Research Article Impact Factor: 0.621 ISSN: 2319-507X
PB Pawar, IJPRET, 2014; Volume 2 (8): 562-573 IJPRET

INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

PERFORMANCE EVALUATION OF SIGNATURE AND SIGNATURE FREE BUFFER OVERFLOW DETECTION FOR IMAGE FILE FORMAT

PROF. P. B. PAWAR, MR. K. D. BODHE, MR. A. D. ISALKAR

Department of Computer Sci. & Engg, SGBAU, Babasaheb Naik College of Engg., Pusad, India

Accepted Date: 15/02/2014 ; Published Date: 01/04/2014

Abstract: Internet threat have different form of attacks, considering individual users to obtain control over data and network. The Buffer Overflow which is one of the most frequently occurring security vulnerabilities on network. Buffer Overflow occurs while writing data to a buffer and it overruns the buffer's threshold and overwrites it to neighbouring memory. The techniques to avoid buffer overflow vulnerability vary per architecture, Operating system and memory region. The Signature based buffer overflow detection finds the particular Signature and if that found it blocks it to protect form malicious attack. The remaining request are consider for checking against the buffer size and grant to server if the buffer of request is less than or equal to defined threshold value of buffer. Signature free first filters and extracts instruction sequences from a request. Finally it compares the number of useful instructions to a threshold to determine if this instruction sequence contains code. Signature free thus it can block new and unknown buffer overflow attacks, Signature free is also immunized from most attack-side code obfuscation methods. Since Signature free *is transparent to the servers that protected, it is efficient for economical Internet wide deployment with very low deployment and maintenance cost. We are proposing novel techniques for preventing buffer overflow during the transmission of images of different formats. In this paper we have discuss and evaluate certain tools and techniques which prevent buffer overflows. We have also discussed some modern tools and techniques with their pros and cons.*

Keywords: *Buffer-Overflow, Signature, Signature free, Malicious code, Intrusion, vulnerabilities.*

PAPER-QR CODE

Corresponding Author: PROF. P. B. PAWAR

Access Online On:

www.ijpret.com

How to Cite This Article:

PB Pawar, IJPRET, 2014; Volume 2 (8): 562-573

INTRODUCTION

Computer Security includes the protection of information and property from hacker, corruption while allowing the information to remain accessible to its intended user. Computer security means valuable information and services are protected from access, 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. Buffer overflow attack is an attack in which a malicious user exploits an unchecked buffer in a program and overwrites the program code with own data. If the program code is overwritten with new executable malicious code, the effect is to change the program's operation as dictated by the hacker. If overwritten with other data, the likely effect is to direct the program to crash. Today‘s software has been widely targeted by buffer overflows. 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. There are several different approaches for finding and preventing buffer overflows. These include enforcing secure coding practices, statically examine source code, halting exploits via operating system support, and detecting buffer overflows at runtime [1]. 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.

1. MOTIVATION

Buffer overflow is a problem where a program tries to store a string of arbitrary length in a fixed size buffer, without checking for whether the string can fit inside the buffer. It results in inadvertently overwriting the memory location that follows the buffer. If the buffer resides in the data section, it could corrupt other global variables [1]. If the buffer is heap allocated, it could corrupt data structure used by memory management routines. If the buffer resides on the stack the overflow could overwrite the stack frame, including the return address, which can alter the program's control flow. In the context of networked programs, buffer overflow allows the execution of arbitrary code injected by a remote client or peer. This could result in compromise of sensitive data. If the buffer overflow happens in an operating system kernel, it could lead to privilege escalation. When remote code injection is combined with privilege escalation, it could result in the compromise of the whole system.

Today's world totally work through important asset i.e. information and there should protective shield to protect such asset with good performance. Therefore Security issue is the important and elicited topic among IT professionals [10]. Taking into consideration the various issues, we propose a model which is capable of detecting the possible security violate in network environment.

2. LITERATURE REVIEW

Xinran Wang, Chi-Chun Pan, Peng Liu, and Sencun Zhu proposed work on Signature free: A Signature-Free Buffer Overflow Attack Blocker [1]. There experimental study shows that the dependency-degree-based Signature free could block all types of code-injection attack packets (above 750) tested in their experiments with very few false positives. Moreover, Signature free causes very small extra latency to normal client requests when some requests contain exploit code.

Eric Haugh and Matt Bishop discussed work on Testing C Programs for Buffer Overflow Vulnerabilities [3] 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 [4]. They consider which combinations of techniques can eliminate the problem of buffer overflow vulnerabilities, while sustaining the functionality and performance of existing systems.

Hassen Sallay, Khalid A. AlShalfan, Ouissem Ben Fred proposed work on A scalable distributed IDS Architecture for High speed Networks [5]. 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 Signature and signature Free (Signature free) for detecting buffer overflow with jpg and gif file formats. This technique capable of detecting malicious code by applying Pattern matching scheme on Signature, here IP address is chosen as a signature in our study. The log file from server contains the all information about the requests along with necessary metadata. We separate out the original image from malicious image by detecting the buffer overflow occurred during transmission over the network with Signature and Without Signature. Each time during receiving end we check for buffer overflow and if found then we are blocking that images and forwarding the remaining images to server for processing.

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 [10]. 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 modification to the source code and recompilation to work in addition to the requirement of source availability. Stack Guard [7] 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 executed by inserting a random word value immediately following the return address for the process on the stack. This value is confirmed 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 study suggests that the protection mechanism may still be circumvented by exploiting function pointers or "long jumps" [8]. If an attacker can overflow an array or buffer that resides beside a target-address variable in an application, shell can then modify this target address, and eventually take control of the application after the dynamic branch instruction using the target-address variable is executed.

3. NETWORK-BASED ATTACKS DETECTION ANDPREVENTION TECHNIQUES

Network-based attacks can exploit, launch and propagate without human interference. Network-based attacks on the host predominantly exploit vulnerabilities in protocols and network defined 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. 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 observing the use of Stack and Heap system memory. 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 charge, buffer overflow exploit prevention will provide the necessary insurance for overflows against unknown vulnerabilities.

A. Principles of Buffer Overflow Attacks

In a typical buffer overflow attack, the attacker injects an instruction sequence into the victim application and transfers the control of the application to the injected code. As an application's

text segment is typically read-only, the only way to hijack the control of an application is to dynamically modify the target address of its branch instructions whose target is not fixed at compilation time. Such dynamic branch instructions include function returns, pointer-based function calls, and Cstyle switch statements. These branch instructions typically have their target addresses stored in some stack or heap.

B. Methodology

Since remote exploits are basically 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[5]. To overcome the problem of buffer overflow we proposed the Signature and Signature free buffer overflow blocker technique. The background behind the Signature free 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]. In the Signature based detection, a Signature on the server is defined and if the request containing that Signature comes on the server it get blocked. Thus the code is protected from unwanted request. We take an IP as a signature, if a particular request containing the defined IP comes then it get blocked there itself, the remaining requests are get allowed to forwarded.

C. Block diagram

Figure 1 shows the architecture of our study. The request from client is verified on Signature and Signature free and the valuable request are send to server. The detail of Signature and Signature free detection of request is discussed is section 5.

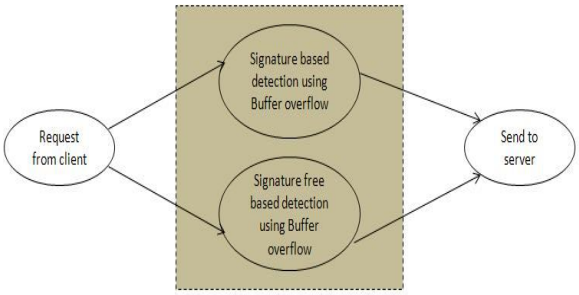


Fig. 1: Block Diagram of System.

D. Algorithm

Input: Request from Client

Step 1: Check request and encode URL

Step 2: Remove unwanted request

Step 3: Check for signature

Step 4: If Signature found

Block the request containing Signature and

go to Step 7

Step 5: Split URL and remove duplicates.

Step 4: Decode URL and distill instruction

Step 5: ASCII Conversion and filter the request

Step 5: Decide Buffer Size

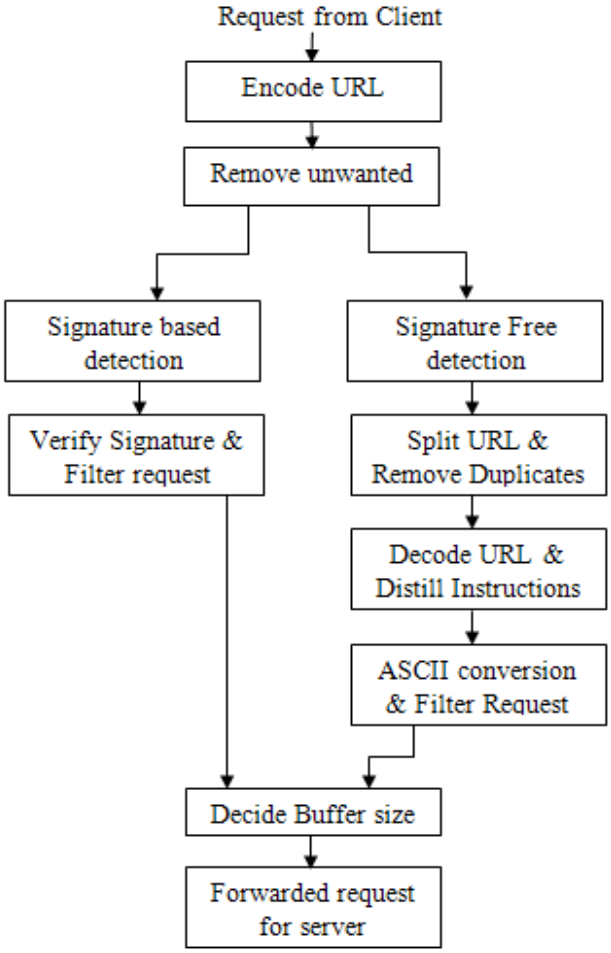
Step 6: If Buffer Size > Size of Request

Blok the request

Step 7: Supply forwarded request to Server.

Step 8: Stop.

E. Flowchart



4. PROPOSED METHODOLOGY

Network-based attacks can penetrate, launch and propagate without human interface. Network-based attacks on the host predominantly exploit vulnerabilities in protocols and network known 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

One of the newest host protection technologies available is buffer overflow exploit prevention, also known as memory protection. We focused on finding the buffer overflow occurred during the transmission of request from client to server. The request first gets encoded and the URL is split which separate metadata from it. The unwanted requests here the request containing the format other than image format get split and blocked so we just allowed the requests which contain the gif and jpg file extension. Further module is divided into two parts: with signature

buffer overflow detection and without signature (Signature free) buffer overflow detection. The signature based detection just check the request against the IP address maintained on server. If the request containing the particular IP address (Which is defined on server in our study) comes, then it will get blocked and all the request are allowed to be forwarded in this technique. The Signature free buffers overflow detection first split the URLs by removing the all other metadata and the split the URL. The duplicates from the splitting are removed afterword. The encoded URL gets decoded to check any unwanted request and the get distilled into protocol, domain, and instruction. The ASCII conversion of request is done to remove unwanted code; we consider the code which is not is standard UTF-8 format. Now by deciding the size of buffer manually depending the situations of requests, we called is threshold buffer size in our study. If the buffer size required by request is greater than the defined threshold value buffer the request thereby blocked itself and the request having the buffer size below threshold value get forwarded to server. We are tested more than 200 requests in our study and checked all requests against threshold buffer size manually. The results comes out will be discussed in detail in Section 7.

Specifically in the Signature based detection, first the signature is verified which is defined on server to block unauthorized request. These verified requests then filtered on basis signature. And requests then check against the buffer size depending on the buffer size the requests are then forwarded to server for processing. In the Signature free technique, split URL and remove the duplicate of the requests. The encode URL is now decoded

and then these requests are distilled. The ASCII conversion is done on the distilled requests and the requests are filtered and checked against the Buffer size. If the Buffer size required by request is more than threshold value then the request is blocked; otherwise it is forwarded to server as shown in figure 2.

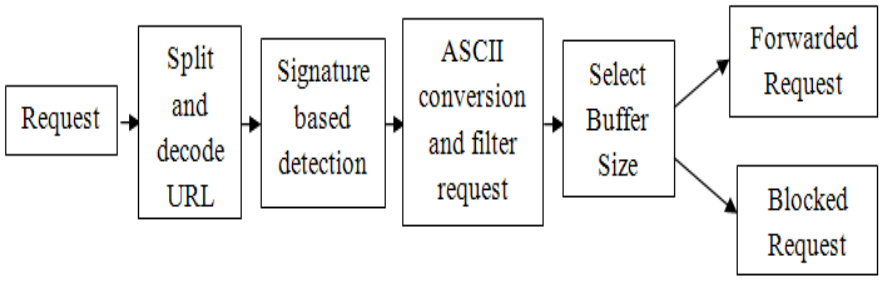


Fig. 2: Working of Signature & Signature Free technique using buffer overflow.

5. LIMITATIONS

Signature free does not detect attacks that just corrupt control flow or data without injecting code. If the buffer being overflowed is inside a jpg or gif system, Signature free cannot be directly applied. Although Signature free can decode the protected file according to the protocols or applications it protects, more domain in 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. The proposed system can be extended for the detection of buffer overflow for all file formats.

6. RESULT

The result is obtained by comparing request from network against the buffer size for signature and without signature methodology. The nature of buffer is changed according to the request from network. Sometimes take more buffer size (buffer overflow occurs) and maximum time average buffer size depending on the nature of request from the network. Figure 3(a) shows behavior of requests against the Buffer size for Signature based detection. Figure 3(b) shows behavior of requests against the Buffer size for Signature free based detection and Figure 4 shows Analysis requests blocked against the Buffer size for Signature and Signature free technique for jpg and gif file format. If the buffer size required by request is more than the threshold value the then requests are blocked; otherwise it is forwarded. Figure 5 shows the analysis of buffer overflow detection with Signature and without Signature for jpg and gif formats.

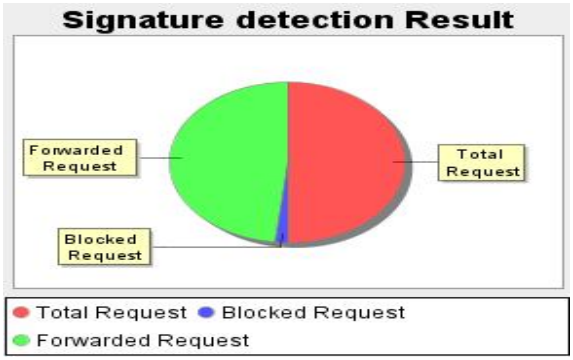


Fig. 3(a): Behavior of request against Signature based detection

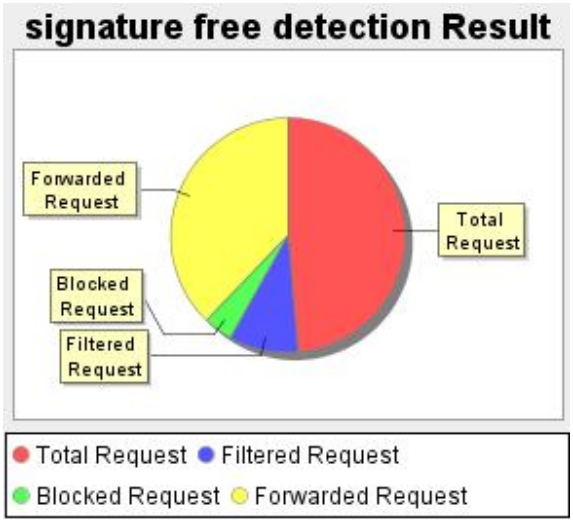


Fig. 3(b): Behavior of request against Signature free based detection

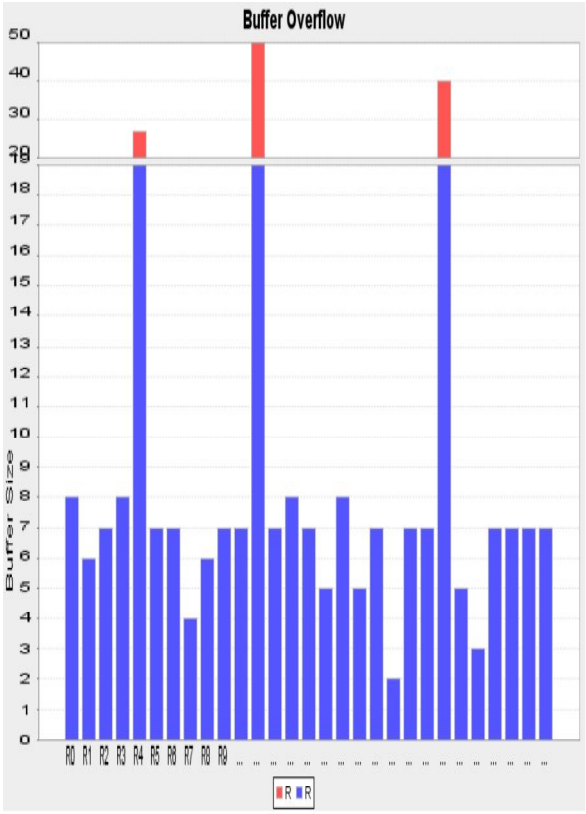


Fig. 4: Analysis of requests block based on Signature and Signature free based detection

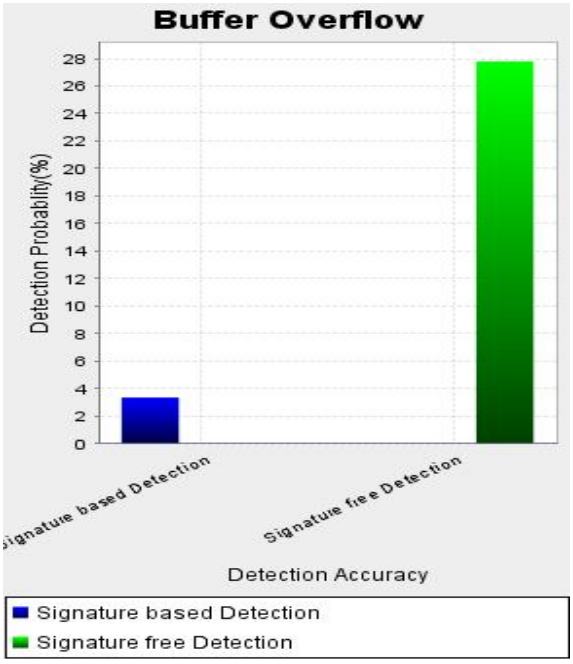


Fig. 5: Analysis of Signature and Signature free detection.

7. CONCLUSION

We have proposed Signature and signature-free buffer overflow detection for jpg and gif systems that can filter code injection buffer overflow attack, one of the most serious cyber security paradigms. The Signature based buffer overflow detection finds the particular Signature and if that found it blocks it to protect form malicious attack. Signature free does not require any signatures, thus it can block new malicious code and provide security for the systems. We can conclude that the maximum requests can be blocked using Signature free technique rather than signature based. Signature and Signature free is less affected from malicious attack, and easy for deployment, low performance overhead with less maintenance cost.

REFERENCES

1. 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.

2. 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.

3. Eric Haugh and Matt Bishop, "Testing C Programs for Buffer Overflow Vulnerabilities".

4. 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.

5. 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.

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. D. Evans and D. Larochelle, "Improving Security Using Extensible Lightweight Static Analysis," IEEE Software, vol. 19, no. 1, 2002.

11. 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.

12. Pankaj Pawar , Malti Nagle and Pankaj Kawadkar "Prevention of Buffer overflow Attack Blocker Using IDS" IJCSN Volume 1, Issue 5, October 2012.

13. [13] P.B. Pawar, M. Nagle "Attack of Buffer-overflow Attack Blocker using IDS" at the International Conference on Electrical, Electronics and Computer Science(ICEECS-2012) held at Nagpur on 16th December,2012.