International Journal of Computer Science and Network (IJCSN)
Volume 1, Issue 5, October 2012 www.ijcsn.org ISSN 2277-5420

Prevention of Buffer overflow Attack Blocker Using IDS

[1]Pankaj B. Pawar, [2]Malti Nagle, [3]Pankaj K. Kawadkar

[1] PIES Bhopal, RGPV University,
Bhopal, M.P., India

[2] PIES Bhopal, RGPV University,
Bhopal, M.P., India

[3] PIES Bhopal, RGPV University,
Bhopal, M.P., India

Abstract

Now a day internet threat takes a blended attack form, targeting individual users to gain control over networks and data. Buffer Overflow which is one of the most occurring security vulnerabilities in a Computer's world. Buffer Overflow occurs while writing data to a buffer and it overruns the buffer's boundary and overwrites it to a adjacent memory. The techniques to exploit buffer overflow vulnerability vary per architecture, Operating system and memory region. There are various exploitation which causes to buffer overflow attack as stack based exploitation, heap based exploitation and choice of programming language and many more. Which may result in erratic program behavior, including memory access errors, incorrect results, a crash or a breach of system security. C and C++ are the two programming languages which do not check that data has overwritten to an array that results to an buffer overflow. There are many more techniques which has been used for protecting the Computer from buffer overflow attack We are proposing a novel techniques for preventing data loss during the transmission of images of different formats. In this paper we have discuss and compare certain tools and techniques which prevent buffer overflows. We have also discuss some modern tools and techniques with their pros and cons.

Keywords: *Buffer Overflow, IDS, Malicious code, Intrusion, thread.*

1. Introduction

Computer Security includes the protection of information and property from theft, corruption while allowing the information to remain accessible to its intended user. Computer Security means valuable information and services are protected from publication, tampering or collapse by unauthorized activities or events. A buffer overflow occurs when data written to a fixed sized buffer, due to insufficient bound checking, corrupts data values in memory addresses adjacent to the allocated buffer.
A Buffer overflow attack is an attack in which a malicious user exploits an unchecked buffer in a program and overwrites the program code with their-own data. If the program code is overwritten with new executable code, the effect is to change the program's operation as dictated by the attacker. If overwritten with other data, the likely effect is to cause the program to crash. Today‘s software has been widely targeted by buffer overflows Such exploits range from arbitrary code execution on the victim‘s computer. to denial of service (DoS) attacks. Detecting and eliminating buffer overflows would thus make software far more secure. There are many more tools and technologies for detecting and preventing buffer overflow and other vulnerabilities but still there are some pros and cons of certain technique.

Network threat can attack through three steps i.e. penetrate, launch and propagate without human intervention. Network-based attacks on the host predominantly exploit vulnerabilities in protocols and network-aware processes. These vulnerabilities are typically the result of programming errors which provide opportunities for a buffer overflow. There are several different approaches for finding and preventing buffer overflows. These include enforcing secure coding practices, statically analyzing source code, halting exploits via operating system support, and detecting buffer overflows at runtime [5]. The general idea is to overflow a buffer so that it overwrites the return address. When the function is done it will jump to whatever address is on the stack. We put some code in the buffer and set the return address to point to it. Network Based Threat can Prevented by using Personal firewalls, Intrusion detection systems and Buffer overflow exploit prevention technique.

2. Existing Study

Xinran Wang, Chi-Chun Pan, Peng Liu, and Sencun Zhu proposed work on SigFree: A Signature-Free Buffer

Overflow Attack Blocker [1]. There experimental study shows that the dependency-degree-based SigFree could block all types of code-injection attack packets (above 750) tested in our experiments with very few false positives. Moreover, SigFree causes very small extra latency to normal client requests when some requests contain exploit code.

Eric Haugh and Matt Bishop proposed work on Testing C Programs for Buffer Overflow Vulnerabilities[2] This evaluation shows that the tool is useful for finding buffer overflow flaws, that it has a low false positive rate, and compares well with other techniques.

Crispin Cowan, Perry Wagle, Calton Pu, Steve Beattie, and Jonathan Walpole proposed work on Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade*[3] . They consider which combinations of techniques can eliminate the problem of buffer overflow vulnerabilities, while preserving the functionality and performance of existing systems.

Hassen Sallay, Khalid A. AlShalfan, Ouissem Ben Fred j proposed work on A scalable distributed IDS Architecture for High speedNetworks [4]. They worked on switch-based splitting approach that supports intrusion detection on high-speed links by balancing the traffic load among different sensors running Snort.

We are proposing technique using IDS (Intrusion Detection System) for detecting malicious code penetrated by attacker while transmission of images. This technique capable of detecting malicious code by applying Pattern matching scheme. We separate out the original image from malicious image by applying IDS. Each time during receiving end we check for malimage and if malimage found then we are applying IDS to it for getting correct image.

3. Network-based attacks Detection and Prevention Techniques

Network-based attacks can penetrate, launch and propagate without human intervention. Network-based attacks on the host predominantly exploit vulnerabilities in protocols and network-aware processes. These vulnerabilities are typically the result of programming errors which provide opportunities for a buffer overflow. Network-based attacks can be protected by using Firewall, Intrusion Prevention System and Buffer Overflow Exploit Prevention.

Personal firewalls are deployed form of host protection, and defend against attacks using network threat vectors in the pre-launch phase before they affect the system. By blocking access to ports, single IP addresses or ranges of IP addresses, protocols and services not needed for legitimate business, personal firewall can reduce, but not eliminate.

Intrusion detection systems (IDS) provide deep packet inspection capabilities that examine the traffic allowed through by personal firewall rules and alert the user to an attack on the host system. IPS technology is with the ability to identify good traffic from malicious traffic in real time. It is categorized into either signature-based methods or protocol analysis-based methods. Signature-based techniques are effective at stopping known exploits, but are often too reactive. As the time decreases between vulnerability disclosure and the release of rapidly propagating, highly infectious worms, signature-based IPS techniques tend to be of less value. In contrast, vulnerability based protocol analysis proves very effective against modern, fast-moving attacks, and since it's based on shielding vulnerabilities, often provides protection even before attacks are released.

One of the newest host protection technologies available is buffer overflow exploit prevention, also known as memory protection. As a high-level rule, code should never be executed from writable areas of system memory. By watching the use of Stack and Heap system memory, BOEP identifies if a buffer overflow has succeeded and attempts to thwart its executable payload.

A personal firewall will block known and unknown attacks against the ports and services you don't need. IPS will filter out known and unknown attacks against known vulnerabilities. At a cost, buffer overflow exploit prevention will provide the necessary insurance for overflows against unknown vulnerabilities.

Methodology

To overcome the problem of buffer overflow we proposed the SigFree attack blocker technique. The background behind the SigFree is motivated by an important observation that "the nature of communication to and from network services is predominantly or exclusively data and not executable code" [12].

Since remote exploits are typically binary executable code, this observation indicates that if we can correctly distinguish (service requesting) messages containing byte code from those containing no byte code, we can protect most Internet services (which accept data only) from code injection buffer overflow attacks by blocking the messages that contain binary code.

Block diagram

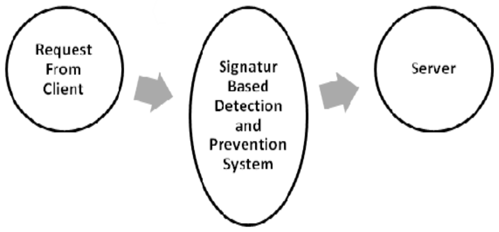
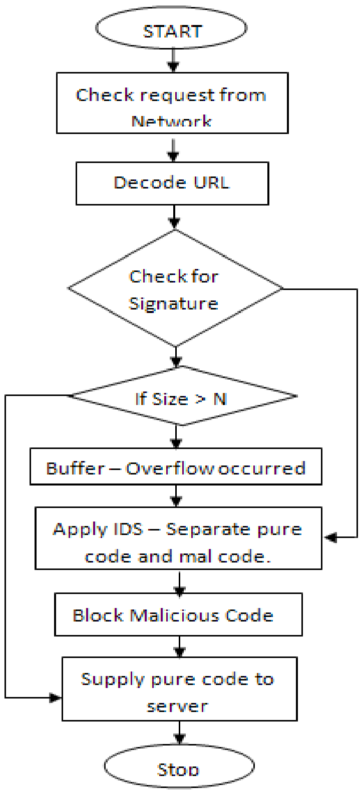


Figure 1: Block Diagram of System.

Algorithm

Input: Data from network

```
Step 1: Check request from Network
Step 2: Decode the URL
Step 3: Check for signature
Step 4: Signature found goto Step 7
Step 5: Check size of Data
Step 4: If size < N (Buffer overflow not occurred)
Step 5:Goto step 7
Step 5: Apply IDS
Step 6: IDS separates pure code and
        malcode
Step 7: Supply pure code to Server.
Step 8: Stop.
```


START
Check request from Network
Decode URL
Check for Signature
If Size > N
Buffer – Overflow occurred
Apply IDS – Separate pure code and mal code.
Block Malicious Code
Supply pure code to server
Stop

Flowchart

We implementing Intrusion Detection System (IDS) to handle the buffer overflow attack occurred during transmission of data. Intrusion detection systems (IDS) provide deep packet inspection capabilities that examine the traffic allowed through by personal firewall rules and alert the user to an attack on the host system. Intrusion Prevention System (IPS) technology is with the ability to identify good traffic from malicious traffic in real time. It is categorized into either signature-based methods or protocol analysis-based methods.

Signature-based techniques are effective at stopping known exploits, but are often too reactive. As the time decreases between vulnerability disclosure and the release of rapidly propagating, highly infectious worms, signature-based IPS techniques tend to be of less value. In contrast, vulnerability based protocol analysis proves very effective against modern, fast-moving attacks, and since it's based on shielding vulnerabilities, often provides protection even before attacks are released. One of the newest host protection technologies available is buffer overflow exploit prevention, also known as memory protection. As a high-level rule, code should never be executed from writable areas of system memory. By watching the use of Stack and Heap system memory, BOEP identifies if a buffer overflow has succeeded and attempts to thwart its executable payload.

4. Proposed Methodology

4.1 Modern techniques to detect and prevent Unknown threat and vulnerability

There are two types of modern threat i.e. Network Attack and Application Attack. There are three phases host attack. They are Penetration, Launch and Prorogation. The compromise of a host which allows further malicious activity to occur. Penetration can occur through e-mail, Web Browsers, remote buffer overflow or various other methods. Launch the execution of the attack's malicious payload. Launch method can range from user double click to remote memory buffer overflow. Prorogation is post compromise activity intended to replicate, retrieve other component, transmit data or enable remote control.

4.2 Network-based attacks Detection and Prevention Techniques

Network-based attacks can penetrate, launch and propagate without human intervention. Network-based attacks on the host predominantly exploit vulnerabilities in protocols and network-aware processes. These vulnerabilities are typically the result of programming errors which provide opportunities for a buffer overflow. Network-based attacks can be protected by using Firewall, Intrusion Prevention System and Buffer Overflow Exploit Prevention.

Personal firewalls are deployed form of host protection, and defend against attacks using network threat vectors in the pre-launch phase before they affect the system. By blocking access to ports, single IP addresses or ranges of IP addresses, protocols and services not needed for legitimate business, personal firewall can reduce, but not eliminate.

Intrusion detection systems (IDS) provide deep packet inspection capabilities that examine the traffic allowed through by personal firewall rules and alert the user to an attack on the host system. IPS technology is with the ability to identify good traffic from malicious traffic in real time. It is categorized into either signature-based methods or protocol analysis-based methods. Signature-based techniques are effective at stopping known exploits, but are often too reactive. As the time decreases between vulnerability disclosure and the release of rapidly propagating, highly infectious worms, signature-based IPS techniques tend to be of less value. In contrast, vulnerability based protocol analysis proves very effective against modern, fast-moving attacks, and since it's based on shielding vulnerabilities, often provides protection even before attacks are released.

One of the newest host protection technologies available is buffer overflow exploit prevention, also known as memory protection. As a high-level rule, code should never be executed from writable areas of system memory. By watching the use of Stack and Heap system memory, BOEP identifies if a buffer overflow has succeeded and attempts to thwart its executable payload.

A personal firewall will block known and unknown attacks against the ports and services you don't need. IPS will filter out known and unknown attacks against known vulnerabilities. At a cost, buffer overflow exploit prevention will provide the necessary insurance for overflows against unknown vulnerabilities.

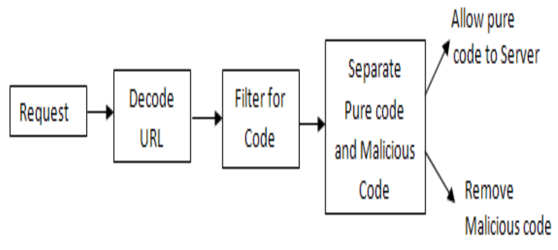

Request
Decode URL
Filter for Code
Separate Pure code and Malicious Code
Allow pure code to Server
Remove Malicious code

Figure 2: Malicious code removal system

5. Limitations

SigFree does not detect attacks that just corrupt control flow or data without injecting code.

If the buffer being overflowed is inside a JPEG or GIF system, ASN.1 or base64 encoder, SigFree cannot be directly applied. Although SigFree can decode the protected file according to the protocols or applications it protects, more details need to be studied in the future.

The mechanism of code abstraction technique and its robustness to obfuscation are not related to any hardware platform. Therefore, we believe that detection capabilities and resilience to obfuscation will be preserved after porting. We will study this portability issue in our future work.

6. Conclusion

We have proposed SigFree, an signature-free malicious code blocker system that can filter code-injection buffer overflow attack, one of the most serious cyber security paradigm. SigFree does not require any signatures, thus it can block new malicious code and provide security for the systems. SigFree is less affected from malicious attack, and economical for deployment with little maintenance cost and low performance overhead.

References

[1] Xinran Wang, Chi-Chun Pan, Peng Liu, and Sencun Zhu, "SigFree: A Signature-Free Buffer Overflow Attack Blocker", Ieee Transactions On Dependable And Secure Computing, Vol. 7, No. 1, January-March 2010.
[2] Eric Haugh and Matt Bishop, "Testing C Programs for Buffer Overflow Vulnerabilities".
[3] Crispin Cowan, Perry Wagle, Calton Pu, Steve Beattie, and Jonathan Walpole "Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade", http://www.cse.ogi.edu/DISC/projects/immunix.
[4] Hassen Sallay, Khalid A. AlShalfan, Ouissem Ben Fred j," A scalable distributed IDS Architecture for High speed Networks", IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.8, August 2009.

[5] E. Barrantes, D. Ackley, T. Palmer, D. Stefanovic, and D. Zovi, "Randomized Instruction Set Emulation to Disrupt Binary Code Injection Attacks," Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03), Oct. 2003.

[6] J. Newsome and D. Song, "Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software," Proc. 12th Ann. Network and Distributed System Security Symp. (NDSS), 2005.

[7] B.A. Kuperman, C.E. Brodley, H. Ozdoganoglu, T.N. Vijaykumar, and A. Jalote, "Detecting and Prevention of Stack Buffer Overflow Attacks," Comm. ACM, vol. 48, no. 11, 2005.

[8] M. Costa, J. Crowcroft, M. Castro, A. Rowstron, L. Zhou, L. Zhang, and P. Barham, "Vigilante: End-to-End Containment of Internet Worms," Proc. 20th ACM Symp. Operating Systems Principles (SOSP), 2005.

[9] J. Pincus and B. Baker, "Beyond Stack Smashing: Recent Advances in Exploiting Buffer Overruns," IEEE Security and Privacy, vol. 2, no. 4, 2004.

[10] G. Kc, A. Keromytis, and V. Prevelakis, "Countering Code-Injection Attacks with Instruction-Set Randomization," Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03), Oct. 2003.