International Journal of Soft Computing 10 (1): 1-5, 2015
ISSN: 1816-9503

A Study of Buffer Overflow Attack Detection Using Artifical Immune System Based Danger Theory

[1]S. Vasanthi and [2]S. Chandrasekar
[1]Department of IT, Sona College of Technology, Salem, Tamil Nadu, India
[2]Gnanamani College of Engineering, Namakkal, Tamil Nadu, India

Abstract: Intrusion attacks are causing major security problems in computer networks. Firewall has been useful for certain attacks but it has its own limitation and can be bypasses. Intrusion detection and prevention is the method of identifying and preventing unauthorized use, misuse and abuse of computer system by both insiders and external attackers. In this approach, the resistant principle of human body is applied to computer security. Existing systems of AIS based IDS suffers lot of drawbacks due to Inefficient Negative Selection algorithm. These systems have high false positive and negative errors as this algorithm mainly detects intrusion based on discrimination of self from non-self. So, Danger theory evolved. According to Danger theory, the immune system does not respond to non-self but to danger. Thus, there is no need to attack everything that is foreign. The foreign entity which causes damage to the cells has to be killed. The proposed system identifies the attack uses this theory. This system runs in a host machine connected to a computer network. Based on this theory, the host machine receives packets from all the nodes connected to a network and analyzes the packets for any dangerous executable components in its payload. If the dangerous executable components are present in the payload it is said to be infected by an intrusion attack and the packet is discarded.

Key words: Intrusion detection and prevention, artificial immune system, Danger theory, non-self, firewall

INTRODUCTION

Biological systems such as human beings can be regarded as sophisticated information processing systems and can be expected to provide inspiration for various ideas to science and engineering. Biologically motivated information processing systems can be classified into: brain-nervous systems (neural networks), genetic systems (Evolutionary algorithms) and immune systems (artificial immune systems). Among these, nervous systems and genetic systems have been widely applied to various fields. There have been a relative few applications of the immune system. One such application is Intrusion Detection System (IDS).

The resistant principle of biology immune system is used in computer security systems. Artificial immune systems are a new computational intelligence method inspired by the biology immune systems. The analogy between Immune Systems and Intrusion Detection Systems encourage the use of Artificial Immune Systems for anomaly detection in computer networks and hosts. During the last decade, the field of Artificial Immune System (AIS) is progressing slowly and steadily as a branch of Computational Intelligence (CI). There has been increasing interest in the development of computational models inspired by several immunological principles. In particular, some are building models mimicking the mechanisms in the Biological Immune System (BIS) to better understand its natural processes and simulate its dynamical behavior in the presence of antigens/ pathogens.

Although, still relatively young, the Artificial Immune System (AIS) is emerging as an active and attractive field involving models, techniques and applications of greater diversity. Some computer security systems were developed based on traditional immunological principles. According to traditional immunology, the lymphocytes in the body generate cells which do not match our body cells (self cells). They are called non-self cells. When antigen (harmful) enters the body the non-self cells which are in closer affinity to the antigen generate antibodies to fight against it. Thus, the body immune system provides protection by discriminating the entered antigen as self or non-self cell. This principle is subject to criticism because all foreign entities that enter the body are not harmful. For, e.g., the food cells researchers intake may not match the body cells but the body immune system does not fight against it. The body immune system confirms whether the entered foreign entity is really harmful and provides protection. This technique is called Danger theory.

Corresponding Author: S. Vasanthi, Department of IT, Sona College of Technology, Salem, Tamil Nadu, India

PREVIOUS RESEARCH

Existing systems which concentrates in Negative Selection algorithms, Clonal Selection algorithms, misinterprets self to a non-self and vice versa hence the false positive and negative error is high. So, Danger theory concept evolved.

Negative Selection algorithm: The algorithm given by Dasgupta (2006) has the following two stages:

- Generation stage
- Detection stage

Generation stage: During this stage detectors are generated by random process. Those candidates that match with the self samples are eliminated and the rest are kept as detectors. Self samples are nothing but cells and molecules in case of human body. In case of computer networks self samples are IP packets from normal nodes.

Detection stage: In the detection stage, the collection of detectors (or detector set) is used to check whether an incoming data instance is self or non-self. If it matches any detector then it is claimed as non-self or anomaly.

Issues in Negative Selection algorithm

Scaling issue of detectors: When the Negative Selection algorithm is applied to a broader range of self it requires generation of exceptionally large amounts of detectors and causes an unacceptably long computation time. Also, when the self definition widened, the string needed to encode a detector lengthened. As the result of the long length of detectors, the number of detectors required to gain an acceptable false negative error, incorrect detection of self, become huge and thus requires an unacceptably long computation time (Iqbal and Maarof, 2003).

Issue of matching rules Another identified drawback of the Negative Selection algorithm is the adoption of the r-contiguous rule to check the match between a given detector and antigen. The Negative Selection algorithm requires an appropriate number of detectors in order to produce acceptable error and detection rates. The established formula that approximates the appropriate number of detectors is applicable only when the algorithm, uses the r-contiguous matching function. However, the r-contiguous matching rule is too simple to be used for determining the match between complex and high-dimensional patterns. Since, the r-contiguous bit matching only measures the contiguous bits of two given strings it is hard to guarantee that it can detect correlations in complex self and non-self patterns.

This self/non-self discrimination viewpoint has the following questionable issues which need attention of AIS researchers: negative selection is bound to be imperfect not having exact matching and therefore auto-reactions (false positives) are inevitable. The self and non-self boundary is blurred since self and non-self antigens often share common regions. Self changes over time. Therefore, one can expect problems with memory cells which later turn out to be inaccurate or even auto-reactive (Dasgupta, 2006).

In evolution induced secondary immunity, AIS the detectors are well trained against a sufficiently diverse set of samples it is possible to encounter a non-self pattern that cannot be detected by the existing detector set. (Krizhanovsky and Marasanov, 2007).

Aickelin *et al.* (2003) outlined the danger project, describing the application of the 'Danger theory' to intrusion detection systems. Researchers suggested a system of detection based around the presence or absence of danger signals as opposed to the pattern matching based approach used in negative selection. Danger signals released as a result of dying cells indicate damage and stimulate the immune system. It was proposed that a system which could differentiate between data collected in a dangerous context with data collected in a safe context. It was suggested that some of the problems with false positives could be alleviated through the incorporation of these two contexts for the purpose of IDS. As dendritic cells are a key cell in the translation of danger signals, they have formed a central part in the development of the danger based IDS, described in this study.

Existing intrusion detection techniques using artificial immune approaches uses Negative Selection algorithms Clonal Selection algorithms. It misinterprets self to non self and vice versa, i.e., it misinterprets normal node as intruder and intruder as a normal node in a computer network. So, Danger theory concept evolved. Human body has a perfect immune system which is capable of defending against any harmful antigen. It can accurately detect a harmful antigen. Similarly, the proposed system has to detect the intrusion attacks accurately.

BUFFER OVERFLOW EXPLOIT

Buffer overflow is the result of writing more data into a buffer than the buffer can hold. This happens when a vulnerable program receives external input and stores the

input to a buffer without checking the buffer's boundary. In order for a buffer overflow attack to succeed it needs to achieve the following two goals; inject the attack code and force the process to execute the injected code. If either goal fails, the attack fails. The most dominant form of buffer overflow exploitation is stack smashing attack.

Existing buffer overflow solutions: Methods for detecting buffer overflow vulnerabilities can be divided into three groups: static or compile time detection, host based detection and network based detection.

A compile time solution has been proposed by Larochelle and Evans (2001). Their solution was the development of a static analysis tool that analyses application source code in search of likely buffer overflow vulnerabilities. This solution is capable of improving an application by eliminating possibilities of successfully executing buffer overflow attacks but it requires modifications to the source code and recompilation to work in addition to the requirement of source availability.

StackGuard (Cowan *et al*., 1998) is an extension to the freely available and very popular gcc compiler that allows detection or prevention of alterations of the return address of a stack frame. Detection is performed by inserting a random word value immediately following the return address for the process on the stack. This value is verified when the process returns. Since, it is difficult to alter the return address without altering the following bytes this method is capable of detecting buffer overflow attacks. Some research suggests that the protection mechanism may still be circumvented by exploiting function pointers or "longjmps".

Toth and Kruegel (2002) have developed the concept of abstract execution for detecting buffer overflow attacks in live networks. This research is highly related to the research as both perform analysis of network data to determine if an attack has been launched. However, their research focuses on the NOP sledge preceding the attack code whereas the research only analyses the attack code itself. Toth and Kruegel (2002) define abstract execution as checking if a sequence of data represents valid machine instructions. Two properties of the sequence are checked: correctness and validity. Correctness refers to valid machine instructions and validity refers to valid memory access. The numbers of valid consecutive instructions are added together for streams of network data. Normal requests tend to have a low number of consecutive valid instructions and buffer overflows have a very high number due to the NOP sledges.

Host based anomaly detection solutions are currently available for detecting buffer overflow attacks. Lindqvist and Porras (1999) present a method of detecting buffer overflow attacks on Solaris hosts using the Basic Security Module (BSM). Their approach performs analysis on the exec call and the parameters passed to the system call as well as the effective and real user ID the process runs with to detect buffer overflow attacks. Similar results have also been achieved with STAT developed at UCSB. The approach differs from this research in that it attempts to detect buffer overflow attacks by only analyzing network traffic.

PROPOSED SYSTEM

Existing system misinterprets self to non self and vice versa, i.e., it misinterprets normal node as intruder and intruder as a normal node in a computer network. So, Danger theory concept evolved. Human body has a perfect immune system which is capable of defending against any harmful antigen. It can accurately detect a harmful antigen. Similarly, the proposed system has to detect the intrusion attacks accurately. The main objective is to:

- Reduce false positive and negative errors
- Maintain high detection accuracy

The proposed system detects remotely executable buffer overflow attacks. A remotely executable component of a buffer overflow attack is a series of machine instructions which will be referred to as "shell code". Attacker will inject this shell code into the remote machine and gets the full control of the system. This shell code will be found in the payload of the IP packet received by the remote machine which is subject to attack. Shell code contains system calls causing the attacks. The proposed system captures the IP packets and extracts the payload information. By finding out the opcodes which are executed for a system call, researchers can detect attacks. It searches for opcodes for system calls and counts the number of system calls and threat level one system calls in a packet and detects the attack.

According to Danger theory, the immune system of human body responds to the antigens which cause necrotic death (abnormal death) of cell or cell distress, i.e., the body gives immune response after confirming the entered antigen is dangerous. Similarly the proposed system analyzes the received IP packet to confirm that the received packet is truly dangerous. The proposed system is a NIDS. It detects all kinds of remotely executable stack based buffer overflow attacks. A remotely executable component of a buffer overflow attack is a series of machine instructions which will be referred to as "shell code".

When an attacker tries to hack a system using this buffer overflow exploit, traces of machine instruction opcodes corresponding to the executable components can be found in the payload. Using these opcodes number of system calls, interrupts and threat level one system calls distributed in a packet can be obtained. System calls such as chmod, chown etc are said to be threat level one system calls because they are used to gain full access to the system.

System model: The architecture of Artificial Immune System based intrusion detection which performs intrusion detection by analyzing the network traffic. This system is implemented in a host machine to detect intrusion attacks (Fig. 1).

Packet capture: The main purpose of this module is to capture IP packets from the real network traffic. It gets the network interface device name from the user or captures the default device from the machine. With the help of this device it captures the IP packets in real network traffic. The proposed system has to be designed such that it has to analyze the packets coming from all the machines in the network. So, the filter expression of the Berkeley packet filter program is set such that the proposed system captures all the packets directed to the host machine in which it runs. It returns the reference to the packet captured to payload extractor.

Payload extractor: This module extracts the payload information from the received IP packet. It gets the header length field of internet protocol and TCP from the received packet header and calculates the size of IP and TCP headers. It also gets the header size of Ethernet header and reference to the packet. Adding the size of IP, TCP and ethernet headers, the total header length of the received packet is obtained. Adding the total header length to the packet reference, payload reference can be obtained.

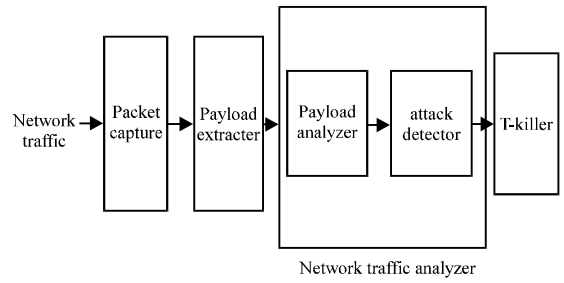

Fig. 1: Architecture of AIS based IDS

Payload analyzer: It receives the payload content from payload extractor and parses the payload character by character. The payload may contain the opcodes of the executable attack components. It calculates the no. of system calls, interrupts and threat level system calls distributed in a packet. It searches for "b0 xx cd 80".These are the opcodes executed for any system call. xx, the number immediately follows b0 is the system call ID in hexa decimal. Using this value the normal and threat level system calls are identified. The number of normal and threat level system calls in a packet is calculated. "cd 80" is the opcode for interrupt. By searching this value interrupt count is obtained.

Attack detector: It receives the interrupt count, system call count and threat level 1 system call count from the payload analyzer and detects buffer overflow attacks. If there are two or more system calls in a packet and one is a threat level one system call, the packet is said to be a part of a buffer over flow attack. If there exists a threat level system call in a packet and there are more interrupts than registered system calls, the packet is part of a buffer overflow attack.

T-killer module: This module discards the packets containing the buffer overflow attacks. As soon as it receives an attack detected signal from the detector it discards the packets containing dangerous components.

IMPLEMENTATION

AIS based IDS can be prototyped in Linux 2.6. Network traffic analysis is done by capturing packets in real time with the help of LIBPCAP, a package for capturing network traffic in Linux. Payloads are extracted from each packet and stored in a buffer. Attack detection is done by identifying the machine instructions of a system call in the payload.

On Linux, system calls are executed by moving the system call ID into the AX register and sending an interrupt signal to kernel. So, move immediate instruction immediately followed by an interrupt instruction represents a system call. The move immediate byte instruction is encoded as "b0". By searching for "b0" instruction and recording the value following it, researchers will be able to find out which system call gets executed. The next pattern to be searched for is the interrupt which is encoded as "cd 80". It maintains an array of threat level system calls and its IDs. It counts the number of system calls when it encounters "b0 xx cd 80" in the received payload. It stores the value next to "b0" and compares it with IDs of threat level one system calls in the database.

If that value matches then the system call received is counted as threat level one system call. It also counts the number of interrupts when it encounters "cd 80" which is machine code for interrupt. If the no. of system calls in a packet is ≥2 and one is a threat level system call or if there is one threat level system call and no. of interrupts is greater than no. of system calls in a packet then that packet is part of a buffer overflow attack.

THE RESULTS OF TESTING

When a remotely executable buffer overflow attack occurs in a remote machine, shell codes causing the attack is found in the payload field of the packet received by that machine. The shell code of this attack contains information about system calls. The proposed system counts the number of threat level one and other system calls. It also counts the number of interrupts in the payload by searching for the opcodes corresponding to them. By using this information the proposed system detects attacks.

The proposed method can able to differentiate both normal and attack packets. The main advantage of the system is, it can detect attacks by searching less number of packets itself. There is no necessity to check all received packets in a message to identify whether the received one is attack or not. During normal session, the system detects all the packets received as normal packets so it has no false negative error and during the attack, the system is able to detect the attack from less number of packets. For example, it can detect the attack from the first 2 packets itself out of 10 received packets. This system is designed to detect remotely executable stack overflow exploits only.

CONCLUSION

The proposed system must have high attack detection accuracy as it uses the immunological principles of human body. In future, this project can be extended to detect any kind of buffer overflow attacks with high detection accuracy by analyzing the host machine at OS kernel parameters this could be achieved. The system calls executed during the state changes of WUFTP daemon can be analyzed because most of the attacks are exploiting this ftp daemon for attacking a remote machine. OS kernel parameters such as Linux kernel signals such as SIGSEGV, SIGILL and SIGBUSER can also be observed to detect all kind of Buffer overflow attacks. In future, this project can be enhanced to detect other attacks such as DOS attack, SYN flood attack, etc.

REFERENCES

Aickelin, U., P. Bentley, S. Cayzer, K. Jungwon and J. McLeod, 2003. Danger theory: The link between ais and ids. Proceedings of the 2nd International Conference on Artificial Immune Systems, September 1-3, 2003, Edinburgh, UK., pp: 147-155.

Cowan, C., C. Pu, D. Maier, J. Walpole and P. Bakke *et al*., 1998. StackGuard: Automatic adaptive detection and prevention of Buffer-overflow attacks. Proceedings of the 7th USENIX Security Symposium, January 26-29, 1998, San Antonio, Texas.

Dasgupta, D., 2006. Advances in artificial immune systems. Comput. Intell. Magazine, 1: 40-49.

Iqbal, N. and M.A. Maarof, 2003. Potential issues in novel computational research: Artificial immune systems. Proceedings of the 7th International Multi Topic Conference, December 8-9, 2003, IEEE, pp: 340-345.

Krizhanovsky, A. and A. Marasanov, 2007. An approach for adaptive intrusion prevention based on the danger. Proceedings of the 2nd International Conference on Availability, Reliability and Security, April 10-13, 2007, Vienna, pp: 1135-1142.

Larochelle, D. and D. Evans, 2001. Statically detecting likely buffer overflow vulnerabilities. Proceedings of the 10th USENIX Security Conference, August 13-17, 2001, Washington, DC.

Lindqvist, U. and P.A. Porras, 1999. Detecting computer and network misuse through the Production-based expert system toolset (P-BEST). Proceedings of the 1999 IEEE Symposium on Security and Privacy, May 9-12, 1999, Oakland, CA., pp: 146-161.

Toth, T. and C. Kruegel, 2002. Accurate buffer overflow detection via abstract payload execution. Distributed Systems Group, Technical University Vienna, Vienna.