional Recurrent Neural Network) algorithms to detect email XSS attack and used the Attention mechanism to improve the training effect of the model. This is the first effective model in the field of Webmail XSS detection.

The rest of the paper is organized as follows. In the Section 2, the research results are introduced on Webmail XSS in recent years. In Section 3, the basic principles of Webmail XSS attacks and various attack methods are discussed. In Section 4, the framework of enterprise email XSS detection is explored. The dataset and two comparison experiments are detailed in Section 5. In Section 6 is the summary of the work.

2. Related Work

Webmail XSS is a headache for the emails of major companies. In the past decade, almost all online Webmails have found XSS vulnerabilities. For example, Yahoo Webmail has exposed to storage XSS vulnerabilities twice in 2013 and 2016 [9], resulting in nearly 40 million users facing the risk of privacy breaches. Gmail was also exposed to XSS vulnerabilities many times between 2012 and 2014 [10]. Although they were repaired in time, it still caused a need for concern.

As a subject of valuable research, XSS attack detection has achieved many results in recent years. Initially, most of XSS attack detection research was regular expression matching. In 2014, Kozik Rafal proposed a regular expressions detection method for HTTP requests [11], which focus on detecting SQLIA (SQL Injection Attacks) and XSS (Cross Site Scripting). In the experiment, the model can effectively run in benchmark CISC'10 database. In 2017, Zalbina proposed a new payload recognition method [12], which investigated the XSS attack detection using regular expression pattern matching and a preprocessing method.

Zalbina built a security system (named Snort NIDC) that effective enough to detect and recognize the payload of XSS Attack, but he concluded that the system can only sufficient as the first defense barrier in the future. Therefore, the regular expression matching method has great limitations, and detection rate for complex or unknown XSS payloads is very low. To make up defects of regular matching, the industry established and expand XSS payloads sample libraries, but researchers began to use machine learning to solve problems.

In 2019, Chen's team proposed an XSS attack detection framework based on machine learning [13]. They experimented with a variety of traditional machine learning algorithms, such as support vector machine, decision tree, naive bayesian and others. They proved that machine learning models can effectively identify XSS payloads. At the end of the article, Chen found that if the word2vec tool was used to retain the semantic relationship between the parts of the attack statement, the deep learning model had better results, and the recall rate reached 97.9%. Besides, Fang's team proposed a XSS attack detection method based on deep learning, which named DeepXSS [14]. They used word2vec to extract payload features, and then used the LSTM algorithm to build the model. In the experiment, the recall of the model reached 0.98, which mean it had great ability to discovered XSS payload. The ideas and methods of Fang's paper had great reference value. However, it only performed simulation detection on the traditional XSS dataset, and did not research the actual data of specific Web applications in depth. Therefore, we paied more attention to XSS attack detection of Webmail and tried to optimize the deep learning model.

China's Tang proposed a Webmail-based XSS finder called L-WMxD [15]. It used a seed and syntax parser to construct numerous test cases in a form similar to fuzz. He inserted the constructed unique XSS statement into webmail to perform the attack, in anticipation of discovering XSS before the application's online service. In the experimental tests, vulnerabilities were discovered in 21 Webmail servers, including large companies such as Yahoo, Mirapoint, and Collaboration. The paper highlighted that most Webmails have had XSS vulnerabilities, especially those with weaker security. Regrettably, the model proposed in the article was more aggressive but did not provide more constructive advice on defensiveness. In the latest research, Xin Wang's team proposed a vulnerability detection technology for the Webmail systems [16]. In the paper, it was considered that the XSS vulnerability is the most common and serious vulnerability in the current email systems. For XSS vulnerabilities, XinWang analyzed the possible injection points in Webmail, which provided theoretical support for our research. In the experiment, they built malicious code through the vulnerability generator and implanted emails to detect XSS vulnerabilities. Eventually, they found a system vulnerability in Tom and Winmail that confirmed the availability of the technology. In the paper published by Ankit Shrivastava, he detailed how to steal Webmail users cookies for identity theft through a reflective XSS vulnerability [17]. Jun Yang's team used a vocabulary-based fuzzy framework to construct a mutated XSS attack vectors [18]. Under the test, there are XSS vulnerabilities available in various Webmail editors.

With the continuous research of deep learning algorithms, Attention mechanism was popular and used to optimize deep learning models. Therefore, researchers gradually tried the structure of deep learning plus Attention mechanism to detect network content security. In 2018, Yu's team proposed an Attention-Based Bi-LSTM (Bidirectional Long Short-Term Memory) model for anomalous HTTP traffic detection [19]. They used Attention mechanisms to help detect abnormal traffic and discover key parts of abnormal traffic. In 2019, Agrawal used Attention mechanism in recurrent neural networks for ransomware detection [20]. He presented specialized recurrent neural networks for capturing local event patterns in ransomware sequences using the concept of Attention mechanism. What's more, Liu's team proposed a real-time Web attack detection via attention-based deep neural network in 2019 [21]. They found the noise induced by irrelevant background strings can be largely eliminated when using Attention mechanisms. Besides, their system can greatly ignore large irrelevant URL content. By analyzing these researches, we believed that the Attention mechanism can be used for email XSS detection.

Most papers are devoted to the exploitation or detection of these vulnerabilities, but the existence of vulnerabilities is difficult to avoid and there is no absolute secure method. The XSS vulnerability discovered by various research can only temporarily find the vulnerabilities in the Webmail system, but cannot provide long-term protection. Therefore, this paper proposes an email XSS detection model based on deep learning and Attention mechanism, which aims to provide a long-term XSS Webmail detection solution for emails.

3. Webmail XSS

3.1. Principle

The XSS (Cross-Site Scripting) is an attack method that injects malicious script codes into web pages viewed by users, which can control the behavior of the user's browser. Hackers often use XSS attacks to steal user's cookies for identity theft. In theory, websites with XSS vulnerabilities can be executed with all front-end code functions. For example, executable functions include keyboard click records, screen captures, page source code, phishing pages, and others. For the XSS attack of Webmail, attackers tend to obtain the login credentials of the victim and use it to login to Webmails and steal personal information. In Webmail, a complete attack process is shown in Figure 1.

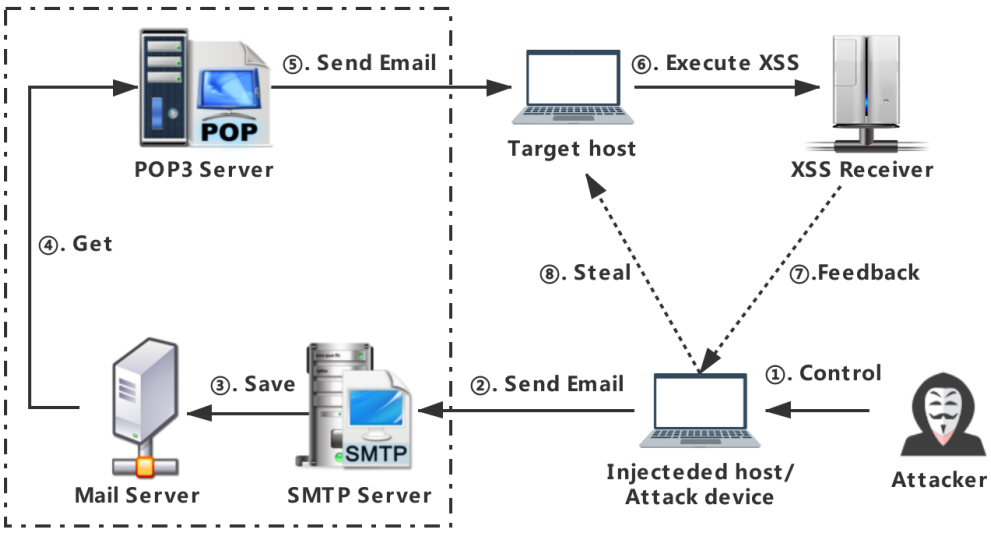

Figure 1. Email Cross-Site Scripting (XSS) attack principle.

In step 1, the attacker attempts to control a device through the intranet. The control method may be a virus infection or a device connection. Because Webmails have the ability to remember passwords or stay logged on for a prolonged session, an attacker may easily access the user's Webmail [22].

In step 2, the attacker edits the email with the XSS code and sends the email to the victim through SMTP Server. In step 3, Mail Server saves the malicious email. Actually, malicious email detection and prevention is usually deployed on Mail Server. In step 4, POP3 Server will obtain emails from Mail Server. Once the victim opens the malicious email on the browser side, the malicious XSS code will be released and the cookie will be sent to the XSS receiving server established by the attacker (steps 5–7). Lastly, the attacker will use the cookies to log in to the Webmail and steal information. If the Webmail is open to the external network, the attacker will be to construct the email directly to proceed the assault. However, relatively speaking, a Webmail opened to the external network will generally set more protective measures. The attacker's initiation to break the environment will be freer, but is often more restricted by the Webmail server.

3.2. Email XSS Injection Type

Referring to the research on Webmail XSS vulnerabilities [16], we concluded that there are several types of Webmail XSS injections. We concluded that there are several types of Webmail XSS injections. Table 1 shows the examples of various Email XSS injection types.

- Sender injection. If the Webmail is available for registration and its filtering is not stringent, an attacker might try to register a username or sender name that contains malicious code. When the victim loads the sender information on the website, the XSS code will be executed.
- Receiver injection. If the Webmail server does not carefully detect the receiver name, the attacker can attempt to construct an illegal receiver. This kind of vulnerability often appears in the email group-send function. If all receivers of the group are displayed on the victim's page, the XSS code will also be performed.
- Subject injection. The subject injection is the attack type in which hackers are most interested, because it will not only appear on the email browsing page but will also appear in the email list bar. Once the malicious XSS code is injected into the subject, the browser will directly execute the XSS code when loading the email or email list.
- Content injection. The content of the email can be inserted into images, links, and other controllable components, which makes the content most susceptible to the XSS vulnerabilities. If the email content editor is not designed properly, the attacker can directly edit the HTML code. Therefore, filtering the content of the email has always been a struggle for the designer, and is the injection point where the attacker spends the most time working.
- Attachment injection. Attachment injection is divided into the name of the attachment (named filename injection) and the attachment content injection (named attachment injection). Filename injection seeks to include the malicious XSS code in the name of the attachment and also attached to the email. If the attachment name is displayed directly on the web page, the XSS code will be executed. Attachment injection injects malicious XSS code into the content of the attached file [23]. If the Webmail has preloaded functions, which display part or all of the attachment content on the page, it will also trigger an XSS vulnerability.

Table 1. Examples for different Email XSS injection type.

Variant Type	Example
Sender injection	From: <"<iframe onload=alert(/xss/);>@attacker.com">
Reciever injection	To: victim1@victim.com;<img src=a onerror=alert("xss")>@victim.com
Subject injection	Subject: hello<script>alert("xss");</script>
Content injection	Content-type: text/html xss<img src="foo" alt="“onerror=alert("xss")">
Attachment injection	Content-Disposition: attachment; filename="<img src=x onerror=alert("xss")>a.jpg"

3.3. Attack Vector

The Webmail system sets basic defenses to resist XSS attacks [24]. Therefore, when conducting cross-site scripting attacks, some mutated attack vectors are usually used to launch these

attacks. We classify mutated XSS vectors into 4 categories—encoding, keyword substitution, position/morphology transformation, and special characters. Table 2 shows the example for these four categories.

- Encoding. The browser can actively decode and escape for some special circumstances, which allows hackers to use this feature to encode the attack vector. In this way, the original attack feature of the XSS vector during transmission can be effectively hidden. For example, HTML, Unicode, UTF-7 encoding for special tags or attributes.
- Keyword substitution. Many tags and keywords are blacklisted by various XSS detection software. Therefore, in order to obtain a better attack effect, it is necessary to replace sensitive words. For example, one could invent sensitive word substitution, "Alert" popup sensitive word replacement, white space replacement, parenthesis replacement, and label replacement for general events.
- Position/morphology transformation. Regular matching and detection models tend to be sequential and usually read strings from left to right. Therefore, the attribute position change or the attribute status change can be performed within the rules allowed by the script tag to achieve the purpose of avoidance detection. For example, the positional transformation of attribute expressions and event expressions, keyword case transformations, alerts, and other bullet-box string morphological changes.
- Special characters. An attacker can add special characters to the constructed XSS code that does not affect code execution but may disrupt the detector. For example, add an assignment expression before the event; add a blank character between the event and the trigger code; add any letter or number in the standard room; add any calculation expression between the function name; add a comment between the function name and the parenthesis; add before and after the attack vector to any character.

Table 2. Examples for different XSS Variant type.

Variant Type	Example
Encoding	%c0u003cimg+src%3d1+onerror%3dalert(/xss/)+%c0u003e
Keyword substitution	<applet code="javascript:confirm(document.cookie);">
Position/morphology transformation	<iMg onErRoR=prompt(1) sRc=1>
Special characters	<scri\x00pt>alert(1);</scri%00pt>

4. Email XSS Dection Framework

This paper proposes a Webmail XSS detection model based on deep learning. The basic framework is shown in Figure 2. At first, it collects the EML format less dedicated to the email. Secondly, the data for each key field will be extracted. The extracted data includes the sender, receiver, subject, content, and attachment. After that, the system performs feature extraction, which mainly provides input for the deep learning algorithm. In the end, the model trained by deep learning can be used to detect Webmail's XSS email.

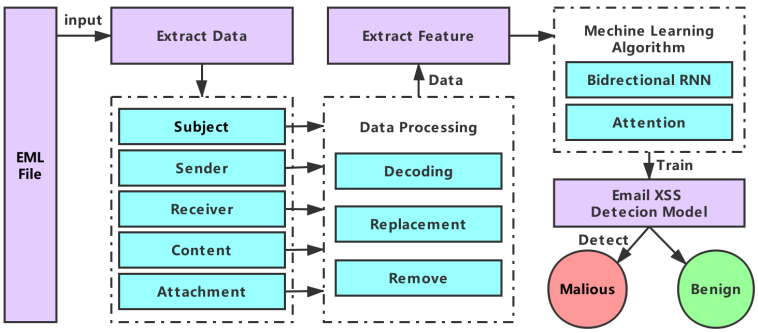

Figure 2. Webmail XSS detection framework.

4.1. EML Source File and Data Extract

The EML (Email) is a common format for all types of email software. We can think of the EML file as a file generated after the email is archived, retaining the original HTML format and title, and so forth. The basis of our detection system is the EML file, which provides us with large email-related information. Fortunately, most mail systems provide a window to download EML files directly. Besides, when the user login the mailbox client, the EML file is automatically downloaded to the local.

In order to get valuable data, the system will parse the original EML file. Each EML file has a standard format, which allows it to load by specified rules. The paper extracts the original sender, receiver, subject, content, and attachment information. In the EML file, some of the information is base64 encrypted, so it needs to be decrypted to get the rawest data. In the process of extracting, the email file that missing field values will be considered as abnormal data and be discarded.

4.2. Data Processing

4.2.1. Type

The data extracted directly from the EML file is the original text. As described in Section 3, attackers often bypass XSS detection by mutating it when constructing the XSS attack vectors. Therefore, it will be necessary to process the original text.

(1) Decoding. Encoding is the most commonly used form of XSS variation. The browser's automatic decoding capability allows many encoded XSS statements to bypass WAF (Web Application Firewall), but still, be automatically parsed. One needs to restore the attack vector to get the original XSS payload. We examined the collected email attack vectors and performed the following common decodings—URL decoding, HTML entity decoding, Unicode decoding, Base64 decoding and JSFuck decoding. The HTML Entity, URL and Unicode can be decode easily by decoding tools, since their decoding principle is mapping. Base64 is an encoding method that can encrypt information, and the encrypted data is completely unreadable. Since this encryption is reversible, we can use the Base64 decryption tool to decode. JSfuck encoding converts the string into a long text of only 6 symbols([]()!+), which can still be directly executed by the browser. The text after JSfuck encoding is like garbled code, so we must use open source tools Jsunfuckit to decode. The decoding example of the real samples are shown in Table 3.

Table 3. Decoding Example for XSS Payload.

	Source Code	After Decode
Html Entity	"><script src="http:// xss.tw/4091">	<script src="http://xss.tw/4091">
URL	%253Cimg%2520src%253D1%2520onerror %253Dprompt%25281%2529%253E	<img src=1 onerror=prompt(1)>
Unicode	<DIV STYLE="background-image:\0075\0072\ 006c\0028\006a\0061\0076\0061\0073\0063 \0072\0069\0070\0074\003a\0061\006c\0065 \0072\0074\0028\0027\0058\0053\0053\005f \0030\0030\0027\0029\0029">	<DIV STYLE="background-image: url(javascript:alert('XSS_00'))">
Base64	<iframe src="dat
a: text/html;base64,PHNjcmlwdD5hbGVyd CgxKTwvc2NyaXB0Pg==">	<iframe src="dat
a:text/ html;base64,<script>alert(1)</script>">
JSFuck	[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+ (![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[] +!+[]+!+[]]+(!![]+[])[+!+[]]][([][(![]+[])[+[]]+ ([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+ (!![]+[])[+[]]+(!..... (show a part)	alert(4)

(2) Equivalent replacement. Direct replacement of keywords actually has less impact on model training, such as label and attribute replacement. When checking the email attack sample, we found some replacements that were more disturbing to the model, which required some equivalent replacement. We mainly performed equivalent replacement for the following situations: alphabet upper/lowercase, Full-angle Character and fromCharCode functions. The examples of equivalent replacement for the real sample as shown in Table 4.

Table 4. Equivalent Replacement Example for XSS Payload.

	Source Code	After Replacement
Alphabet upper/lower	<DIv StYLe="x:exPreSsiOn(alErt(1))"> IE6</dIV>	IE6
String.from CharCode		

(3) Remove disturbing characters. Attackers often add some disturbing characters to the basic attack vector to confuse the detector's judgment. For example, if the detector reads the text in rows to detect the XSS attack, the attacker can bypass the detection by using a newline character which can block the attack vector. We checked the real email attack samples and deleted several types of disturbing characters. Some of disturbing characters is shown in Table 5.

Table 5. Remove Disturbing Characters.

Disturbing Characters
\n; \\00; %00; /**/; NewLine; ...

4.2.2. Procedure

The attack vector may exist in any input area, so we should process the text of all fields. We combined all fields' data into one long string and then process it. Because there may be multiple variations in an attack vector, all types of processing are used for the text. However, in the actual processing, it was found that the processing sequence is very important. For example, if we delete the "\00" character first, then the Unicode format is destroyed and cannot be decoded. After a period of trials, the step summarized is—(1) Alphabet upper to lower; (2) Html Entity Decode; (3) URL Decode; (4) Unicode Decode;(5) Base64 Decode; (6) StringFromCode Decode; (7) JSFuck Decode; (8) All-angle Decode; (9) Remove disturbing characters.

4.3. Exact Feature

To train the model through machine learning, one converts the email sample into a computer-readable digital format. The usual approach is to extract statistical features or word vectorization. Under our observation, we found that statistical features cannot describe well the differences between benign and malicious email samples. Since only a small piece of the email text contains the malicious code, the difference in statistical characteristics between benign and malicious email samples was few. Therefore, we used the word vectorization method to extract the feature of each email text, which paid more attention to the association between words.

Word2Vec is a tool published by Google in 2013, which is specifically used for word vector computing. Word2Vec has two major advantages: it can run efficiently in a million-level corpus; the word vector generated by training can also measure the relationship between words. Therefore, we decided to use Word2Vec as a feature extraction tool for our email samples.

First, we formed a corpus of processed benign and malicious samples. Then, we used Word2Vec to train in the corpus and get the word vector for each word. Referring to Jingchi Zhang's research on XSS detection [25], we set the word vector dimension of Word2Vec to 200. The word vector of the "img" is shown in Figure 3.

```
img:

[0.20706260   0.68060790  0.18821543  0.2935011   0.24210198  0.31016836
−0.41839656  0.06610132 −0.25400597 0.03141395 −0.3452557  0.19232723
−0.00566868  0.27681908  0.24117452 −0.02407036 −0.23219763 −0.14390673
 0.04069300  0.01226758 −0.18251793 −0.24230677 −0.04116664 −0.23572175
 ...
 0.12812933  0.07698229  0.00360328 −0.49380810 0.2380006   0.10403243
 0.2515339   0.14449877  0.44496915  0.16371286 −0.34831035 −0.11835948
−0.07838395  0.20522314  0.03352085 −0.5094921 −0.09658884 −0.18881974
 0.41346255 −0.09298827 0.10227906 −0.4553482 −0.18240954 −0.15324433
−0.02053374  0.49063563]
```

Figure 3. The word vector of "img" training by Word2Vec.

Each word vector represents a feature of a word in the corpus. Since each word has the same dimensional features, we calculated the average of all word vectors in the sample as a final feature of each email sample. Lastly, the sample features are passed as input to the deep learning algorithm.

4.4. Deep Learning Algorithm

4.4.1. Bidirectional-RNN

The RNN (Recurrent Neural Network) is a deep learning algorithm that can be modeled in time series [26]. The characteristic of RNN is the output of the neuron can be passed to itself in the next moment, which breaks the restriction that the neuron can only be transmitted up and down. Bidirectional-RNN adds a reverse transfer mechanism based on the original RNN [27]. In other words, the Bidirectional-RNN not only considers the above impact on the following but also considers the following impact on the above. This makes the Bidirectional-RNN more comprehensive in predicting. Bidirectional-RNN is often used to train language models, which has been well used in the field of natural language processing [28]. Therefore, the paper uses the Bidirectional-RNN algorithm to train the email XSS detection model.

4.4.2. Attention

Attention is an effect-enhancing mechanism based on the RNN model, which named Attention Mechanical [29]. At present, the Attention mechanism is very popular because it gives the model discriminating power. When people observe something, they usually do not look at it as a whole. They tend to observe the important parts of things first, then connect the information of different parts in the series, and finally form the overall impression of the thing. The Attention mechanism refers to the characteristics of human observation. The Attention mechanism can also help the model assign different weights to each part of the input X, extract more important information, which makes the model have more accurate predictions without incurring more computational and storage overhead. In this paper, the global Attention is used to optimize the Bidirectional-RNN model. The basic principle is shown in Figure 4.

In the Attention mechanism, all hidden states are used to calculate the weight of the Context vector. For a variable length alignment vector a_t, the actual length is equal to the length of the input statement in the encoder. At time t, the value of a_t is first calculated according to two hidden states (h_t and h_s). The calculation formula is following:

$$a_t(s) = align(h_t, h'_s) = \frac{exp(score(h_t, h'_s))}{\sum_{s'} exp(score(h_t, h'_{s'}))}. \quad (1)$$

The score is used to evaluate the correlation function between h_t and h_s, and also used as an alignment function. Its calculation formula is as follows:

$$score(h_t, h'_s) = \begin{cases} h_t^\top h'_s, & dot \\ h_t^\top W_a h'_s, & general \\ v_a^\top tanh(W_a[h_t; h'_s]), & concat \end{cases} \quad (2)$$

After the alignment vector a_t is calculated, the context vector c_t can be calculated averaging. An email often contains too much information, and in fact, the effective XSS payload is only a small part. When training email samples, it is only a small part of the information that contains the XSS payload has training value, and the rest are low-value text. Therefore, using the Attention mechanism can make the model more focused on the effective XSS payload in the email, rather than treating all texts equally.

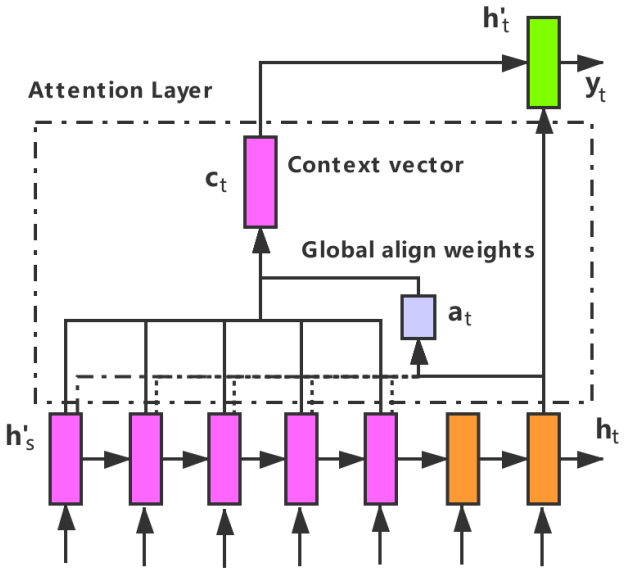

Figure 4. Basic principle of Attention Mechanism.

5. Experiment

5.1. Environment

The experiments designed in this article are all running on the Windows platform. All models are developed in the context of Python 3.6. All the involved Python libraries and their versions are shown in Table 6. The Jsunfuckit library is used to decode JSFuck code, which references the open-source project of clams4shoes on Github [30,31]. The Attention library is used to add the attention mechanism to deep learning, which refers to philipperemy's open source project on Github [32].

Table 6. Experiment Environment.

Module	Library and Version
Mail processing	Email (bulit-in) Numpy == 1.14.0 Pandas == 0.23.4
Data processing	Base64 (bulit-in) Urllib == 1.23.0 Html (bulit-in) Re (bulit-in) Jsunfuckit (Third party
Feature extraction	Gensim == 3.6.0
Traditional machine learning	Scikit-learn == 0.19.1
Deep learning	Tensorflow == 1.3.0 Keras == 2.1.2 Attention (Third party)

5.2. Dataset

5.2.1. Malicious sample

Since there is no mature experimental dataset in the email XSS detection field, we have to spend a lot of time collecting malicious samples. Thanks to China's WooYun Database [33], we could collect XSS vulnerabilities in China's prevalent Webmail since 2012. After search engine filtering, we got 245 Webmail XSS vulnerabilities that were submitted to the WooYun, and they all have detailed instructions, screenshots, and attack payloads.

It took us nearly a month to sort and clean the email XSS records. We removed some XSS vulnerabilities in the Webmail's own function, such as XSS vulnerabilities in login and registration fields, Notepad XSS vulnerabilities, personal information XSS vulnerabilities, and so forth. We checked each Webmail XSS record one by one to determine that it is an XSS vulnerability generated by the email, and the Webmail manufacturer verified that the XSS vulnerability does exist. In order to protect the anonymity of the white-hat hacker that submitted the vulnerability, we format all the attacker's mailbox as "attacker@attacker.com" and format all the victim's mailbox as "victim@(mail).com" (except the XSS attack record occur in the sender/receiver).

Since WooYun was shut down in 2016, we could only collect Webmaill XSS vulnerabilities that were disclosed between 2012 and 2016. Although the vulnerability platform paied more attention to the confidentiality of the vulnerability payload after WooYun, we still analyzed from some leaked vulnerability information that the variability of email XSS payload was little these years. After filtering and sorting, we got 168 EML mail samples carrying malicious XSS attack payloads. The malicious samples are all the Webmail XSS vulnerabilities confirmed by the enterprise from WooYun. We have pushed the compiled data on github [34], hoping to help future generations' research.

5.2.2. Benign Sample

To provide the benign samples for the experiment, we used the Enron Email Dataset for the experiment. The Enron Email Dataset is the most authoritative dataset in email research [25], which documenting Enron's 150 executive email contacts. We chose it because the Enron mail dataset provided a portion of the raw data, which is the standard EML mail format, so we can load and parse it easily.

Due to a large amount of data in the Enron Email Dataset, we randomly select and filtered 1000 email samples as a benign email sample. In order to avoid the model classification bias caused by the excessive gap between benign samples and malicious samples, we controlled the length of

the positive sample that less than 1000 characters, and it contained some network links and labels. That made our classification models achieve a better balance.

5.2.3. Dataset and analysis

We integrated the malicious and benign samples, and put them into data processing module. After processing, we used word2vec to train the corpus and then extracted features to form the experimental dataset.

Analyzing the malicious sample of our experimental dataset, we discovered some interesting conclusions. The number of Webmail XSS samples contributed by enterprises is shown in Table 7. We found that qq Webmail provided the most samples, which is 19; The sina Webmail provided 16 samples; Tom, 21cn, 139mail, 189mail, sohu, 163mail, wo Webmail have provided more than 10 samples. It warns us that even Internet enterprises who focus on cybersecurity still have so many XSS vulnerabilities.

Table 7. Malicious Sample Number for Webmail.

Webmail	Sample Number
qq	19
sina	16
tom	15
21cn	14
139mail	13
189mail	13
sohu	13
163mail	10
wo	10

We had statistics on the types of malicious samples, and the statistical results are shown in Figure 5. We found that most attacks are initiated in the content of the email. This is because the content is the body of the email and allows us to insert HTML tags, which makes it more possible to exist XSS vulnerabilities and easier to hide attack payloads. The Sender and Subject fields are also a part of attack payload. We observed the sample and found that only two cases occurred in the Receiver field, which both used the Webmail's own vulnerability, so we decided to ignore this field. The attachment is still noteworthy when design the Webmail system, and the number of attacks initiated by the filename and attachment has reached 20% in the malicious sample. If combining the time to analyze, we found that in the early days, most enterprises' Webmail XSS vulnerabilities are evenly distributed in various fields. As time passed and the bug is fixed, the XSS vulnerability is slowly concentrated in the content field. This is understandable because the filtering of other fields are easier than the content field.

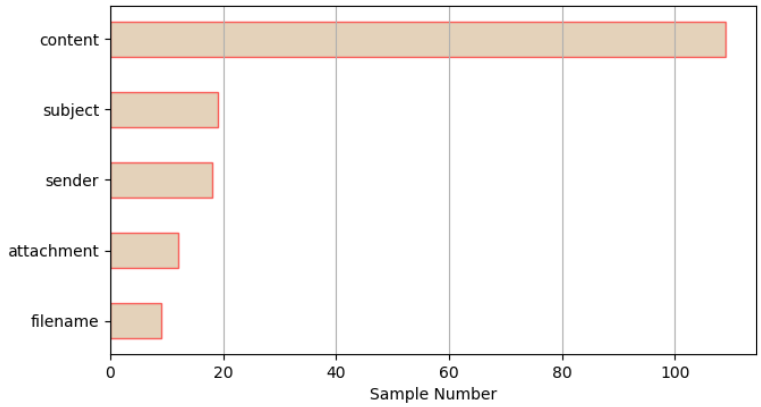

Figure 5. Field types in malicious samples.

For experiments, we divided the experimental dataset into a train-dataset and a test-dataset (6:4) by Scikt-learn tools. Due to the amount of data, it is not appropriate to separate the verification-dataset from the original dataset. Therefore, we used the 10-fold cross verification to verify the dataset. We used SVM (Support Vector Machine) as the verification algorithm and calculated the average value of 10-fold cross-validation. Finally, the accuracy obtained by verification reached 0.9968, which means the dataset has extremely high training value. In the first experiment, we compared the effects of deep learning (Bidirectional-RNN) and general machine learning. In the second experiment, we studied whether the Bidirectional-RNN model with attention mechanism preforms better.

5.3. Experiment and Result Analysis

In the first experiment, we used a Bidirectional-RNN to train the dataset and compare it with traditional machine learning algorithms. We set the hidden layer number of RNN to 64, the epochs to 200, the batch_size to 72, the loss function to "mae", the optimization function to "adam", and other parameters to default. After training, we used the test-dataset to evaluate the model, plotted the ROC curve and calculated the AUC value. We selected the SVM algorithm, GMM (Gaussian Mixed Model) algorithm, Naive Bayes algorithm, and Random Forest algorithm for comparison. The experimental results are shown in Figure 6.

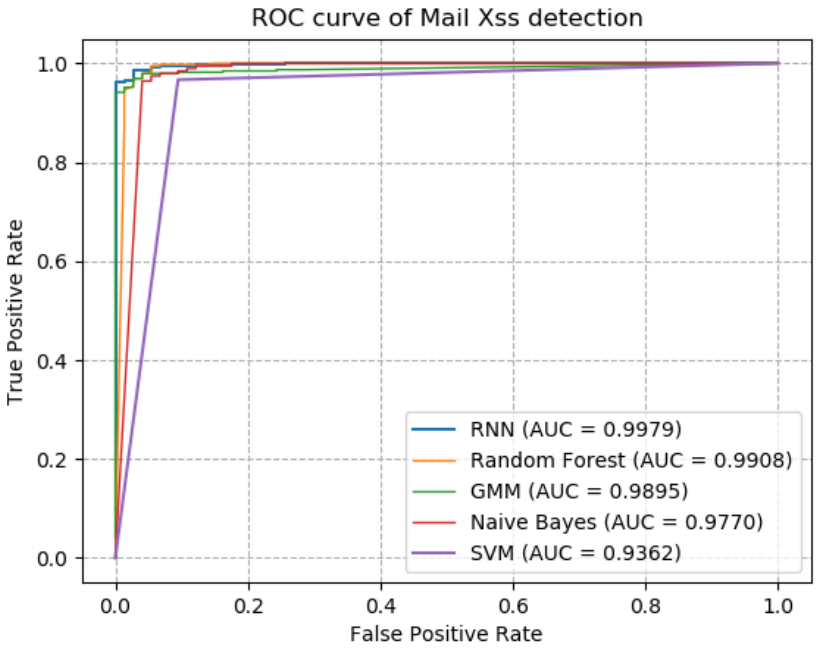

Figure 6. Bidirectional-Recurrent Neural Network (RNN) compared with traditional machine learning in ROC curve.

From Figure 6, we saw that the AUC value of the Bidirectional-RNN reaches 0.9979, which is much higher than other traditional machine learning algorithms. Only the random forest who is a multi-classifier can be similar to AUC value to the Bidirectional-RNN algorithm. It showed that the integrated learning algorithm and the deep learning algorithm were better than the traditional machine learning algorithm when training the email XSS dataset. This was because the deep learning algorithm and the random forest algorithm were more efficient in working under a high-dimensional dataset with higher accuracy. A more detailed assessment is shown in Table 8.

In the second experiment, we attempted to add the Attention mechanism to the Bidirectional-RNN algorithm to compare whether it can improve the performance of the original model. First, we use the Bidirectional-RNN algorithm to configure two identical neural networks. The parameters of network 1 are the same as the previous experiment, and network 2 adds a layer of Attention mechanism based

on network 1. We used all the collected email XSS dataset to train the two neural networks, and the training results are shown in Figure 7.

Table 8. More detailed evaluation for Bidirectional-RNN and other Machine learning algorithms.

	Precision	Recall	F1-Score
Bidirectional-RNN	0.99	0.98	0.99
Random Forest	0.98	0.94	0.96
GMM	0.93	0.90	0.91
Naive Bayes	0.83	0.94	0.88
SVM	0.84	0.86	0.85

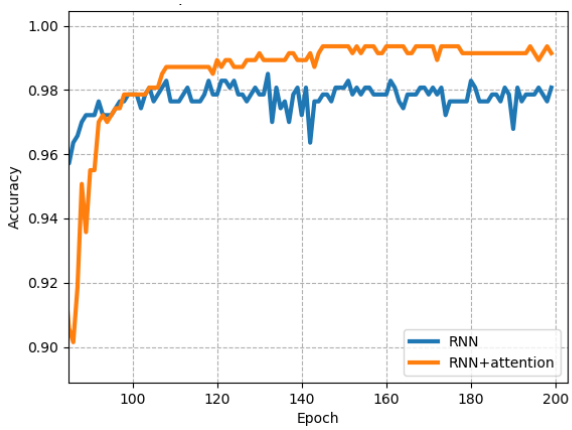

Figure 7. The accuracy comparison of RNN and RNN with Attention Mechanism.

It can be analyzed from Figure 7 that the accuracy of the two neural networks gradually stabilizes after the number of training epoch reaches 100. The highest accuracy of the neural network directly trained by the Bidirectional-RNN is up to 0.9828, and its average value is 0.98. The highest accurate value of the neural network added the attention mechanism is up to 0.9936, and its average value is 0.99. It shows that the Attention mechanism optimizes the process of Bidirectional-RNN training, increases its accuracy limit and has better performance. At the same time, we evaluated the loss of two neural networks during the training process, and the results are shown in Figure 8.

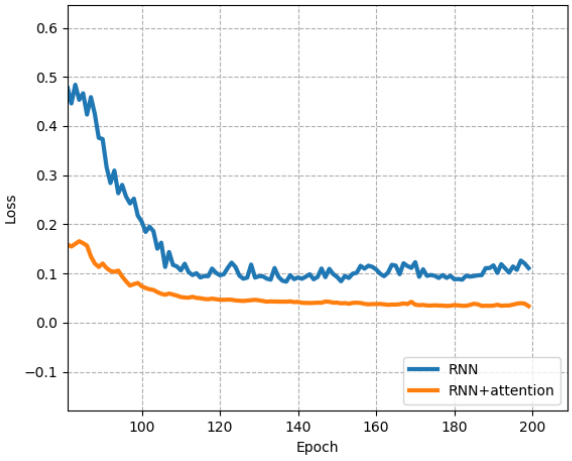

Figure 8. The loss comparison of RNN and RNN with Attention Mechanism.

It can be seen from Figure 8 that the loss of the two neural networks gradually stabilizes after the training epoch reaches 100. The loss of the neural network directly trained by the Bidirectional-RNN is about 0.1, and the loss of the neural network added to the Attention mechanism is about 0.03. It shows

that the Attention mechanism can effectively reduce its loss value during the Bidirectional-RNN training process.

In the experiments of Chen and Fang, the recall rate of the deep learning model reached 98%. Comparing with their experimental results, our experiments proved that the Bidirectional-RNN model is completely available for email XSS detection. After adding the Attention mechanism, the model was optimized, which had higher accuracy and lower loss than the original model.

6. Conclusions

This paper proposes an email XSS detection model based on deep learning. This model can be used for email XSS detection in enterprise intranet, or as a built-in email XSS attack payload identification in Webmail. The email XSS detection model establishes a neural network through a two-way RNN and uses the Attention mechanism to optimize performance. First, the paper gave an overview of the main principles and variants of Webmail attacks. Second, the paper elaborated on the email XSS detection framework proposed in this paper, which includes five levels of specific content. The main contribution of the paper is to collect and analyze the email XSS dataset and prove that the model proposed by the paper has excellent performance. In the experiment, the AUC value of the deep learning model reached 0.9799, which is superior to the traditional machine learning algorithm. After adding the Attention mechanism, the upper limit of the accuracy of the deep learning algorithm is improved, and the loss of the training process is effectively reduced.

Author Contributions: Conceptualization, Y.F. and C.H.; Methodology, C.H. and Y.X.; Software, Y.X. and P.J.; Validation, Y.X., P.J. and C.H.; Formal Analysis, P.J.; Investigation, Y.X.; Resources, C.H.; Data Curation, P.J.; Writing-Original Draft Preparation, Y.X.; Writing-Review & Editing, Y.F. and C.H.; Supervision, C.H.; Funding Acquisition, C.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by National Natural Science Foundation of China (61902265), National Key Research and Development Program (2018YFB0804503, 2019QY1405), and the Key Research and Development Plan Project of Sichuan Province (No. 2020YFG0374).

Acknowledgments: We thank those anonymous reviewers whose comments/suggestions helped improve and clarify this manuscript.

Conflicts of Interest: The authors declare that there is no conflict of interest.

References

1. Nusser, S.; Cerruti, J.; Wilcox, E.; Cousins, S.; Schoudt, J.; Sancho, S. Enabling efficient orienteering behavior in webmail clients. In Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology, Newport, RI, USA, 7–10 October 2007; pp. 139–148.
2. Zhang, Y.; Yang, Y. A secure webmail system based on FreeBSD UNIX. In Proceedings of the 2013 6th International Conference on Intelligent Networks and Intelligent Systems—ICINIS, Shenyang, China, 1–3 November 2013; pp. 212–214.
3. Liu, G.; Cui, X.; Wang, Z.; Wang, X.; Fang, Y.; Li, X. MaliceScript: A novel browser-based intranet threat. In Proceedings of the 2018 IEEE 3rd International Conference on Data Science in Cyberspace—DSC, Guangzhou, China, 18–21 June 2018; pp. 219–226.
4. Yang, C.; Hu, H.; Cheng, G. A software-defined intranet dynamic defense system. In Proceedings of the International Conference on Communication Technology—ICICT, Aachen, Germany, 27–29 September 2019; pp. 849–854.
5. Hameed, S.; Asif, M.; Khan, F. PiMail: Affordable, lightweight and energy-efficient private email infrastructure. In Proceedings of the 2015 11th International Conference on Innovations in Information Technology—IIT, Dubai, UAE, 1–3 November 2015; pp. 320–325.
6. Akinyelu, A.A. Machine Learning and Nature Inspired Based Phishing Detection: A Literature Survey. *Int. J. Artif. Intell. Tools* **2019**, *28*, 1930002. [CrossRef]
7. Wang, X.; Zhang, C.; Zheng, K. Detecting Spear-phishing Emails Based on Authentication. In Proceedings of the IEEE 4th International Conference on Computer and Communication Systems—ICCCS, Singapore, 23–25 February 2019; pp. 450–456.

8. Zhuorao, Y.; Chen, Q.; Wanling, K. Phishing Email Detection Based on Hybrid Features. *IoP Conf. Ser. Earth Environ. Sci.* **2018**, *252*, 042051.
9. Yahoo Mail stored XSS [EB/OL]. 2016. Available online: https://klikki.fi/adv/yahoo2.html (accessed on 14 May 2020).
10. Cross-Site-Scripting in Google Mail [EB/OL]. 2012. Available online: https://www.nilsjuenemann.de/2012/06/11/cross-site-scripting-in-google-mail-html/ (accessed on 14 May 2020).
11. Kozik, R.; Choraś, M.; Renk, R.; Hołubowicz, W. Modelling HTTP requests with regular expressions for detection of cyber attacks targeted at web applications. *Adv. Intell. Syst. Comput.* **2014**, *299*, 527–535.
12. Zalbina, M.; Septian, T.; Stiawan, D.; Idris M Heryanto, A.; Budiarto, R. Payload recognition and detection of Cross Site Scripting attack. In Proceedings of the 2nd International Conference on Anti-Cyber Crimes—ICACC, Abha, Asir, Saudi Arabia, 26–27 March 2017; pp. 172–176.
13. Chen, X.; Li, M.; Jiang, Y.; Sun, Y. A Comparison of Machine Learning Algorithms for Detecting XSS Attacks. In Proceedings of the Artificial Intelligence and Security—5th International Conference—ICAIS, New York, NY, USA, 26–28 July 2019; pp. 214–224.
14. Yong, F.; Yang, L.; Liang, L.; Cheng, H. DeepXSS: Cross Site Scripting Detection Based on Deep Learning. In Proceedings of the 2018 International Conference on Computing and Artificial Intelligence—ICCAI, Chengdu, China, 12–14 March 2018; pp. 47–51.
15. Tang, Z.; Zhu, H.; Cao, Z.; Zhao, S. L-WMxD: Lexical based Webmail XSS Discoverer. In Proceedings of the 2011 IEEE Conference on Computer Communications Workshops—INFOCOM WKSHPS, Shanghai, China, 10–15 April 2011; pp. 976–981.
16. Wang, X.; Wu, R.; Ma, J.; Long, G.; Han, J. Research on vulnerability detection technology for web mail system. In Proceedings of the International Conference of Information and Communication Technology—ICTC, Jeju Island, Korea, 17–19 October 2018; Volume 131, pp. 124–130.
17. Shrivastava, A.; Choudhary, S.; Kumar, A. XSS Vulnerability Assessment and Prevention in Web Application. In Proceedings of the 2016 2nd International Conference on Next Generation Computing Technologies, Dehradun, Uttrakhand, India, 14–16 October 2017; pp. 850–853.
18. Yang, J.; Tang, Q. RTF editor XSS fuzz framework. *Adv. Intell. Syst. Comput.* **2017**, *612*, 941–951.
19. Yu, Y.; Liu, G.; Yan, H.; Li, H.; Guan, H. Attention-Based Bi-LSTM Model for Anomalous HTTP Traffic Detection. In Proceedings of the 15th International Conference on Service Systems and Service Management—ICSSSM, Hang Zhou, China, 21–22 July 2018.
20. Agrawal, R.; Stokes, J.; Selvaraj, K.; Marinescu, M. Attention in Recurrent Neural Networks for Ransomware Detection. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing—ICASSP, Brighton, UK, 12–17 May 2019; pp. 3222–3226.
21. Liu, T.; Qi, Y.; Shi, L.; Yan, J. Locate-then-DetecT: Real-time web attack detection via attention-based deep neural networks. In Proceedings of the International Joint Conference on Artificial Intelligence—IJCAI, Macao, China, 10–16 August 2019; pp. 4725–4731.
22. Onaolapo, J.; Mariconti, E.; Stringhini, G. What happens after you are Pwnd: Understanding the use of leaked webmail credentials in the wild. In Proceedings of the ACM SIGCOMM Internet Measurement Conference—IMC, Los Angeles, CA, USA, 31 October–2 November 2016; pp. 65–79.
23. Cohen, C.; Hendler, D.; Rubin, A. Detection of malicious webmail attachments based on propagation patterns. *Knowl.-Based Syst.* **2018**, *141*, 67–79. [CrossRef]
24. Heiderich, M.; Spath, C.; Schwenk, J. DOMPurify: Client-side protection against XSS and markup injection. In Proceedings of the Computer Security—22nd European Symposium on Research in Computer Security—ESORICS, Oslo, Norway, 11–15 September 2017; pp. 116–134.
25. Jingchi, Z.; Yu-Tsern, J.; Xiangyang, L. Cross-Site Scripting (XSS) Detection Integrating Evidences in Multiple Stages. In Proceedings of the 52nd Hawaii International Conference on System Sciences—HICSS, Grand Wailea, Maui, 8–11 January 2019; pp. 7166–7175.
26. Sutskever, I.; Vinyals, O.; Le Quoc, V. Sequence to sequence learning with neural networks. In Proceedings of the 28th Annual Conference on Neural Information Processing Systems—NIPS, Montreal, QC, Canada, 8–13 December 2014; pp. 3104–3112.
27. Schuster, M.; Paliwal Kuldip, K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **1997**, *45*, 2673–2681. [CrossRef]

28. Kombrink, S.; Mikolov, T.; Karafiat, M.; Burget, L. Recurrent neural network based language modeling in meeting recognition. In Proceedings of the Annual Conference of the International Speech Communication Association—INTERSPEECH, Florence, Italy, 28–31 August 2011; pp. 2877–2880.
29. Bahdanau, D.; Cho, K.; Karafiat, M.; Bengio, Y. Neural machine translation by jointly learning to align and translate. In Proceedings of the 3rd International Conference on Learning Representations—ICLR, San Diego, CA, USA, 7–9 May 2015; pp. 2877–2880.
30. Wooyun Vulnerability Database [EB/OL]. 2016. Available online: https://shuimugan.com/ (accessed on 16 May 2020).
31. Encodes or Decodes Javascript obfuscated using JSFuck [EB/OL]. 2017. Available online: https://github.com/clams4shoes/jsunfuckit (accessed on 20 May 2020).
32. Attention mechanism Implementation for Keras. [EB/OL]. 2019. Available online: https://github.com/philipperemy/keras-attention-mechanism (accessed on 20 May 2020).
33. Wooyun Email XSS Dataset [EB/OL]. 2019. Available online: https://github.com/WhiteRabbitc/Wooyun-Email-XSS-Dataset/tree/master/malious-sample (accessed on 22 May 2020).
34. Enron Email Dataset [EB/OL]. 2015. Available online: https://www.cs.cmu.edu/~enron/ (accessed on 22 May 2020).