ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
*A Peer Reviewed Open Access International Journal*

# Detection of Malicious Behavior with NDIS Intermediate Drivers

**Vimal Kumar Gangwar**
**Department of Computer Science and Engineering,**
**ICAR- Indian Institute of Farming System and Research,**
**Meerut, Uttar Pradesh 250 001, India.**

***ABSTRACT***
***In the progression of technology world, the malicious harmful code is also improved their evolution which often hides its malicious behavior in various methods which might be bind with other code. This will mainly effect to the network communication. This cover-up technique poses difficulties to security mechanisms, which will detect the malicious harmful behavior codes. In this paper, we give an overview of the detection of malicious data and blocking, a new approach to computer security via malicious data detection and automatic blocking software. In particular, this project focuses on building a united executable program analysis platform and using it to provide unique solutions to an aware the future loss of their data and other security issues. We propose a technique for the Network Driver Interface Specification (NDIS) integrate together with a united malicious software detection platform. The NDIS model supports hybrid network transport NDIS drivers, called NDIS intermediate drivers. This driver lies between transport driver and NDIS driver. This can be see the entire network traffic taking place on a system as the drivers lie between protocol drivers and network drivers. By introducing security-related approaches from network traffic directly, our project permits a principled, root cause based method to computer security and provides an effective solutions.***

***Keywords—malicious behavior; network driver interface specification; code abstraction; data security; vulnerability; code injection attack.***

## Introduction

The main Objective of this project is to develop a real time Signature free buffer overflow attack model, to protect from buffer overflow attacks in Internet services. Buffer overflow attacks typically contain executable whereas legitimate client requests never contain executable in most Internet services such as Web Services, SQL Services, BIND, SNMP, and other remote access services. Based on this observation, Sig Free blocks attacks by detecting the presence of code. Throughout the history of cyber security, buffer overflow is one of the most serious vulnerabilities in computer systems. Buffer overflow vulnerability is a root cause for most of the cyber-attacks such as server breaking-in, worms, zombies etc. A buffer overflow occurs during program execution when a fixed-size buffer has had too much data copied into it. This causes the data to overwrite into adjacent memory locations, and, depending on what is stored there, the behavior of the program itself might be affected.” (Note that the buffer could be in stack or heap.)Sig Free is an application layer blocker that typically stays between a service and the corresponding firewall. When a service requesting message arrives at Sig Free, Sig Free first uses a new O (N) algorithm, where N is the byte length of the message, to disassemble and distil all possible instruction sequences from the message’s payload, where every byte in the payload is considered as a possible starting point of the code embedded.Sig Free can filter out code-injection buffer overflow attack messages targeting at various Internet services such as web service. Sig Free blocks attacks by detecting the presence of code. Sig Free first blindly dissembles and extracts instruction sequences from a request. It then applies a novel technique called code abstraction, which

**Cite this article as:** Vimal Kumar Gangwar, "Detection of Malicious Behavior with NDIS Intermediate Drivers", International Journal & Magazine of Engineering, Technology, Management and Research, Volume 7 Issue 2, 2020, Page 29-33.

uses data flow anomaly to crop useless instructions in an instruction sequence. Finally it compares the number of useful instructions to a threshold to determine if this instruction sequence contains code. Sig Free is signature free, thus it can block new and unknown buffer overflow attacks and also update the new threats. Sig Free is transparent to the servers being protected; it is good for economical Internet wide deployment with very low deployment and maintenance cost. Sig Free would block all types of code injection attack packets that are tested in our project.

## Exiting system

Detection of Data Flow Anomalies there is static or dynamic methods to detect data flow anomalies in the software reliability and testing field. Static methods are not suitable in our case due to its slow speed; dynamic methods are not suitable either due to the need for real execution of a program with some inputs.

## Drawbacks

- Existing systems does not identifies new and unknown threats.
- Rely on string-matching and uses limited rules.
- Requires changes to legacy systems required
- Existing system slows down the system.
- Existing systems requires updating and so increases the cost.

## Proposed system

Their scheme is rule-based, whereas Sig Free is a genericapproach which does not require any pre-known patterns. Then, it uses the found patterns and a data flow analysis technique called program slicing to analyze the packet's payload to see if the packet really contains code Four rules (or cases) are discussed in their project: Case 1 not only assumes the occurrence of the call/jmp instructions, but alsoexpects the push instruction appears before the branch; Case 2 relies on the interruptinstruction; Case 3 relies on instruction *ret*; Case 4 exploits hidden branch instructions. Besides, they used a special rule to detect polymorphic exploit code which contains a loop. Although they mentioned that the above rules are initial sets and may require updating with time, it is always possible for attackers to bypass those pre-known rules. Moreover, more rules mean more overhead and longer latency in filtering packets. In contrast, Sig Free exploits a different data flow analysis technique, which is much harder for exploit code to evade

## Advantages:

- Sig Free is Signature Free recognises new and unknown threats.
- Does not rely on string-matching and very resistant to attack code altering.
- Uses generic code-data separation criteria instead of limited rules.
- No changes to legacy systems required and do not slow you down the system.
- Sig Free is an economical deployment with extremely low maintenance cost.
- Sig Free is the first technique that can detect self-modifying code without any runtime analysis.
- Sig Free can handle polymorphism, encryption, metamorphism, self-modifying, and anti-disassembly, anti-emulation.
- We proposed Sig Free, a real-time, signature free, out of- the-box blocker that can filter code-injection buffer overflow attack messages, one of the most serious cyber security threats, to various Internet services. Sig Free does not require any signatures, thus it can block new, unknown attacks.

## Feasibility study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- Economical feasibility
- Technical Feasibility
- Social Feasibility

**Economical Feasibility:**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

**Technical Feasibility:**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

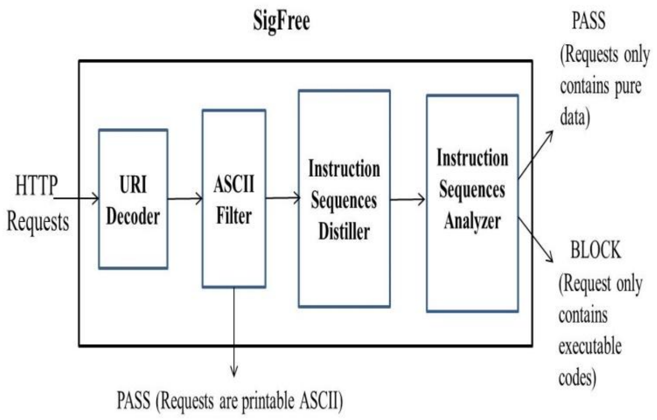
**Social Feasibility:**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

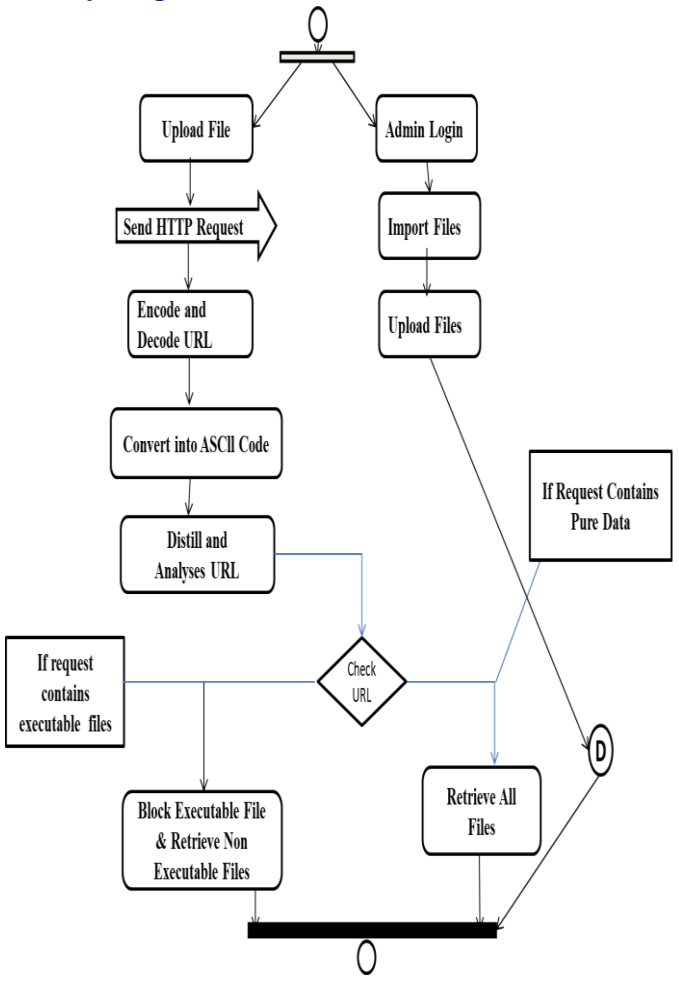
**Main Modules**

- Sig Free Attack Model
- URI decoder.
- ASCII Filter.
- Instruction sequences distiller (ISD).
- Instruction sequences analyzer.

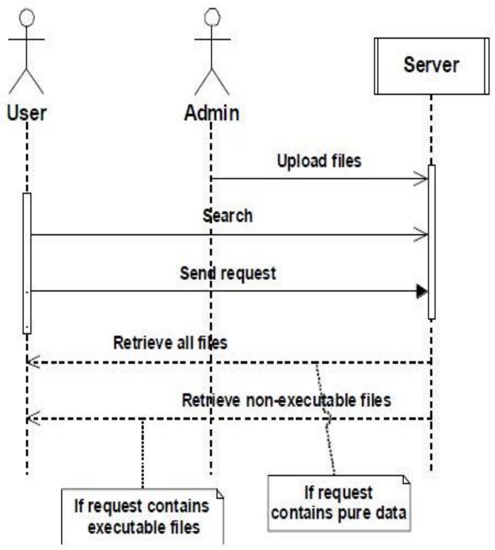
**Architecture:**


SigFree
HTTP Requests
URI Decoder
ASCII Filter
Instruction Sequences Distiller
Instruction Sequences Analyzer
PASS (Requests only contains pure data)
BLOCK (Request only contains executable codes)
PASS (Requests are printable ASCII)

**Activity Diagram:**

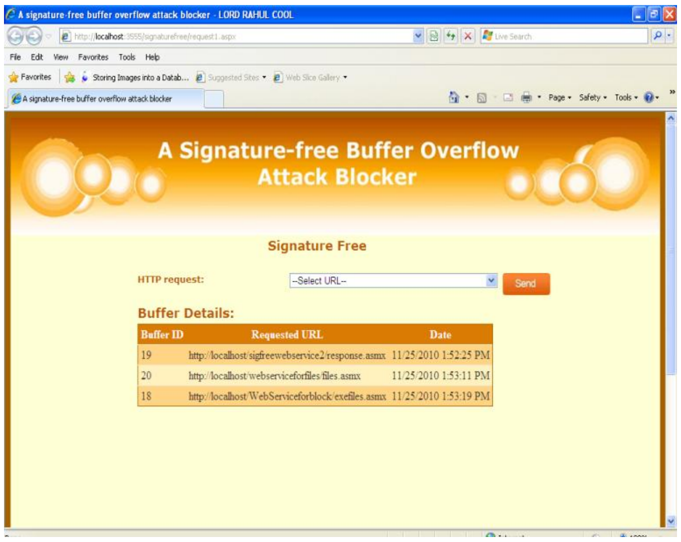


**Sequential Diagram:**


User
Admin
Server
Upload files
Search
Send request
Retrieve all files
Retrieve non-executable files
If request contains executable files
If request contains pure data

**Sig Free Attack Model:**

- A malicious payload may be embedded in the Request-URL field as a query parameter.
- An attacker can use any request method and embed the malicious code data in any field. So we create all possible Attack Model in this module to test the proposed system.


A signature-free buffer overflow attack blocker - LORD RAHUL COOL
A Signature-free Buffer Overflow Attack Blocker
Send

Signature Free

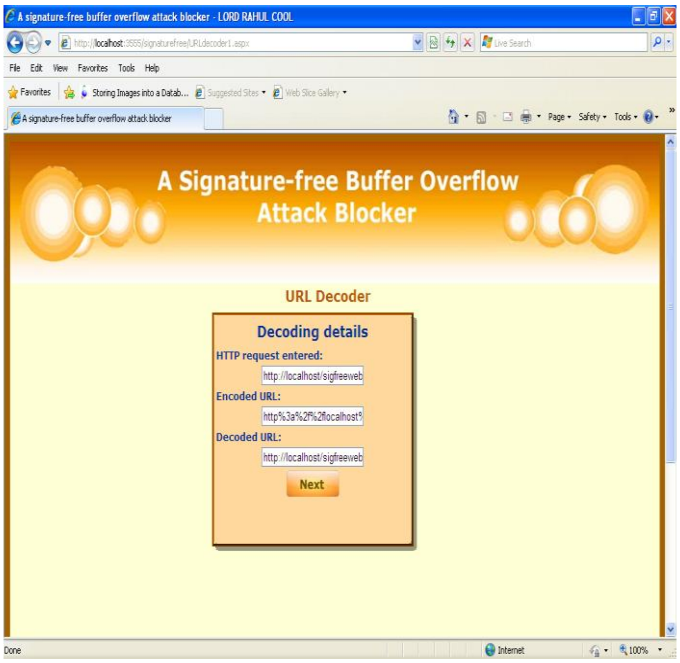
HTTP request: --Select URL--

Buffer Details:

| Buffer ID | Requested URL | Date |
|---|---|---|
| 19 | http://localhost/sigfreewebservice2/response.asmx | 11/25/2010 1:52:25 PM |
| 20 | http://localhost/webserviceforfiles/files.asmx | 11/25/2010 1:53:11 PM |
| 18 | http://localhost/WebServiceforblock/exefiles.asmx | 11/25/2010 1:53:19 PM |

**URL Decoder:**

URL decoder is used because a malicious payload may be embedded in the Request-URL as a request parameter.The first step of Sig Free is to decode the Request-URL.


A signature-free buffer overflow attack blocker - LORD RAHUL COOL
A Signature-free Buffer Overflow Attack Blocker
Next

URL Decoder

Decoding details

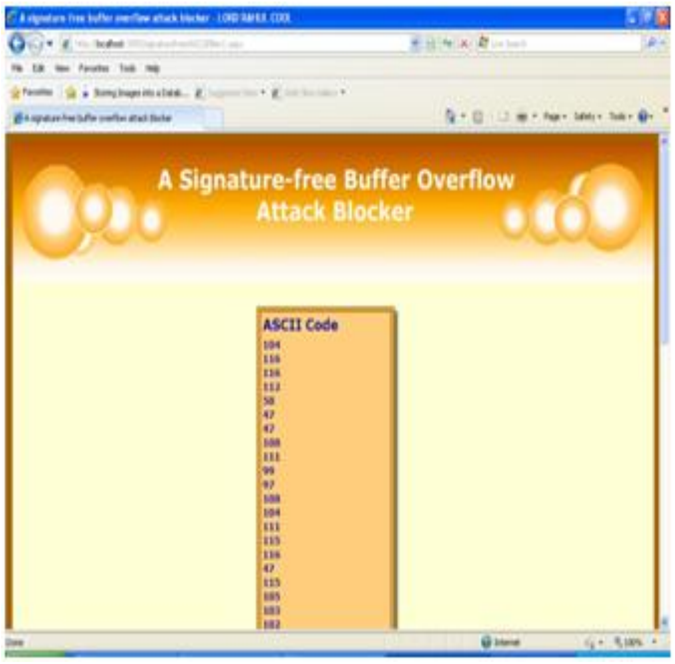
HTTP request entered: http://localhost/sigfreeweb

Encoded URL: http%3a%2f%2flocalhost%

Decoded URL: http://localhost/sigfreeweb

**ASCII Filter:**

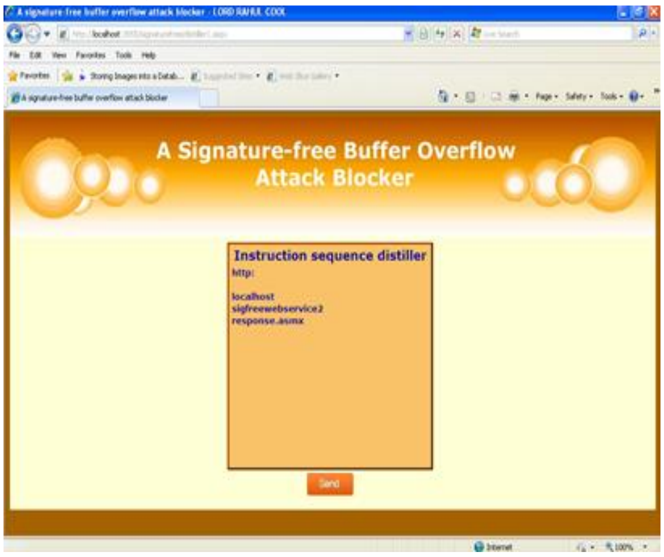
- Malicious executable code is normally binary strings.
- In order to guarantee the throughput and response time of the protected web system, if the query parameters of the Request-URL and Request-body of a request are both printable ASCII codes.


A signature-free buffer overflow attack blocker - LORD RAHUL COOL
A Signature-free Buffer Overflow Attack Blocker

ASCII Code
104
116
116
112
58
47
47
108
111
99
97
108
104
111
115
116
47
115
105
103
102

### Instruction Sequences Distiller(ISD):

This module distills all possible instruction sequences from the query parameters of Request-URL and Request-body.

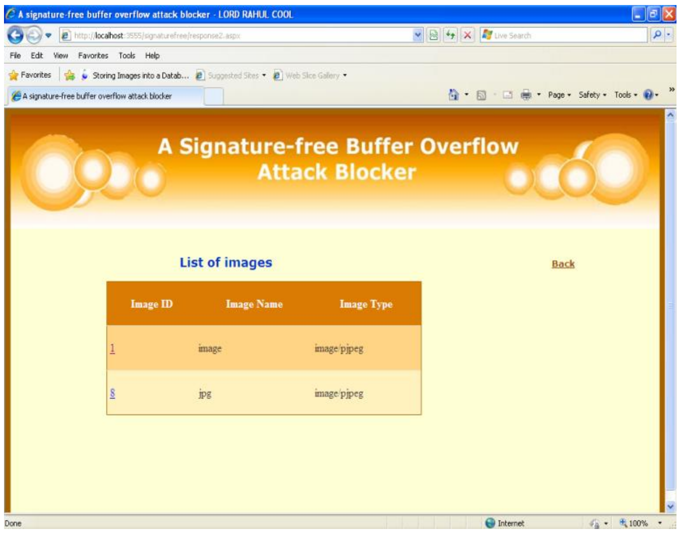

A signature-free buffer overflow attack blocker - LORD RAHUL COOL
A Signature-free Buffer Overflow Attack Blocker
Instruction sequence distiller
Send

http:

localhost
sigfreewebservice2
response.asmx

### Instruction Sequences Analyzer (ISA):

Using all the instruction sequences distilled from the instruction sequences distiller as the inputs, this module analyzes these instruction sequences to determine whether one of them is a program.


A signature-free buffer overflow attack blocker - LORD RAHUL COOL
A Signature-free Buffer Overflow Attack Blocker
List of images
Back

| Image ID | Image Name | Image Type |
|---|---|---|
| 1 | image | image/pjpeg |
| 8 | jpg | image/pjpeg |

### Future Enhancements

As future enhancements the Sig Free should be upgraded up to the level of zero false negatives, which must also detect more number of attacks with higher accuracy and throughput.

### Conclusion

We proposed Sig Free, a real-time, Signature Fee, out of- the-box locker that can filter code-injection buffer overflow attack messages, one of the most serious cyber security threats, to various internet application services used by day to day work. Sig Free does not require any signatures, which can block new, unknown attacks. It is immunized from most attack-side code obfuscation methods, good for economical Internet wide deployment with less maintenance cost and negligible throughput degradation and can also handle encrypted SSL messages. Sig Free is transparent to the servers being protected. The Sig Free DLL was tested extensively throughout its development, in both lab and real-world scenarios. With every test, improvements were made in methods of detection and Sig Free efficiency. Improvements included the development of unique techniques to defeat polymorphism, encryption, metamorphism, and self-modification, and in the development of real life applications

### References

[1] Lee Ling Chuan, Chan Lee Yee, Mahamod Ismail and Kasmiran Jumari, "Automated Blocking of Malicious Code with NDIS Intermediate Driver", ICACT 2011, IEEE February 2011.

[2] MSDN Library, Microsoft Corporation, "NDIS-Supplied Packet and Buffer Handling Functions (NDIS 5.1)," March 6, 2010.