An Overview of Buffer Overflow and Network Intrusion Detection and Prevention Systems

Nwokeji, C. E.
ICT Unit, Chukwuemeka Odumegwu Ojukwu University Uli, Nigeria.
Email: ce.nwokeji@coou.edu.ng

Abstract

Cyber-crime is becoming a big business run by organised and sophisticated syndicate with varying degree of Information Technology skills targeting computer applications and programs by running arbitrary code execution into the memory of a program to take over the program. The aim of this work is to review how traditional host based protection mechanism like Data Execution Prevention (DEP), Address Space Layout Randomization (ASLR) and Canaries help to detect and prevent these buffer overflow attacks and how their shortcoming led to the development of Network Intrusion Detection and Prevention Systems (NIDPS). Secondary data was used to reviewed relevant available materials in their effort to examine in-depth performance these technologies. It is recommended that combining anomaly-based with fuzzy logic will be a good technology for intrusion detection and efficient performance that can reduced false positive alerts.

Keyword: Buffer Overflow, Data Execution Prevention, NIDPS, ASLR, Canaries, Snort.

1. Introduction

Applications and software are always under serious and sophisticated attacks from both remote and local attackers who try to run arbitrary code execution into the memory of a program. While traditional host based protection mechanisms like Data Execution Prevention (DEP), Address Space Layout Randomization (ASLR) and Stack Canaries are important mitigating mechanism, they still miss some attacks due to severity and sophistication of current attack mechanism. The cyber warfare is becoming an "arms race" where network administrators and hackers are battling to outsmart each other by developing the best technologies for their various uses. As severity of attacks increases, introduction of more sophisticated security technologies like Network Intrusion Detection and Prevention systems became inevitable.

This work discussed buffer overflow attacks, the traditional host based protection mechanisms against buffer overflow like DEP, ASLR and Canaries, how they help to detect and prevent buffer overflow attacks and their shortcomings. Also, Network Intrusion Detection and Prevention Systems (NIDPS) was extensively discussed, Snort as a type of NIDPS was examined and Signature and Anomaly based NIDPS with their pros and cons were all examined and analysed. The work also looked at the combination of anomaly-based NIDPS with Fuzzy logic as a solution to reduce false positive alert and improve performance in anomaly-based NIDPS.

2. Buffer Overflow Attacks

Buffer Overflow attack is where some section of the memory is over written with more data than required and this will make the data to spills over to other sections of the memory.

Buffers were meant to store a specified amount of data, any extra data written on it will go elsewhere thereby overflowing to adjacent buffer and either corrupt or overwrite valid stored data [1]. In mitigating against buffer overflow attack [2] believed that strongly and securely coded software with automatic bound checks and stack execute invalidation will minimise buffer overflow attacks. In their work [3] also agreed that buffer overflow is more of human error problem, and the solution lies on the use of good language and good software programming paradigm. But warned that the problem will continue if programmers keep

focusing more on software performance and less on security. Data Execution Prevention, Address Space Layout Randomization & Canary were the first mechanisms introduced to fight buffer overflow attack.

2.1 Data Execution Prevention (DEP)

DEP is a combination of hardware and software security system in the modern operating systems that makes additional checks on memory to help to avert malicious codes from executing on a system. According to [1] DEP is an enforcement technology that helps to protect computer systems against any insertion of malicious code into computer memories kept for non-executable codes. DEP helps to prevent code execution from default heap, stack and memory pool pages. Hardware enforcer DEP monitors codes running in these locations and according to [4] raises exception if any code tries to execute. If the exception was not handled promptly, [5] opined that the process will stop. DEP can block programs that want to leverage exception handling mechanism in window.

DEP functions by marking memory pages being indented to hold data only and not executable codes. [6] opined that DEP does this by setting separate bit in the page table entry known as NX (No eXecute). The OS sets the NX bit for the heap & stack memory areas, should faulty program or malware try to execute code from NX identified memory page the CPU will trigger an interrupt which will cause the OS to shut down the application.

DEP cannot be managed and deployed centrally by group policies, it must be configured on the local machines and windows rebooted on each case to take effect, this according to [6] is among the weak points. Also, in DEP, the attacker knows the address of the WinExec without any guess.

2.2 Address Space Layout Randomization (ASLR)

ASLR is a security system that randomly arranges the location of key data areas, it is an exploit mitigation procedure where the memory mapping of a process is placed in random locations. It randomises where the program and its dependencies are loaded in the memory, it means that the program is loaded in a deterministic way. In their work, [1] argued that the randomness of the addresses by ASLR helps in preventing attacker from guessing correctly the address of a shell code and this prevents him from executing any code that will allow him take over the application. An attacker attempting to execute return to libC attack must calculate correctly the location of the target function [7]. It helps to protect against low level attacks like Buffer Overflows, but it can't help to protect high level attacks like SQL Injection, Cross-Site Scripting etc.

2.3 Stack Canaries

When a return address is stored in a stack, a canary value is written before it, any attempt to modify the address by buffer overflow will also modify the canary and an overflow will be detected and program will terminate. It was argued in [8] that canaries work by modifying opening and concluding areas of every function to place and check a value on the stack accordingly so that if stack buffer is overwritten by the time of memory copy operation, error will be observed before execution returns from the copy function. As this occurs an exception will be raised and will be passed back to the exception handler hierarchy up to OS's default exception handler.

Canary, DEP and ASLR do not prevent buffer overflow attacks, as argued by [9] they only try to cope with the consequences after it has happened. They explained that Canaries tries to establish the case of an overflow which overwrite the return address in the stack. DEP goes further to assume that the return address has been overwritten and shadowed so it limits the areas where execution could jump by making the stack

space non-executable. While ASLR goes further than DEP to shuffle around the area where execution is allowed. DEP, ASLR & Canaries are all Host based IDS that according to [5] relies on operating system audit data to monitor and analyse the events generated by programs or users.

2.4 Bypassing DEP, ASLR & Canary

ASLR and DEP can be bypassed by making your shellcode from return oriented programming (ROP) which is an advanced return to LibC attack and jump oriented programming (JOP). Explaining further [5] asserts that ROP and JOP are techniques to bypass these protection mechanisms. This type of attacks doesn't inject executables codes rather they recycle any available codes of that application and use them maliciously. ROP uses an unchecked application stack while JOP involves sets of device addresses and data values loaded into memory, with the device addresses being comparable to opcodes within a new jump oriented machine. JOP attacks are more complex to run and more difficult to prevent. In their opinion ROP chains produced by automated scripts are not suitable for some kinds of vulnerabilities, so using ROP one can develop shellcode. To bypass DEP and Canary the attacker must look for overflows that will allow to overwrite pointers to function to make execution jump into standard library code [6]. The ability to overwrite an exception handler in the stack and force it to point to your code, can lead to comfortable bypassing canary check. These limitations led to the development of advanced NIDPS.

3. Network Intrusion Detection and Prevention System (NIDPS)

Network Intrusion Detection and Prevention System (NIDPS) is a hardware or software technology that monitors network traffic for malicious activity or policy violations and prevents any vulnerability exploits. In their work [10] argued that intrusion occurs when a hacker tries to get into the network through the back door to cause damages or take over the system; NIDPS is there to help prepare for and deal with the attack. It achieves this by collecting data from inbound and outbound traffics, monitor and analyse the data to establish likely security issues. It also documents the existing threats, provides information about intrusion diagnosis, recovery and stops an attack by disconnecting network connection or user session or blocking the target accessibility. Also, [11] opined that IDPS is programmed to automatically detect any intruder, alert network administrator and block the intruder or change the router configuration, thereby providing a real-time corrective response to any attack. NIDPS is common technique of mitigating against buffer overflow attack as argued by [3], it inspects network traffic for patterns showing likely exploits. Once traffic has been marked as suspicious, [11], [12] stated that all traffics from that session will be terminated with little extra processing. NIDPS generates four types of alerts, true positive, true negative, false positive and false negative, but we will only discuss false positive and false negative.

3.1 NIDPS False Positive

This is a condition where legitimate traffic was mistakenly marked as illegitimate. Detecting bad traffics with speed and accuracy is what is needed of every IDPS. When it sees good traffic as bad traffic, it adds extra burden on the network administrators who will spend valuable time investigating false alarm reducing the time to investigate real threats [13]. False positive leads to legitimate traffic being dropped. However, Hubballi and Suryanarayanan [14] argued that IDPS can generate false positive for reasons like -

- ✓ Activity that deviates from normal and sometimes hard to differentiate.
- ✓ An activity that is normal today may be abnormal under another circumstance, a network scan by administrator is normal but abnormal if scanned by another person.
- ✓ A multistage attack which eventually failed at some stage for some other reasons may have raised false alarm.

- ✓ Non-security issues such as misconfigured network equipment, faulty hardware and badly written network-aware software can trigger false positive alarm.

3.2 NIDPS False Negative

This is when an alert that is supposed to happen didn't happen and malicious traffic managed to pass through the IDPS unnoticed. The problem with false positive is that it gives false sense of security and that is not what network administrators want. According to [12] it may be impossible for IDPS to detect every attack, but it should at least bring number of undetected attacks to the barest minimum. Failure to detect bad traffics means that the system has failed to perform optimally and should not be trusted. However, [15] suggested some of the causes of false negative as-

Complexity of network infrastructure can cause snort / IDS sensor to miss some traffic. In a multipath network, a sensor may not be able to inspect all incoming and outgoing traffic.

Signature-based system may not recognise new attacks or signature written tightly to catch only subset of attack vector, also some attack toolkits can obfuscate the attack on the fly.

Poorly written signature in the case of user-created signatures and un-updated signature can allow malicious traffic in.

3.3 IDPS Strengths and Limitations

IDPS has its strengths and weaknesses, as identified by [12] they include the following.

Strengths: NIDPS helps network administrators to monitor very large network with very few devices. It can be deployed with easy into an existing network with minimal network disruption. Also, it is not easily detectable to attackers so it is not vulnerable to direct attack

Limitations: NIDPS can be overwhelmed by network volume and may fail to identify attack. Must have access to all traffic to monitor them, it cannot be able to detect attacks that involved fragmented packets.

4. Snort

Snort is a free and open source network intrusion detection and prevention systems (NIDPS). According to [15] snort is a signature-based network intrusion detection and prevention systems that runs in multiple platform. Its signatures must be updated regularly to keep up with new exploits and attacks, it also requires continuous fine-tuning to minimise false positives. With the use of set rules, [16] argued that snort performs content searching, protocol analysis with predefined patterns to detect variety of attacks like port scanning, SMB probe, buffer overflow, OS fingerprinting attempts etc and generate alert to the administrator. Snort has an advantage of flexible rules and easy configuration. Rules can be quickly added to the rule base in little or no time as soon as new exploits are detected. Again, snort allows for complete raw packet analysis to determine why the alert, the cause of any alert and whether any action is needed. Argued that snort uses conventional mining algorithm like apriori to generate its rules which [17] opined brings about low computational cost. They explained that signature apriori captures packets of unknown signature inputs and uses the input to generate new attack signatures.

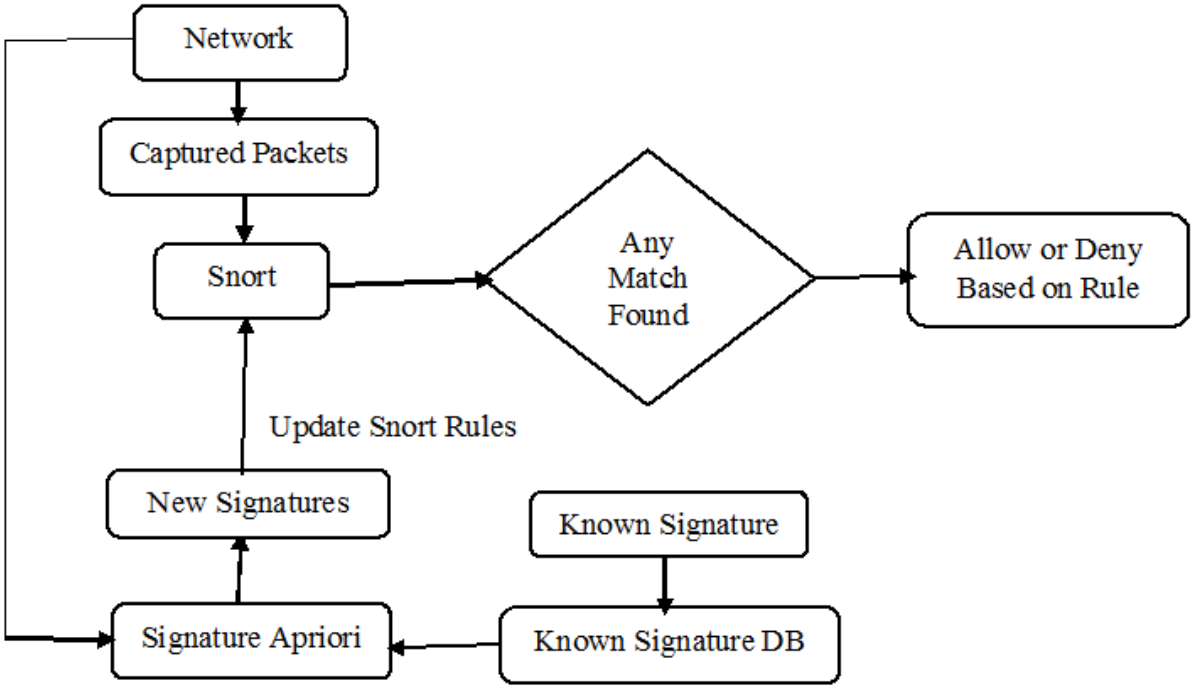

Figure 1: Snort workflow module. Source [17]

External and internal network packets are captured using snorts which allows / deny the packets based on its rules. Unknown signatures, captured packets which are in signature database as input in the signature apriori algorithm which creates new likely signatures and update them as snort rules.

5. Signature-based Intrusion Detection System (IDPS)

Signature-based detection method refers to database for past attacks signatures and known system vulnerabilities. Because every intrusion leaves a footprint, signature-based intrusion has documented evidence of every intrusion. These footprints are known as signature and [11] augured will be used to recognise and prevent a similar attack in the future. Signature-based is highly efficient in detecting known threats, has low false positive and is widely used because most attackers use clear and discrete signatures. In their work, [14] states why signature-based IDPS can give false positive-

- ✓ Analysing the wrong application data or irrelevant traffic portion for a signature match can lead to false positive.
- ✓ Writing a good signature is highly technical, this technical proficiency is prone to error.

5.1 Advantages of Signature-Based Detection

- ✓ Signature-based detection records low false positive alert
- ✓ The rate and accuracy of detecting intrusion is high with Signature-based detection
- ✓ The low false positive saves network administrators time spend in investigating alarms.
- ✓ Detecting causes of alarms are easy because of log files it keeps.

5.2 Disadvantages of Signature-Based Detection

- ✓ Signature-based detection is weak in detecting unknown or new threats.
- ✓ The signature databased must be regularly updated to cover these new and unknown threats.
- ✓ Updating of signature database for new attacks can take days and this delay may cause security breach.

- ✓ With huge amount of traffic, the system may find it difficult to inspect every packet thereby forcing some packets to drop.

6. Anomaly-based Intrusion Detection Prevention System

Anomaly-based detection method analyses packet flow in the network to identity abnormalities in the normal pattern or acceptable internet communication rules. The mission is to identify traffics that violets the rules and once this is done, it makes a network security decision. It makes use of established pattern of the normal system activities or behaviours to recognise active intrusion attempts, according to [11] any deviation from the normal pattern will trigger the alert. Anomaly-based IDPS do have normal behaviour profile in areas like hosts, users and network connections, this profile [13] argued is developed by monitoring the features of the network activities over a long period. Anomaly-based IDPS has three categories

Behavioural analysis checks for any anomalies in the types of already known behaviour in relation to packets already sent to the network.

Traffic-pattern analysis looks for definite pattern in the network traffic.

Protocol analysis is for identifying likely attacks that may have not been known. Protocol analysis [18] stated is good at preventing buffer overflow which try to exploit programming error that permits infinite number of data to be read to a fixed-size memory.

6.1 Advantages of Anomaly-Based Detection

- ✓ With anomaly-based detection new and unknown threats can be detected without waiting to update the database.
- ✓ As the system works it keeps studying the network traffic patterns and builds its profiles automatically.
- ✓ The longer the system works the better it becomes in threat identification.

6.2 Disadvantages of Anomaly-Based Detection

- ✓ Anomaly-based is always associated with high false positive
- ✓ The network maybe vulnerable during the time the system is building its profile.
- ✓ Any malicious activity that is disguised to look normal may not be detected.
- ✓ High false positive will lead to network administrators spending too much time investigating alarms.

7. Combination of Anomaly-Based NIDPS with Fuzzy Logic

Analysis of current NIDPS level of protection against buffer overflow and other attack shows the need to address some of its shot comings to improve performance. Anomaly-based is better in detecting unknown threats than signature-based but it is associated with high false positive alert thus, network administrators prefer signature-base which is associated with low false positive alert. However, effective combination of anomaly-based NIDPS with fuzzy logic can minimise the high false positive alerts and enhance its performance.

Fuzzy logic is a technology that induces computers to make decisions like a human being do by using its own precise quantities more like human brain. Fuzzy logic relies on fuzzy sets which [19] argues allows it to deal with situation that are not precise. It is based on degree of truth rather than true or false Boolean logic which is the bases of our modern NIDPS. It allows members to belong to two groups and have a membership value of each group. This functions [20] argued enable systems to overcome the difficulty of

having very different control actions for a small change in the measured variable. Its ability to handle huge amount of complex and dynamic dataset can help it in detecting known and unknown threats with high accuracy without dropping packets. Fuzzy logic sets and rules were designed to make the system more accurate for detecting attacks using the fuzzy inference approach, this will be more reliable in detecting intrusion in a network, reduce the false positive alerts and prioritise the alert by telling network administrators which alert that needs urgent attention.

Conclusion

This work reviews buffer overflow and other attacks, how DEP, ASLR and Canary mitigate against them and its limitations. How the limitations brought about the development of other technologies like NIDPS and how they can monitor and detect the almost common attack such as buffer overflow. NIDPS false positive and false negative alerts were examined and its strengths and limitations considered. Snorts as a type of NIDPS was explained in full and signature and anomaly-based NIDPS were critically reviewed highlighting their differences, advantages and disadvantages. The work recommended research work on combining anomaly-based IDPS with fuzzy logic for efficient performance and reduced false positive alerts.

Reference

[1] S. Alouneh, M. Kharbutli, and R. AlQurem, "Stack memory buffer overflow protection based on duplication and randomization," *Procedia Comput. Sci.*, vol. 21, pp. 250–256, 2013.

[2] A. A. Ghorbani, W. Lu, and M. Tavallaee, *Network Intrusion Detection and Prevention: Concepts and Techniques*. New York: Springer, 2009.

[3] D. J. Day and Z. X. Zhao, "Protecting against address space layout randomisation (ASLR) compromises and return-to-libc attacks using network intrusion detection systems," *Int. J. Autom. Comput.*, vol. 8, no. 4, pp. 472–483, 2011.

[4] V. Iyer, A. Kanitkar, P. Dasgupta, and R. Srinivasan, "Preventing overflow attacks by memory randomization," *Proc. - Int. Symp. Softw. Reliab. Eng. ISSRE*, pp. 339–347, 2010.

[5] V. Katoch, "Bypassing ASLR/DEP."

[6] N. Stojanovski, M. Gusev, D. Gligoroski, and S. J. Knapskog, "Bypassing Data Execution Prevention on MicrosoftWindows XP SP2," in *The Second International Conference on Availability, Reliability and Security (ARES'07)*, 2007, pp. 1222–1226.

[7] H. Marco-Gisbert, "On the Effectiveness of Full-ASLR on 64-bit Linux," 2014.

[8] M. Bishop, D. Howard, S. Engle, and S. Whalen, "A Taxonomy of Buffer Overflow Preconditions," *Security*, vol. 9, no. July 2006, pp. 1–18, 2010.

[9] L. Jai and R. Shah, "Safe Stack : Automatically Patching Stack Based Buffer Overflow Vulnerabilities," vol. 4, no. 2, pp. 139–144, 2014.

[10] A. Sharifi, F. F. Zad, F. Farokhmanesh, A. Noorollahi, and J. Sharif, "An Overview of Intrusion Detection and Prevention Systems (IDPS) and Security Issues," *IOSR J. Comput. Eng.*, vol. 16, no. 1, pp. 47–52, 2014.

[11] H. El-Taj, F. Najjar, H. Alsenawi, and M. Najjar, "Intrusion Detection and Prevention Response based on Signature-Based and Anomaly-Based: Investigation Study."

[12] H. R. El-Taj, "Intrusion Detection and Prevention Systems (IDPS) State of Art: IDPS Challenges," *Int. J. Comput. Sci. Inf. Secur.*, vol. 13, no. 9, p. 28, 2015.

[13] A. Kundu, T. K. Kundu, and I. Mukhopadhyay, "Survey on Intrusion Detection and Prevention System: A MANET Perspective," *Int. J. Sci. Eng. Res.*, vol. 3, no. 9, 2012.

[14] N. Hubballi and V. Suryanarayanan, "False alarm minimization techniques in signature-based intrusion detection systems: A survey," *Comput. Commun.*, vol. 49, pp. 1–17, 2014.

[15] K. Cox and C. Gerg, *Managing Security with Snort and IDS Tools*. Califonia: O Reilly, 2004.

[16] A. Alhomoud, R. Munir, J. P. Disso, I. Awan, and A. Al-Dhelaan, "Performance evaluation study of Intrusion Detection Systems," *Procedia Comput. Sci.*, vol. 5, pp. 173–180, 2011.

[17] C. N. Modi, D. R. Patel, A. Patel, and M. Rajarajan, "Integrating Signature Apriori based Network Intrusion Detection System (NIDS) in Cloud Computing," *Procedia Technol.*, vol. 6, pp. 905–912, 2012.

[18] A. A. Nasr, M. M. Ezz, and M. Z. Abdulmaged, "An Intrusion Detection and Prevention System based on Automatic Learning of Traffic Anomalies," *I.J. Comput. Netw. Inf. Secur.*, vol. 1, no. 1, pp. 53–60, 2016.

[19] S. Partha Sarathi Bhattacharjee and S. Ara Begum Associate Professor, "A Study on Fuzzy Rules for Intrusion Detection System," *Int. J. Res. Stud. Comput. Sci. Eng.*, vol. 2, no. 1, pp. 2349–4840, 2015.

[20] P. Sarathi Bhattacharjee and S. Ara Begum, "Fuzzy Approach for Intrusion Detection System: A Survey," *Int. J. Adv. Res. Comput. Sci.*, vol. 4, no. 2.