National College of Ireland

Dedicated resources for the software puzzle framework for more efficient way of overcoming DDoS attacks

MSc Research Project
MSc in Cloud Computing

Ashish Chinnappa Thirutera Madappa
Student ID: X17155801

School of Computing
National College of Ireland

Supervisor: Dr. Sachin Sharma

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ashish Chinnappa Thirutera Madappa
Student ID:	X17155801
Programme:	MSc in Cloud Computing **Year:** 2018/2019
Module:	MSc Research Project
Supervisor:	Dr. Sachin Sharma
Submission Due Date:	20/12/2018
Project Title:	Dedicated Resources for the Software Puzzle Framework for more Efficient Way of Overcoming DDoS Attacks
Word Count:	6753 **Page Count:** 19

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: .....

Date: 20th December 2018

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Dedicated resources for the software puzzle framework for more efficient way of overcoming DDoS attacks

Ashish Chinnappa Thirutera Madappa
X17155801

Abstract

Denial-of-service (DoS) and Distributed Denial-of-service (DDoS) attacks contribute themselves to be the biggest threats to the cloud environment. The current countermeasure to this security issue is given by a dynamically generated client puzzle called the software puzzle, which acts as a real-time solution and requests the client to conduct computationally expensive operations by solving the puzzle before being served with the requests by the server. The normal client-server computations interfere with the software puzzle computations running in the same architecture, affecting the efficiencies of either of them. This research proposal suggests a dedicated server for the software puzzle to execute and communicate with the actual client-server architecture using a communication network with a centralized database. This helps the two servers to perform their computations without being disturbed by either of them, enabling a better resource utilization and a more efficient way of detecting resource inflated DDoS attack.

1 Introduction

Cloud computing is one of the utility-oriented computing system which provides the clients with storage facilities and several solutions over the internet through remotely placed servers. Cloud being a distributed system, has pushed itself to various attacks and security glitches by provisioning dynamically generated resources, which is caused due to the increase in the number of users. Among the several security issues, DDoS attacks become highly dangerous as they manipulate the availability and reliability of the servers by making their services unavailable for the end users or the legitimate clients (Buyya et al; 2012). This occurs by launching an attack against the cloud service providers and bombarding the server with several garbage requests. This enables a sudden spike in the network traffic, which the server incorrectly assumes to be the legitimate requests by genuine clients. This results in the resources being scaled up and thus leading to a very high consumption of power and a misuse of various server resources (Chonka et al; 2012).

As a countermeasure to the above issue and to help in detecting and overcoming resource-inflated DDoS attacks, Software puzzle was proposed. It is a dynamically generated client puzzle, which generates the puzzle only after the server receives the client request such that: 1. An attacker finds it extremely difficult to solve the puzzle in advance, as an attacker would be unprepared to do so and 2. The attacker require high volume of resources to computationally translate a CPU version of the puzzle software into its respective GPU version which is highly impossible to be carried out in real time (Wu et al; 2015).

Considering a scenario where the server integrated with software puzzle, is engaged in serving the genuine requests of the clients and if the software puzzle might as well have to execute its operations of detecting and authenticating the client requests, then the resources of that particular server have to be shared between the two processes or rather be given to the one with highest priority. This makes either of the processes to wait for the other to finish, affecting their time and resource efficiencies (Malet et al; 2010). Therefore, allocating a dedicated server for the software puzzle by moving it away from the normal client-server, enables the software puzzle server to carry out only the puzzle related work and not interfere with the operations of the general client-server. A dedicated architecture for the software puzzle upholds the efficient use of both resources and time and consists of a very simple server. This server accommodates a code block warehouse and other computational resources needed to generate, encrypt and validate the client puzzle (Wu et al; 2015). Once the software puzzle architecture is separated from that of the usual client-server architecture, there exists a requirement for the two servers to communicate between each other. Using various available wireless communication networks, the proposed architecture introduces a communication network which enables the packet transmission between the software puzzle server and the client-server through a centrally placed database. The client-server updates the database with the client credentials to be used by the software puzzle and the software puzzle updates the puzzle result inference into the same database, which the client-server uses to decide whether to serve the client or not. This becomes extremely simple for the packet transmission in the proposed architecture, such that the communication between the software puzzle architecture and the client-server architecture adheres to a highly efficient network topology and becomes technologically updated.

The remaining part of this paper has been organized as follows: The literature review explains the background and consequences of DDoS attacks demonstrating the different ways the system gets prone to these attacks and its effects, emergence and drawbacks of client puzzles, existence of software puzzle demonstrating the overview of various countermeasures for DDoS attacks and how software puzzle stands best among the others and finally the methodology, research implementation of the proposed system, conclusion and the future work.

2 Related Work

2.1 Consequences of DDoS attacks

Cloud computing deals with performing various computations and providing other software and storage services to the clients from remotely placed servers where the end-users have absolutely no knowledge about the physical location of their data and configurations of the system (Jadeja and Modi; 2012). These developments in the cloud environment are suppressed to a large extent by various security issues. Among them, DoS and DDoS attacks play a major role in affecting the availability of the cloud services for the clients or the end-users. In these type of security attacks, an attacker attempts to make the system and network resources unavailable to the genuine users by flooding the target with a large volume of garbage packets along with tampered source IP addresses. Due to machine networks being compromised by malware, there has been a drastic growth in the rate of DDoS attacks in the cloud computing environment (Bicakci and Tavli; 2009).

Some of the other reasons triggering the rate of these attacks are given below by Yan et al. (2015):

a. The broadcast nature of the wireless networks leads to a lot of suspicious attacks resulting in other vulnerabilities of confidentiality and authentication.
b. On-demand feature of cloud environment results in the outbreaking of botnets caused by malware.
c. Quantified measured services with rapid elasticity gives birth to the new breed of DDoS attacks.
This leads to shutting down of the active servers and destructing the accessibility of its services by consuming the service provider's resources like storage and network bandwidth. The network placement and a centralized logical controller supports in building a consistent security change. In this context, TCP SYN flooding results in misusing the weakness in the three-way handshake protocol through a TCP network establishment. The SYN packet sent by the client will be received by the server to allocate a state for holding the half-open connection information which would be completed by the client after receiving the SYN-ACK packet from the server. As a result of the SYN flooding attacks, the genuine client's requests get blocked from being served as the server's half-open waiting queue gets exploited by the attackers (Wang and Reiter; 2003). The layered approach of the cloud network architecture welcomes potential malicious distributed denial of service attacks on them. In an application layer DDoS attacks, the application programming interface of the system is affected as the attacks on any one of the applications become contagious to rest of them within the same system (Xie et al; 2009). In a control layer DDoS attack, the flooded packets arrived at the data plane are sent directly to the control plane to resolve the query. But this transmission leads to high bandwidth consumption. Thereby, risking the single point of failure for the network and being the target to these attacks (Sheinidashtegol et al; 2017). Finally, in an infrastructure layer DDoS attacks, the attacker sends packets with insufficient information, so that they are stored in the node memory until the incomplete information is returned. This opportunity is used by the attacker to flood the switch memory, disabling the data plane to store the resources for the normal network flows (Agha et al; 1993). The network layer attacks take place through ICMP flooding, SYN flooding and UDP flooding intending to deny services to legitimate users by consuming the network bandwidth (Xie et al; 2009).

Upon a research conducted, the performance of web application and web servers in a virtual machine with different operating systems can decrease by up to 23% where as a non-virtualized server built on the same hardware as the other shows a decrement by only 8% (Shea et al; 2013). This proves that virtualization is yet another important asset of cloud computing that is being affected by DDoS attacks and other security issues.

2.2 Evolution of client puzzle, as an initial countermeasure to DoS attacks

In the path of my chosen research topic, client puzzles have been evolved as the initial countermeasure for DDoS attacks. According to (Wang and Reiter; 2003), the idea of client puzzles can help in fighting against the attacks associated with the zombie-computers. The existing DDoS tools are designed in such a way that they fail to awaken these zombie computers in order to avoid alerting the machine owners about their existence. This motivates the server or the system owner to stop the attack. TCP-based and TLS-based puzzle schemes are known to be the pioneer implementations of the client puzzles against the resource-

inflated DDoS attacks. Though authentication creates a pathway for DoS attacks by storing the session-specific state data, some of the authentication security frameworks such as SYN-cookie protection act as a weaker strategy against the SYN attacks. Here, the client is sent a nonce which must be sent back in the upcoming message, in order to verify that the return address is non-fictional. This strategy becomes unreliable as the protocol takes longer to run, making those large number of messages difficult to analyse (Aura et al; 2000).
A basic client puzzle consists of two algorithms, one to randomly generate or create client solutions and the other one is to verify the client solution to its correctness or authenticity. A complete sequence of generating and verifying the client solutions using the two algorithms is known as a trial.

According to (Wang and Reiter; 2003), the resource allocation problem of the client puzzle is represented as a tuple (C,A,S,V,R) where,

C is a set of legitimate clients
A denotes a set of adversaries
S is the server
R denotes the server resources
V refers to the valuation function giving the total number of trials to be performed to combat the attack with reasonable waiting period.

An important logic runs with the rule of thumb that, the resource efficiency cost of the client should be greater than that of the server to make a DoS authentication strategy successful and to combat it. Thus, client puzzles came into existence where the malicious clients creating a traffic to the server, are sent server generated puzzles to solve it. These clients use their resources to solve the puzzles and send the puzzle results back to the servers for verification (Fraser et al; 2007). Only after authenticating the puzzle results, the clients are served. But unfortunately, these puzzles were generated well in advance and sent to the clients. The clients therefore had the puzzle answers ready with them and they started to inflate the resources by solving the puzzle much faster with the help of built-in GPUs. This gradually pulled down the puzzle effectiveness, which significantly weakened the client puzzles. These drawbacks lead to the emergence of software puzzles (Waters et al; 2004).

2.3 Existence of Software Puzzle

An experimental result portrayed software puzzle to be a more evolved client puzzle that dynamically generates the puzzle only after the server receives the client requests such that execution time of the clients become more than the computation time of the servers. The author Wu et al. (2015) denotes this ratio to be γ, obtained by the resource time consumption by the client to solve the puzzle (*tc*) and the resource time consumed by the server to carry out puzzle generation and validation (*ts*).

ie, $\gamma = tc / ts$

This ratio must be increased to overcome the resource-inflation and this can be achieved only by increasing the client computational cost and decreasing the server execution cost through an important puzzle scheme called the hash-reversal. This forces the client to get through a one-way hash instance such as a block cipher AES or SHA-1 and thereby increases the *tc* value. A puzzle challenge '*x*' which is randomly generated by the server, is then sent to the

client to solve it. The client's response *(x,y)* is then verified by the server with the help of a generated public puzzle function *P*. Since an attacker cannot perform the DDoS attacks by wasting more time than the server in solving the puzzle, he effortlessly replies to the puzzle by sending an arbitrary value *~y* so that the server exhausts a large amount of resources in verifying the result. There is a huge misconception that the client uses only the traditional CPU resources for solving the puzzle. This is however shown in Fig-1 that the client also uses a multi-core GPU along with an integrated CPU-GPU duo which is required to inflate the resource computations. This gradually decreases the ratio γ, negatively affecting the significance of the current client puzzles (Wu et al; 2015). The probability of the DDoS attacks increases when an attacker utilizes its multi-core GPU to solve each of the puzzle using independent core, irrespective of the puzzle function being parallelizable or non-parallelizable (Waluni et al; 2017).

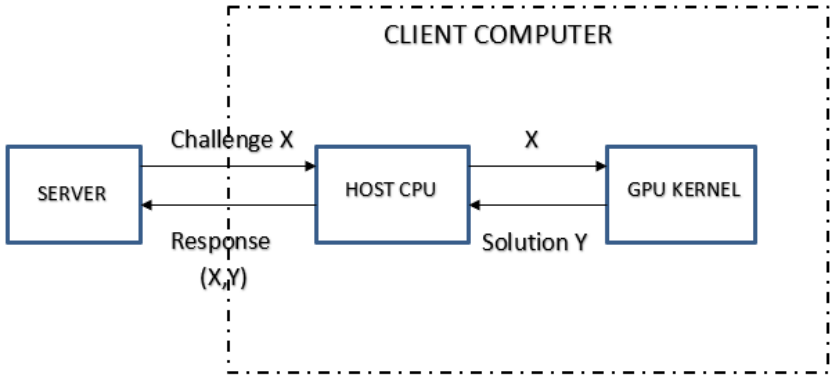

Figure 1: Overview of the GPU-inflated DDoS attacks (Wu et al; 2015).

Therefore, the software puzzle is created in a way that, an attacker must translate the CPU version of the puzzle into its equivalent GPU version as both of them have a completely different internal architecture and instruction sets used for various applications. Also, this rewriting or translation of the software puzzle from CPU to GPU is highly time consuming and complicated which might take even longer than solving the puzzle directly on the host CPU, making software puzzle to easily get rid of GPU-inflated DDoS attack (Sorte et al; 2016).

Fig-2 demonstrates the working of the software puzzle embedded within the client-server architecture. According to this concurrent framework, both the software puzzle application and the client-server applications are sharing the same resources for their computations. The clients send several malicious requests to the server, creating an unnecessary network traffic. The software puzzle within the server gets activated and performs the respective puzzle generation and verification operations using the available resources or wait until the resources are made free. While the software puzzle is running, the client-server applications might have to wait for the resources which are being used by the software puzzle in order to serve the genuine clients. It could also be that the software puzzle must wait until the client-server releases the required resources for them to generate the puzzle. This, sometimes results in a deadlock situation where the servers might crash, affecting the real-time cloud services. This also severely reduces the efficiency of the resources such as memory, CPU and network bandwidth in terms of time and performance (Wu et al; 2015).
Also, the previous related work scrutinizes the issues of growing cloud technology and gives a countermeasure to the DDoS attacks that runs on the server side. This in turn has a loop hole

of exhausting the server resources by not being implemented in the client systems. The study has also showed that adopting a new communication algorithm in wireless transmission will result in the efficient growth of cloud networks and best utilization of its resources.

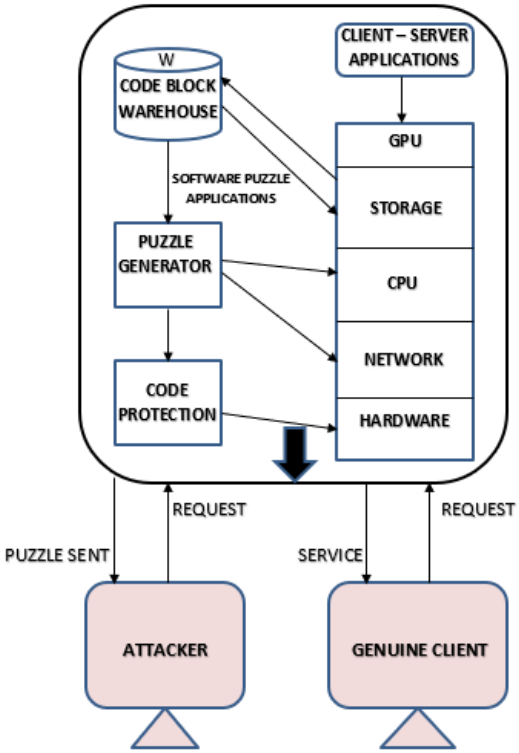

Figure 2: Concurrent Software puzzle architecture embedded within the client-server.

All these drawbacks and their considerations has led this research paper to propose a dedicated software puzzle framework with a centralized database communication network which enables to adopt and enhance the existing work along with certain analytical approaches, to produce a better architecture in terms of efficiency and performance of the system.

3 Research Methodology

3.1 Objective

A detailed analysis on the performance and resource utilization of the server with and without software puzzle running in it, is obtained by measuring the execution time, memory consumed, CPU usage and the network bandwidth utilization individually. This shows us the necessity to allocate a separate server for the software puzzle. The objective here is to allow the two servers to dedicate their resources and time only to perform their job and not get deviated by other services. It also motivates the two servers to communicate using a wireless communication network through a centralized database, to improve the network efficiency of the system and therefore the entire data centre.

3.2 Dedicated Software Puzzle Architecture

The current architecture consists of a concurrent server within which the software puzzle framework is imbibed. Whereas the proposed architecture signifies the separation of the software puzzle framework into a dedicated server placed away from the actual client servers within the same data centre or in different data centres.

The proposed architecture consists of 4 major components:

1. Client Systems – Clients are the users of cloud services who use the various services of the cloud providers by requesting for it through internet. They can be genuine clients or even attackers who flood the servers by sending high volumes of malicious requests. The genuine clients who send requests with a required motive, get served by the servers. But the suspicious clients who create unnecessary malicious network traffic, are served only after being verified by the software puzzle. These types of clients will have communications with both the client-servers and the software puzzle servers.

2. General Client-Server – These are the usual servers executing the requests of the clients and granting them their required services. They are made of various resources and computational components like storage units, hardware, CPU, GPU and the network components. This paper introduces a wireless communication algorithm through a centralized database, for the transmission of data between the normal client-server and the dedicated software puzzle server.

3. Centralized Database – When the software puzzle is separated from the normal client-server, there evolves a need for the communication between these two servers. This communication is mediated through a centralized database placed within the same data centre or away from the existing one. The normal client-server feeds the database with the client credentials which would be used by the software puzzle server for its puzzle execution. Similarly, the puzzle server stores the puzzle result inference, which the client-server uses to serve the clients.

4. Dedicated software puzzle server – This is a very simple server with low memory resources needed to accommodate a code block warehouse and higher computational resources required to dynamically generate a client puzzle, encrypt them and validate the results sent by the client. Both the software puzzle server and the normal client server work with a wireless communication network which is used to communicate between the two servers and with the clients who are remotely placed.

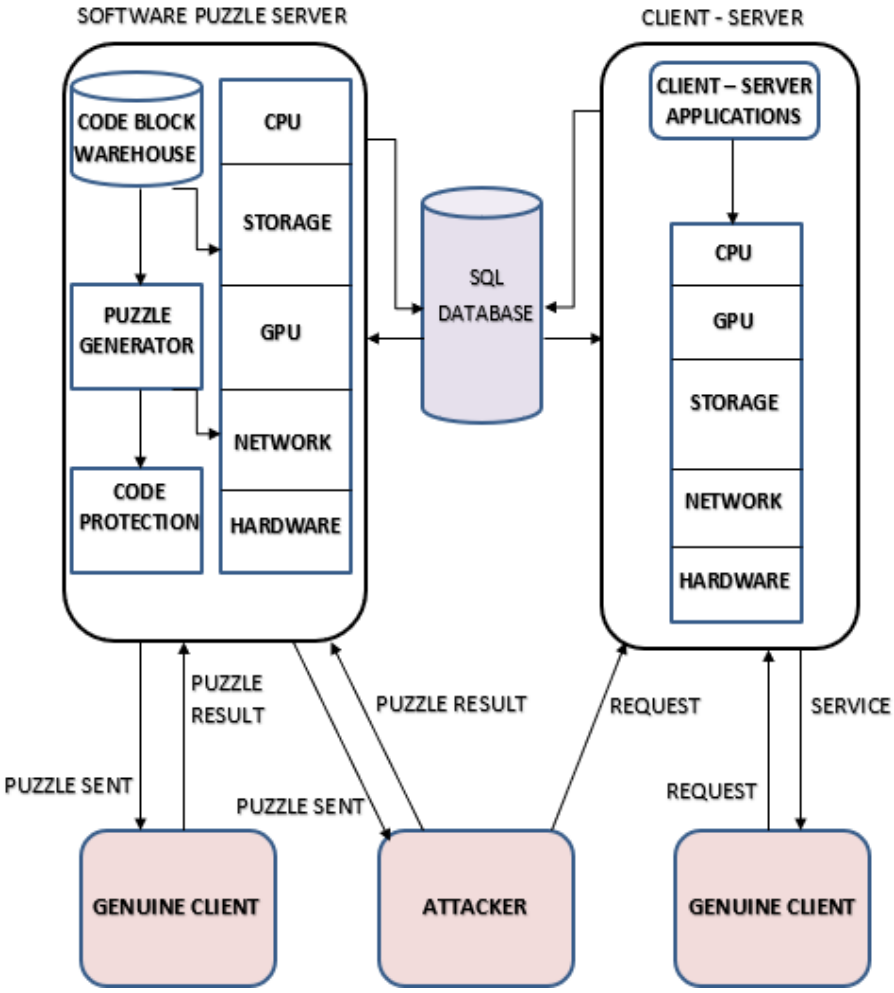

Figure 3: Dedicated software puzzle architecture with a centralized SQL database.

According to Fig-3, initially the genuine clients send requests to the server. These requests would be served by the client-server without any hassle. In a scenario where the malicious clients generate a suspicious network traffic with garbage requests, the client-server stores the packet header credentials of the malicious clients in the centralized database. The database is centrally placed between the two servers so that when the client or an attacker sends the client-server with large amount of request packets that crosses a specified number, the client-server stores those client credentials into the database which is received by the software puzzle for its execution. The software puzzle server executes only when malicious high-volume requests are received and not all the time. When it does, it occupies considerable amount of resources, temporarily distracting the execution of client-server framework.

Fig-4 describes the role of the centralized database in the proposed architecture. The software puzzle conducts a periodic check into the centralized database to see if there are any client credentials updated by the client-server into the database. If the database is updated, then the software puzzle takes those credentials and use this information to compute the puzzle operations. Similarly, the puzzle server generates a random puzzle and sends it to the client. If the client is genuine, he would solve the puzzle and send the result back to the puzzle server for verification. If it is an attacker, then he would either randomly send a wrong puzzle result or drop the puzzle packets, as it would cost him more computational resources to genuinely solve the puzzle than the computational resources of the server in generating and verifying the puzzle. The puzzle server after carrying out its process and obtaining the puzzle

result, updates the database with the puzzle result inference, indicating whether the client is genuine or not and if the client-server must serve that particular client or drop its packets. The client-server checks the database for the puzzle result inference updated by the software puzzle and uses the data to serve the clients.

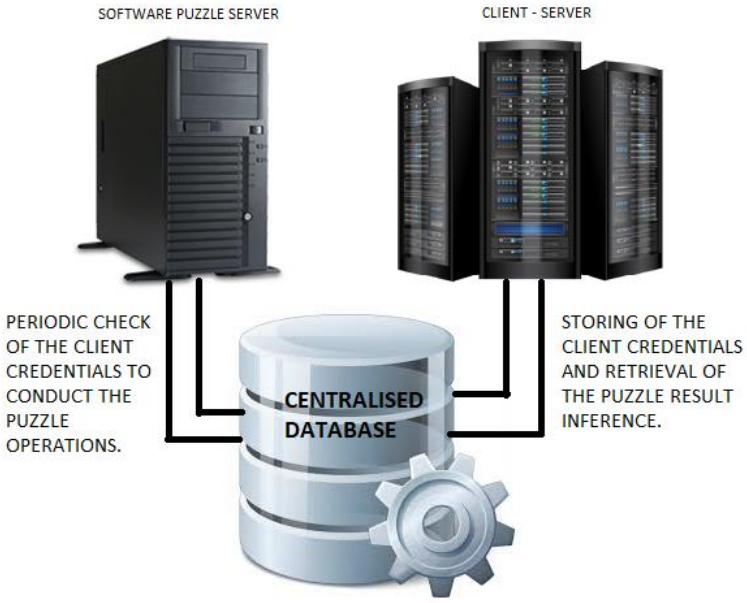

Figure 4: Communication between the two servers through a centralized database in a proposed dedicated architecture.

When the software puzzle application would be performing its executions with the suspicious clients, the client-server would be made to stay idle without serving the genuine clients as the resources are being made use by the software puzzle application. But the software puzzle does not utilize large amount of time and resources as compared to the client-servers and also this scenario does not occur frequently. Therefore, this condition only constitutes to a very small part in supporting the need for proposed architecture. But, when genuine requests are being served by the client-server application, the software puzzle becomes idle as it would be waiting for the resources that is being used by the client-server application. This condition constitutes to a major downfall in the performance and time efficiency of the entire system in overcoming DDoS attacks and therefore it strongly supports the need for dedicated servers and encourages the proposed architecture which clearly signifies the separation of the software puzzle framework from the client servers, duly by allocating required amount of dedicated resources for both the client-server and the software puzzle.
This way, the dedicated architecture becomes highly resource efficient than that of the concurrent architecture.

4 Design Specification

4.1 Amazon Web Services EC2 Instance

Amazon EC2 is a simple web service interface which enable us to develop applications and run various software in a secure and a scalable compute capacity platform. It allows us to

choose the necessary amount of instance storage, memory, CPU and a boot partition size which can be retained and restarted remotely using web service APIs (Amazon EC2; 2018). In this paper, for my experiments, I have used an AWS EC2 instance with type t2.micro server which is generated with a Private IP address 172.31.28.213, one virtual CPU and 1 GB total memory. It uses the eu-west-1a availability zone.

4.2 Microsoft Azure Cloud Service

Microsoft Azure is another cloud service provider which provides its customers with a varied number of services. Among them, I have used Azure Database for MySQL by name 'softwarepuzzle' in my experiments with 4 virtual cores and 100GB memory. It is a fully managed relational database service with an open source MySQL Server Engine, having the capacity to handle cross-platform mission-critical workloads and ensure predictable performance (Microsoft Azure; 2018).

4.3 Implementation Functionalities

The implementation design specification consists of 5 major functions written in Java and executed on NetBeans IDE – 8.2. This is described in Fig-5 as follows:

a) PuzzleServer function - This function is internally executed when the server function is called during the integrated or the concurrent server implementation. But, this function in the integrated server, gets suspended when the standalone PuzzleServer function is executed. The puzzle function realizes 3 major functionalities such as Getoperand, Getoperator and EvaluateExpression. Both the Server function and the PuzzzleServer function can invoke these functionalities any number of times to randomly generate and evaluate puzzles.

b) Server function – This function contains the datagram sockets functionalities to connect to the PuzzleServer and the client. These connections among the two servers, centralized database and the client take place purely through the UDP socket communications in both standalone and integrated configurations. Both the PuzzleServer function and the Server function share a common SoftwarePuzzle Library which stores the puzzle related functionalities such as code block warehouse.

c) Client function – The client function executes to send login requests to the server. The puzzle sent by the puzzle function is handled and solved by the 'handlechallenge' and 'handlerequest' functionalities of the client.

d) Database server – The centralized database mediating the two servers is in the form of an SQL database obtained from the MySQL workbench and SQLyog database environments. The MySQL database server is generated in the Microsoft Azure cloud platform and linked to the client-server generated in AWS cloud platform.

e) JFreeChart function – This is a java library function used to plot line and bar graphs using the data obtained from the experiments conducted. It realizes 2 major functionalities; 'category.BarRenderer' renders a bar graph using the data from the file Perfg4.txt to obtain the average time efficiency of the two processes and 'xy.XYLineAndShapeRenderer' produces line graphs using the data obtained from

the files Perfg1.txt, Perfg2.txt and Perfg3.txt to obtain the memory, network and CPU utilization respectively.

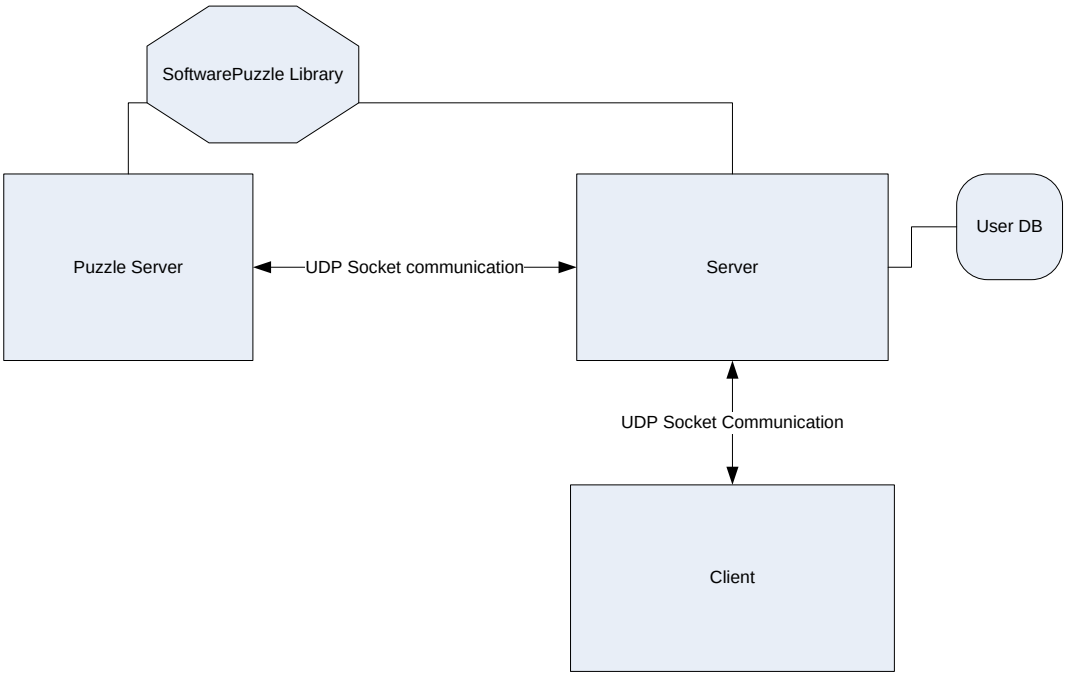

Figure 5: Implementation Design Architecture.

5 Implementation

As part of an implementation, a normal client-server with required amount of memory and other resources is created on an Amazon Web Server. For the experimental purpose, the software puzzle application and other client-server applications are coded in Java and made to run on NetBeans IDE. A MySQL database is created on Microsoft Azure and is linked to the servers placed on the AWS platform.

5.1 Concurrent Software Puzzle Execution

Initially, the software puzzle application will be present within the client-server by sharing the same resources as that of the client-server. In this scenario, normal low volume requests as well as the malicious garbage requests such as log-in requests are sent to the client-server. The performance and time parameters of this whole process for different number of login requests are stored in separate files for memory consumption, CPU usage, average time and network bandwidth.

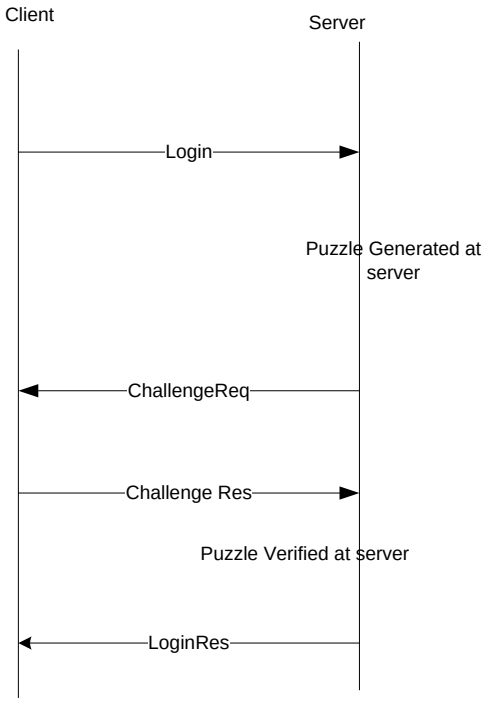

Figure 6: Sequence diagram of Integrated server computations with software puzzle embedded within the client-server.

5.2 Dedicated software Puzzle execution

Later, as per the proposed architecture, the software puzzle is separated from the normal client-server into its own individual server by allocating required amount of resources to it. Now there exists two servers which communicate using UDP. In this scenario, both genuine low volume requests as well as a high-volume garbage requests are generated and sent to the client-server as login-requests. This client-server serves the genuine low volume requests and sends the credentials of the malicious client (such as its source address, destination address and other information) to the software puzzle server, for it to check whether the client is genuine or not. For the transmission of credentials between the two servers, a communication network is developed through a centralized SQL database. In the experiments, Mysql database is used with SQLyog as the front-end platform.

While the client-server performs its usual functions, the software puzzle makes use of its newly allocated resources to generate the puzzle and sends it to the client. The client's puzzle result is checked for its authenticity and correctness. Depending on this, the software puzzle server communicates back to the normal client-server intimating it to either serve the client or deny its request. The communication takes place by storing and retrieving the essential data by both the client-server and the software puzzle server. The performance and time parameters of this whole process of the second scenario are stored in the same respective memory, CPU, average time and network bandwidth files as that of the integrated server.

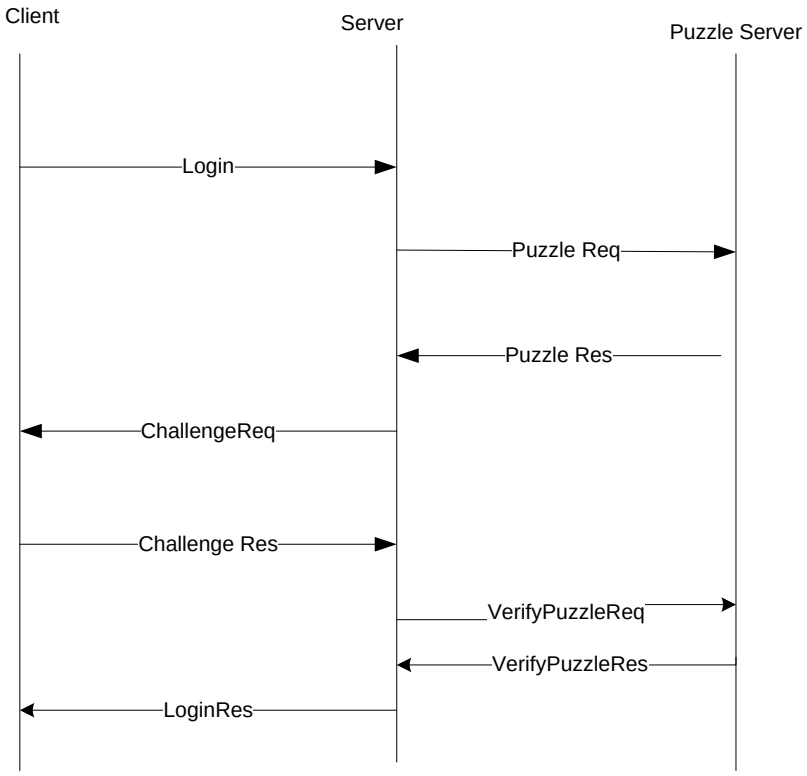

Figure 7: Sequence diagram of Standalone server computations with software puzzle server separated from the client-server.

Finally, the results of the two scenarios are analyzed and compared with each other. Using these parameters, four graphs are plotted each comparing the memory usage, CPU time, network bandwidth and the average time efficiencies for integrated server against dedicated server. This shows that the concurrent architecture takes more time and executes with poor resource management than that of the proposed architecture with dedicated servers allocated with individual resources.

6 Evaluation

An experiment was conducted just the way as mentioned in the implementation section by executing the code nearly 25 times for each 1, 500, 1000 and 1500 login requests. For each of these 25 executions of the varied input login requests, the output of the average time, memory, network and CPU time were almost equal to each other, with minute differences. An average result among the 25 executions is shown below in terms of a graphical representation.

6.1 Experiment 1 – Memory Utilization

The chosen AWS EC2 instance has 1 GB of total memory. Complete memory of 1 GB is made available for the client-server, while executing the integrated software puzzle code. But when the puzzle server is made to run separately, both the client-server and the puzzle server used only the dedicated amount of 0.5 GB memory which is pre-allocated to it. By conducting an experiment with these memory allocations, the below graph was obtained.

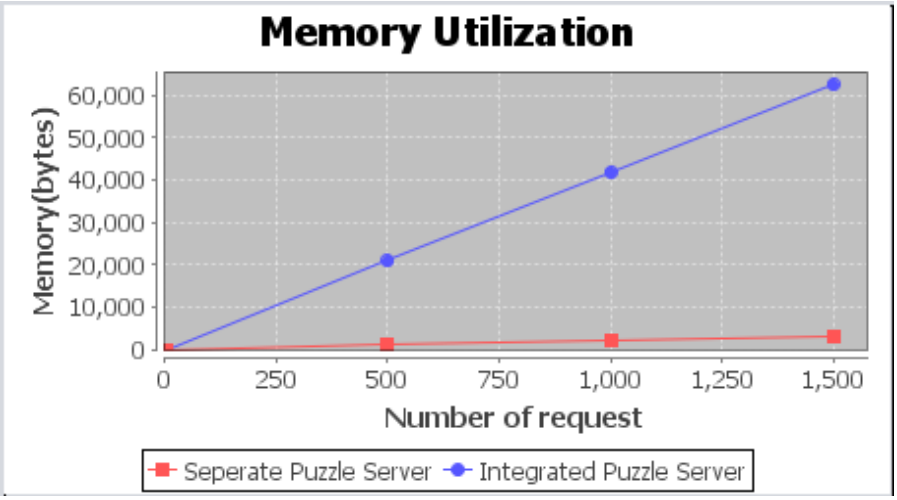

Figure 8: Memory utilization comparison of the two architectures.

The above graph clearly shows that the memory consumed in bytes by the integrated puzzle server is much higher than that of the separate software puzzle server. For example, with an input of 1500 login requests, the integrated server uses approximately 65000 bytes of memory, which is consumed by both the software puzzle operations as well as the normal client-server operations. In this case, the amount of memory which is being used by the normal client-server might had to be used by the software puzzle on high priority or the client-server with high priority might have to wait until the memory is made free. But the software puzzle server takes approximately 3000 bytes from its locally allocated memory of 0.5GB by fetching the redundant data from the same allocated memory.

6.2 Experiment 2 – CPU Usage

Fig-9 describes the amount of time slots that a CPU consumes just to carry out the software puzzle function (puzzle generation and puzzle verification) in both the integrated architecture as well as the dedicated architecture. Unlike the elapsed time, this CPU usage gives only the time taken by the software puzzle and its interaction with the database as in case of the dedicated architecture.
Therefore, the CPU usage for every value of the input is greater than that of the integrated architecture. But when it is compared with the response time which is the total elapsed time (shown in experiment 4) that includes the wait time and other client-server interruptions, it shows that even with a greater CPU time of the dedicated architecture, it stands efficient in terms of proper usage of its resources (Zhanikeev 2015).

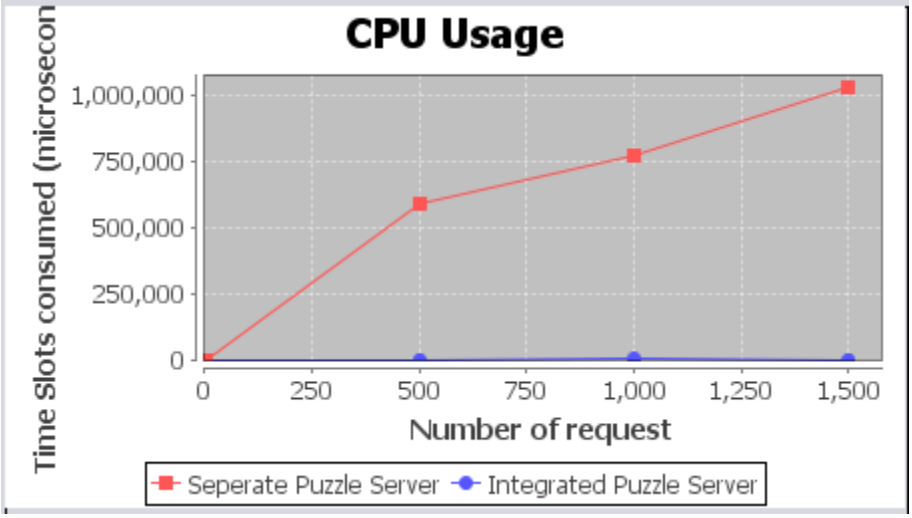

Figure 9: CPU usage time comparison of the two architectures.

6.3 Experiment 3 – Network Utilization

The network utilization is given by the total consumption of network bandwidth which includes packets transferred between the client and the server and between the two servers (integrated and dedicated) through the centralized database. From Fig-10, we can deduce that the network consumption of the dedicated server is higher than that of the integrated server for every input of the login requests. This result was well expected as the dedicated architecture defines an extra communication with the client-server after being separated from it. This exchange of packets between the two servers through a centralized database, constitutes to the higher network consumption by the proposed architecture.
This drawback in the dedicated architecture can be rectified by using the visible light communication instead of the usual radio waves communication which is currently being used. A much-detailed explanation of this technology is included in the 'Future Works' section of this paper.

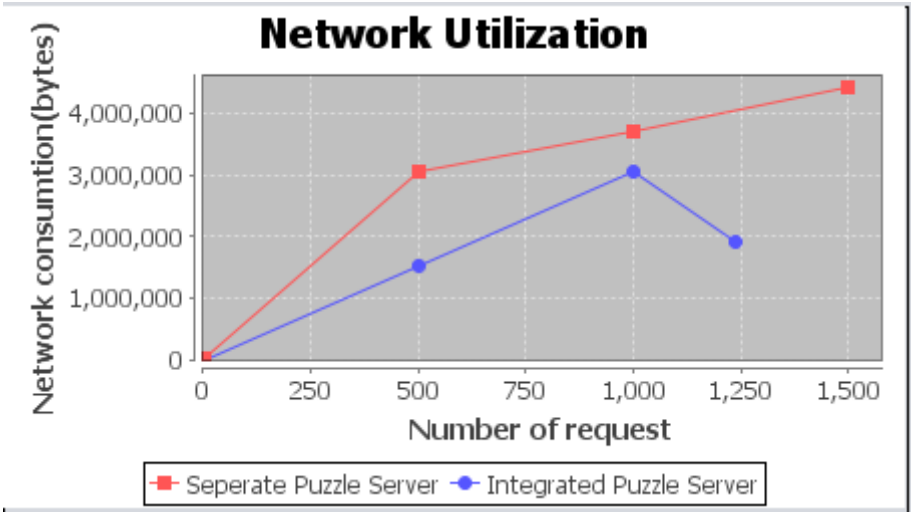

Figure 10: Network utilization comparison of the two architectures.

6.4 Experiment 4 – Total Response Time

The total response time is the elapsed time of the complete experimental process of each of the architectures (integrated and dedicated). This includes the time taken by the puzzle operations, client-server operations and other miscellaneous wait time. From Fig-11, it can be deduced that for 500 login requests, the response time of the integrated architecture (4000 ms) is higher than the response time of the dedicated architecture (approx..1500 ms). This is mainly due to the interruptions and various preemptions caused by the other processes and also due to the priority handling of the resources by either the software puzzle or the client-server operations while the other keeps waiting for the resources to be released in the current integrated architecture as both the software puzzle and client-server operations are embedded within the same server, using the same resources. In case of the dedicated architecture, software puzzle is allotted required amount of its own resources and the client-server is allotted its own resources, executing themselves without interfering with each other.

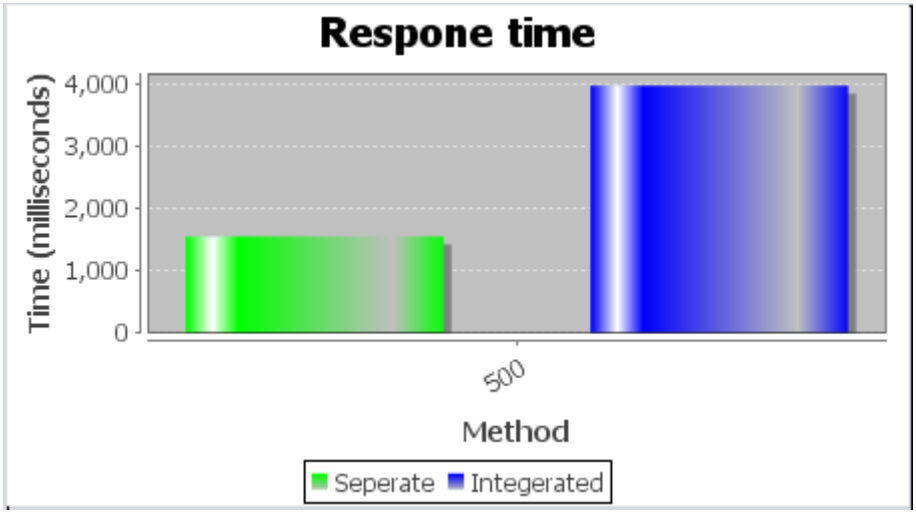

Figure 11: Total Response Time between the two architectures.

6.5 Discussion

With the development of a new architecture to the software puzzle framework, we have improved its time and resource utilization to a greater extent. The memory utilization of the dedicated architecture has drastically reduced as the puzzle software uses its entire memory to store only the code block warehouse and the non-redundant puzzle data. In the experiments conducted, we haven't used a dedicated CPU for each of the architecture. But we have certainly shown that the CPU efficiency of the dedicated architecture is greater than that of the integrated one, by comparing the difference between the CPU time of the software puzzle operation and the total response time of the two architectures. Even though the CPU time for the proposed architecture exceeds, the response time decreases. This proves that the raise in the integrated architecture response time is duly by the client-server preemptive processes and other miscellaneous wait time. The network bandwidth utilization has showed a slight drawback to the proposed architecture which is caused by the introduction of the network communication between the two separated servers, through the centralized database. But we have suggested a new technology of visible light communication to replace the existing radio frequency communication between the two servers, as part of the future work. With all these

experimental developments observed in the dedicated software puzzle architecture, we are able to minimize the misuse of time and resources and efficiently combat the resource-inflated DDoS attacks.

7 Conclusion and Future Work

Software puzzle portrays itself to be one of the current best countermeasures to resource-inflated Distributed Denial of Service attacks. But after enough research on the effects of DoS attacks, its initial countermeasure of client puzzles and the development of software puzzle, it was shown that it holds its own set of disadvantages within the client-server in which it is embedded in, resulting in an inefficient use of resources between the client-server and puzzle processes. This gave us the motivation to come up with a proposal emphasizing the separation of software puzzle from the client-server and allocating dedicated resources to it, thereby enhancing an efficient functioning of both the client-server and the software puzzle. As part of the implementation, various experiments conducted strived to exhibit the performance and time consumption of both architectures. Our approach supported the proposed architecture to a major extent, leaving behind some corrections and suggestions that can be extensively carried out in future for the betterment of the proposed architecture. The inclusion of a communication network between two servers in the dedicated framework, has led to the increase in the network bandwidth utilization of this new server.

As part of the future work, introducing the visible light communication for the packet transfer between the two servers enables us to overcome the network drawback. This technology which uses 300 THz of unused visible light spectrum for its wireless communication, can be implemented only to small-scale data centres, where the two servers (dedicated and integrated) are placed within the same room, under one single roof without any opaque obstruction between them. According to Wang et al. (2015), the wireless data throughput is achieved utilizing the simple visible light spectrum by several small Light-Fidelity attocells that acts as a transmission medium for the data and at the same time provide illumination to the place where these servers are placed. Replacing the radio frequency networks with these Li-Fi networks, can increase the speed of data transmission by 10 folds with a larger network bandwidth and also there will be a minimal investment on the network set-up, as the major required resource such as light medium is already available within the data centres where these servers are placed.
With all these above-mentioned parametric considerations and developments on the new technology implementations, we can conclude that the proposed dedicated software puzzle framework is certainly a better architecture than the existing integrated framework.

References

Agha, G., Frolund, S., Kim, W., Panwar, R., Patterson, A. and Sturman, D., (1993). Abstraction and modularity mechanisms for concurrent computing. *IEEE Parallel & Distributed Technology: Systems & Applications*, *1*(2), pp.3-14.

Amazon EC2 (2018) *Secure and resizable compute capacity in the cloud. Launch applications when needed without upfront commitments* [Online] Available at: https://aws.amazon.com/ec2/ [Accessed 28 November 2018].

Aura, T., Nikander, P. and Leiwo, J., (2000), April. DOS-resistant authentication with client puzzles. In *International workshop on security protocols* (pp. 170-177). Springer, Berlin, Heidelberg.

Bicakci, K. and Tavli, B., (2009). Denial-of-Service attacks and countermeasures in IEEE 802.11 wireless networks. *Computer Standards & Interfaces*, *31*(5), pp.931-941.

Buyya, R., Calheiros, R.N. and Li, X., (2012), November. Autonomic cloud computing: Open challenges and architectural elements. In *Emerging Applications of Information Technology (EAIT), 2012 Third International Conference on* (pp. 3-10). IEEE.

Chonka, A. and Abawajy, J., (2012), September. Detecting and mitigating HX-DoS attacks against cloud web services. In *Network-Based Information Systems (NBiS), 2012 15th International Conference on* (pp. 429-434). IEEE.

Fraser, N.A., Kelly, D.J., Raines, R.A., Baldwin, R.O. and Mullins, B.E., (2007), June. Using client puzzles to mitigate distributed denial of service attacks in the tor anonymous routing environment. In *Communications, 2007. ICC'07. IEEE International Conference on* (pp. 1197-1202). IEEE.

Jadeja, Y. and Modi, K., (2012), March. Cloud computing-concepts, architecture and challenges. In *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on* (pp. 877-880). IEEE.

Malet, B. and Pietzuch, P., (2010), November. Resource allocation across multiple cloud data centres. In *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science* (p. 5). ACM.

Microsoft Azure (2018) *Azure Database for MySQL Documentation* [Online] Available at: https://docs.microsoft.com/en-us/azure/mysql/ [Accessed 2 December 2018].

Shea, R. and Liu, J., (2013). Performance of virtual machines under networked denial of service attacks: Experiments and analysis. *IEEE systems journal*, *7*(2), pp.335-345.

Sheinidashtegol, P. and Galloway, M., (2017), April. Performance Impact of DDoS Attacks on Three Virtual Machine Hypervisors. In *Cloud Engineering (IC2E), 2017 IEEE International Conference on* (pp. 204-214). IEEE.

Sorte, P.R., Chougule, S. and Lingam, C., (2016). NOVEL SOFTWARE PUZZLE SCHEME TO DE-AMPLIFY RESOURCE INFALTED DDOS ATTACKS.

Walunj, V. and Pawar, V., (2017). Software Puzzle Approach: A Measure to Resource-Inflated Denial-of-Service Attack. *International Journal of Computer Applications*, *161*(10).

Wang, X. and Reiter, M.K., 2003, May. Defending against denial-of-service attacks with puzzle auctions. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on* (pp. 78-92). IEEE.

Wang, Y. and Haas, H., (2015). Dynamic load balancing with handover in hybrid Li-Fi and Wi-Fi networks. *Journal of Lightwave Technology*, *33*(22), pp.4671-4682.

Waters, B., Juels, A., Halderman, J.A. and Felten, E.W., (2004), October. New client puzzle outsourcing techniques for DoS resistance. In *Proceedings of the 11th ACM conference on Computer and communications security* (pp. 246-256). ACM.

Wu, Y., Zhao, Z., Bao, F. and Deng, R.H., (2015). Software puzzle: A countermeasure to resource-inflated denial-of-service attacks. *IEEE Transactions on Information forensics and security*, *10*(1), pp.168-177.

Xie, Y. and Yu, S.Z., (2009). Monitoring the application-layer DDoS attacks for popular websites. *IEEE/ACM Transactions on Networking (TON)*, *17*(1), pp.15-25.

Yan, Q. and Yu, F.R., (2015). Distributed denial of service attacks in software-defined networking with cloud computing. *IEEE Communications Magazine*, *53*(4), pp.52-59.

Zhanikeev, M., (2015). A holistic community-based architecture for measuring end-to-end QoS at data centres. *International Journal of Computational Science and Engineering*, *10*(3), pp.315-324.