1. INTRODUCTION

Distributed denial of service attack (DDoS) is an attempt by malicious hosts to overload website, network, e-mail servers, applications, network resources, bandwidth, etc. Challenges involved in taking counter measures against DDoS attacks are network infrastructure, identifying legitimate traffic from polluted traffic, attacker anonymity, large problem space, nature of attacks, etc.

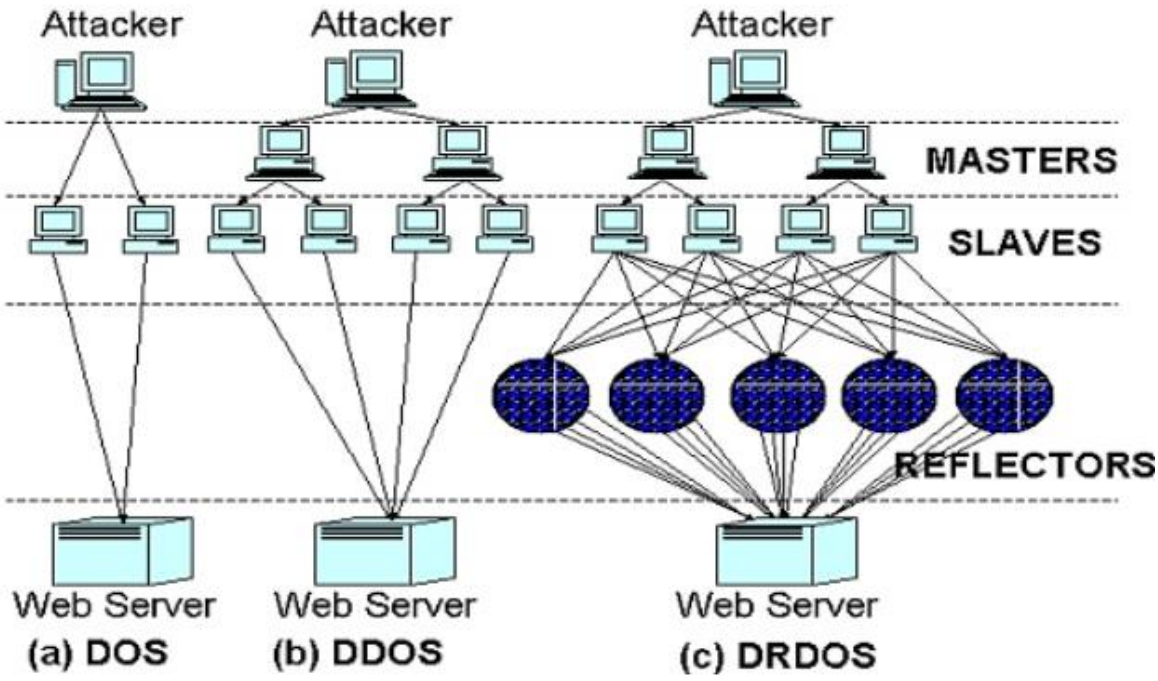

Fig.1.1 DDOS Attack Description

In a distributed denial-of-service attack (DDoS attack), the incoming traffic flooding the victim originates from many different sources. This effectively makes it impossible to stop the attack simply by blocking a single source.

A DDoS attack is analogous to a group of people crowding the entry door of a shop, making it hard for legitimate customers to enter, disrupting trade.

1.1 AREA OF OCCURRENCE

It overloads the website, network, e-mail servers, applications, network resources, bandwidth, etc. with the help of botnets/zombies, used by the attacker.

1.2 IMPACT

The DDoS attack impact varies, depending on the size and nature of the attack i.e. the number of servers that are used to launch the attack and how effective the attack is at consuming all available website or server (and network) resources. The impact can vary from a single website being slowed or down for a few minutes or hours, to an entire network.

Another serious concern is the number of Internet-connected systems and devices that either form part of or are connected to industrial control systems. As organizations become increasingly reliant on the convenience of Internet accessibility, the potential attack surface for damaging cyber-attacks, including DDoS, increases.

1.3 COMMON DDOS ATTACKS TYPES

1.3.1 UDP Flood

A UDP flood, by definition, is any DDoS attack that floods a target with User Datagram Protocol (UDP) packets. The goal of the attack is to flood random ports on a remote host. This causes the host to repeatedly check for the application listening at that port, and (when no application is found) reply with an ICMP 'Destination Unreachable' packet. This process saps host resources, which can ultimately lead to inaccessibility.

1.3.2 ICMP (Ping) Flood

Similar in principle to the UDP flood attack, an ICMP flood overwhelms the target resource with ICMP Echo Request (ping) packets, generally sending packets as fast as possible without waiting for replies. This type of attack can consume both outgoing and incoming bandwidth, since the victim's servers will often attempt to respond with ICMP Echo Reply packets, resulting a significant overall system slowdown.

1.3.3 SYN Flood

A SYN flood DDoS attack exploits a known weakness in the TCP connection sequence (the "three-way handshake"), wherein a SYN request to initiate a TCP connection with a host must be answered by a SYN-ACK response from that host, and then confirmed by an ACK response from the requester. In a SYN flood scenario, the requester sends multiple SYN requests, but either does not respond to the host's SYN-ACK response or

sends the SYN requests from a spoofed IP address. Either way, the host system continues to wait for acknowledgement for each of the requests, binding resources until no new connections can be made, and ultimately resulting in denial of service.

1.3.4 PING of Death

A PING of Death ("POD") attack involves the attacker sending multiple malformed or malicious pings to a computer. The maximum packet length of an IP packet (including header) is 65,535 bytes. However, the Data Link Layer usually poses limits to the maximum frame size – for example 1500 bytes over an Ethernet network. In this case, a large IP packet is split across multiple IP packets (known as fragments), and the recipient host reassembles the IP fragments into the complete packet. In a Ping of Death scenario, following malicious manipulation of fragment content, the recipient ends up with an IP packet which is larger than 65,535 bytes when reassembled. This can overflow memory buffers allocated for the packet, causing denial of service for legitimate packets.

1.3.5 NTP Amplification

In NTP amplification attacks, the perpetrator exploits publically-accessible Network Time Protocol (NTP) servers to overwhelm a targeted server with UDP traffic. The attack is defined as an amplification assault because the query-to-response ratio in such scenarios is anywhere between 1:20 and 1:200 or more. This means that any attacker that obtains a list of open NTP servers (e.g., by a using tool like Metasploit or data from the Open NTP Project) can easily generate a devastating high-bandwidth, high-volume DDoS attack.

1.3.6 HTTP Flood

In an HTTP flood DDoS attack, the attacker exploits seemingly-legitimate HTTP GET or POST requests to attack a web server or application. HTTP floods do not use malformed packets, spoofing or reflection techniques, and require less bandwidth than other attacks to bring down the targeted site or server. The attack is most effective when it forces the server or application to allocate the maximum resources possible in response to each single request.

One of the most common attacks is the PING FLOOD (ICMP FLOOD) on which we intend to propose our solution.

1.4 PING FLOODING ATTACK (using ICMP FLOOD)

1.4.1 WHAT IS A PING FLOOD ATTACK?

Ping flood, also known as ICMP flood, is a common DDoS attack in which an attacker takes down a victim's computer by overwhelming it with ICMP echo requests, also known as pings. The attack involves flooding the victim's network with request packets, knowing that the network will respond with an equal number of reply packets. Additional methods for bringing down a target with ICMP requests include the use of custom tools or code, such as hping and scapy.

These strains both the incoming and outgoing channels of the network, consuming significant bandwidth and resulting in a denial of service.

1.4.2 ATTACK DESCRIPTION

Normally, ping requests are used to test the connectivity of two computers by measuring the round-trip time from when an ICMP echo request is sent to when an ICMP echo reply is received. During an attack, however, they are used to overload a target network with data packets.

Executing a ping flood is dependent on attackers knowing the IP address of their target. Attacks can therefore be broken down into three categories, based on the target and how its IP address is resolved.

- A targeted local disclosed ping flood targets a single computer on a local network. An attacker needs to have physical access to the computer in order to discover its IP address. A successful attack would result in the target computer being taken down.
- A router disclosed ping flood targets routers in order to disrupt communications between computers on a network. A successful attack would result in all computers connected to the router being taken down.

There are a number of ping commands that can be used to facilitate an attack, including:

- The –n command, which is used to specify the number of times a request is sent.

- The –l command, which is used to specify the amount of data sent with each packet.
- The –t command, which is used to continue pinging until the host times out.

1.4.3 How does a Ping flood attack work?

The Internet Control Message Protocol (ICMP), which is utilized in a Ping Flood attack, is an Internet layer protocol used by network devices to communicate. The network diagnostic tools traceroute and ping both operate using ICMP. Commonly, ICMP echo-request and echo-reply messages are used to ping a network device for diagnosing the health and connectivity of the device and the connection between the sender and the device.

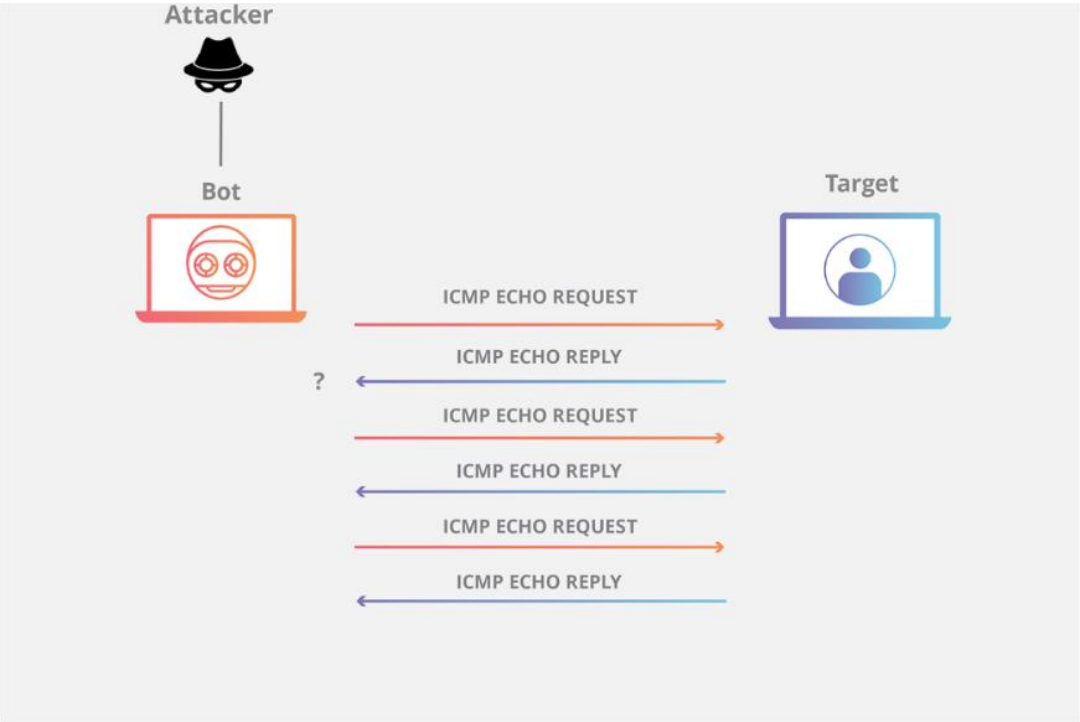

Fig.1.2 ICMP Echo Request and Reply in Ping Flood Attack

An ICMP request requires some server resources to process each request and to send a response. The request also requires bandwidth on both the incoming message (echo-request) and outgoing response (echo-reply). The Ping Flood attack aims to overwhelm the targeted device's ability to respond to the high number of requests and/or overload the network connection with bogus traffic. By having many devices in a botnet target the same Internet property or infrastructure component with ICMP

requests, the attack traffic is increased substantially, potentially resulting in a disruption of normal network activity. Historically, attackers would often spoof in a bogus IP address in order to mask the sending device. With modern botnet attacks, the malicious actors rarely see the need to mask the bot's IP, and instead rely on a large network of un-spoofed bots to saturate a target's capacity.

The DDoS form of a Ping (ICMP) Flood can be broken down into 2 repeating steps:

The attacker sends many ICMP echo request packets to the targeted server using multiple devices. The targeted server then sends an ICMP echo reply packet to each requesting device's IP address as a response.

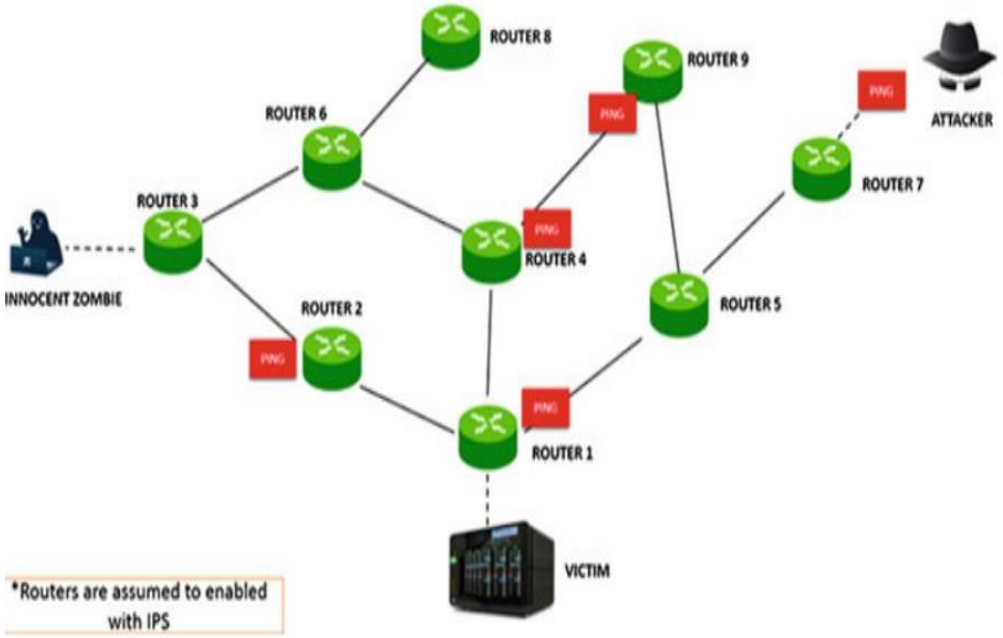

Fig.1.3 Ping Flood Attack

The damaging effect of a Ping Flood is directly proportional to the number of requests made to the targeted server. Unlike reflection-based DDoS attacks like NTP amplification and DNS amplification, Ping Flood attack traffic is symmetrical; the amount of bandwidth the targeted device receives is simply the sum of the total traffic sent from each bot.

Note that for a ping flood to be sustained, the attacking computer must have access to more bandwidth than the victim. This limits the ability to carry out a DoS attack, especially against a large network.

Additionally, a Distributed Denial of Service (DDoS) attack executed with the use of a botnet has a much greater chance of sustaining a ping flood and overwhelming a target's resources.

1.4.4 METHODS OF MITIGATION

Recognise your perimeter firewall to disallow pings will block attacks originating from outside your network, albeit not internal attacks. Still, the blanket blocking of ping requests can have unintended consequences, including the inability to diagnose server issues.

The Imperva DDoS protection provide blanket protection against ICMP floods by limiting the size of ping requests as well as the rate at which they can be accepted.

1.5 PROBLEM STATEMENT

The PING flood attack apart from consuming bandwidth, also in some occasions consumes file space, memory as well as CPU. The server is overwhelmed by overloaded traffic making it unable to respond to the legitimate users till countermeasures are taken. On the other hand, Distributed Denial of Service attack is launched indirectly through compromised computing systems that facilitate coordinated attack on network or system. This attack attempts to disrupt the targeted system using the internet principles thus overwhelming legitimate users by denying them access to the system resources. There is need to identify these attacks, manage them as well as prevent them and look into ways of differentiating between malicious and legitimate users as it is a common persistent problem faced up to date.

1.6 OBJECTIVE

In our project we have proposed a solution which enhances server abilities through traffic pattern observation and attack detection before the attack occurs and taking adaptive measure of banning the attacker IP and thus preventing DDoS PING flooding attack.

2. Literature Survey

2.1 Existing Approaches

2.1.1 Apiary: Easy-to-Use Desktop Application Fault Containment on Commodity Operating Systems.

It introduces the Virtual Layered File System to make instantiating containers fast and space efficient, and to make managing many containers no more complex than a single traditional desktop. We have implemented Apiary on Linux without any application or operating system kernel changes.

2.1.1.1 Apiary Usage Model

Apiary desktop session: (1) application menu, (2) window list, (3) composited display directory. This enables containers to enforce isolation without the creation of any isolation rules. Apiary isolates individual applications, not individual programs. An application in Apiary is a software appliance made up of multiple programs that are used together in a single environment to accomplish a specific task.

Apiary lets users select whether an application should launch within an ephemeral or a persistent container. Ephemeral containers provide a powerful mechanism for protecting desktop security and user privacy. Users will typically run multiple ephemeral containers, even for the same application, at the same time. They provide important benefits for a wide range of uses. Ephemeral containers prevent compromises because exploits cannot persist. For example, a malicious PDF document that exploits an ephemeral PDF viewer will have no persistent effect on the system because the exploit is isolated in the container and will disappear when the container finishes executing. Ephemeral containers protect user privacy when using the Internet.

Apiary provides every desktop with two ways to share files between containers. First, containers can use standard file system share concepts to create directories that can be seen by multiple containers. This has the benefit of any data stored in the shared directory being automatically available to the other containers that have access to the share. Second, Apiary supplies every desktop with a special persistent container with a file explorer. The explorer has access to all of the user's containers

and can manage all of the user's files, including copying them between containers. This is useful if the user wants to preserve a file from an ephemeral container or move a file from one persistent container to another, as, for instance, when emailing a set of files. The file explorer container cannot be used in an ephemeral manner. Its functionality cannot be invoked by any other application on the system and no other application is allowed to execute within it. This prevents an exploited container from using the file explorer container to corrupt others. It should be noted that both of these mechanism break, to some degree, the container's isolation. File system shares can be used by an exploited container as a vector to infect other containers by tricking a user into moving a malicious file between containers. However, this is a tension that will always exist in security systems that are meant to be usable to a diverse crowd of users. To mitigate this, Apiary lets documents stored in a persistent manner be viewable, by default, in an ephemeral container. For example, PDF files can be stored persistently, but always viewed in ephemeral containers. However, to persistently edit a PDF file, it would still have to be opened within a persistent container such that a malicious PDF would have a persistent effect.

2.1.1.2 Conclusion

Apiary introduces a new compartmentalized application desktop paradigm. Instead of running one's applications in a single environment with complex rules to isolate the applications from each other, Apiary enables them to be easily and completely isolated while retaining the integrated feel users expect from their desktop computer. The key innovations that make this possible are the introduction of the Virtual Layered File System and the ephemeral containers they enable. The Virtual Layered File System enables the multiple containers to be stored as efficiently as a single regular desktop, while also allowing containers to be instantiated almost instantly. This functionality enables the creation of the ephemeral containers that provide an always fresh and clean environment for applications to run in. Ephemeral containers prevent malicious data from having any persistent effect on the system and isolate faults to a single application instance. We have implemented Apiary on Linux without requiring any operating system kernel or application changes. Our results demonstrate that Apiary's containerized desktop severely reduces the threat posed by malicious files and plugins by isolating them in ephemeral containers and enabling users to quickly recover if they penetrate a persistent container.

2.1.2. Toward Automated Detection of Logic Vulnerabilities in Web APP

Application logic vulnerabilities are an important class of defects that are the result of faulty application logic. These vulnerabilities are specific to the functionality of particular web applications, and, thus, they are extremely difficult to characterize and identify.

2.1.2.1 Execution Model

To find all entry points, our system inspects the application deployment descriptor (typically, the web.xml file), which defines how URLs requested by a user are mapped to servlets. When analyzing the URL-to-servlet mapping, we take into account that not all servlets are directly accessible to users (those servlets that are not directly accessible are typically invoked internally by other servlets). Following the standard servlet invocation model, all URLs that point to accessible (public) servlets are assumed to be possible entry points. Once the application's entry points are determined, the Application Controller systematically explores the state space of the application. To this end, it initiates execution of servlets by simulating all possible user choices of URLs. For example, if the application has three servlets mapped to the URLs /login, /cart, and /checkout, the application controller attempts to execute all possible combinations (sequences) of these servlets. The actual order in which servlets are explored depends on the chosen search strategy. JPF offers a limited depth-first search (DFS) and a heuristics-based breadth-first search (BFS) strategy. We found that DFS works better for our system because it requires significantly less memory during model checking. With DFS, a path is explored until the system reaches a specific (configurable) limit on the number of entry points that are executed.

2.1.2.2 Limitations

Our approach aims at detecting logic vulnerabilities in a general, application-independent way. However, the current prototype version of Waler has a number of limitations, many of which, we believe, can be solved with more engineering. First,

the types of vulnerabilities that can be identified by Waler are limited by the set of currently-implemented heuristics. For example, if an application allows the user to include a negative number of items in the shopping cart, we would be able to identify this issue only if the developer checked for that number to be non-negative on at least one program path leading to that program point. In addition, this check needs to be in a direct if-comparison6 between variables. Conditions deriving from switch instructions or resulting from complex operations (such as regular expression matching) are not currently implemented. Another limitation stems from the fact that we need a tool to derive approximations of program specifications. As a result, the detection rate of Waler is bounded by the capabilities of such a tool. In the current implementation, we chose to use Daikon. While Daikon is able to derive a wide variety of complex relationships between program variables, it has a limited support for some complex data structures. For example, if the isAdmin flag value is stored in a hash table, and it is not passed as an argument to any application function, Daikon will not be able to generate invariants based on that value. This limitation could be improved by implementing a smarter exploration technique for complex objects and/or by tracing 12 local and temporary variables for the purpose of likely invariant generation.

2.1.3 Regular Expressions Considered Harmful in Client-Side XSS Filters

We analyze the best existing filters and find them to be either unacceptably slow or easily circumvented. Worse, some of these filters could introduce vulnerabilities into sites that were previously bug-free. We propose a new filter design that achieves both high performance and high precision by blocking scripts after HTML parsing but before execution. Compared to previous approaches, our approach is faster, protects against more vulnerability, and is harder for attackers to abuse.

2.1.3.1 Threat Model

Attacker Abilities. Client-side XSS filters are designed to mitigate XSS vulnerabilities in web sites without requiring the web site operator to modify the web site. We assume the attacker has the following abilities: • The attacker owns and operates a web site. • The user visits the attacker's web site. • The target web site lets the attacker inject an arbitrary sequence of bytes into the entity-body of one of its HTTP responses. Also, we consider mitigating injections at a single location only and do not seek to provide protection for so-called "double injection" vulnerabilities in which the attacker can inject content at multiple locations simultaneously. Covering vulnerabilities is useful because the filter will protect a web site that contains only covered vulnerabilities. However, covering attacks is of less utility. If an attacker can evade the filter by constructing a convoluted attack string (e.g., by injecting script via CSS expressions or via obscure parser quirks), then the filter does not actually prevent a sophisticated attacker from attacking the site. Each filter, then, defines a set of vulnerabilities that are in-scope, meaning the filter aims to prevent the attacker from exploiting these vulnerabilities to achieve his or her goals. Attacker Goals. We assume the attacker's goal is to run arbitrary script in the user's browser with the privileges of the target web site. Typically, an attacker will run script as a stepping stone to disrupting the confidentiality or integrity of the user's session with the target web site.

In the limit, the attacker can always inject script into a web site if the attacker can induce the user into taking arbitrary actions. In this paper, we consider attackers who

seek to achieve their goals with zero interaction or a single-click interaction with the user.

2.1.3.2 Conclusion

We propose an improved design for a client-side XSS filter. Our design achieves high performance and high fidelity by interposing on the interface between the browser's HTML parser and JavaScript engine. Our implementation is enabled by default in Google Chrome. Most existing client-side XSS filters simulate the browser's HTML parser with regular expressions that produce unnecessary false positives. These filters can be bypassed by exploiting differences between the simulation and the actual parser. Worse, when they detect an attack, the filters resort to mangling the HTTP response in a way that introduces vulnerabilities into otherwise vulnerability-free sites. Our post-parser design examines the semantics of an HTTP response, as interpreted by the browser, without performing a time-consuming, error-prone simulation. We block suspected attacks by preventing the injected script from being passed to the JavaScript engine rather than performing risky transformations on the HTML. Cross-site scripting attacks are among the most common classes of web security vulnerabilities, and this trend shows no signs of reversing. Fixing every XSS vulnerability in a large web application can be a daunting task. Every browser should include a client-side XSS filter to help mitigate unpatched XSS vulnerabilities.

2.1.4. Effective Anomaly Detection with Scarce Training Data

Learning-based anomaly detection has proven to bean effective black-box technique for detecting unknown attacks. However, the effectiveness of this technique crucially depends upon both the quality and the completeness of the training data. Unfortunately, in most cases, the traffic to the system (e.g., a web application or daemon process) protected by an anomaly detector is not uniformly distributed. Therefore, some components (e.g., authentication, payments, or content publishing) might not be exercised enough to train an anomaly detection system in a reasonable time frame. We run our experiments on a real-world data set containing over 58 million HTTP requests to more than 36,000 distinct web application components. The results show that by using the proposed solution, it is possible to achieve effective attack detection even with scarce training data.

2.1.4.1 Related Work

The use of Bayesian networks is proposed to compose models and express inter-model dependencies in order to further reduce false positive rates. A system for clustering anomalies and classifying clusters according to the type of attack is proposed in to improve the explanatory power of web application anomaly detection as well as further reduce their false positive rate. In this work, a sanitization phase is first performed to remove suspected attacks and other abnormalities from the data. Instead of creating one model instance, a set of “micro-models” is trained against disjoint subsets of the training data. These micro-models are then subject to one of several voting schemes to recognize and cull outliers that may represent attacks. While this work is somewhat related to ours in that slices of training data are considered, the application of micro-models is intended to ensure that attacks are not present in training data. In contrast, our approach is complementary in the sense that it is focused on addressing the lack of training data. Though this system shares the element of profile clustering with our work, the scope of the clustering is limited to end hosts connected to a single switch, while our work clusters across a large population of web application profiles. Also, the models operate over coarse network statistics such as the average number of hosts contacted per hour, the average number of packets exchanged per hour, and the average length of packets exchanged per hour. Our system, on the other hand, considers fine-grained features specific to

web applications. Finally, our system does not require the use of a distributed voting scheme, which incurs additional overhead. A recent proposal for the anomaly-based detection of web-based attacks is presented. In this work, a mixture of Markov chains incorporating n-gram transitions is used to model the normal behaviour of HTTP request parameters. The resulting system attains a high detection accuracy for a variety of web-based attacks. In contrast to our work, however, a single mixture is learned for an entire web application. Additionally, the proposed system utilizes a supervised learning algorithm (i.e., attacks must be labelled as such in the training set), whereas ours operates on unlabelled training data.

2.1.4.2 Conclusion

In this work, we have identified that non-uniform web client access distributions cause model undertraining. This is an issue that must be addressed by web application anomaly detection systems in order to provide a reasonable level of security. We propose the use of global knowledge bases of well-trained, stable profiles to remediate a local scarcity of training data by exploiting global similarities in web application parameters. We have evaluated the efficacy of this approach over an extensive data set collected from real-world web applications. We found that although using global profiles does result in a small reduction in detection accuracy, the resulting system, when given appropriate parameters, does provide reasonably precise modelling of otherwise unprotected web application parameters. As future work, we plan to investigate the extension of global model clustering to other types of models, particularly HTTP response and session models. An additional line of future work is to investigate the application of different clustering methods in order to improve the efficiency of the querying procedure for high-cardinality knowledge bases.

3.SYSTEM ANALYSIS

3.1. Problems with Existing System

3.1.1General Techniques

3.1.1.1 Disabling unused services

The less there are applications and open ports in hosts, the less there are chance to exploit vulnerabilities by attackers. Therefore, if network services are not needed or unused, the services should be disabled to prevent attacks, e.g. UDP echo, character generation services.

3.1.1.2 Install latest security patches

Today, many DDoS attacks exploit vulnerabilities in target system. So, removing known security holes by installing all relevant latest security patches prevents re-exploitation of vulnerabilities in the target system.

3.1.1.3 Disabling IP broadcast

Defense against attacks that use intermediate broadcasting nodes e.g. ICMP flood attacks, Smurf attacks etc. will be successful only if host computers and all the neighbouring networks disable IP broadcast.

3.1.1.4 Firewalls

Firewalls can effectively prevent users from launching simple flooding type attacks from machines behind the firewall. Firewalls have simple rules such as to allow or deny protocols, ports or IP addresses. But some complex attack e.g. if there is an attack on port 80 (web service), firewalls cannot prevent that attack because they cannot distinguish good traffic from DoS attack traffic.

3.1.1.5 Global defense infrastructure

A global deployable defense infrastructure can prevent from many DDoS attacks by installing filtering rules in the most important routers of the Internet. As Internet is administered by various autonomous systems according their own local security policies, such type of global defense architecture is possible only in theory.

3.1.1.6 IP hopping

DDoS attacks can be prevented by changing location or IP address of the active server proactively within a pool of homogeneous servers or with a pre-specified set of IP address ranges. The victim computer's IP address is invalidated by changing it with a new one. Once the IP addresses change is completed all internet routers will be informed and edge routers will drop the attacking packets. Although this action leaves the computer vulnerable because the attacker can launch the attack at the new IP address, this option is practical for DDoS attacks that are based on IP addresses. On the other hand, attackers can make this technique useless by adding a domain name service tracing function to the DDoS attack tools.

3.1.2 Filtering Techniques

3.1.2.1Ingress/Egress filtering

Ingress Filtering, proposed by Ferguson et al., is a restrictive mechanism to drop traffic with IP addresses that do not match a domain prefix connected to the ingress router. Egress filtering is an outbound filter, which ensures that only assigned or allocated IP address space leaves the network. A key requirement for ingress or egress filtering is knowledge of the expected IP addresses at a particular port. For some networks with complicated topologies, it is not easy to obtain this knowledge. One technique known as reverse path filtering can help to build this knowledge. This technique works as follows. Generally, a router always knows which networks are reachable via any of its interfaces. By looking up source addresses of the incoming traffic, it is possible to check whether the return path to that address would flow out the same interface as the packet arrived upon. If they do, these packets are allowed. Otherwise, they are dropped. Unfortunately, this technique cannot operate effectively in real networks where asymmetric Internet routes are not uncommon. More importantly, both ingress and egress filtering can be applied not only to IP addresses, but also protocol type, port number, or any other criteria of importance. Both ingress and egress filtering provide some opportunities to throttle the attack power of DoS attacks. However, it is difficult to deploy ingress/egress filtering universally. If the attacker carefully chooses a network without ingress/egress filtering to launch a

spoofed DoS attack, the attack can go undetected. Moreover, if an attack spoofs IP addresses from within the subnet, the attack can go undetected as well. Nowadays DDoS attacks do not need to use source address spoofing to be effective. By exploiting a large number of compromised hosts, attackers do not need to use spoofing to take advantage of protocol vulnerabilities or to hide their locations. For example, each legitimate HTTP Web page request from 10,000 compromised hosts can bypass any ingress/egress filtering, but in combination they can constitute a powerful attack. Hence, ingress and egress filtering are ineffective to stop DDoS attacks.

3.1.2.2 Router based packet filtering

Route based filtering, proposed by Park and Lee, extends ingress filtering and uses the route information to filter out spoofed IP packets. It is based on the principle that for each link in the core of the Internet, there is only a limited set of source addresses from which traffic on the link could have originated. If an unexpected source address appears in an IP packet on a link, then it is assumed that the source address has been spoofed, and hence the packet can be filtered. RPF uses information about the BGP routing topology to filter traffic with spoofed source addresses. Simulation results show that a significant fraction of spoofed IP addresses can be filtered if RPF is implemented in at least 18% of ASs in the Internet. However, there are several limitations of this scheme. The first limitation relates to the implementation of RPF in practice. Given that the Internet contains more than 10,000 Ass, would need to be implemented in at least 1800 ASs in order to be effective, which is an onerous task to accomplish. The second limitation is that RPF may drop legitimate packets if there has recently been a route change. The third potential limitation is that RPF relies on valid BGP messages to configure the filter. If an attacker can hijack a BGP session and disseminate bogus BGP messages, then it is possible to mislead border routers to update filtering rules in favour of the attacker. RPF is effective against randomly spoofed DoS attacks. However, the filtering granularity of RPF is low. This means that the attack traffic can still bypass the RPF filters by carefully choosing the range of IP addresses to spoof. Hence, RPF is ineffective against DDoS attacks. The router-based packet filter is vulnerable to asymmetrical and dynamic Internet routing as it does not provide a scheme to update the routing information.

3.1.2.3 History based IP filtering

Generally, the set of source IP addresses that is seen during normal operation tends to remain stable. In contrast, during DoS attacks, most of the source IP addresses have not been seen before. Peng et al. relies on the above idea and use IP address database (IAD) to keep frequent source IP addresses. During an attack, if the source address of a packet is not in IAD, the packet is dropped. Hash based/Bloom filter techniques are used for fast searching of IP in IAD. This scheme is robust and does not need the cooperation of the whole Internet community [36]. However, history-based packet filtering scheme is ineffective when the attacks come from real IP addresses. In addition, it requires an offline database to keep track of IP addresses. Therefore, Cost of storage and information sharing is very high. 4) Capability based method Capability based mechanisms provides destination a way to control the traffic directed towards itself. In this approach, source first sends request packets to its destination. Router marks (pre-capabilities) are added to request packet while passing through the router. The destination may or may not grant permission to the source to send. If permission is granted then destination returns the capabilities, if not then it does not supply the capabilities in the returned packet. The data packets carrying the capabilities are then send to the destination via router. The main advantage achieved in this architecture is that the destination can now control the traffic according to its own policy, thereby reducing the chances of DDoS attack, as packets without capabilities are treated as legacy and might get dropped at the router when congestion happens. However, these systems offer strong protection for established network flows, but responsible to generate a new attack type known as DOC (Denial of Capability), which prevents new capability-setup packets from reaching the destination, limits the value of these systems. In addition, these systems have high computational complexity and space requirement. 5) Secure overlay Service (SOS) Secure Overlay Service proposed by Keromytis et al. defines an architecture called secure overlay service (SOS) to secure the communication between the confirmed users and the victim. All the traffic from a source point is verified by a secure overlay access point (SOAP). Authenticated traffic will be routed to a special overlay node called a beacon in an anonymous manner by consistent hash mapping. The beacon then forwards traffic to another special overlay node called a Filtering Technique Benefits Limitations Ingress/ Egress -Prevents IP

Spoofing -Need global development - Attacks with real IP addresses can not be prevented RPF (Route based Packet Filtering) -Work well with static routing - Problem when dynamic routing is used -Need wide implementation to be effective History based -Does not require cooperation of whole Internet Community. -Gives priority to the frequent packets in case of congestion or attack - Ineffective when the attacks come from real IP addresses - Requires an offline database to keep track of IP addresses -Depend on information collected Capability based -Provides destination a way to control the traffic it desires -Incremental deployment -Attacks against the request packets cannot prevented (e.g. ROC attack) -High computational complexity and space requirement SOS -Works well for communication of predefined source nodes -Solution has limited scope e.g. not applicable to web servers -Require introduction of a new routing protocol that itself another security issue SAVE -Filtering improperly addressed packets is worthwhile -incremental deployment -During the transient period valid packets can be dropped secret servlet for further authentication, and the secret servlet forwards verified traffic to the victim. The identity of the secret servlet is revealed to the beacon via a secure protocol and remains a secret to the attacker. Finally, only traffic forwarded by the secret servlet chosen by the victim can pass its perimetric routers. Secure Overlay Service (SOS) addresses the problem of how to guarantee the communication between legitimate users and a victim during DoS attacks. SOS can greatly reduce the likelihood of a successful attack. The power of SOS is based on the number and distribution level of SOAPs. However, wide deployment of SOAPs is a difficult DoS defense challenge. Moreover, the power of SOS is also based on the anonymous routing protocol within the overlay nodes. Unfortunately, the introduction of a new routing protocol is in itself another security issue. If an attacker is able to breach the security protection of some overlay node, then it can launch the attack from inside the overlay network. Moreover, if attackers can gain massive attack power, for example, via worm spread, all the SOAPs can be paralyzed, and the target's services will be disrupted.

3.1.3 Limitations

The main problem is that there are still many insecure machines over the Internet that can be compromised to launch large-scale coordinated DDoS attack. One promising direction is to develop a comprehensive solution that encompasses several defense

activities to trap variety of DDoS attack. If one level of defense fails, the others still have the possibility to defend against attack. A successful intrusion requires all defense level to fail

3.2 Proposed System

- It employs a lightweight virtualization technique to assign each user's web session to a dedicated container, an isolated virtual computing environment.
- It uses the container ID to accurately associate the web request with the subsequent DB queries. Thus, Application can build a causal mapping profile by taking both the webserver and DB traffic into account.
- In addition, there are web services that permit persistent back-end data modifications. These services, which we call dynamic, allow HTTP requests to include parameters that are variable and depend on user input.

3.2.1 Advantages

- Application was able to identify a wide range of attacks with minimal false positives.
- Easy to find the Direct DB Attack.
- Light weight virtualization technique
- User friendly
- Economic

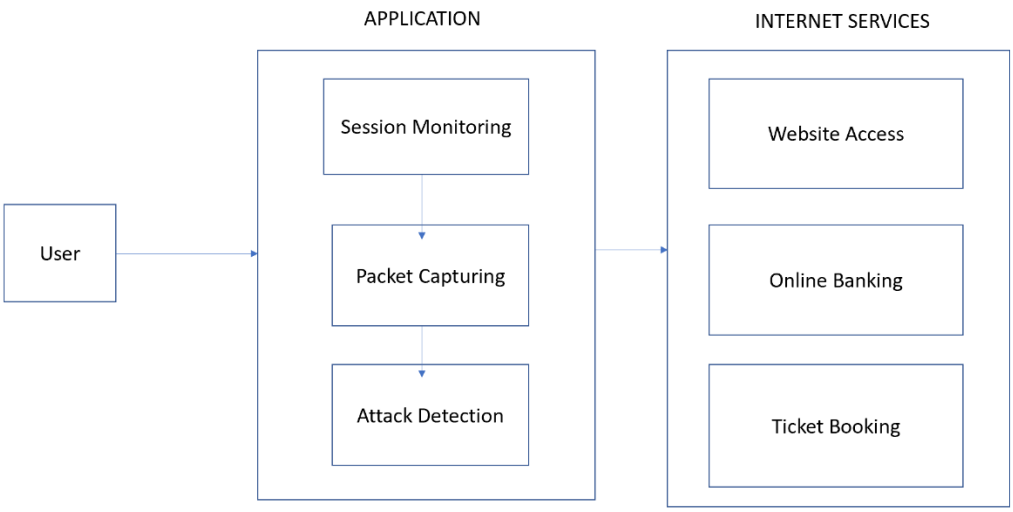

Fig. 3.2 Proposed System

3.3 Feasibility Study

Feasibility study is the test of a system proposal according to its workability, impact on the organization, ability to meet user needs, and effective use of recourses. It focuses on the evaluation of existing system and procedures analysis of alternative candidate system cost estimates. Feasibility analysis was done to determine whether the system would be feasible.

The development of a computer-based system or a product is more likely plagued by resources and delivery dates. Feasibility study helps the analyst to decide whether or not to proceed, amend, postpone or cancel the project, particularly important when the project is large, complex and costly.

Once the analysis of the user requirement is complement, the system has to check for the compatibility and feasibility of the software package that is aimed at. An important outcome of the preliminary investigation is the determination that the system requested is feasible.

3.3.1 Types of Feasibility

- **Technical Feasibility**
- **Operational Feasibility**
- **Economical Feasibility**

3.3.1.1 Technical Feasibility

The application can be developed with the current equipments and has the technical capacity to hold the data required by the system.

- This technology supports the modern trends of technology.
- Easily accessible, more secure technologies.

3.3.1.1.1 Software and Platforms Used

- **JDK 1.6** The **Java Development Kit** (**JDK**) is an implementation of either one of the Java Platform, Standard Edition, Java Platform, Enterprise Edition, or Java Platform, Micro Edition platforms released by Oracle Corporation in the form of a binary product aimed at Java developers on Solaris, Linux, macOS or Windows.
- **Apache Tomcat**, often referred to as **Tomcat Server**, is an open-source Java Servlet Container developed by the Apache Software Foundation (ASF). Tomcat implements several Java EE specifications including Java Servlet, JavaServer Pages (JSP), Java EL, and WebSocket, and provides a "pure Java" HTTP web server environment in which Java code can run.
- **MySQL server 5** is an open source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language.
- **NetBeans** is an integrated development environment (IDE) for Java. NetBeans allows applications to be developed from a set of modular software components called modules.
- Operating system:**Windows 10** is a series of personal computer operating systems produced by Microsoft as part of its Windows NT family of operating systems

Technical feasibility on the existing system and to what extend it can support the proposed addition.

We can add new modules easily without affecting the Core Program. Most of parts are running in the server using the concept of stored procedures.

3.3.1.2 Operational Feasibility

This proposed system can easily be implemented, as this is based on JSP coding (JAVA) & HTML. The database created is with MySql server which is more secure and easy to handle. The resources that are required to implement/install these are

available. The personal of the organization already has enough exposure to computers. So, the project is operationally feasible and user friendly.

3.3.1.3 Economical Feasibility

If benefits outweigh costs, then the decision is made to design and implement the system. An entrepreneur must accurately weigh the cost versus benefits before taking an action. This system is more economically feasible and budget friendly which assess the brain capacity with quick & online test. So, it is economically a good project.

3.4 EFFORT AND COST ESTIMATION:

COCOMO (Constructive Cost Model) is a regression model based on LOC, i.e. number of Lines of Code. It is a procedural cost estimate model for software projects and often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time and quality. It was proposed by Barry Boehm in 1970 and is based on the study of 63 projects, which make it one of the best-documented models.

The key parameters which define the quality of any software products, which are also an outcome of the COCOMO are primarily Effort & Schedule:

- **Effort:** Amount of labour that will be required to complete a task. It is measured in person-months units.
- **Schedule:** Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put. It is measured in the units of time such as weeks, months.

Different models of COCOMO have been proposed to predict the cost estimation at different levels, based on the amount of accuracy and correctness required. All of these models can be applied to a variety of projects, whose characteristics determine the value of constant to be used in subsequent calculations. These characteristics pertaining to different system types are mentioned below.

Boehm's definition of organic, semidetached, and embedded systems:

1. **Organic –** A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.
2. **Semi-detached –** A software project is said to be a Semi-detached type if the vital characteristics such as team-size, experience, knowledge of the various programming environment lie in between that of organic and embedded. The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience and better guidance and creativity. Eg: Compilers or different Embedded Systems can be considered of Semi-Detached type.
3. **Embedded –** A software project with requiring the highest level of complexity, creativity, and experience requirement fall under this category. Such software requires

a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.

3.4.1 Basic COCOMO Model

E=a*(KLOC)^b

D=c*€^d

Where E=Effort applied in person-months, D is the development time in chronological months and KLOC is estimated number of delivered lines of code for project expressed in per thousand. The coefficients a, b, c, d come from the given table considering our project to be organic.

SOFTWARE PROJECTS	A	B
Organic	2.4	1.05
Semi Detached	3.0	1.12
Embedded	3.6	1.20

Table 3.4.1 Coefficients for Basic COCOMO model

The KLOC for our project are as follows:

1. Frame Test =0.14
2. Packet Test =0.217
3. PieGraph =0.039
4. PieGraph1=0.040
5. PieTest=0.020
6. Test1= 0.037

Total=0.755

E=2.4*(0.493)^1.05

=1.14 person-months

D=2.5*(1.14)^0.38

= 6 months

The number of people required to complete this project were 2.

Software Requirements Specification

INDEX

Software Requirements Specification

1.Introduction

1.1 Purpose

In our proposed solution we focus on enhancing server abilities to observe attack pattern and detect the attack before it occurs and take adaptive measures to handle and prevent DDoS PING flooding attack.

In our proposed system, we developed a user interface that detects the malicious traffic on the network. It differentiates the legitimate traffic from the malicious traffic and aims in preventing the malicious traffic from reaching the target server before affecting the server.

1.2 Scope

In our proposed system, we developed a user interface that detects the malicious traffic on the network. It differentiates the legitimate traffic from the malicious traffic and aims in preventing the malicious traffic from reaching the target server before affecting the server. It is used by network administrators and service providers to detect the anomaly in the traffic.

2. Overall Description

2.1 Product Perspective

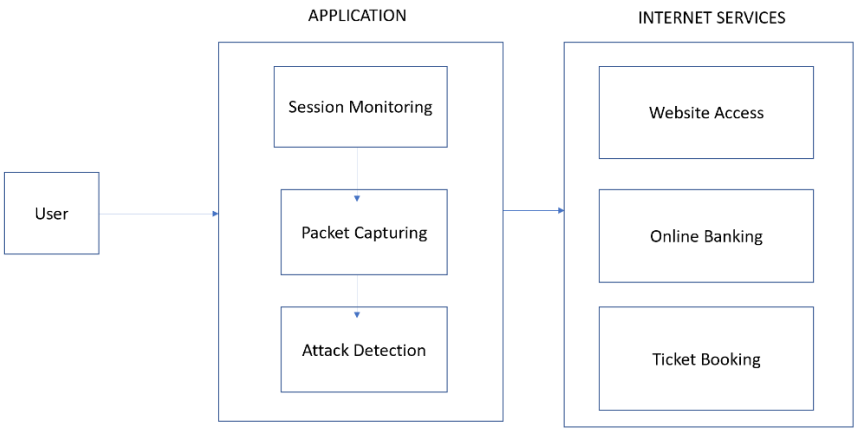

Fig 2.1 Architecture Description

2.2 Product Functions:

2.2.1 Service Registration

A registered user is a user of a website, program, or other system who has previously registered. Registered users normally provide some sort of credentials (such as a username or e-mail address, and a password) to the system in order to prove their identity: this is known as logging in. Systems intended for use by the general public often allow any user to register simply by selecting a register or sign up function and providing these credentials for the first time. Registered users may be granted privileges beyond those granted to unregistered users.

2.2.2 Session Monitoring

In our application, we chose to assign each user session into a different container; however, this was a design decision. For instance, we can assign a new session per each new IP address of the client. In our implementation, sessions were recycled based on events or when sessions time out. Thus, we could maintain a large number of parallel-running instances similar to the threads that the server would maintain in the scenario without session containers. If a session timed out, the instance was

terminated along with its container. In our application, we use a time-out to end the session.

2.2.3 Attack Detection and Prevention

The attacker visits the website as a normal user aiming to compromise the webserver process or exploit vulnerabilities to bypass authentication. At that point, the attacker issues a set of privileged (e.g., admin-level) DB queries to retrieve sensitive information. We log and process both legitimate web requests and database queries in the session traffic, but there are no mappings among them. Application separates the traffic by sessions.

DB queries will not have any matching web requests during this type of attack. On the other hand, as this traffic will not go through any containers, it will be captured as it appears to differ from the legitimate traffic that goes through the containers. Application is designed to detect and prevent DDoS attacks. These attacks can occur in the server architecture without the back-end database

2.3User Classes and Characteristics

- An application service provider (ASP) is a business providing computer-based services to customers over a network; such as access to a particular software application (such as customer relationship management) using a standard protocol (such as HTTP).The need for ASPs has evolved from the increasing costs of specialized software that have far exceeded the price range of small to medium-sized businesses. As well, the growing complexities of software have led to huge costs in distributing the software to end-users.
- A network administrator is responsible for installing, maintaining and upgrading any software or hardware required to efficiently run a computer network. The IT or computer network may extend to a local area network, wide area network, the Internet and intranets. A network administrator is the person designated in an organization whose responsibility includes maintaining computer infrastructures with emphasis on networking. Responsibilities may vary between organizations, but on-site servers, software-network interactions as well as network integrity/resilience are the key areas of focus.

2.4 Operating Environment

- Apache Tomcat, often referred to as Tomcat Server, is an open-source Java Servlet Container developed by the Apache Software Foundation (ASF). Tomcat implements several Java EE specifications including Java Servlet, JavaServer Pages (JSP), Java EL, and WebSocket, and provides a "pure Java" HTTP web server environment in which Java code can run.
- **MySQL server 5** is an open source relational database management system (RDBMS).
- **NetBeans** is an integrated development environment (IDE) for Java. NetBeans allows applications to be developed from a set of modular software components called modules.
- Operating system:Windows 10 is a series of personal computer operating systems produced by Microsoft as part of its Windows NT family of operating systems

3.External Interface Requirements

3.1 User Interfaces

It provides the types of protocol used, the ranges of packet size. It gives us the traffic description in the form of a pie chart and displays the frequency of thresholds based on the previous analysis of the attack. It also displays whether an attack is detected or not.

3.2 Software Interfaces

- **JDK 1.6** The **Java Development Kit** (**JDK**) is an implementation of either one of the Java Platform, Standard Edition, Java Platform, Enterprise Edition, or Java Platform, Micro Edition platforms released by Oracle Corporation in the form of a binary product aimed at Java developers on Solaris, Linux, macOS or Windows.
- **Apache Tomcat**, often referred to as **Tomcat Server**, is an open-source Java Servlet Container developed by the Apache Software Foundation (ASF). Tomcat implements several Java EE specifications including Java Servlet, JavaServer

Pages (JSP), Java EL, and WebSocket, and provides a "pure Java" HTTP web server environment in which Java code can run.

- **MySQL server 5** is an open source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language.
- **NetBeans** is an integrated development environment (IDE) for Java. NetBeans allows applications to be developed from a set of modular software components called modules.
- Operating system: **Windows 10** is a series of personal computer operating systems produced by Microsoft as part of its Windows NT family of operating systems

4.System Features

4.1 Attack Detection and Prevention

4.1.1 Description and Priority

This feature involves performing detection of the attack by using the application. It is a high priority function as it is the main functionality and purpose of the system-to recognize the attack so that we will be able to prevent it.

4.1.2 Stimulus/Response Sequences

- The service provider is registered which triggers the application service.
- Monitoring is performed to gain session information
- If there is an attack, the attack is therefore detected and prevented from further loss**.**

4.1.3 Functional Requirements

REQ-1: The JDK includes a private JVM and a few other resources to finish the development of a Java Application.

REQ-2: NetBeans is an integrated development environment (IDE) for Java. It allows our application to be developed from a set of modular software components called modules.

5.Other Non-functional Requirements

5.1 Performance Requirements

The software system to able to fulfill its purpose i.e, detecting the attack and preventing the attack with the best possible utilization of all necessary resources (time, storage, transmission channels, and peripherals)

5.2 Software Quality Attributes

5.2.1 Usability

Our application can be viewed as a tool for network admins to monitor any suspicious activity in the network. The application compares the results of the session with the previous data and detects whether a DDOS attack has occurred or not.

5.2.2 Efficiency

The software system to able to fulfill its purpose i.e., detecting the attack and preventing the attack with the best possible utilization of all necessary resources (time, storage, transmission channels, and peripherals)

5.2.3 Robustness

Robustness reduces the impact of operational mistakes, erroneous input data, and hardware errors. Our user interface-based application is robust and the consequences of an error in its operation, in the input, in relation to the application, are inversely proportional to the probability of the occurrence of this error in the given application.

4. SYSTEM DESIGN

4.1 System Architecture

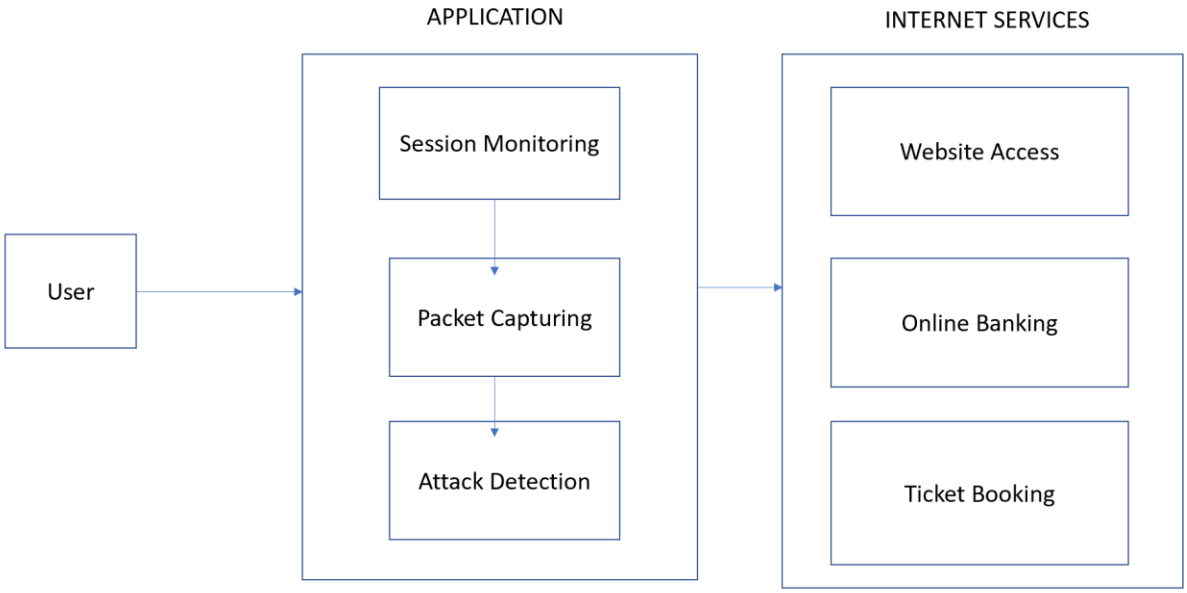

Fig 4.1 System Architecture

4.2 UML Diagrams

4.2.1 Use Case Diagram

4.2.1.1 Use C for Service Provider

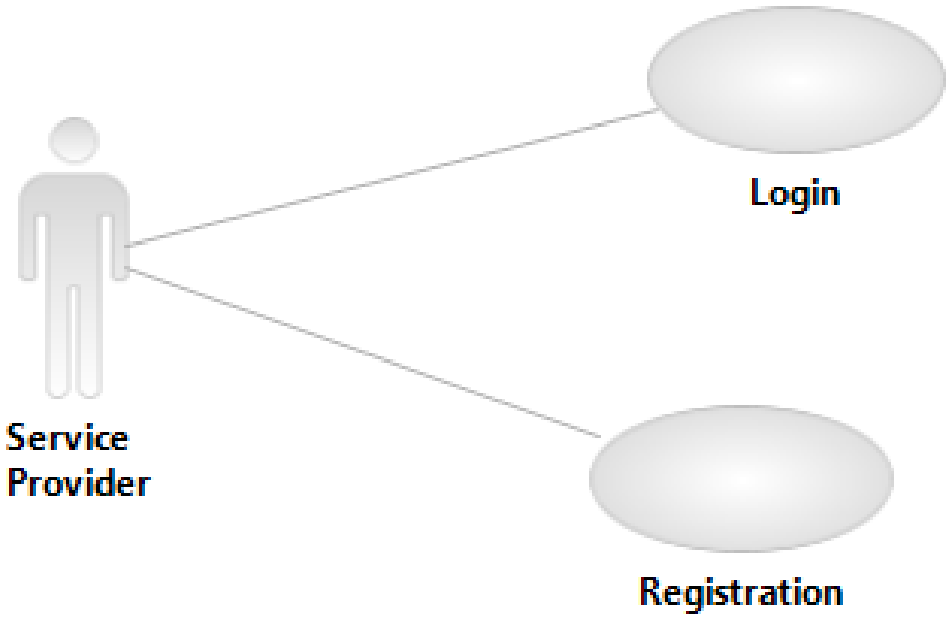

Fig 4.2.1.1 Use Case Diagram for Service Provider

4.2.1.2 Use Case Diagram for Network Admin

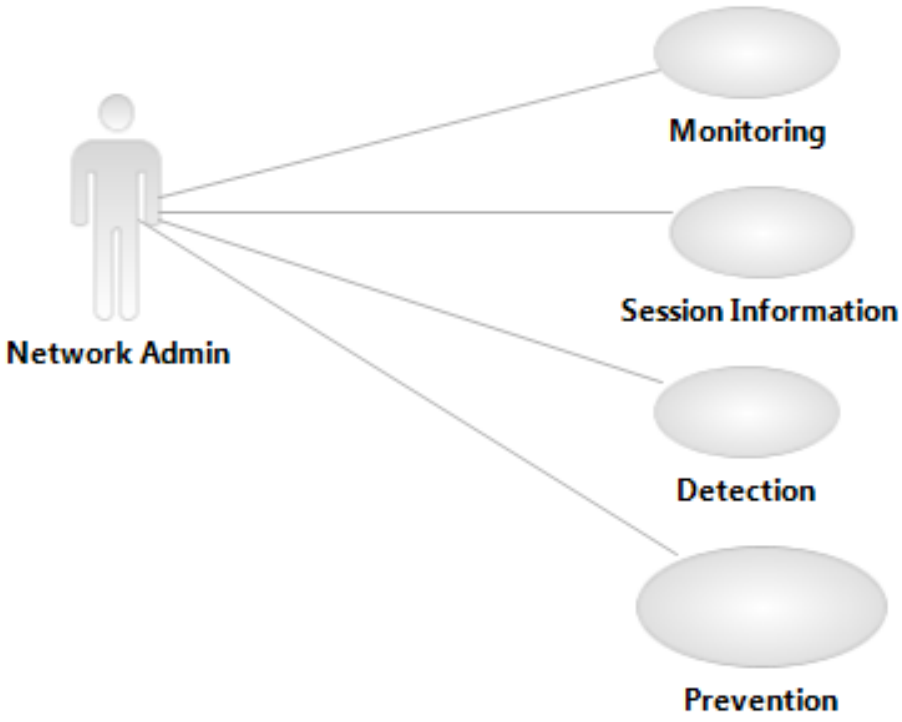

Fig 4.2.1.2 Use Case Diagram for Network Admin

4.2.2 Activity Diagram

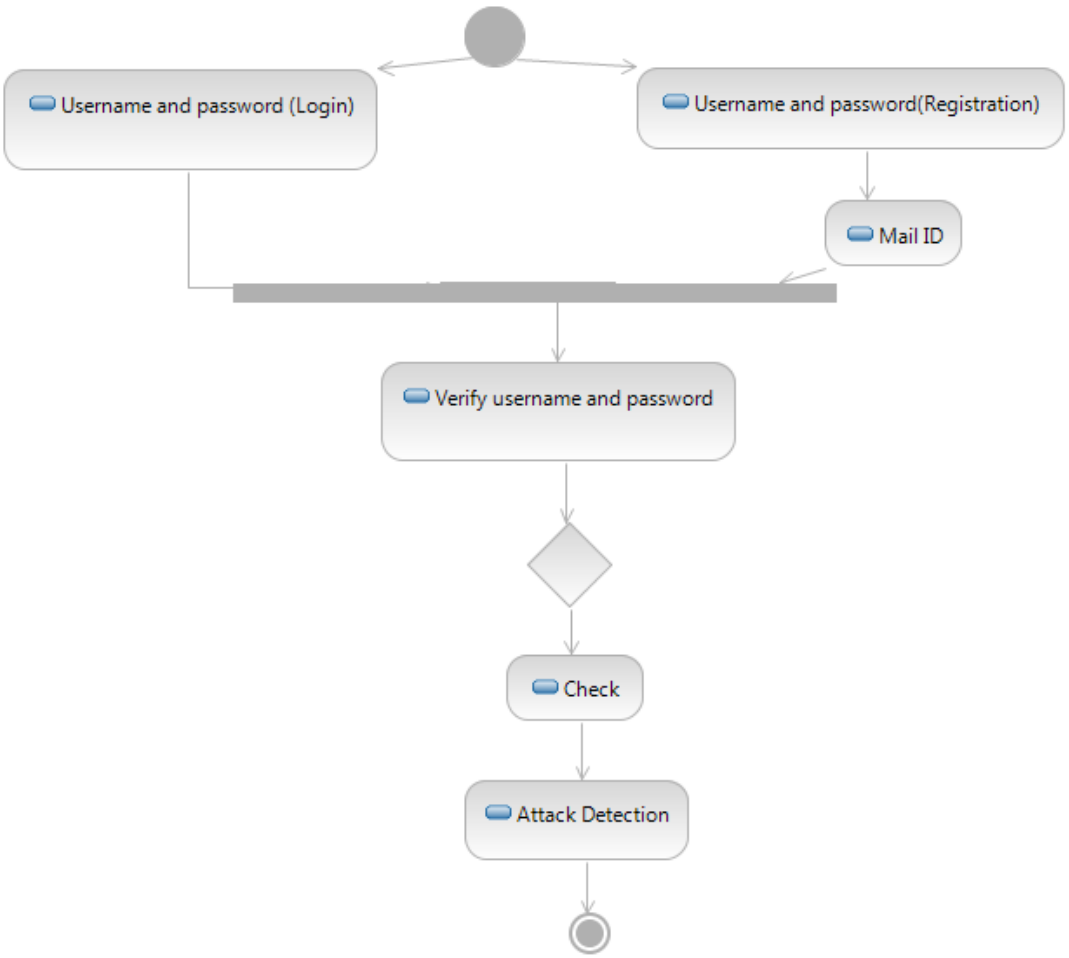

Fig 4.2.2 Activity Diagram

4.2.3 Sequence Diagram

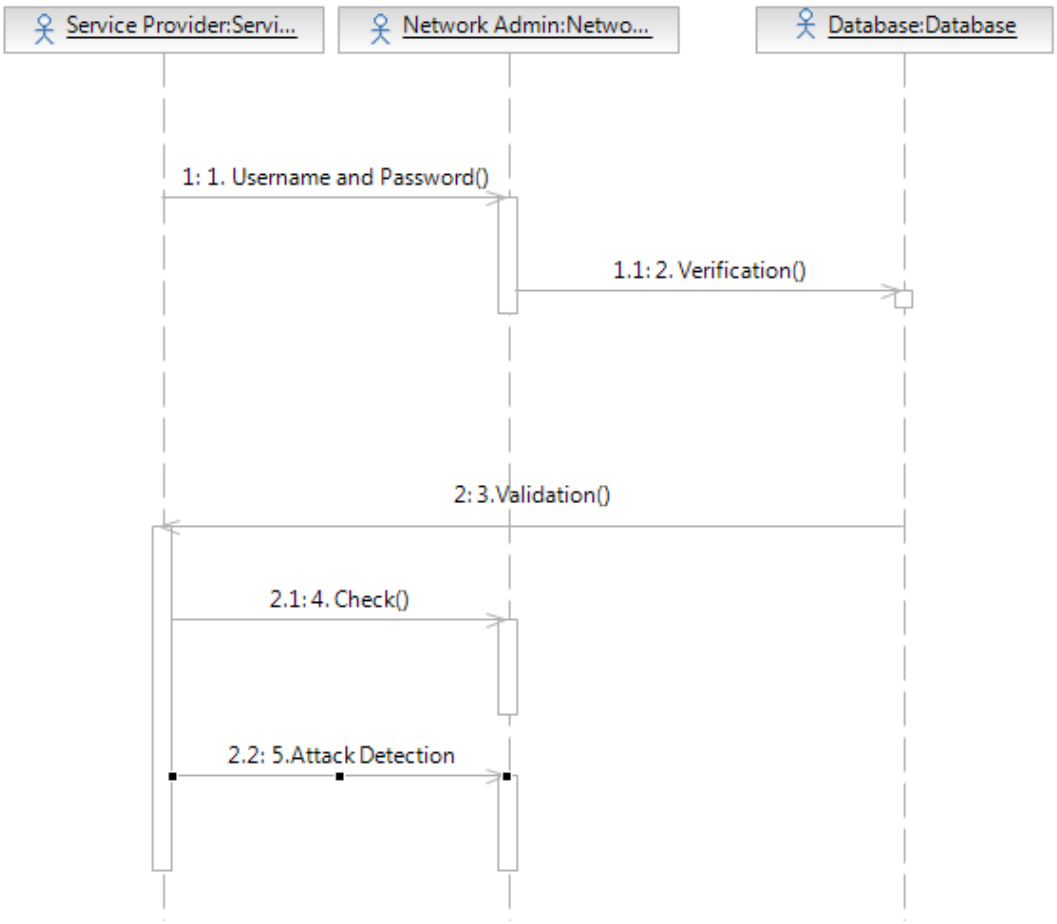

Fig 4.2.3 Sequence Diagram

5.IMPLEMENTATION

5.1 MODULES

List of Modules

- Service Registration
- Session Monitoring
- Attack Detection and Prevention

5.1.1 Service Registration

A **registered user** is a user of a website, program, or other system who has previously registered. Registered users normally provide some sort of credentials (such as a username or e-mail address, and a password) to the system in order to prove their identity: this is known as logging in. Systems intended for use by the general public often allow any user to register simply by selecting a register or sign up function and providing these credentials for the first time. Registered users may be granted privileges beyond those granted to unregistered users.

5.1.1.1 Privacy concerns

Registration necessarily provides more personal information to a system than it would otherwise have. Even if the credentials used are otherwise meaningless, the system can distinguish a logged-in user from other users and might use this property to store a history of users' actions or activity, possibly without their knowledge or consent. While many systems have privacy policies, depending on the nature of the system, a user might not have any way of knowing for certain exactly what information is stored, how it is used, and with whom, if anyone, it is shared. A system could even sell information it has gathered on its users to third parties for advertising or other purposes. The subject of systems' transparency in this regard is one of ongoing debate.

5.1.1.2 User inconvenience

Registration may be seen as an annoyance or hindrance, especially if it is not inherently necessary or important (for example, in the context of a search engine) or

if the system repeatedly prompts users to register. A system's registration process might also be time-consuming or require that the user provide information they might be reluctant to, such as a home address or social security number.

5.1.2 Session Monitoring

In our application, we chose to assign each user session into a different container; however, this was a design decision. For instance, we can assign a new session per each new IP address of the client. In our implementation, sessions were recycled based on events or when sessions time out. Thus, we could maintain a large number of parallel-running instances similar to the threads that the server would maintain in the scenario without session containers. If a session timed out, the instance was terminated along with its container. In our application, we use a time-out to end the session.

Our application takes the input of training data set .For each unique HTTP request and database query, we assign a hash table entry, the key of the entry is the request or query itself, and the value of the hash entry is AR for the request or AQ for the query, respectively. The application generates the mapping pattern by considering all mapping patterns that would happen in websites.

5.1.4 Attack Detection and Prevention

The attacker visits the website as a normal user aiming to compromise the webserver process or exploit vulnerabilities to bypass authentication. At that point, the attacker issues a set of privileged (e.g., admin-level) DB queries to retrieve sensitive information. We log and process both legitimate web requests and database queries in the session traffic, but there are no mappings among them. Application separates the traffic by sessions.

If it is a user session, then the requests and queries should all belong to normal users and match structurally. Using the mapping pattern that we created during the training phase, Application aims to capture the unmatched cases. For a ping flood attack to be successful, it must change the structure (or the semantics) of the query, which our approach can readily detect.

DB queries will not have any matching web requests during this type of attack. On the other hand, as this traffic will not go through any containers, it will be captured as it appears to differ from the legitimate traffic that goes through the containers. Application is designed to detect and prevent DDoS attacks. These attacks can occur in the server architecture without the back-end database.

6. TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and /or a finished product.

6.1 Types of Testing

6.1.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests endure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Test Cases:

A test case is a set of conditions or variables under which a tester will determine whether an application, software system or one of its features is working as it was originally established for it to do.

6.1.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Test Case: Attack Detection and Prevention.

Test Objective: To check whether the attack is detected thoroughly, and the IP address is banished from the network.

Test Description: The user (service provider) checks for any anomaly. If attack is detected it discards the attacker from disrupting the service.

Test Results: All the test cases mentioned above passed successfully. No defects encountered leading to the detection of the attack.

Image Result:

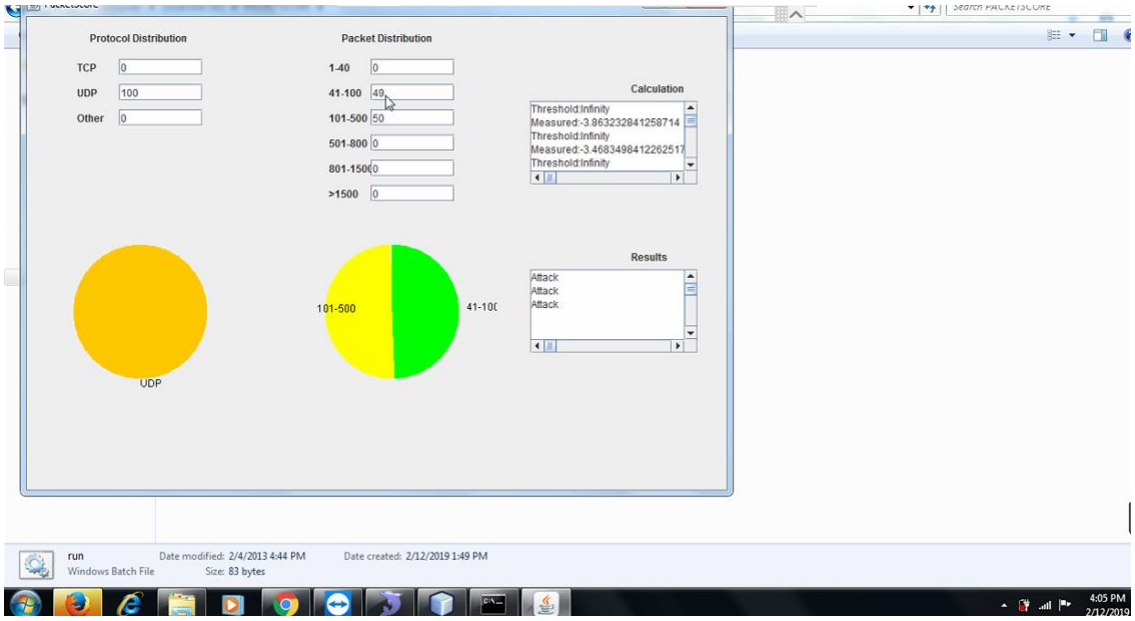

Fig 6.3 Image Result

6.1.3 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner working, structure or knowledge of the module being tested. These tests can be functional or non-functional, though usually functional. This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories.

Test Results:

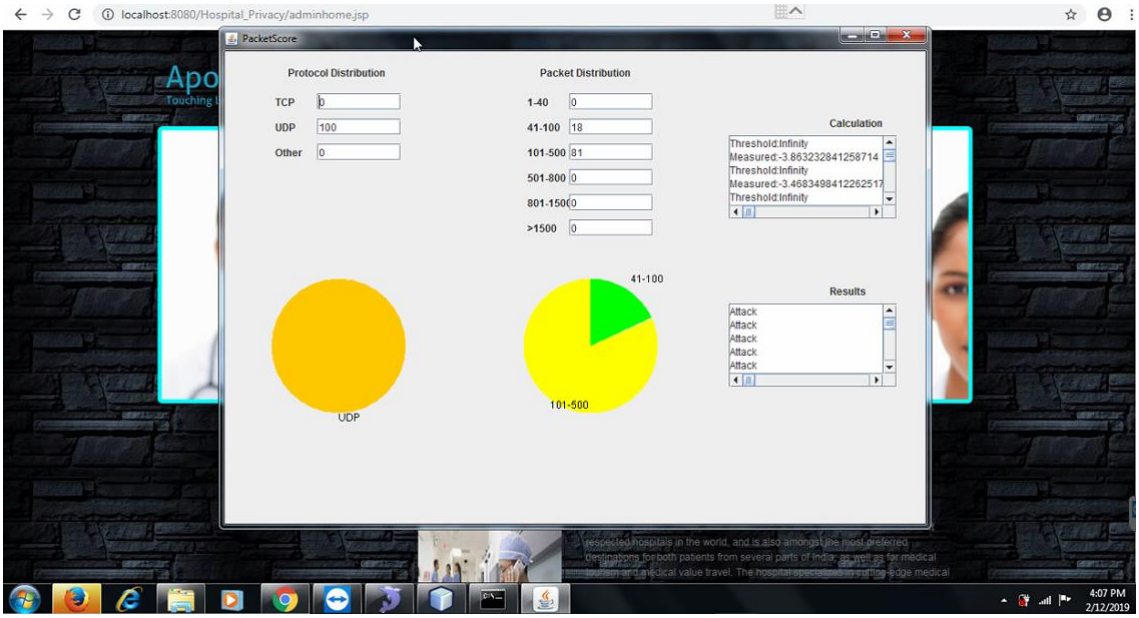

Fig.6.4 Detailed analysis of packet

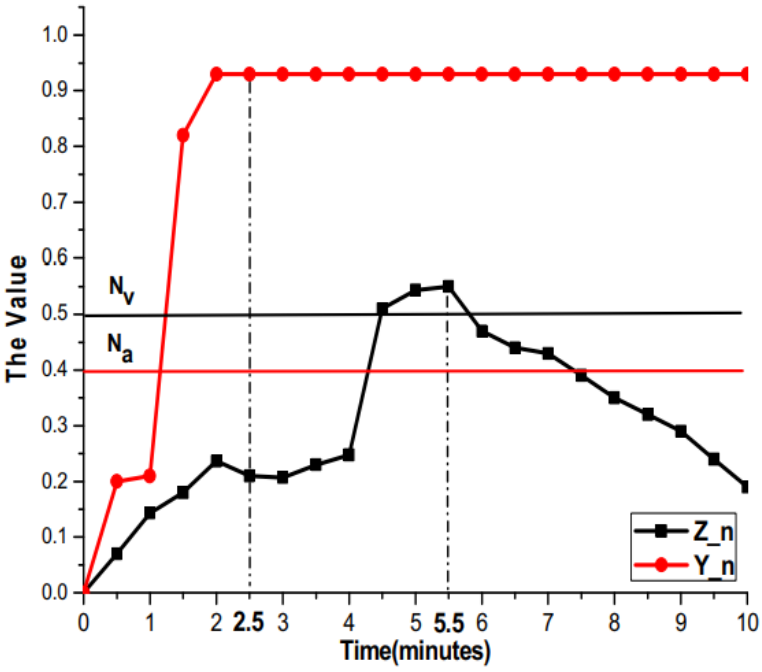

Fig 6.5 Traffic analysis of network

Test Case Id	Test Case Name	Test Case Desc.	Test Steps		
			Input	Expected	Requirement verified
01	Service Registration	To check whether the service provider is registered with the user interface-based application.	Username and password.	Commencement of service	yes
02	Session monitoring	To monitor the network traffic generated by different user requests and to detect any suspicious activity.		Monitoring of the packet traffic.	yes
03	Attack Detection	The user-interface based application starts to monitor the network traffic. If an anomaly is observed, then it notifies the network admin and further		Detection of the malicious IP address (attacker).	yes

		measures are taken to ban the malicious IP (attacker).			
04	Attack Detection and Prevention.	The user (service provider) checks for any anomaly. If attack is detected it discards the attacker from disrupting the service.		All the test cases mentioned above passed successfully. No defects encountered leading to the detection of the attack.	yes
05	Performance	We see here the different ranges of the packet size, the protocol used and distribution of these packet sizes.			yes

Table.6 Test cases

7. SCREENSHOTS

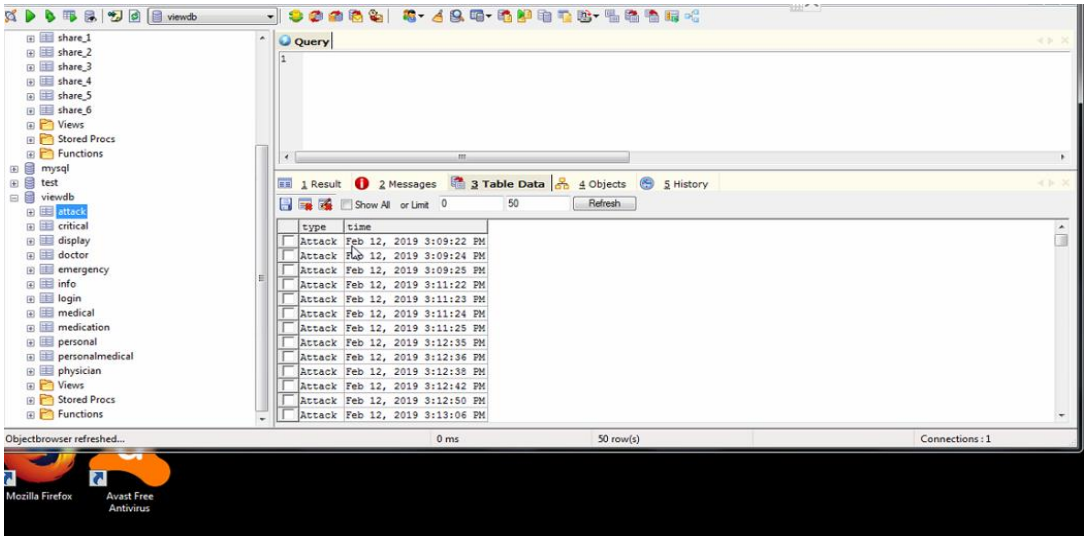

Fig.7.1 Screenshot of MYSQL GUI

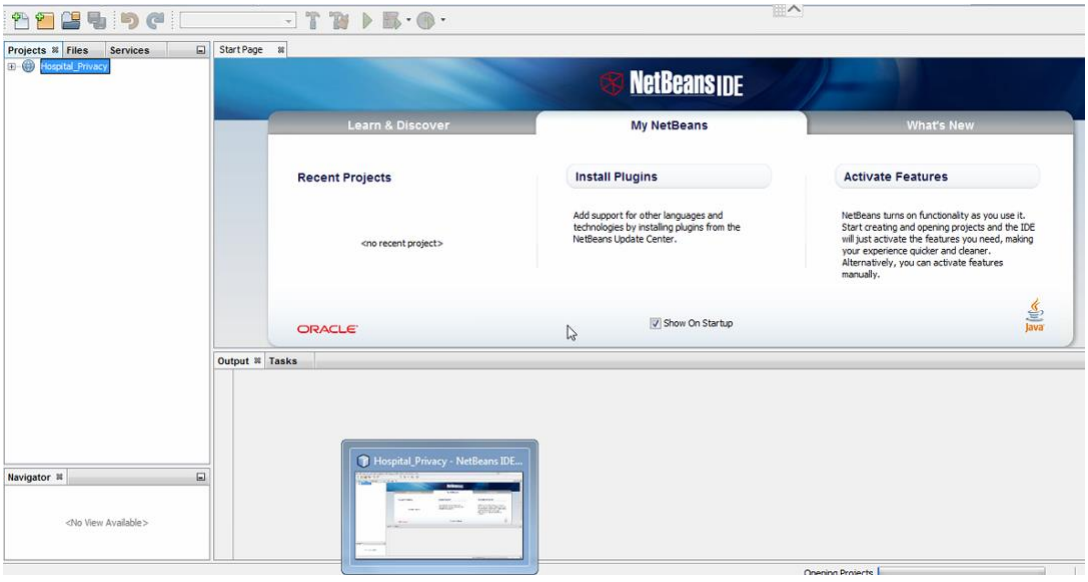

Fig.7.2 Screenshot of NetBeans IDE

Fig.7.3 Screenshot of static website

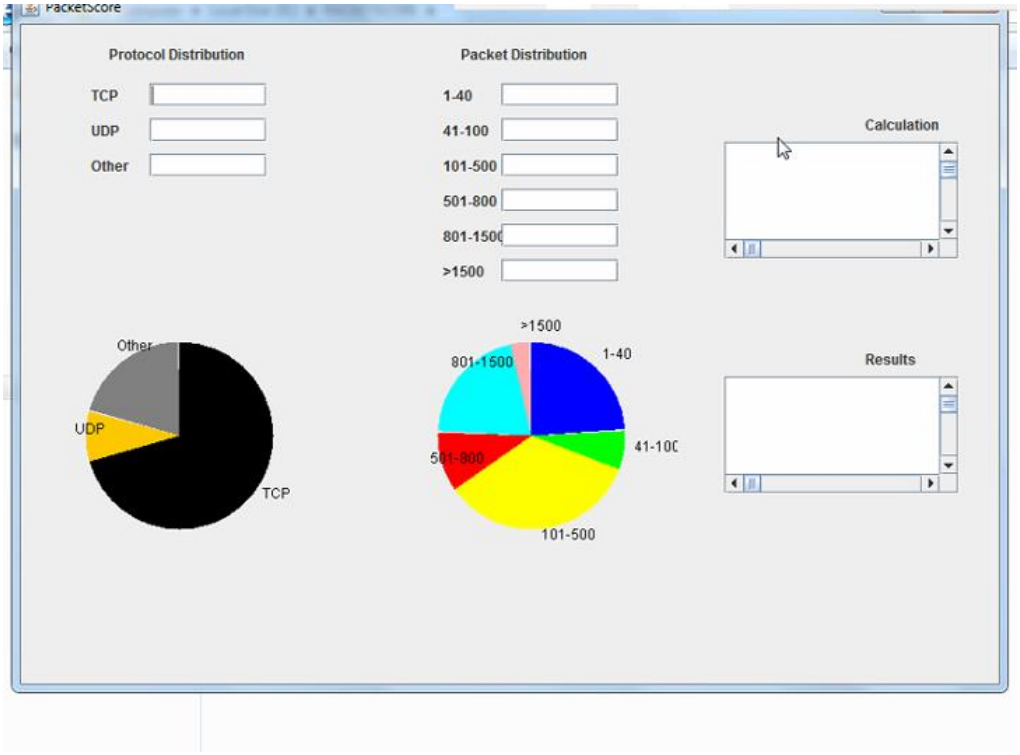

Fig. 7.4 Screenshot of user interface-based application

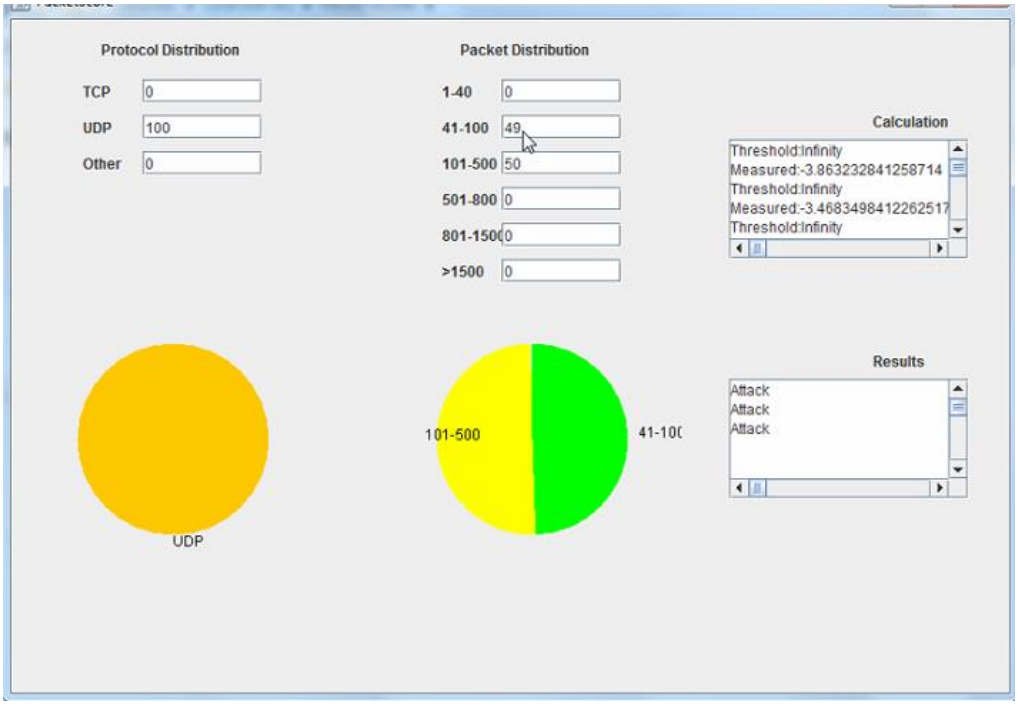

Fig.7.5 Screenshot of analysis of packet traffic

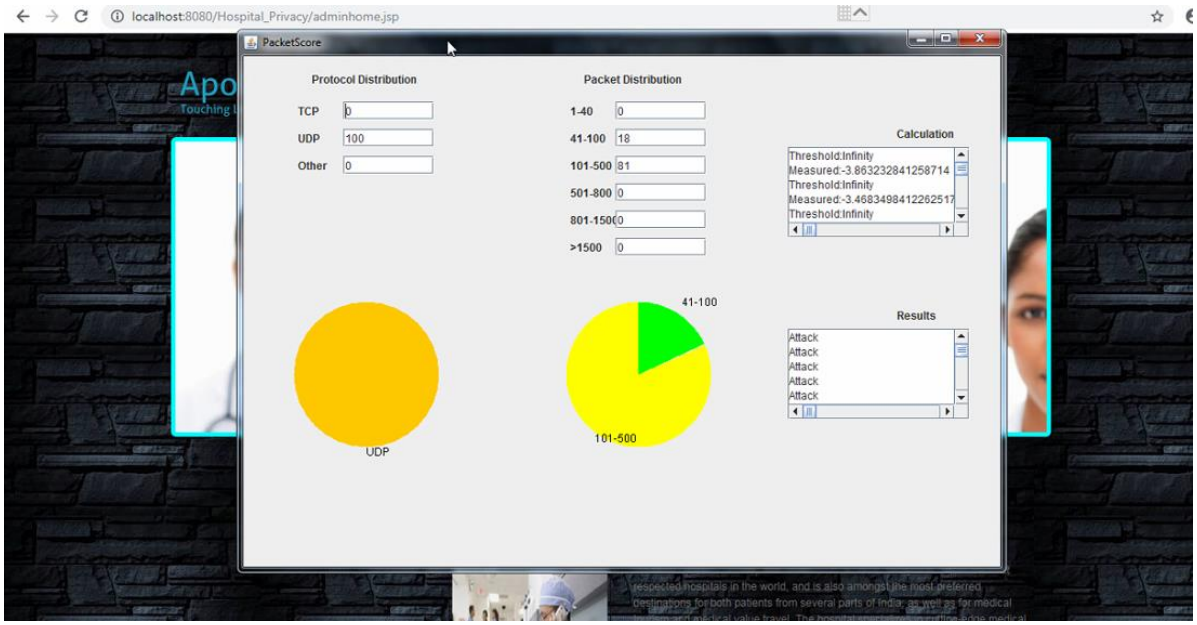

Fig.7.6 Screenshot of attack detection

8. CONCLUSION

A DDOS attack is one of the complicated attacks to defend against. There is need to identify these attacks, manage them as well as prevent them and look into ways of differentiating between malicious and legitimate users. It is a common persistent problem faced till date. In the solution provided it has been possible to observe suspicious activity in the network and the corresponding traffic flow, analyse it and then restrict the attacker from downing the service. In the proposed application solution, the services successfully register and post registration any malicious traffic on the network directed towards it is monitored and successfully restricted thus ensuring that the services is almost always up.

9. REFERENCES

[1] Autobench, http://www.xenoclast.org/autobench/, 2011

[2] “Common Vulnerabilities and Exposures,” http://www.cve

[3] “Five Common Web Application Vulnerabilities,” http://www.symantec.com/ connect/articles/ five-common-web-applicationvulnerabilities,2011.

[4] greensql, http://www.greensql.net/, 2011.

[5] httperf, http://www.hpl.hp.com/research/linux/httperf/, 2011.

[6] http_load, http://www.acme.com/software/http_load/, 2011.

[7] Joomla cms, http://www.joomla.org/, 2011.

APPENDIX I

RELEVANCE OF PROJECT TO POs and PSOs

PROJECT TITLE	CATEGORY
Detection cum Prevention of DOS/DDOS Attack	Application

ABSTRACT

The term denial of Service (DOS) refers to form an attacking computer over a network. The denial of service attack is an explicit attempt by an attacker to prevent the legitimate users not to access the services. When this attack is made at a larger amount that is by using multiple computers than it's known as Distributed Denial of Service Attack (DDoS). An attacker can use many techniques for denial of service like flooding technique is to flood a network and reduce the legitimate user bandwidths to disrupt the services of the users. In DDoS attack, the attacker tries to interrupt the services of a server and utilizes its CPU and Network. Flooding DDOS attack is based on a huge volume of attack traffic which is termed as a Flooding based DDOS attack. Flooding-based DDOS attack attempts to congest the victim's network bandwidth with real-looking but unwanted IP data. Due to which Legitimate IP packets cannot reach the victim because of lack of bandwidth resource. ICMP FLOOD initiated by sending a large number of ICMP packets to a remote host. As a result, the victimized system's resources will be consumed with handling the attacking packets, which eventually causes the system to be unreachable by other clients. In this research firstly, we detect the ICMP Flood by using various methods and tools and then find out the prevention techniques for DDOS attack. In our project we have proposed a solution which enhances server abilities through traffic pattern observation and attack detection before the attack occurs and taking adaptive measure of banning the attacker IP and thus preventing DDoS PING flooding attack.

PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
3	3	3	2	2	3	1	2	3	2	3	2

Table [put table no] Mapping to PSOs: (Mapping Scale [1-3]: 1-Slight 2-Moderate 3-Strong)

PSO1	PSO2	PSO3
3	2	3

RELEVANCE DETAILS	
Implementation details The project is implemented using JAVA language, Tomcat Server and NetBeans IDE	
PO and PSO Justification	
PO1	It is moderately mapped as the captured traffic for the service is analysed using the basic concepts of mathematics.
PO2	It is strongly mapped as the captured traffic analysis formulation is based on the review literature.
PO3	It is strongly mapped as availability, responsiveness and efficiency of the application service are considered which cater to the safety needs.
PO4	It is moderately mapped as investigations were conducted on the different samples of captured traffic.
PO5	It is moderately mapped as the NetBeans IDE has been used to develop the application.
PO6	It is strongly mapped as contextual knowledge of the captured traffic is used to prevent service denial.
PO7	It is weakly mapped as it addresses service application availability.
PO8	It is strongly mapped because the plagiarism tool can be used to check the authenticity and originality of the work done by the students
PO9	Projects are used to inculcate group work and to manage a team for promoting knowledge, conceptualization and delivering same with varied complexity, therefore the mapping is strong.
PO10	Demonstrate versatile and effective communication skills, both verbal and written with team members and present the product to the audience in comprehensive manner. Therefore, the mapping is moderate.
PO11	Basic knowledge and understanding of engineering principles are applied in the development of this project hence a moderate mapping.
PO12	The mapping is moderate as projects are executed based on the self-learning or self-efforts put in by the group.
PSO1	It is strongly mapped as understanding of the principles and working of the hardware and software aspects of computer systems and the networking environment is required to decompose the system into phases and workflows.
PSO2	It is moderately mapped as the application the tools used for development, operation and maintenance are minimal.
PSO3	It is strongly mapped as self-learning skills are applied for a real-world problem by using content knowledge and acquired intellectual skills. Our application is used for detecting and preventing the PING flooding attack.

Academic Year	2018-19	
Project Title	Detection cum Prevention of DOS/DDOS Attack	
Project Supervisor	Mrs. Maniza Hijab	
Start Date	4-Apr-18	
End Date	12-Apr-19	

Overall Progress

S.No.	Tasks	Start	End	Days	Status
1	**Project and Batch Registration**	**4-Apr-18**	**27-Apr-18**	**23**	**Completed**
2	Problem Definition	28-Apr-18	7-May-18	9	Completed
3	Abstact	20-Jul-18	9-Aug-18	20	Completed
4	Literature Survey	9-Aug-18	11-Sep-18	33	Completed
5	Requirements Analysis (including H/w and S/w)	9-Aug-18	11-Sep-18	33	Completed
6	**Project Seminar**	**18-Aug-18**	**16-Sep-18**		**Completed**
7	**Project Review 1**	**6-Oct-18**	**10-Oct-18**	**2**	**Completed**
8	Fesibility Study	11-Sep-18	29-Sep-18	18	Completed
9	Detailed Design / Module Identification	16-Sep-18	30-Sep-18	14	Completed
11	Implementation	25-Sep-18	10-Mar-19	166	Completed
12	Testing	1-Jan-19	10-Mar-19	68	Completed
13	Result Analysis	1-Feb-19	10-Mar-19	37	Completed
14	**Project Review2**	**11-Mar-19**	**16-Mar-19**	5	Completed
15	Report Compilation	17-Mar-19	28-Mar-19	11	Completed
16	Viva Voice	15-Apr-19	27-Apr-19	12	---