PassEye: Sniffing Your Password from HTTP Sessions by Deep Neural Network

Zhiqing Rui[1], Jingzheng Wu[1](✉), Yanjie Shao[1], Tianyue Luo[1], Mutian Yang[1,2], Yanjun Wu[1], and Bin Wu[1]

[1] Institute of Software, Chinese Academy of Sciences, Beijing, China
{zhiqing,jingzheng08,yanjie,tianyue,mutian,yanjun, wubin}@iscas.ac.cn

[2] Beijing ZhongKeWeiLan Technology, Beijing, China

Abstract. Passwords are the most widely used method for user authentication in HTTP websites. Password sniffing attacks are considered a common way to steal password. However, most existing methods have many deficiencies in versatility and automation, such as manual analysis, keyword matching, regular expression and SniffPass. In this paper, to better describe the problem, we propose a HTTP Sessions Password Sniffing (HSPS) attack model which is more suitable in HTTP environment. Furthermore, we propose PassEye, a novel deep neural networkbased implementation of HSPS attack. PassEye is a binary neural network classifier that learns features from the HTTP sessions and identifies Password Authentication Session (PAS). We collected 979,681 HTTP sessions from the HTTP and HTTPS websites for training the binary classifier. The results show that PassEye is effective in sniffing the passwords with an accuracy of 99.38%. In addition, several measures are provided to prevent HSPS attacks in the end.

Keywords: Password sniffing attack · Deep neural network · Website security · Network traffic analysis

1 Introduction

Password is a traditional identity authentication method [1]. However, this authentication method has many security problems, which has been criticized for a long time. Some more secure methods have been proposed for the same purpose, such as fingerprint, asymmetric key, 2-step verification, one-time password, but password is still the most widely used one due to its convenience, simplicity, and user habits. This gives attackers the opportunity to perform brute force attacks, password sniffing attacks and password reuse attacks. The widespread use of plain text password transmission and weakly encrypted password transmission in HTTP websites makes password sniffing attacks more easily.

This work was supported by National Key Research and Development Program of China (2017YFB0801900), National Natural Science Foundation of China (61772507) and the Key Research Program of Frontier Sciences, CAS (ZDBS-LY-JSC038).

Traditional methods of password sniffing attacks include manual analysis, keyword matching, regular expression and automatic tool [2, 3], which can attack some HTTP websites. Session is the basic unit of communication between the client and the server in the HTTP protocol [4] including request and response messages. HTTP websites usually perform password authentication through sessions. Because of the diversity of websites, manual analysis is probably the most common and effective measure. For examples, attackers listen to the network traffic packets and search for PAS, Keyword matching is also fast and effective, but experiments show that it has a high false-positive rate, and regular expression is an upgraded version of the former two. Attackers can write several regular expressions to match the PAS of some websites. However, writing regular expressions for all websites is an impossible task. Therefore, some automatic password sniffing tools have been proposed, e.g., SniffPass [5] and Password Sniffer Spy [6]. These tools support some protocols, such as POP3, IMAP4, SMTP, FTP, and HTTP Basic authentication, and do not support password authentication in HTTP webpage form, resulting in low availability in HTTP website password sniffing attacks. Overall, the current methods have many deficiencies in terms of versatility and automation.

Currently, more and more websites use HTTPS protocol to protect the security of data transmission and prevent man-in-the-middle attacks, thereby greatly enhancing the security of websites. However, since the user may try to install the root certificate in the web browser due to the temptation of the attacker or the request of the network administrator, the attacker can track the user's web browsing request by setting a transparent proxy server. In this paper, we propose an HSPS attack model if an attacker can obtain unencrypted traffic logs of users browsing the web. We define PAS as a session containing a password authentication request message. And the attacker intents to sort out PAS for users, so that any website can be accessed from numerous of traffic logs.

To overcome the shortcomings of previous methods, we have developed a password sniffing attack tool based on deep neural networks, called PassEye. Firstly, PassEye takes the HTTP session as input and uses designed rules to extract the features from the HTTP session. Preprocessing feature data is required: getting the invalid items removed, and the feature data normalized and one-hot encoded. The correlation rate between each feature and the plaintext password feature can be calculated by XGBoost algorithm [7], and the features with high rates can then be selected. Secondly, the basic architecture of PassEye is a neural network. The loss function, the number of layers and neurons, and the activation function are elaborately designed to build the network. The selected feature data is used to train the neural network model. Finally, PassEye can perform password sniffing attacks on network traffic.

In the experiments, an approach was first designed to collect labeled training data. 979,681 HTTP sessions were collected as our raw dataset and 7,697 were labeled as PAS. Secondly, the designed feature extraction and selection methods of PassEye were used to collect features from the raw data. 58 features were extracted and the top 20 were selected for the subsequent training. Thirdly, python and TensorFlow are used to build a deep learning neural network for binary classification, and it was trained by using the selected data and features. Experimental results show that the accuracy, f1-score, precision and recall of PassEye reach 0.9931, 0.9931, 0.9932 and 0.9931 respectively, which successfully proves the superiority of PassEye.

In summary, our **contributions** are as follows:

- A new HSPS attack model is proposed for website traffic password sniffing in HTTP and HTTPS protocols.
- We design and implement PassEye, a practical HSPS attack tool based on deep neural networks.
- We also show that PassEye is effective deep neural networks in HSPS attack.

Outline. The rest of this paper is organized as follows. In Sect. 2, we provide background on password attacks, password sniffing attacks, and the application of neural network to network traffic classification. In Sect. 3, we define the HSPS attack model. In Sect. 4, we show the design of PassEye. In Sect. 5, we present an evaluation of PassEye. Finally, we provide conclusions and future work in Sect. 6.

2 Background

2.1 Password Attack

Due to the vulnerability of password authorization, password attacks have been the focus of many scholars. Current research on password authentication is mainly focus on the evaluation of password security [8] and the optimization of password guessing methods [8–12]. Traditional password guessing methods are based on dictionary, Markov model or probabilistic context-free grammar (PCFG) [11]. Melicher et al. [8] use a neural network for password guessing attacks for the first time, and the evaluation results show outstanding performance. Following Melicher's work, neural network methods for password guessing have developed rapidly in recent years.

2.2 Password Sniffing Attack

Compared with password guessing attacks, there is little research on password sniffing attacks, since it does not have good versatility currently. In fact, it can directly capture plain text passwords from network traffic without guessing, which is more time-saving and of greater practical value. This is an motivation for our research.

There are four traditional methods of password sniffing attacks, such as manual analysis, keyword matching, regular expression, and automatic tools. These methods are also applicable to HTTP website attacks. Manual analysis is based on traffic dump tools (e.g. Wireshark, TcpDump) or man-in-the-middle (MITM) proxy tools (e.g. fiddle, Burp Suite, mitmproxy). Attackers manually search and filter the raw network traffic logs and find which packet contain plain passwords. This can be the most common and effective method due to the complexity of websites. Keyword matching is fast, which uses password's keywords (e.g. 'password', 'pwd', 'passwd') to match the content of the network traffic. However, experiments show that it has a high false positive rate. Compared with these methods, regular expression can bring more accurate results. According to the patterns of the site's PAS, attackers can write regular expressions to match the usernames and passwords. However, since regular expressions are usually specifically designed and do not support a wider range of websites, attackers need to learn the pattern of PAS for each website. Therefore, it is indeed a time-consuming method for attackers. Since SniffPass [5] and Password Sniffer Spy [6] are two automatic tools that support only a few patterns, such as POP3, IMAP4, SMTP, FTP, and HTTP basic authentication, and do not support password authentication in HTTP webpage form, their availability in HTTP website password sniffing attacks is quite low. In summary, current methods have many deficiencies in terms of versatility and automation.

2.3 Neural Network in Network Traffic Classification

The deep neural network has shown superior performance in software developing and analysis [13, 14] and has been widely used for classification and detection of network traffic logs in recent years [15, 16], such as attack detection, traffic content classification. Liu et al. [16] use a two-step neural network, Payload Locating Network and Payload Classification Network, for web attack detection, and the precision of the evaluation results reaches 100%.

Traffic content type identification is also an important application of deep neural networks [17]. Lotfollahi et al. [18] take the extracted first 20 bytes of the IP header, first 20 bytes of the TCP/UDP header and first 1460 bytes of payload as the input to the CNN/SAE deep neural network classification model, and the classification precision for Tor, webpage, audio and video reaches 95%.

3 Attack Model

The ultimate goal of a passive attacker in a traditional password sniffing attack is to intercept the user's password. The attack model is shown in Fig. 1.

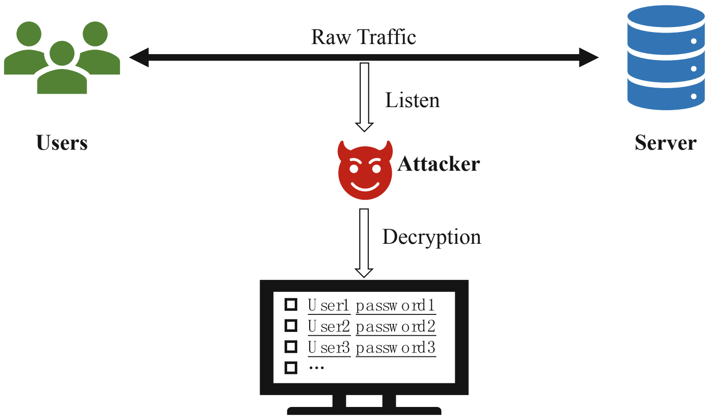

Fig. 1. Traditional password sniffing attack model.

HTTPS is a common and effective method to prevent MITM attacks on websites. Mi et al. [19] analyze the insecurity of IP proxy and Krombholz et al. [20] reveal the problems encountered by HTTPS in practice. Their research shows that HTTPS is not absolutely secure. In addition to the above work, there are many attack methods for MITM attacks in HTTPS. Users may be tempted to install a root certificate in a web browser, and the attackers can set a transparent proxy server to track users' web browsing requests. DNS hijacking is also effective in HTTPS MITM attack.

In this paper, we propose an HSPS attack model, focusing on the classification of the HTTP sessions, and assuming that HTTPS traffic has been perfectly decrypted into HTTP sessions. Figure 2 shows the HSPS model. Users use browsers to surf the internet, and the browsers send HTTP/HTTPS requests to the webserver and receive the response.

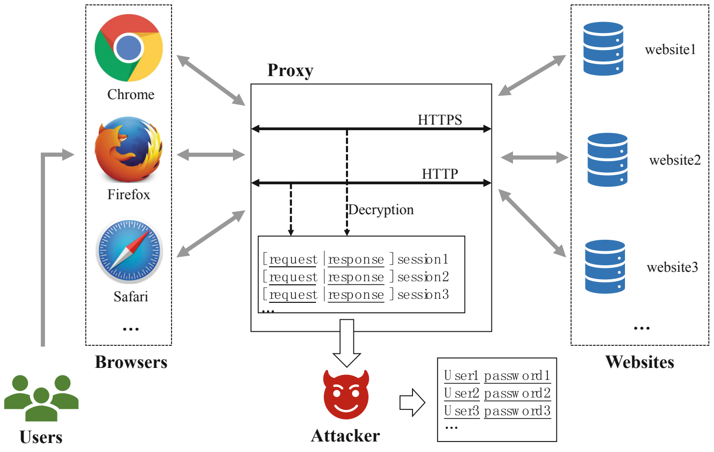

Fig. 2. HTTP session password sniffing (HSPS) attack model.

In the process of messages transposition, there is a transparent proxy that can perfectly decrypt HTTPS traffic and parse the traffic into request and response. For some reason, attackers can monitor the decrypted HTTP sessions. The goal of the attackers is to filter out PAS, and then parse the user's password information as much and as accurately as possible from a large amount of HTTP sessions.

4 PassEye Design

Due to the lack of versatility of previous methods, this paper proposes PassEye, a password sniffing attack tool based on deep neural networks, which can steal password in numerous HTTP traffic logs.

4.1 Overview

Figure 3 shows an overview of the PassEye design. The input of the PassEye was the HTTP sessions containing request and response messages. Then we designed a feature extraction method that could extract feature data from the HTTP sessions, which was helpful for PAS. The invalid items in the feature data were removed, and the feature data was normalized and one-hot encoded. The correlation rate between each feature and the plaintext password feature was calculated using the XGBoost [7] features with high correlation were selected features with high correlation. After these steps, the HTTP sessions was transformed into feature vectors, which could be used to train the deep neural network model, PassEye, designed in this paper. The results show that this method can perform password sniffing attacks in the HTTP sessions.

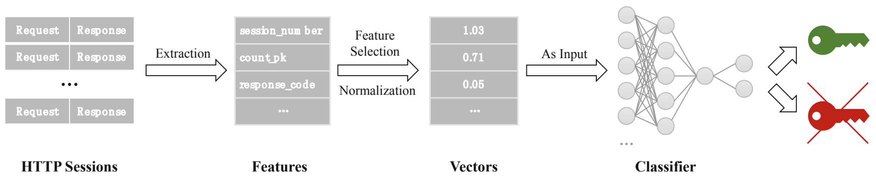

Fig. 3. An overview of the PassEye.

4.2 Feature Extraction and Selection

Feature extraction is very important for neural network models. The more important the features can represent the PAS, the more accurate and generalized the machine learning model can be.

In this paper, a total of 21 plain passwords related features extracted from the HTTP sessions are listed in Table 1.

Table 1. Features extracted from the http session.

Name	Meaning	Type	Name	Meaning	Type
Session number	The session number in each record	Int	Response set cookie	Whether the response header has the 'Set Cookie'Field	Bool
Count pk	The occurrences of password keywords in the request message	Int	Response cookie len	The length of 'Set Cookie' field in the response header	Int
Count uk	The occurrences of username keywords in the request message	Int	Response code	Response status code	Enum

(*continued*)

Table 1. (*continued*)

Name	Meaning	Type	Name	Meaning	Type
Request content len	The length of the request content	Int	Time request	Time taken for the browser sending the request to the server	Float
Response content len	The length of the response content	Int	Time response	Time taken for the server sending the response to the browser	Float
Request header len	The length of the request header	Int	Time all	Time taken from the beginning of the request to the end of the response	Float
Response header len	The length of the response header	Int	Content type request	The list of content types in the request header 'Content-Type' field	List
Request header count	The number of the request header fields	Int	Content type Response	The list of content types in the response header 'Content-Type' field	List
Response header count	The number of the response header fields	Int	Content type accept	The list of content types in the request header 'Accept' field	List
Request cookie len	The length of the request header cookie field	Int	Is https	Whether this session uses HTTPS protocol	Bool
Request cookie count	The number of key-value pairs in the request header cookie field	Int			

It is worth mentioning that the "count pk" feature counts the number of times that the password keywords appear in the request messages. Password keywords are words with high frequency around passwords in statistics. In other words, we take the keyword matching method as a feature in PassEye method.

After the step of feature extraction, a HTTP session is abstracted into a list of features. To better show the correlation between discrete features and plain passwords, PassEye uses one-hot encoding to convert discrete feature variables into multiple Boolean features. Z-score standardization is used to keep the features within a similar numerical range, which can be described as follows:

$$z = \frac{x - \mu}{\sigma}$$

where x denotes the eigenvalue to be normalized, μ denotes the arithmetic mean, σ denotes the standard deviation of the feature, and z denotes the target value.

To quantify the impact of each feature on the plain password, PassEye calculates its correlation using the XGBoost algorithm. The top k of the above features are selected.

Through the above steps, we can obtain a $1 * k$ feature vector, which can be used as the input of the neural network. The feature vector can well keep the information of the session itself and its correlation with the PAS, so that the performance of the neural network can be improved.

4.3 Neural Network Model

Our model consists of 5 layers, including an input layer, 3 hidden layers, and an output layer. The input layer has k nodes, which correspond to the feature vectors on a one-to-one basis. The three hidden layers contain 5 nodes, 5 nodes, and 1 node, respectively. The activation function in hidden layers 1 and 2 is ReLU, while that in hidden layer 3 is Sigmoid. The output layer has 2 nodes, corresponds to the two classification results: PAS and non-PAS. The optimizer is Adam, the learning rate is 0.001, and the loss function is Binary Cross Entropy.

During the training process, the random value of the initialization of the model weights ranges from -1 to 1. The batch size is 32, the number of steps per epoch is 100, and the maximum epoch is 10,000. An early stop condition is set to prevent over-fitting. The training will stop if the model does not show any improvement in 20 consecutive epochs.

5 Evaluation

We demonstrate the effectiveness of PassEye by answering the following questions:

Q1. Does PassEye perform better than keyword matching and regular expression methods?
Q2. What are the characteristics of PassEye compared to traditional methods in HSPS attacks?

5.1 Environment Setup

The hardware environment and main softwares are as followed.

Hardware: (1) CPU: Intel E7 4809v4 * 2; (2) Memory: 128G; (3) Disk: 8T SSD.
Software: (1) OS: Ubuntu Linux 18.04 LTS; (2) Python 3.6.9; (3) TensorFlow 1.0; (4) XGBoost 0.9.0; (5) Docker 19.03.2; (6) mitmproxy 4.0.4; (7) Selenium 141.0; (8) Chrome 78.0.3904.108.

5.2 Dataset

We designed a new approach to collect labeled training data: using selenium and chrome to simulate browsing and logging into a website, and then using mitmproxy as a middleman proxy to collect HTTP traffic logs. The experiment target site was Alexa China's top 500 website [21]. 224 of the sites use HTTPS while 276 do not. We wrote a script for each of these websites, and several browsing and login records could be captured by executing each script. Each record generated a set of usernames and passwords randomly as well as several HTTP sessions. As a result, a total of 43,619 records (with corresponding usernames and passwords) and 979,681 HTTP sessions were collected as our raw dataset. Text search was used to see if the plaintext password corresponding to the HTTP sessions exists in the request message of the sessions, and a PAS was labeled when the answer was yes. In the end, 7,697 PAS were obtained. sample set: Due to the disparity in proportion between PAS and non-PAS samples, we used weighted random sampling to select a subset from the raw dataset as the sample set for training, which contains 5,875 PAS and 11,058 non-PAS.

We then divided the sample set into a training set, a validation set, and a test set at a ratio of 0.64:0.16:0.20.

5.3 Effectiveness of PassEye

We then extracted features from the training set described in PassEye Design. After one-hot encoding, 58 features were collected. XGBoost was used to calculate the correlation of the features, and the top 20 were selected to train the deep neural network, as shown in Fig. 4.

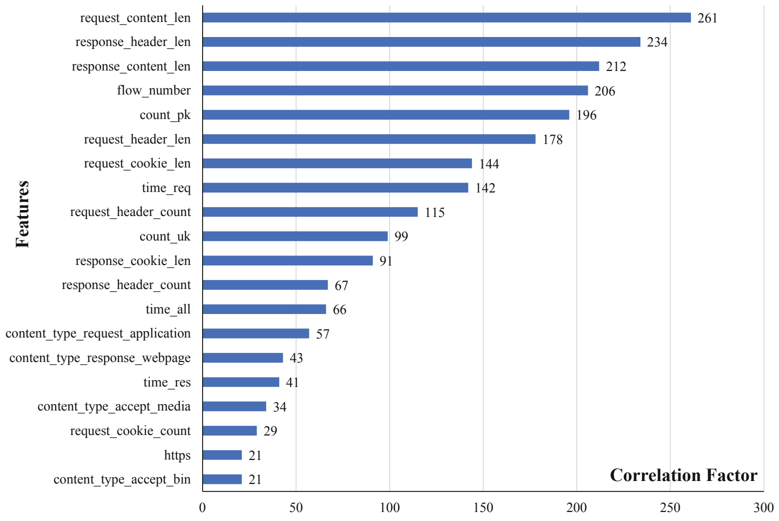

Fig. 4. Correlation of the top 20 features.

The training and validation sets were used to train the machine learning model described in PassEye Design. The test set was used to test the performance of the trained model.

For comparison, we also implemented keyword matching and regular expression methods as our baselines, and the test set was the same one.

Performance of PassEye

Table 2 shows the accuracy, precision, recall, and f1-score results for the three methods. As can be seen from the table, all the performance metrics of PassEye are over 99.2%. Furthermore, all the metrics of PassEye are the highest, followed by the regular expression, and the performance of the keyword matching is the worst. It can be concluded that PassEye significantly surpasses these traditional methods in terms of the performance.

Table 2. The performance of the three methods.

Method	Accuracy	Precision	Recall	F1-score
Keyword matching	81.87%	85.67%	82.92%	81.65%
Regular expression	97.40%	96.50%	97.89%	97.13%
PassEye	**99.38%**	**99.46%**	**99.20%**	**99.33%**

Characteristics of PassEye

Table 3 shows the characteristics of different password sniffing attack methods. Manual analysis, keyword matching, regular expression, and SniffPass, which can be seen as the representative of automatic tool, are presented in it for comparison, along with PassEye. The evaluation metrics include automaticity, versatility, scalability, independence, fastness, and robustness. Automaticity refers to whether it can run completely automatically without human intervention. Versatility refers to whether it can be used on any website. Scalability refers to whether the method supports extensions for use on new websites. Independence refers to whether this method can perform password sniffing attacks independently. Fastness refers to whether the method can run fast enough. Robustness refers to whether the method is effective enough in the face of unknown situations.

As can be seen from Table 3, PassEye has the characteristics of automaticity, versatility, scalability, fastness and robustness, except for independence. All other methods are not robust. Despite that SniffPass owns the independence, PassEye is still the best choice after the comprehensive consideration of all characteristics.

Therefore, it can be summarized that PassEye has the best characteristics among all these methods.

Table 3. The characteristics of different password sniffing attack methods.

	Manual analysis	Keyword matching	Regular expression	SniffPass	PassEye
Automaticity	×	✓	✓	✓	✓
Versatility	✓	×	×	×	✓
Scalability	×	✓	✓	×	✓
Independence	×	×	×	✓	×
Fastness	×	✓	✓	✓	✓
Robustness	×	×	×	×	✓

5.4 Discussion

It can be seen from the experiment results that PassEye has brilliant performance and best characteristics compared with some other traditional methods.

In the experiments, we also calculated the correlation between each feature and whether it is PAS. The correlation is shown in Fig. 4. The figure shows that the five features that have the most influence on the classifier are request_content_len, response_header_len, response_content_len, session_number and count_pk. This has given us some implications for preventing against HSPS attacks.

To prevent HSPS attacks, websites can make the following changes to the above features:

- Randomly change the length of the request content, the length of the response header, and the length of the response content by padding arbitrary characters.
- Have several unrelated random sessions between the browser and the server before the former sending the password authentication request messages to the latter. The goal is to change the session number. – Obfuscate and encrypt the fields of PAS.

In addition, there are some conventional ways to prevent password sniffing attacks. Websites can asymmetrically encrypt or hash passwords before sending login requests. Using self-built algorithms to obfuscate the content of requests is also an effective way. Changing the way of password authentication can be a solution as well, such as using one-time password, 2-step verification, etc.

6 Conclusion and Future Work

This paper proposed an HSPS attack model, which is a perfect expression for the problem of website password sniffing. PassEye, a tool based on deep neural networks, was proposed to implement the attack. We also designed a feature extraction method for HSPS attacks. In Evaluation, the experiment results verified the effectiveness of PassEye and deep neural networks in HSPS attacks. Some prevention strategies for websites were provided as well.

In the future, we will explore methods to make PassEye more robust, such as CNN, RNN or other machine learning models. The classification of obfuscated and hashed passwords can be considered and added to make PassEye more practical.

References

1. Wang, D., Wang, P., He, D., Tian, Y.: Birthday, name and bifacial-security: understanding passwords of Chinese web users. In: 28th USENIX Security Symposium (USENIX Security 19), pp. 1537–1555. USENIX Association, Santa Clara (2019)
2. Jammalamadaka, R.C., Van Der Horst, T.W., Mehrotra, S., Seamons, K.E., Venkasubramanian, N.: Delegate: a proxy based architecture for secure website access from an untrusted machine. In: 2006 22nd Annual Computer Security Applications Conference (ACSAC 2006), pp. 57–66. IEEE, Miami Beach (2006)
3. Password Sniffing Attack. In: SSH.COM (2020). https://www.ssh.com/attack/password-sniffing. Accessed 3 Dec 2019
4. Mozilla: a typical HTTP session. In: MDN Web Docs (2019). https://developer.mozilla.org/en-US/docs/Web/HTTP/Session. Accessed 20 Oct 2019
5. SniffPass Password Sniffer - Capture POP3/IMAP/SMTP/FTP/HTTP passwords. In: NirSoft. https://www.nirsoft.net/utils/password_sniffer.html. Accessed 22 Oct 2019
6. SecurityXploded: Password Sniffer Spy : Free Tool to Sniff and Capture HTTP/FTP/POP3/SMTP/IMAP Passwords (2020). https://www.SecurityXploded.com. Accessed 1 Jan 2020
7. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 2016, pp. 785–794. ACM Press, San Francisco (2016)
8. Melicher, W., et al.: Fast, lean, and accurate: modeling password guessability using neural networks. In: 25th USENIX Security Symposium (USENIX Security 16), pp. 175–191. USENIX Association, Austin (2016)
9. Hitaj, B., Gasti, P., Ateniese, G., Perez-Cruz, F.: PassGAN: A Deep Learning Approach for Password Guessing. arXiv:170900440 [cs, stat] (2017)
10. Pal, B., Daniel, T., Chatterjee, R., Ristenpart, T.: Beyond credential stuffing: password similarity models using neural networks. In: 2019 IEEE Symposium on Security and Privacy (SP), pp. 417–434. IEEE, San Francisco (2019)
11. Liu, Y., et al.: GENPass: a general deep learning model for password guessing with PCFG rules and adversarial generation. In: 2018 IEEE International Conference on Communications, ICC 2018, May 20, 2018–May 24, 2018. Institute of Electrical and Electronics Engineers Inc., p Cisco; et al.; Huawei; National Instruments; Qualcomm; Sprint (2018)
12. Muliono, Y., Ham, H., Darmawan, D.: Keystroke dynamic classification using machine learning for password authorization. Proc. Comput. Sci. **135**, 564–569 (2018). https://doi.org/10.1016/j.procs.2018.08.209
13. Duan, X., et al.: VulSniper: focus your attention to shoot fine-grained vulnerabilities. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence. International Joint Conferences on Artificial Intelligence Organization, Macao, China, pp. 4665–4671 (2019)
14. Yang, M., Wu, J., Ji, S., Luo, T., Wu, Y.: Pre-Patch: find hidden threats in open software based on machine learning method. In: Yang, A., et al. (eds.) SERVICES 2018. LNCS, vol. 10975, pp. 48–65. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94472-2_4
15. Prasse, P., Machlica, L., Pevny, T., Havelka, J., Scheffer, T.: Malware detection by analysing network traffic with neural networks. 2017 IEEE Security and Privacy Workshops (SPW), pp. 205–210. IEEE, San Jose (2017)

16. 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 Twenty-Eighth International Joint Conference on Artificial Intelligence. International Joint Conferences on Artificial Intelligence Organization, Macao, China, pp. 4725–4731 (2019)
17. Yao, Z., et al.: Research review on traffic obfuscation and its corresponding identification and tracking technologies. Ruan Jian Xue Bao/J. Softw. **29**(10), 3205–3222 (2018). (in Chinese). http://www.jos.org.cn/1000-9825/5620.htm
18. Lotfollahi, M., Zade, R.S.H., Siavoshani, M.J., Saberian, M.: Deep packet: a novel approach for encrypted traffic classification using deep learning. arXiv:170902656 [cs] (2018)
19. Mi, X., et al.: Resident evil: understanding residential IP proxy as a dark service. In: 2019 IEEE Symposium on Security and Privacy (SP), pp. 1185–1201. IEEE, San Francisco (2019)
20. Krombholz, K., Busse, K., Pfeffer, K., Smith, M., von Zezschwitz, E.: "If HTTPS were secure, i wouldn't need 2FA" - end user and administrator mental models of HTTPS. In: 2019 IEEE Symposium on Security and Privacy (SP), pp. 246–263. IEEE, San Francisco (2019)
21. Alexa China Siterank. http://www.alexa.cn/siterank. Accessed 28 Nov 2019