A Machine Learning-based Method for Detecting Buffer Overflow Attack with High Accuracy

Shubin Li[1,a], Rongfeng Zheng[2,b], Anmin Zhou[1,c] and Liang Liu[1,d,*]

[1]College of Cybersecurity, Sichuan University, Chengdu, Sichuan, China
[2]College of Electronics and Information Engineering, Sichuan University, Chengdu, Sichuan, China
a. lishubin00@sina.cn, b. qswhs@foxmail.com
c. zhouanmin@scu.edu.cn, d. liangzhai118@163.com
**corresponding author: Liang Liu*

Keywords: Buffer overflow, machine learning, network traffic detection, feature extraction.

Abstract: Buffer overflow attack is one of the typical attacks over the internet, it aims to make the overflow overwrite the legitimate data. How to detect Buffer overflow attack becomes a hot topic in research. Protocol uncertainties and varying attack modes will seriously affect the efficiency and the accuracy of attack detection. Recently, machine learning is widely applied in network traffic detection and data processing, meanwhile tradition buffer overflow detection method based on feature matching is difficult to detect the attack load hidden in network traffic. In order to address these challenges, we apply machine learning to detect remote buffer overflow attack, to enhance the classification ability of machine learning models, we propose a unique set of feature extraction rules after analyzing a large number of attack codes. This method is not only capable of identifying various attack forms and unknown attack types but also has the ability of supporting protocol independent detection and identifying attacks using various protocols based on TCP or UDP. In order to evaluate the performance of our method, three machine learning algorithms are selected to establish detection models and the model base on random forest algorithm perform best., moreover, comparison experiments with other detection methods are also carried out based on the same dataset. The experimental results show our method can detect remote buffer overflow attack with adaptability, efficiency and accuracy.

1. Introduction

Remote network attack as the main form of attack in the Internet has attracted wide attention [1]. Due to the openness and interactivity of the network and the insufficient security of the code,the devices in the network are vulnerable facing to network attack. Buffer overflow attack is one of the typical network attacks, it is an attack that exploits buffer overflow vulnerabilities, which exists widely in various applications and even operating systems. It can not only cause the program to crash, but also make attackers obtain the control of the system.

The essence of buffer overflow is that the computer fills the buffer with more data bits than the buffer itself, and the overflow data overrides the legal data. In the ranking of vulnerabilities since 2003, buffer overflows have always been "famous on the list." Well-known shock waves, big worm

shocks, etc. have all achieved attacks through buffer overflows. Buffer overflow attack can be generated in two ways: remote buffer overflow attack [2] and local buffer overflow [3]. In a remote buffer overflow attack, the attacker usually uses common protocols to disguise the attack code, so that the program's buffer overflows to achieve the purpose of the attacker; see Figure 1. Local buffer overflow attack is different from remote buffer overflow attack, it is not a network attack. Normally, the target is a local application, an attacker exploits a buffer overflow vulnerability in the source code of a local application to crack the program or gain developer permissions.

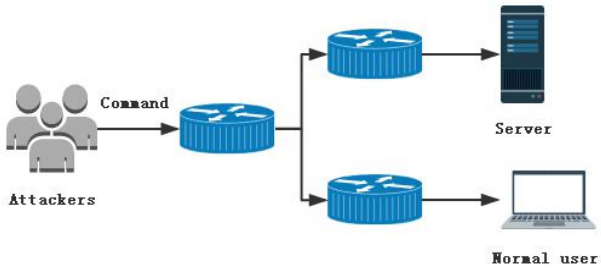

Figure 1: A diagram of remote buffer overflow attack.

DPI (Deep Packet Inspection) technology is one of the commonly used traditional methods. Common tools such as 'snort' and 'bro' have been developed on the basis of this technology. But once hackers deliberately avoid using some common features, it is difficult for the traditional network intrusion detection system to identify attack payloads contained in data packets. Although many efforts have been made to detect remote buffer overflow attack, we still have some problems that have not been solved. We summarize these problems as follows: (1) The remote buffer overflow method based on feature matching does not have the ability to detect attack code with insignificant features and unknown attack types. The essence of this detection method is to analyze the characteristics of a large number of attack codes, and establish a corresponding feature library based on these characteristics. However, as the network size continues to increase, the types of attack code are constantly changing. Attackers can change the type of code as needed to be used for immune signature detection. For unknown attack codes, because the feature database contains only the features of known attack types, this method also does not have the ability to detect unknown types of attack codes; (2) lack of protocol independent detection methods [4]. There are many kinds of remote buffer overflow attacks based on multiple network protocols. We do not know which type of attacks will be handled. Each attack type has its own characteristics and could exploit multiple protocols to launch attacks. So, a protocol-independent method is highly expected to detect remote buffer overflow attack. According to the discussion above, a protocol-independent buffer overflow detection method is urgently needed, which has the ability of handling variable attack types and unknown attack types.

In this paper, we propose a remote buffer overflow attack detection method based on machine learning [5,6] that can support protocol independence and handling of unknown attack types. First, with the purpose of handling changing attack patterns and unknown attack types, we collecte and reproduce the attack code that exploited the remote buffer overflow vulnerability in the past ten years, by analyzing the header fields and time distribution of the packets containing the attack payload and the necessary conditions for buffer overflow, we propose a novel feature extraction rules to identify changing attack types with the help of machine learning. To make our method protocol independent, when designing features, time-related statistical features are our key candidates for feature selection, such as the rate of data packets. In addition, compared to the unique

fields of various application layer protocols, we are more inclined to choose the common fields of the underlying protocols to ignore the differences between different application layer protocols. By doing so, our method can not only retain the same features of the traditional detection methods, but also has ability of handling variable attack types and supporting protocol independence.

The structure of this paper is as follows. Section II gives a brief introduction on related work. The theoretical basis of our method is given Section III. In Section IV, we present the system architecture of our method in detail. The performance evaluation results of comparative tests are showed in Section V. Finally, conclusions are covered in the last section.

2. Related Works

Buffer overflow vulnerability is one of the common and high-risk vulnerabilities. There are three different types of methods that are widely used to detect buffer overflow vulnerabilities or aggressive behavior. The first is the buffer overflow vulnerability detection technology based on source code [7]. It is a method of detecting whether a buffer overflow vulnerability exists in the source code by analyzing the semantics and syntax of the source program. Shahriar, H. and Haddad, H.M. [8] used source-based buffer overflow detection technology to detect vulnerabilities and repair vulnerable programs without modifying application functionality. This method relies on the source code of the program and researchers can effectively find some obvious buffer overflow vulnerabilities, but its high false alarm rate is major shortcomings. The second is the detection technology based on the target code. The essence of this method is to analyze the target code in a black box and use random strings of different lengths to test the target code so that we can find buffer overflow vulnerabilities in software, Terry Bruce Gillette [9] proposed a detection method based on object code. The main idea is to use disassembly tools to process the object code, and then use the source-based detection and analysis technology to process the obtained results. Joao Duraes [10] proposed a method to find buffer overflow vulnerabilities without source code. The essence is to analyze executable code to locate suspicious functions. Then input some test parameters to the suspicious function for robustness test to determine the authenticity of the suspicious function. However, because the execution path cannot be completely traversed, this method has a high rate of false negatives. Not only that, the position of the vulnerability in the code cannot be determined. The third is the detection technology based on feature matching. This technology is mainly used for remote buffer overflow attack detection. By analyzing the characteristics of the buffer overflow attack code, a feature database is established, and the matching technology is used to detect whether the data flow contains attack packets. Ivan Homoliak and Ladislav Sulak [11] proposed an approach: signature matching against packets' payload versus analysis of packets' headers with the behavioral analysis of the connection's flow. This method is characterized by simple implementation and low consumption, but it cannot detect attack code without obvious features.

There are numerous detection methods for buffer overflow attack detection. However, few of existing methods considered the variability of attack forms and protocol independence. In order to solve these problems, we propose to integrate machine learning into remote buffer overflow attack detection and standardize the data in the packets.

3. Preliminaries

In this section, we introduce the principle of buffer overflow attacks, which is the theoretical preliminaries of our method.

In memory, the growth direction of memory is from low address to high address. In contrast to the growth direction of memory, the growth direction of stack is opposite to the former, as shown in Figure 2. The main function is first pushed onto the stack in the direction of stack growth. Before

called function is pushed onto the stack, the address of the next instruction stored in the EIP register is first pushed onto the stack, and then the base address value in the real-time EBP register is pushed onto the stack. After the called function is pushed onto the stack, if the length of the passed string is longer than the length of the buf array, a buffer overflow will occur. More seriously, if the length of the string is calculated by the hacker, the return address will be accurately overwritten, and the program will jump to the shellcode [12] pointed to by the new address. At this point, the victim's computer will run shellcode compiled by the hacker.

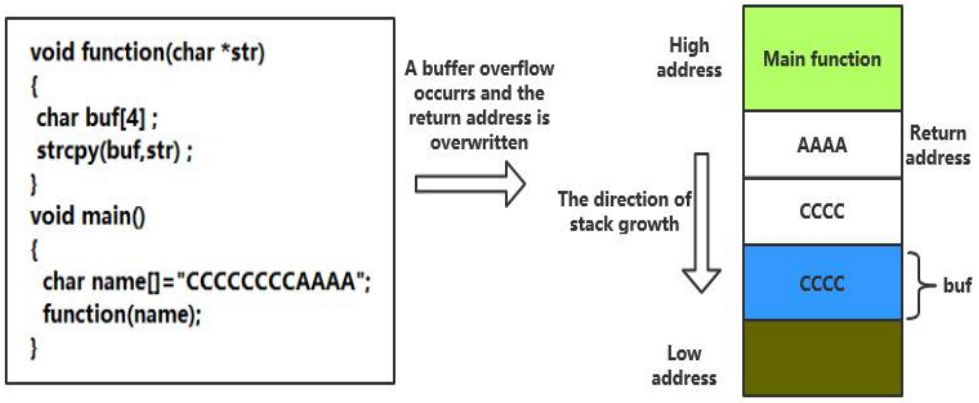

Figure 2: A diagram of buffer overflow.

4. Method

In this section, we introduce the architecture of a remote buffer overflow attack detection method based on machine learning. Then, we describe our feature extraction rules and some of them in detail.

4.1.The Overall Architecture of Our Method

Figure 3 shows the system architecture of our detection method. In the initialization module, we first merge the packets into a network stream according to five tuple information. In the data preprocessing module, we extract the features of according to the unique feature extraction rules, and then these data will be normalized. After these numerical features are identified by the classifier model, malicious traffic will be identified and its IP address will be blacklisted for the coarse filtering module, and normal traffic will be released. The remote buffer overflow attack detection is described as Algorithm I.

4.2.Feature Extraction

In this subsection, we describe our feature extraction rules and some of them in detail. The features we used is depicted in Table 1.

The three types of features, 'Type of Service', 'Time to Live', and 'Port numbers', are calculated based on the network flow and the header of the data packet. In order to make our method more protocol-independent, we only select a small amount of header field information and focus on statistical information.

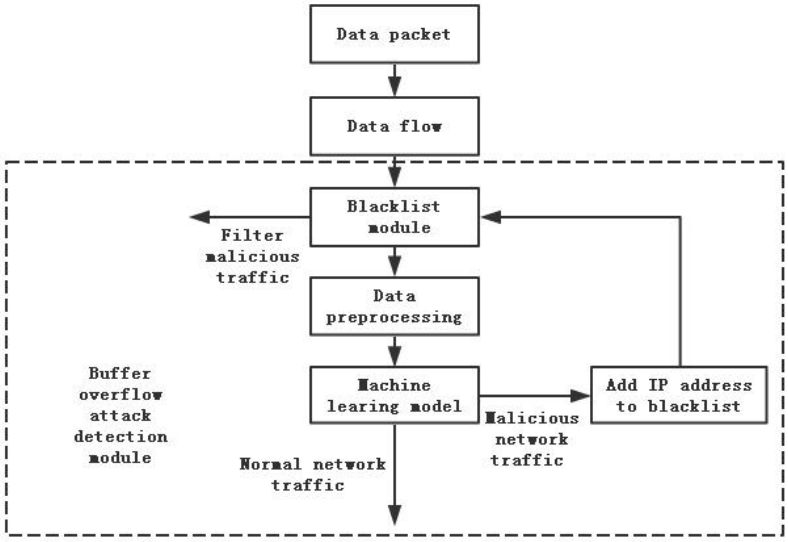

Figure 3: System architecture.

Algorithm I Remote buffer overflow detection

1. Use the filtering module to find out if the IP address of the newly arrived data packet or data flow is in the blacklist. If it is, the data packet will be filtered, otherwise the data packet will go to the next step.
2. Use mathematical methods to record statistical features, such as the number of incoming and outgoing data packets and so on, and selectively record the header fields of the protocol at each layer of the data packet.
3. Normalize features
4. Identify by classifier, normal traffic will be allowed, malicious traffic will be filtered, and its IP address will be added to the blacklist.
5. Return step 1.

Table 1: Summary of the features used in experiments.

Num	Source	Feature name
1	Type of Service	Mode
2	Time to Live (TTL)	Minimum of TTL
3	Time to Live (TTL)	Maximum of TTL
4	Time to Live (TTL)	Sum of TTL
5	Time to Live (TTL)	Mean of TTL
6	Time to Live (TTL)	Standard deviation of TTL
7	Port numbers	Source port number
8	Port numbers	Destination port number
9	Packet counting	Number of incoming packets
10	Packet counting	Number of outgoing packets
11	Packet counting	Number of all packets
12	Packet counting	Number of non-data packets
13	Payload lengths	Sum of payload lengths
14	Payload lengths	Mean of payload lengths
15	Payload lengths	Standard deviation of payload lengths
16	Fragmentation	Number of fragmented packets
17	Fragmentation	Number of non-fragmented packets
18	Fragmentation	Fragmented packet payload lengths
19	Transfer time	Time of connection
20	Transfer time	Bits per second
21	Transfer time	Total time
22	Transfer time	Total time for no packet transfer
23	Other	The number of consecutive identical characters
24	Other	The existence of '\0'
25	Other	The existence of jump addresses

A large number of padding characters is a necessary condition for buffer overflow. To facilitate operations, hackers often use a large number of identical characters or a large number of regular characters to fill the buffer, which is an important feature of buffer overflow attacks. The length of the payload will be significantly longer, and it will also cause a lot of fragmentation of the network data stream.

We know that '\ 0' represents the end of this string. In shellcode, the strcpy function will be truncated if it encounters '0x00' when it is executed, causing the shellcode to be incomplete, so that the return address cannot be overwritten, so there is no '0x00' in the shellcode.

In addition, the inclusion of jump addresses in the payload is also an important feature of the buffer overflow attack code. Traditionally, the return address is modified to the address where the shellcode is located. When the function execution ends, the return address is popped from the stack, which corresponds to the address of the shellcode. The final shellcode code is executed by the system. Just like reading a file with an absolute path, the address will occasionally be wrong. To solve this problem, hackers invented a method called 'jmp esp', whose main purpose is to overwrite

the return address with the address of the 'jmp esp' instruction. The instruction is then executed by the CPU so that the execution pointer jumps to the location pointed to by ESP register, the code at the location pointed to by the ESP register will be executed. As shown in Figure.4, the location pointed to by the ESP register is where the shellcode is located.

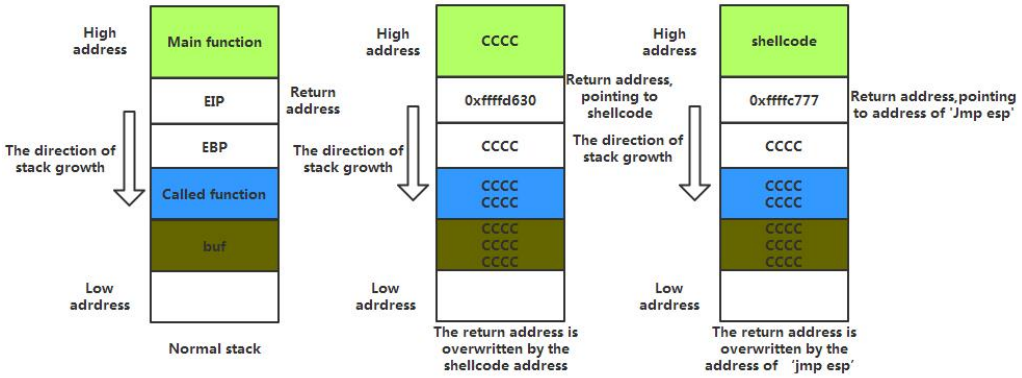

Figure 4: 'Jmp esp' schematic.

5. Experiments and Performance Evaluation

To evaluate the availability of our method, two comparative experiments were conducted on the same data set. First, in order to find out the impact of different machine learning models on the experimental results, we selected three different models to train and test the experimental data. Second, we choose the best performing machine learning model as a classifier for comparison with traditional detection methods. Our method performs better than traditional detection methods in both Recall and Precision.

5.1.Dataset

To evaluate the effectiveness of our method, we collected and reproduced about 500 exploits of buffer overflow vulnerabilities in the past 10 years from major authoritative vulnerability websites, which included exploits of major vulnerabilities such as Eternal Blue. Not only that, we collected nearly 3000 normal network data traffic from the laboratory server and saved them as pcap files for whitelisting. These 3,500 pcap samples constitute a complete data set. In addition, assessment indicators such as Precision, Recall and F1-score are used to evaluate the performance of our method.

5.2.Experimental Design

Different from the traditional feature-based detection method, constructing excellent features is one of the necessary conditions for good performance of the machine learning-based attack detection method. The feature construction rules have been introduced in Section 4. In addition, suitable machine learning models are also very important, so we choose three different machine learning models: Random forest [13], Bayesian network [14], AdaBoost [15], and then train and test them separately. In order to make an intuitive comparison between our method and the traditional method, we choose the Random forest model with the best test results as the final classifier and compare it with the traditional method on the same dataset. Finally, the Precision, Recall and F1-score are evaluated and compared.

5.3.Experimental Result

Three different machine learning models are used for training and testing on the same data set according to the ten-fold cross-validation method, as shown in Table 2, we can see that the random forest model has a better performance in the Precision, Recall, and F1-score.

Table 2: Evaluation indicators of different machine learning models.

Models	Precision	Recall	F1-score
Random forest	99.4%	99.2%	99.3%
Bayesian	98.3%	98.0%	98.1%
SVM	97.4%	98.1%	97.6%
AdaBoost	96.0%	97.1%	96.4%
Random tree	94.7%	95.2%	94.8%

We select the best-performing random forest model as a classifier to compare with the feature-based method called HBVDS [16]. As can be seen from Table 3, compared with traditional methods, our method has higher accuracy and lower false alarm rate, and the overall index is better than the detection method based on feature matching.

Table 3: Results between our method and traditional methods.

Methods	Precision	Recall	F1-score
Our method	99.4%	99.2%	99.3%
HBVDS	97.2%	96.8%	96.9%

6. Conclusions

This paper proposed a remote buffer overflow attack detection method based on machine science. This method combines machine learning with remote buffer overflow detection, not only that, we also proposed a specific feature extraction rule for machine learning based on the rules of attack code. It can be seen from the results of multiple comparative experiments that compared with traditional attack detection methods based on feature matching, our method not only has better performance in accuracy and recall, but also has the characteristics of protocol independence. The network traffic at the network nodes is heavy, and a very low false alarm rate will also cause a large number of normal traffic to be falsely reported. In the future, we will further modify the feature rules to reduce the false alarm rate to achieve better detection results.

References

[1] X.Y.Jing, Z.Yan, W.Pedrycz, Security data collection and data analytics in the internet: a survey, IEEE Commun. Surv. Tutor. (2018), doi:10.1109/COMST.2018.2863942.

[2] Zhang X, Liu H Y. Design and Implementation of Remote Buffer Overflow and implanted Backdoor[J]. 2012.

[3] Kuperman B A, Brodley C E , Ozdoganoglu H, et al. Detection and prevention of stack buffer overflow attacks[J]. Communications of the ACM, 2005, 48(11): 50-56.

[4] Swinnen, Arne, et al. "ProtoLeaks: A Reliable and Protocol-Independent Network Covert Channel." International Conference on Information Systems Security Springer, Berlin, Heidelberg, 2012.

[5] Imamverdiyev, Yadigar , and L. Sukhostat . "Anomaly detection in network traffic using extreme learning machine." 2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT) IEEE, 2016.

[6] Stevanovic, Matija , and J. M. Pedersen. "An efficient flow-based botnet detection using supervised machine learning." 2014 International Conference on Computing, Networking and Communications (ICNC) IEEE, 2014.

[7] Zheng Y, Li H, Chen K. Buffer Overflow Detection on Binary Code[J]. Journal of Shanghai Jiaotong University(Science), 2006(02): 107-112.

[8] Shahriar, H.,Haddad, H.M.. Rule-Based Source Level Patching of Buffer Overflow Vulnerabilities[P]., 2013.

[9] Terry Bruce Gillette,A Unique Examination of the Buffer Overflow Condtion [D].Florida: Engineering Florida Institute of Technology, 2002:5-34.

[10] Joao Duraes,Henrique Madeira.A Methodolog for Automated Identification of Buffer Overflow Vulnerabilities in Executable Software Without Source code [J]. Lecture Notes in Computer Science, 2005:3746-3747.

[11] Homoliak I, Sulak L, Hanacek P. Features for Behavioral Anomaly Detection of Connectionless Network Buffer Overflow Attacks[C]// 17th International Workshop on Information Security Applications (WISA 2016). Springer, Cham, 2016.

[12] Huang H L, Liu T J, Chen K H, et al. A Polymorphic Shellcode Detection Mechanism in the Network[C]// Proceedings of the 2nf International Conference on Scalable Information Systems, Infoscale 2007, Suzhou, China, June 6-8, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007.

[13] Wang Y, Xiang Y, Zhang J. Network traffic clustering using Random Forest proximities[C]// Communications (ICC), 2013 IEEE International Conference on. IEEE, 2013.

[14] Xu Y, Cheng P, Chen Z, et al. Mobile Collaborative Spectrum Sensing for Heterogeneous Networks: A Bayesian Machine Learning Approach[J]. IEEE Transactions on Signal Processing, 2018, 66(21):5634-5647.

[15] Souza E N D, Matwin S, Fernandes S. Network traffic classification using AdaBoost Dynamic.[J]. 2013.

[16] Han W. Research on buffer overflow attack code detection and defense technology [D]. PLA information engineering university, 2012. (in Chinese).