Wireless Networks (2021) 27:5491–5501
https://doi.org/10.1007/s11276-019-01937-4

An OpenStack based cloud testbed framework for evaluating HTTP flooding attacks

A. Dhanapal[1] · P. Nithyanandam[1]

Published online: 12 January 2019

Abstract

The cloud computing has inherent challenges to detect the *Hyper Text Transfer Protocol* (HTTP) flooding *Distributed Denial of Service* (DDoS) attack due to its natural characteristics like virtualization, elasticity and multi-tenancy. The usage of cloud computing is user-friendly, but the implementation of the cloud infrastructure such as compute node, networking, cloud storage is very complex in order to achieve its various characteristics. Similarly, detecting the HTTP flooding attack in the cloud is also very complex as it requires an understanding of various potential attack paths in such a complex environment. So, designing the cloud testbed framework to detect the HTTP flooding attacks is a challenging problem to be solved. The cloud testbed framework has to consider several aspects of attack scenarios while accounting the cloud characteristics. This paper reviews the existing DDoS attack detection framework and their gaps and proposes a cloud testbed framework for evaluating the HTTP flooding DDoS attack solution. The proposed framework is implemented using the OpenStack cloud environment. The *Fédération Internationale de Football Association* (FIFA) World Cup 1998 real-time dataset is used to generate the HTTP flooding attack to the OpenStack cloud testbed framework for the experimentation.

Keywords DDoS · Cloud testbed framework · HTTP flooding · OpenStack · Cloud computing security · Layer 7 attacks

1 Introduction

Cloud computing supports startups, small and medium level enterprises to subside their initial investment cost of the infrastructure and helps to make use of those investments to their core business purposes [1]. *National Institute of Standards and Technology* (NIST) defines cloud computing [2] is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources such as networks, servers, storage, applications, and services that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes the availability and exhibits the following five characteristics: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service [2].

✉ A. Dhanapal
Dhanapal.a2013@vit.ac.in

P. Nithyanandam
Nithyanandam.P@vit.ac.in

[1] School of Computing Science and Engineering, VIT University, Chennai, India

1.1 Threats in the cloud computing

Cloud computing is getting matured every day. The threats applicable to the computing world like *Distributed Denial of Service* (DDoS), Data Security, Data Integrity, etc., are also applicable to the cloud computing [3, 4] as well. The organizations have a big concern on safety and security aspects of the cloud computing. The enterprises are running their online business and/or business critical applications like financial services, banking services, etc., are slow in the adoption of the cloud due to these security issues. The DDoS is one of the major security concern in the cloud environment.

1.2 DDoS attack types

The DDoS attacks broadly classified [5–7] into three types:

- *Volumetric Attacks*—The aim of this attack is to saturate the bandwidth. Examples: *User Datagram*

Protocol (UDP) floods, *Internet Control Message Protocol* (ICMP) floods, etc.
- *Protocol Attacks*—This type of attack targets actual server resources. Examples: Ping of Death, SYN floods, etc.
- *Application Layer Attacks*—The objective of this attack is to bring down the application services so that legitimate user cannot be able to access the application. Examples: HTTP flooding attack, *Extensible Markup Language* (XML) attack, etc.

The Competition, revenge, political reasons, proving the ability, etc., are [8, 9] some of the reasons for the various DDoS attack.

1.3 The significance of detecting the HTTP flooding DDoS attacks in the cloud

The HTTP flooding attack detection is one of the challenging attacks to detect in the cloud and it gained greater attention with the research community.

The CNN report on 21st Oct 2016 [10], says that the number of websites such as Twitter, Netflix, Github, etc., is affected by flooding attacks. The CNN report [10] adds that one of the leading public cloud service *Amazon Web Services* (AWS) also experienced the issue.

The website Tripwire [11] captures the top 5 most significant application level DDoS attacks of the year 2016. The list is as follows:

- Attacks on services provided by Dyn Inc.
- Attacks on blogs of American journalist and investigative reporter Brain Krebs.
- Attacks on Hillary Clinton and Donald Trump campaign sites.
- Attacks on Rio Olympics websites.
- The attack on Russian Banks such as Sberbank and Alfabank.

The article “Application attacks against clouds up 45%” [12] states that 45% increase in application layer attacks in the cloud as per cloud security firm Alert Logic, Inc. This report is prepared based on the security incidents over 3000 enterprise customers. The article “Denial-of-Service Attacks Meet the Cloud” [13] covers the experts view on the DDoS attack over the cloud. This emphasis on the importance to detect the HTTP flooding attacks on the cloud environment.

The following characteristics [2, 14–16] create special needs and complexities associated with the HTTP flooding detection in the cloud environment.

Multi-tenancy Multiple customers may co-locate in the same cloud environment. If anyone of the customers affected by the HTTP flooding attack creates a greater impact on the rest of the clients as well as the cloud environment itself.

Elasticity The cloud computing gives the advantages of growing and shrinking of the cloud resource dynamically. If the HTTP flooding attack not detected, the cloud platform tends to add more resources to the customer who is under attack and eventually reaches resources starvation. This will be a disaster to the cloud service provider and all cloud customers.

Virtualization Virtualization is the core concept of cloud computing. This helps cloud computing to achieve its characteristics like on-demand services, multiple tenants under the same infrastructure, etc. There shall be multiple virtual instances of the same customer is running across the different geographical locations of the cloud service provider. So, there should also exist corresponding virtual routers at the multiple locations to route the traffics to the respective virtual instances of the customers. This requires a special kind of HTTP flooding DDoS detection solution integrated within the cloud provider environment at the virtual routers and virtual instance level rather than typical intrusion detection system lies at the gateway of the network.

Computing model The computing nodes abstraction over virtualization and responsible for running virtual machines of the customers. Each computes node can have multiple virtual machines. These compute nodes also spread across various locations of the cloud service provider. Each compute node has the capability to route the traffic within the compute node itself, across the compute nodes and to the external world. The HTTP flooding DDoS attack has to be detected at the each computes node level. This mandates special kind of DDoS detection approach integrated into the cloud environment.

The above points discussed underscores the importance of detecting the application layer HTTP flooding attacks in the cloud environment. While designing the cloud framework, it has to include all those characteristics mentioned above. Also, the cloud testbed should cover all possible scenarios. So, designing the cloud testbed for evaluating the HTTP flooding attack solution is very interesting and challenging problem to be solved.

The research communities are working actively on providing solutions for detecting and mitigating HTTP flooding attacks in the cloud. The solution for HTTP flooding attack cannot be verified in the real-time, as it brings down the victim services. So, usually, it is verified in the controlled lab environment. The various solutions proposed so far has significant gaps in covering the different aspects of the cloud environment. This paper addresses the challenges of designing the cloud framework which covers all possible options, different cloud characteristics and scenarios. The proposed work uses the

OpenStack cloud environment which provides *Infrastructure as a Service model* (IaaS). The realistic HTTP flooding generated using the FIFA World Cup 1998 datasets [17, 18].

The rest of the paper organised as follows: Sect. 2 covers the existing work and their critical review. Section 3 contains various options available for testbed framework design based on the cloud service model. Section 4 captures various choices available for modelling the HTTP flooding attacks. Section 5 explains the architecture model of the proposed framework. Section 6 gives the OpenStack cloud environment implementation details, and Sect. 7 discusses the comparison of the proposed works with other works. Section 8 narrates the experimental results. Subsequently, Sect. 9 carries the conclusion and future enhancement of the work.

2 Related works

The DDoS attack has been evolving over the period of time. The enormous research is going on, the solutions being proposed to detect and mitigate DDoS attacks. The recent hot topic in DDoS is the application layer HTTP flooding attacks in the cloud environment. There are different solutions proposed in this space. Each solution has its own limitation due to lack of the standard evaluation of the cloud tested framework. We have reviewed various works related to DDoS detection and mitigation. This section restricts the discussion to most relevant papers, their framework, the HTTP flooding attack generation model and their critical reviews.

Smirnov et al. [19], used the OpenStack environment to verify their solutions. This work used the tools to generate HTTP flooding attack to validate the solution. This has a limitation on capturing the OpenStack cloud framework details and parameters like a multi-tenancy model, etc.

Kobayashi et al. [20], classified the *Denial of Service* (DoS) attack into multiple types like the method of attack, service quality impact, attack scenario, etc. This work provides the solution to HTTP flooding attacks. The framework of this paper is to introduce intermediate server named as control machine between the actual application server and the outside world. This control machine has the intelligence of detecting the HTTP flooding and if there is any flooding attack happens then such attacks requests are forwarded to the decoy machines and legitimate requests are served by application servers. This model has the limitation of capturing cloud parameters like multi-tenancy, computing model, etc.

Shruthi et al. [21], proposed a solution for safeguarding application in the cloud computing. This paper provides a very generic framework. The work lacks in adding the details elasticity, the real-time dataset for cloud framework.

Shahanaz Begum et al. [22], discussed the DDoS detection and prevention in the private cloud. The experimental setup consists of three virtual machines. Two virtual machines act as a bot and a web server running on one virtual machine. There is a significant gap in addressing cloud characteristics like elasticity, computing model, etc.

Kiruthika Devi et al. [23], analyzed the various DDoS solutions proposed and their pitfalls. This paper captured the DDoS detection mechanism but lacks in adding details on how the cloud framework used in defining the solution. This paper focused only on virtualization from the cloud perspective.

Tarun Karnwal et al. [24], talked about vulnerabilities in each service model, proposed the defense architecture model against XML and HTTP flooding attacks. It failed to explain the deployment of defense mechanism. The framework missed capturing the details of the cloud environment. These details are essential, as the solution evaluated in the cloud environment.

Osanaiye et al. [25], proposed the DDoS detection solution in the cloud environment using packet inter-arrival time. This work explained the framework for the HTTP flooding detection but has a gap in defining the overall cloud framework where the solution fits.

The discussion and critical review of the various work done so far, still have the gap in defining an effective cloud testbed framework to evaluate the HTTP flooding attack.

3 Potential design considerations for cloud testbed framework

The authors evaluated the various possible options to design the cloud testbed framework and the HTTP flooding attack generation. The aim of this work is to design an effective testbed framework for evaluating the HTTP flooding attack in a cloud environment and the corresponding HTTP flooding DDoS attack generation. Each of the cloud service models considered while designing the testbed.

- *Software as a Service* (*SaaS*) model has the option of providing application, and this application can act as target service. This may not be suited for the designing framework as the cloud infrastructure entirely hidden behind by the cloud service provider. Designing the testbed using this model may look similar to regular web application in attack scenario and dilutes the cloud environment.
- *Platform as a Service* (*PaaS*) option looks better when compared to SaaS model as it provides a platform for

developing applications as well as the development environment. Still, there is some constraint exists like very limited control to the user. The cloud monitoring and administration are under control of the cloud service provider.

- *Infrastructure as a Service* (*IaaS*) is the best suitable option while comparing other two models. The IaaS provides complete control of monitoring, administrating and maintaining of the cloud environment. It gives complete control for designing and fine tuning the testbed framework and fulfils the requirement perfectly. Most importantly, there is an open source software OpenStack is available at the free of cost. This OpenStack is popular among the cloud community.

Due to the reasons mentioned above, the authors selected the OpenStack Infrastructure as a Service model for designing the testbed framework for this research work.

4 The HTTP flooding DDoS attack modelling alternatives

There are various methods to generate the HTTP flooding DDoS Attack. It can be like generating flooding using the performance measurement tools, the HTTP DDoS attack generation tools, proprietary in-house developed tools or using available real-time datasets. The authors analyzed all the possible options for generating the HTTP flooding attack.

Table 1 captures some of the important lists of performance testing tools available. The tools can be used to generate the HTTP flooding attack. Generally, the webserver performance tested with the help of tools mentioned in Table 1.

Table 2 covers the details on the various DDoS attack generation tools available for research work. This table organized with details such as tool name, online reference, availability and usability of *Graphical User Interface* (GUI) based or *Command Line Interface* (CLI) based.

Next, the possible option is to generate the HTTP flooding attack from the real-time datasets available on the internet. Table 3 depicts the details like name of the dataset, an online link to the dataset, and the number of requests and time during which those datasets collected.

The user has multiple options to choose different datasets based on the requirement. The number of HTTP requests varies from 28 thousand to 1.3 billion logs. The collected HTTP logs stored in a processed format in order to hide the confidential details. There should be some mechanism required to formulate an HTTP request from the dataset.

The various tools depicted in Table 1 are generally used to measure the performance of the webserver. Usually, there are different parameters evaluated for the performance measurements.

- The response time of the user.
- The response time of the server.
- CPU usage.
- The validity of the returned data.
- HTTP errors.
- Maximum open connections with the webserver, etc.

The performance tools mentioned in Table 1 help to generate the HTTP flooding attack to some extent, but may not be able to simulate the real-time attacks.

The various tools depicted in Table 2 are open source based and available on the internet for the purposes of simulating DDoS attacks. Though it helps to simulate HTTP flooding better when compared to tools in Table 1, still there are some constraints associated such as generating the HTTP flooding with some of the routine parameters like using the same set of repeated user agents, the continuous request to the same URL or invalid URL, etc. The real-time scenario HTTP flooding attack characteristics vary dynamically.

Table 1 The webserver performance testing tools

Tool name	Online link	Availability	Ease of use	Replay option
Siege	https://www.joedog.org/siege-home/	Open source	Yes	Limited
Apache bench	http://httpd.apache.org/docs/current/programs/ab.html	Open source	Yes	Limited
Locust.io	http://locust.io/	Open source	Complex	Liberal
Http-perf	https://www.npmjs.com/package/http-perf	Open source	Yes	Limited
Apache JMeter	http://jmeter.apache.org/download_jmetercgi	Open source	Yes	Limited
Rational performance tester	https://www.ibm.com/developerworks/downloads/r/rpt/	Commercial	Medium	Liberal
Loadstorm	https://loadstorm.com/	Commercial	Medium	Limited
WAPT	https://www.loadtestingtool.com/	Commercial	Medium	Liberal

Table 2 The DDoS attacking tools available on the internet

DDoS tool name	Online link	Availability	GUI based
HOIC	https://sourceforge.net/projects/highorbitioncannon/	Free	Yes
LOIC	https://sourceforge.net/projects/loic/	Free	Yes
Golden Eye	https://github.com/jseidl/GoldenEye	Free	No
HULK	https://packetstormsecurity.com/files/112856/HULK-Http-Unbearable-Load-King.html	Free	No
DDOSIM	https://sourceforge.net/projects/ddosim/	Free	No
RUDY	https://code.google.com/archive/p/r-u-dead-yet/	Free	No

Table 3 The real-time HTTP flooding dataset available on the internet

Real time datasets	Online link	Number of HTTP requests	Collection period
FIFA World Cup 1998	http://ita.ee.lbl.gov/html/contrib/WorldCup.html	1.3 Billions	30th April 1998–26th July 1998
Environmental Protection Agency (EPA)	http://ita.ee.lbl.gov/html/contrib/EPA-HTTP.html	46,014	29th August 1995–30th August 1995
San Diego Supercomputer Center (SDSC)	http://ita.ee.lbl.gov/html/contrib/SDSC-HTTP.html	28,338	22nd August 1995
University of Calgary's Computer Science Department web logs	http://ita.ee.lbl.gov/html/contrib/Calgary-HTTP.html	726,739	24th October 1994–11th October 1995
ClarkNet Internet Service Provide Web server logs	http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html	3,328,587	28th August 1995–03rd September 1995 & 04th September 1995–10th September 1995
NASA Kennedy Space Center Florida web server logs	http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html	3,461,612	1st July 1995–31st July 1995 & 1st August 1995–31st August 1995
University of Saskatchewan's web logs	http://ita.ee.lbl.gov/html/contrib/Sask-HTTP.html	2,408,625	1st June 1995–31st December 1995

Though the tools in Tables 1 and 2 helps to evaluate the solution to some level, it is always wise to use any of the real-time datasets in Table 3.

5 Proposed architecture model

The proposed cloud testbed framework depicted in Fig. 1. The cloud environment has multiple tenants. The different tenants represented in different colours for easy understanding. Each tenant has one or more virtual instances/machines running for multiple purposes in the cloud environment. The virtual machines of the tenant shall be running on the same or different compute nodes in the cloud. The compute node may be running on the same or different physical machines of the cloud. These physical machines can present anywhere in the world.

In Fig. 1, the tenant with green colour has multiple virtual machines running on different compute node of the cloud. The same tenant also runs the web server on the one of the virtual machine. There are multiple compromised virtual machines running on the various compute nodes. These compromised virtual machines are known as BOT VM's and used by the attackers to generate HTTP flooding attack to bring down the web server running in the cloud environment.

Similarly, there are other compromised BOT machines running over the internet by the attacker to generate the HTTP flooding attack to the same web server running in the cloud. Now, based on the placement of the compromised BOT machines used by the attackers, multiple HTTP flooding attack path exist to the web server in the cloud environment.

The different possibilities of the HTTP flooding attack paths in the cloud environment captured in Fig. 1. The description of the same as follows.

- The HTTP flooding attacks by the compromised virtual instance of the same tenant (1).
- The HTTP flooding attacks from a compromised virtual instance of the other tenants. Fundamentally, attacks from other tenants running in the same cloud (2).
- The HTTP flooding attacks by the compromised virtual instance within the same compute node (3).

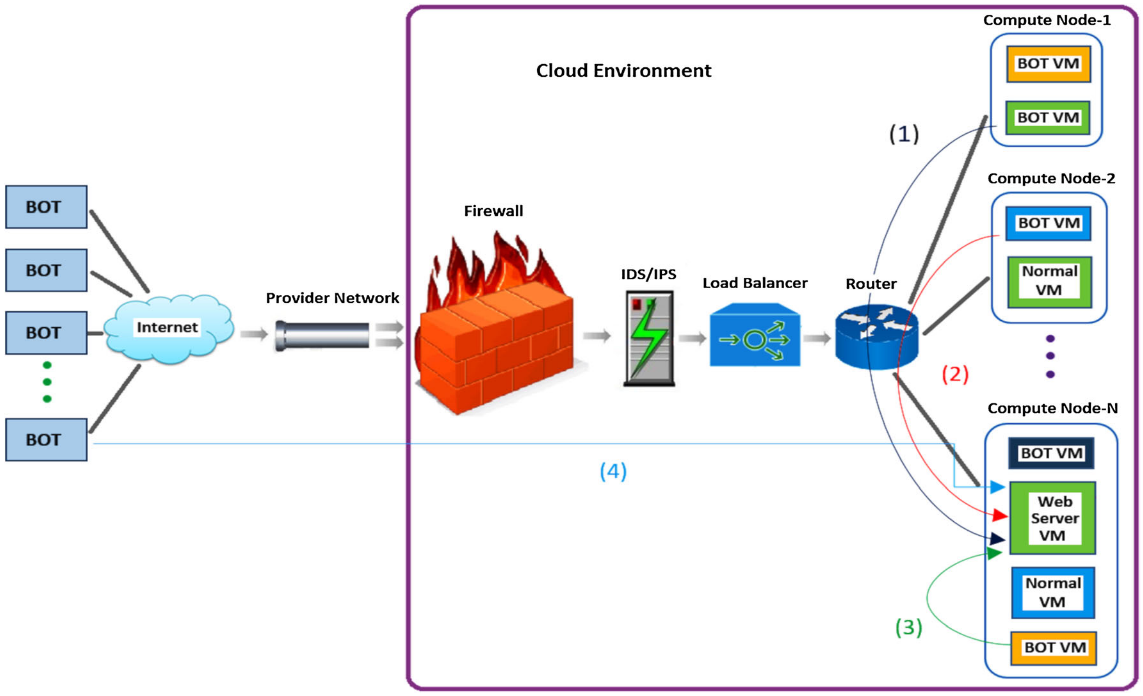


Fig. 1 Proposed cloud testbed framework architecture (Color figure online)

- The HTTP flooding attacks by the compromised system from the internet (4).

The HTTP flooding attacks from the internet to the cloud web server comes through the various components shown in Fig. 1. The components include the internet service provider network, firewall, *Intrusion Detection System* (IDS) or *Intrusion Prevention Systems* (IPS), load balancer and router.

The above scenarios cover the perspective of the HTTP flooding attack within the cloud environment itself as well as an attack from the outside of the cloud environment. The outside attack indicates that the compromised system or bot are flooding HTTP requests over the internet to the victim web server running in the cloud environment.

The typical HTTP flooding attack paths to the web server are different from attack over the web server in cloud environment due to the facts explored here. The framework has been developed to explore all possibilities in the cloud to make sure that all attack paths discovered so that the web services running in the cloud environment safeguarded in a better manner.

6 The OpenStack cloud framework implementation details

The proposed solution implemented in the OpenStack cloud environment. Figure 2 shows the list of virtual instances running in the cloud environment.

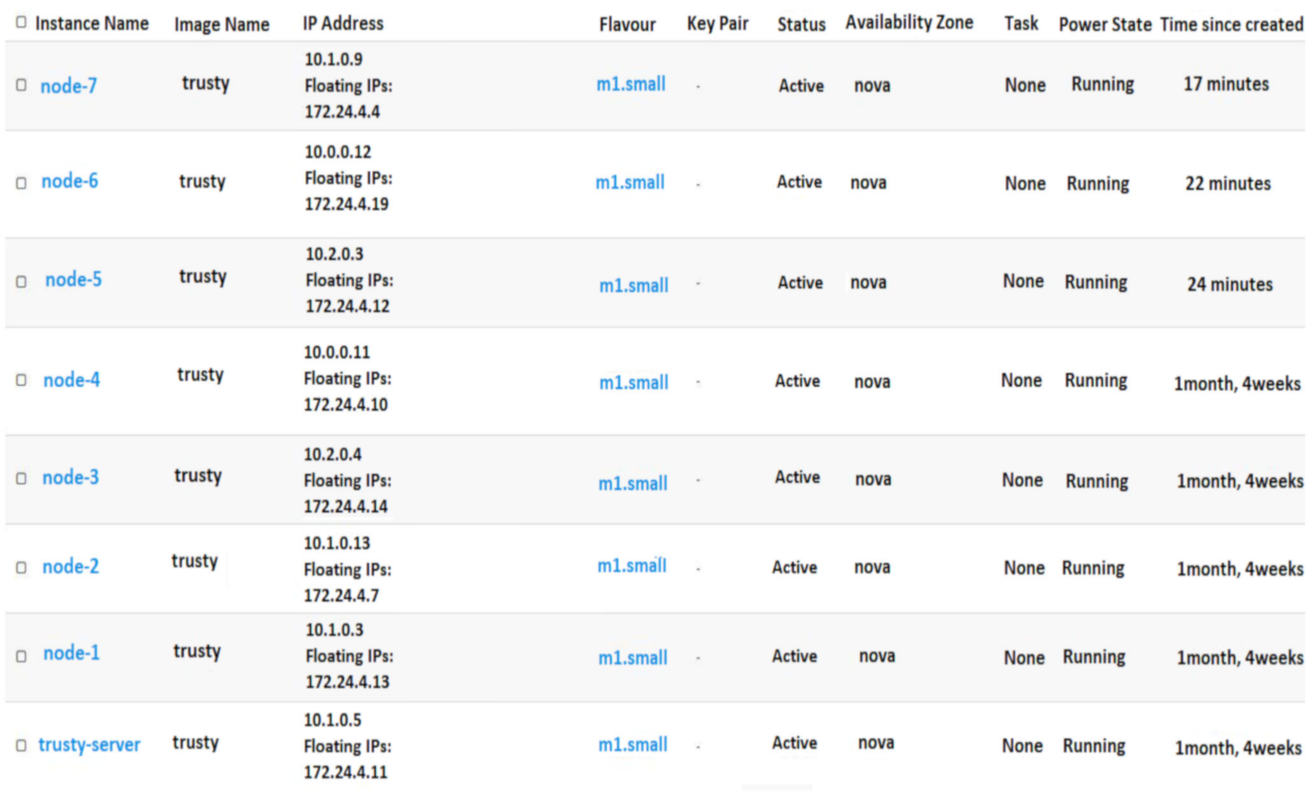
Each virtual instance associated with internal/private *Internet Protocol* (IP) address for communication between virtual instances of the same the tenants. For example, different departments of the same organisation in the cloud communicates using the internal/private IP address.

Similarly, floating/external IP address assigned to each virtual instances for communicating from/to the internet. The clients/customers of the organisation shall access the services of the organisation using the floating IP address from the outside world. The IP address starts with 10.x.x.x is an internal/private address, and IP address starts with 172.24.x.x is floating/external IP address.

Each tenant separated by means of different internal networks, the same depicted using OpenStack network topology in Fig. 3. The tenant-1 allotted by the orange network, tenant-2 placed on the green networks and tenant-3 is on the red network. The Nginx web server launched by tenant-2 virtual instance named as trusty-server.

All these tenants connect to the outside world with the help of router-1 using the public network. The HTTP flooding targeted to the webserver running on the green network.

The webserver under HTTP flooding attack belongs to the tenant-2. The tenant-2 also has virtual instances such as node-1, node-2 and node-7 which are spread across multiple compute nodes. The tenant-1 virtual instance node-3 and node-5 generate the HTTP flooding to the webserver of tenant-2 placed over the green network.



▢ Instance Name	Image Name	IP Address	Flavour	Key Pair	Status	Availability Zone	Task	Power State	Time since created
▢ node-7	trusty	10.1.0.9 Floating IPs: 172.24.4.4	m1.small	-	Active	nova	None	Running	17 minutes
▢ node-6	trusty	10.0.0.12 Floating IPs: 172.24.4.19	m1.small	-	Active	nova	None	Running	22 minutes
▢ node-5	trusty	10.2.0.3 Floating IPs: 172.24.4.12	m1.small	-	Active	nova	None	Running	24 minutes
▢ node-4	trusty	10.0.0.11 Floating IPs: 172.24.4.10	m1.small	-	Active	nova	None	Running	1month, 4weeks
▢ node-3	trusty	10.2.0.4 Floating IPs: 172.24.4.14	m1.small	-	Active	nova	None	Running	1month, 4weeks
▢ node-2	trusty	10.1.0.13 Floating IPs: 172.24.4.7	m1.small	-	Active	nova	None	Running	1month, 4weeks
▢ node-1	trusty	10.1.0.3 Floating IPs: 172.24.4.13	m1.small	-	Active	nova	None	Running	1month, 4weeks
▢ trusty-server	trusty	10.1.0.5 Floating IPs: 172.24.4.11	m1.small	-	Active	nova	None	Running	1month, 4weeks

Fig. 2 The OpenStack virtual instances detail

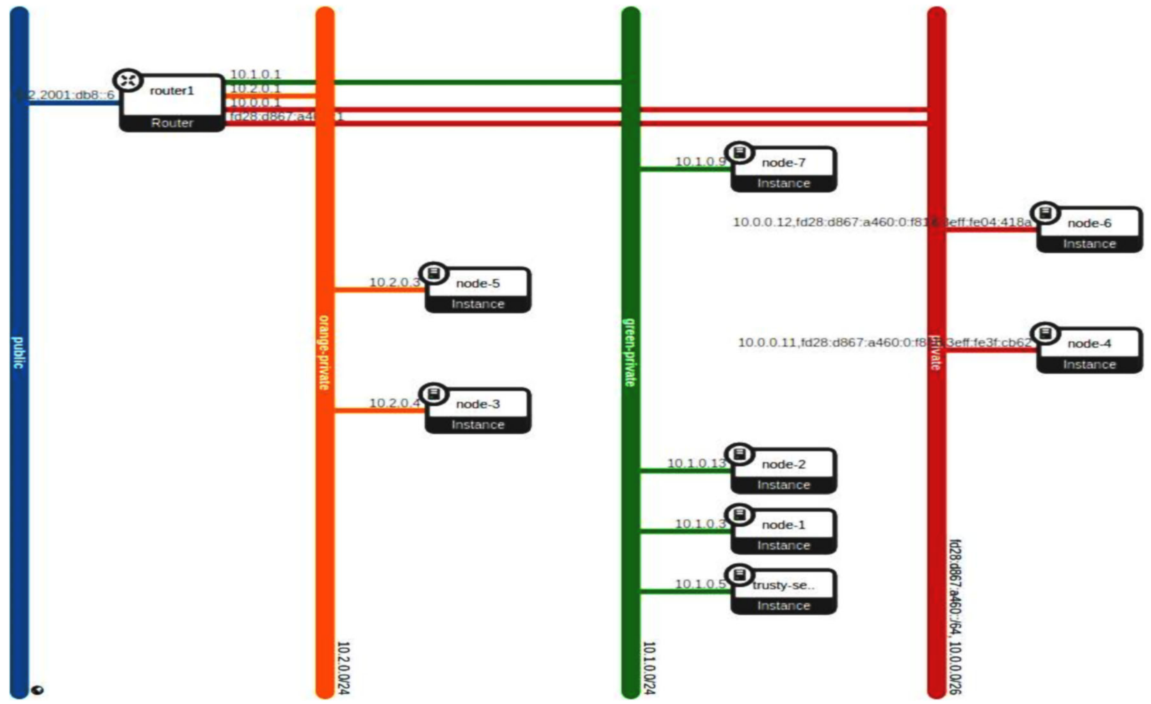


Fig. 3 The OpenStack network topology diagram

Similarly, node-5 and node-6 of the tenant-3 also generate the flooding attack to the webserver. These attacks launched using both internal/private IP address as well as floating/external IP address. This helps to cover all the case of HTTP flooding attacks within the cloud of the same tenant as well as other tenants and attacks from outside world.

7 Comparisons of proposed work with other works

This section compares the various relevant existing works and their gaps with proposed framework model.

Smirnov et al. [19], proposed an architecture model, but it has gaps in capturing the cloud specific information such as elasticity, multi-tenancy and related attack paths in their work.

Kobayashi et al. [20], covered the architecture model which introduced an intermediate server called control machines. This generic model covered the attack path from the internet, and it has gaps in exploring the various aspects of the cloud.

Shruthi et al. [21], captured the generic framework model for cloud computing and it has gaps to identify the various potential attacks on the cloud environment and the real-time dataset not used for the experimentation.

Shahanaz Begum et al. [22], discussed and proposed DDoS detection in the private cloud environment. This work captures some aspects of the attack within the cloud but has gaps in clearly defining multiple attack paths within the cloud as well as from outside of the cloud.

Kiruthika Devi et al. [23], proposed the cloud framework which captured the virtualization aspects of the cloud. This has gaps in explicitly defining the other properties of the cloud such as multi-tenancy, computing model as well as discovering the attack paths in the cloud.

Karnwal et al. [24], explained the cloud architecture model and used the real-time dataset for their experimentation. This has gaps in capturing cloud environment details, and other cloud characteristics like multi-tenancy, elasticity, computing model. It also missed covering the various attack paths not covered as well.

Osanaiye et al. [25], defined the cloud framework model. This paper also used real-time dataset for their experimentation. It has gaps in exploring the potential attacks within the cloud environment as well as from the internet. This is very critical when defining the solution to cloud environment as possibilities of attacks are different from the conventional web server.

The proposed architecture model comprises cloud architecture model along with considering the different aspects of cloud computing. It also investigates the various possibilities of attack paths to the web server in the cloud environment which includes both attack paths within the cloud as well as outside of the cloud. The real-time dataset also used in the experimentation to cover all possible attack paths to the victim web server running in the cloud environment. The summary of the various work discussed and their comparison showed in Table 4.

8 Results

The author considered different options for designing the testbed framework and chosen OpenStack Infrastructure as a Service model due to advantages discussed in Sect. 3 to implement the solution. Equally, analyzed the various options for generating the HTTP flooding attack as discussed in Sect. 4, and due to the fact of simulating the real-time attacks and obtain more granular results, the author decided to use one of the real-time datasets discussed in Table 3.

From the perspective of the richness of collections, and varieties of resources in requests, the authors decided to use the FIFA World CUP 1998 dataset for generating the HTTP flooding in this research work. The FIFA World CUP 1998 dataset is in the processed logs format to prevent the leakage of the confidential details. This dataset requires conversion of the logs into the HTTP request in order to recreate the HTTP flooding attacks. The process explained in the paper [18] used to regenerate the HTTP flooding attack generation from the FIFA World CUP 1998 dataset.

The snapshot for the normal HTTP request scenarios to the web server running the cloud environment depicted in Fig. 4. The picture in Fig. 4 captured with the help of Wireshark input/output graph. The number of requests to the web server seen during normal scenario varies from seven requests to sixty requests per second.

The HTTP flooding attack has been launched to the victim web server running in the cloud environment. The corresponding results captured with the help of the Wireshark tool.

Figure 5 shows the Wireshark packet capture of the HTTP flooding attack using the FIFA World CUP 1998 dataset in the OpenStack cloud environment. The web server is running with private IP address 10.1.0.5 and floating IP address as 172.24.4.11 as pictured in Fig. 2.

The HTTP requests flooding from the various bot machines with IP addresses 172.24.4.10, 172.24.4.201, 172.24.4.7, 172.24.4.14, 10.2.0.4, 10.1.0.13 to the web server is shown in the captured scenario. The IP address mapping to tenant nodes is captured in Figs. 2 and 3.

The HTTP attack paths to the web server with IP address 10.1.0.5/172.24.4.11 are

- The HTTP flooding attacks from different tenant node-4 with IP address 172.24.4.10 and node-3 with IP address 172.24.4.14 to the web server.
- The HTTP flooding attacks from same tenant node-2 with IP address 172.24.4.7 to the web server.

Table 4 Comparison of the various work

Reference number	Architecture (or) framework proposed	Multi-tenancy model considered	Elasticity and virtualization considered	Computing model considered	Real-time DDoS dataset discussed (or) used
[19]	✔	✗	✗	✔	✗
[20]	✔	✗	✔	✗	✗
[21]	✔	✗	✔	✗	✗
[22]	✗	✗	✗	✗	✔
[23]	✔	✗	✔	✗	✗
[24]	✔	✗	✗	✗	–
[25]	✔	✗	✗	✗	✔
Proposed work	✔	✔	✔	✔	✔

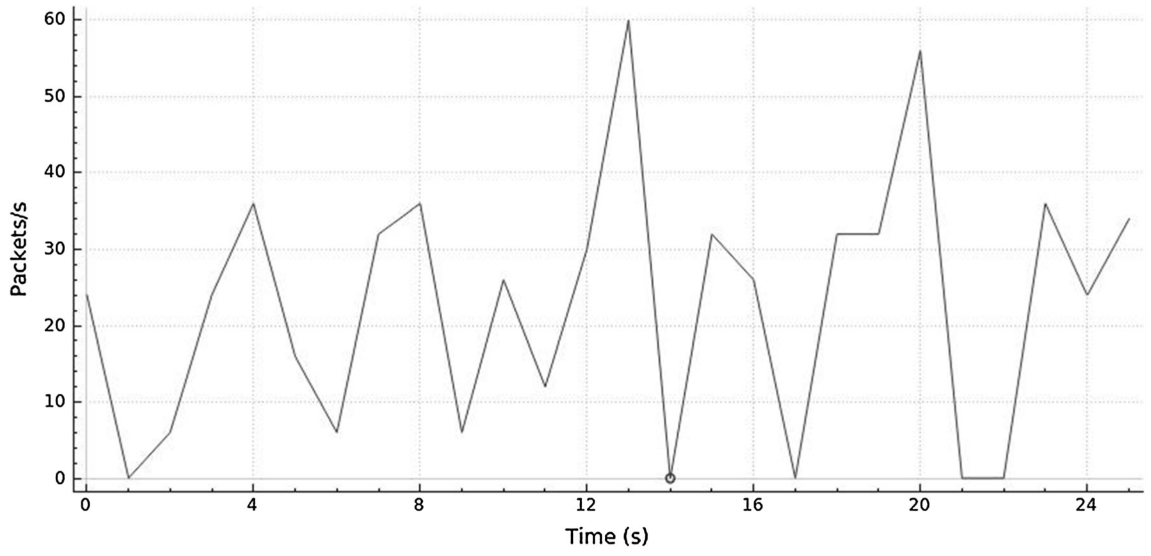


Fig. 4 The HTTP request normal scenario input/output graph

- The HTTP flooding attacks within the same compute node by the tenant with IP address 10.1.0.13 to the web server.
- The HTTP flooding attack from outside of the cloud environment with IP address 172.24.4.201 to the web server.

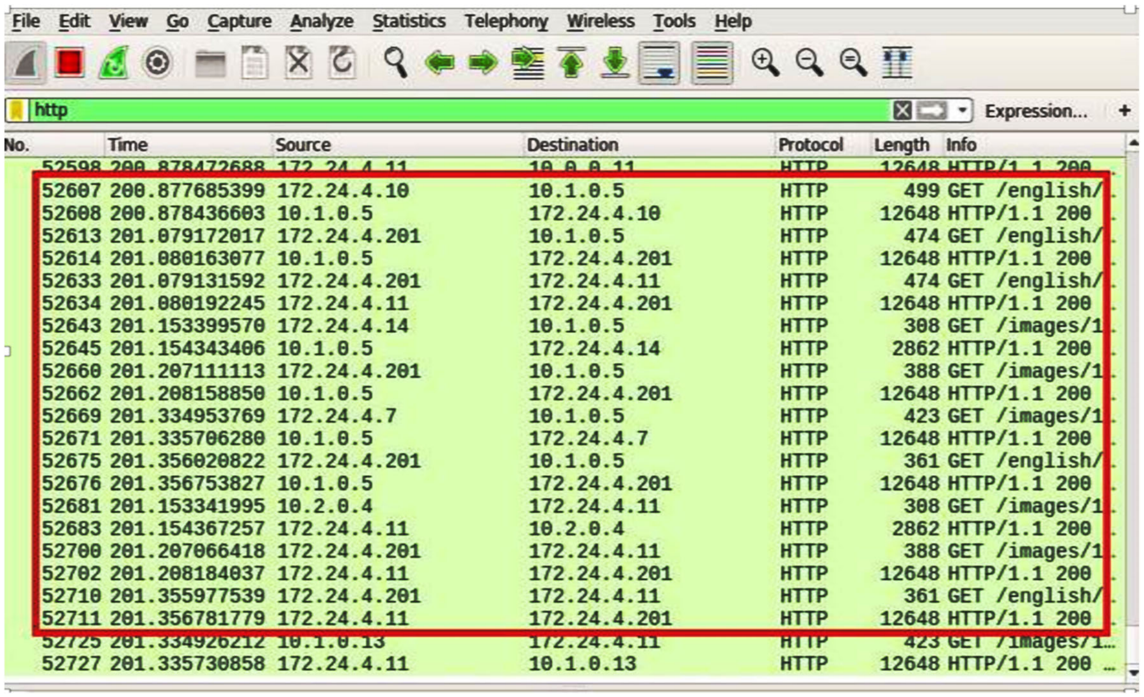
The experiment covers all possible attack scenarios discussed in Sect. 5. The equivalent input/output performance graph in the OpenStack environment showed in Fig. 6.

It is evident from Fig. 6 that the number of HTTP requests continuously flooded to the web server is really huge when compared to the normal scenario. The number of requests to the web server ranges from 400 to 1000 packets per second during a captured attack situation. The time axis interval of 10 s in Fig. 6 represents the constant flooding of HTTP requests to the web server when comparing the same time axis interval of 1 s in Fig. 4 during the normal situation.

9 Conclusion and future direction

The authors discussed several issues in the cloud computing, details of the DDoS attacks, classification of DDoS attacks. The HTTP flooding attack is one of the renowned and topmost DDoS issues in the cloud. This paper reviewed the various cloud testbed framework, architectures proposed and their gaps. Proposed a new cloud testbed framework architecture to covers all possible scenarios of the HTTP flooding DDoS attack.

The experimental details of the proposed model and its implementation using the OpenStack cloud environment discussed. The various possibilities to generate the HTTP flooding attack explained. The comparison of the existing


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
Expression...
+

```
No.   Time           Source        Destination   Protocol Length Info
52598 200.878472688 172.24.4.11   10.0.0.11     HTTP     12648 HTTP/1.1 200
52607 200.877685399 172.24.4.10   10.1.0.5      HTTP       499 GET /english/…
52608 200.878436603 10.1.0.5      172.24.4.10   HTTP     12648 HTTP/1.1 200 …
52613 201.079172017 172.24.4.201  10.1.0.5      HTTP       474 GET /english/…
52614 201.080163077 10.1.0.5      172.24.4.201  HTTP     12648 HTTP/1.1 200 …
52633 201.079131592 172.24.4.201  172.24.4.11   HTTP       474 GET /english/…
52634 201.080192245 172.24.4.11   172.24.4.201  HTTP     12648 HTTP/1.1 200 …
52643 201.153399570 172.24.4.14   10.1.0.5      HTTP       308 GET /images/1…
52645 201.154343406 10.1.0.5      172.24.4.14   HTTP      2862 HTTP/1.1 200 …
52660 201.207111113 172.24.4.201  10.1.0.5      HTTP       388 GET /images/1…
52662 201.208158850 10.1.0.5      172.24.4.201  HTTP     12648 HTTP/1.1 200 …
52669 201.334953769 172.24.4.7    10.1.0.5      HTTP       423 GET /images/1…
52671 201.335706280 10.1.0.5      172.24.4.7    HTTP     12648 HTTP/1.1 200 …
52675 201.356020822 172.24.4.201  10.1.0.5      HTTP       361 GET /english/…
52676 201.356753827 10.1.0.5      172.24.4.201  HTTP     12648 HTTP/1.1 200 …
52681 201.153341995 10.2.0.4      172.24.4.11   HTTP       308 GET /images/1…
52683 201.154367257 172.24.4.11   10.2.0.4      HTTP      2862 HTTP/1.1 200 …
52700 201.207066418 172.24.4.201  172.24.4.11   HTTP       388 GET /images/1…
52702 201.208184037 172.24.4.11   172.24.4.201  HTTP     12648 HTTP/1.1 200 …
52710 201.355977539 172.24.4.201  172.24.4.11   HTTP       361 GET /english/…
52711 201.356781779 172.24.4.11   172.24.4.201  HTTP     12648 HTTP/1.1 200 …
52725 201.334926212 10.1.0.13     172.24.4.11   HTTP       423 GET /images/1…
52727 201.335730858 172.24.4.11   10.1.0.13     HTTP     12648 HTTP/1.1 200 …
```

Fig. 5 The HTTP flooding attack Wireshark capture

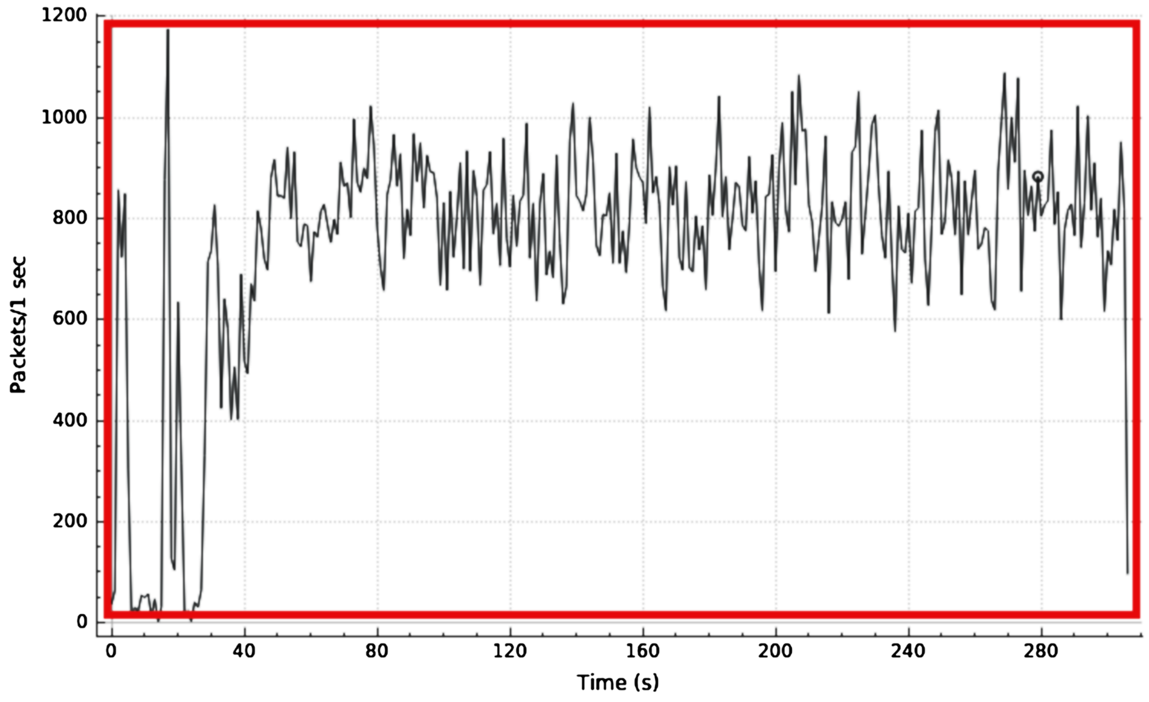


Fig. 6 The HTTP flooding scenario input/output performance graph

work with proposed model discussed. Finally, the HTTP flooding attack using FIFA World CUP 1998 dataset used to flood the webserver running in the OpenStack, the corresponding snapshots captured and discussed.

The future direction is to enhance the proposed framework to detect and mitigate the HTTP flooding attacks in the cloud environment.

References

1. The 5 motives for DDoS attack. https://arch.simplicable.com/arch/new/the-5-motives-for-DDoS-attack. Accessed 9 Jan 2019.
2. NIST Cloud Computing Program—NCCP. https://www.nist.gov/programs-projects/nist-cloud-computing-program-nccp. Accessed 9 Jan 2019.
3. Top 10 security concerns for cloud-based services. https://www.incapsula.com/blog/top-10-cloud-security-concerns.html. Accessed 9 Jan 2019.
4. Dhanapal, A., et al. (2013) Data usage security, accounting and auditing in cloud computing. In National Conference on Networking and Communication Systems (NCS) (vol. 2, pp. 226–229).
5. Denial of service attack: what is a dos attack? https://security.radware.com/ddos-knowledge-center/ddospedia/dos-attack/. Accessed 9 Jan 2019.
6. DDoS attack types and mitigation methods. https://www.incapsula.com/ddos/ddos-attacks/. Accessed 9 Jan 2019.
7. What is a DDoS attack and how do you protect against DDoS attacks? https://www.arbornetworks.com/research/what-is-ddos. Accessed 9 Jan 2019.
8. Why move to the cloud? 10 benefits of cloud computing. https://www.salesforce.com/uk/blog/2015/11/why-move-to-the-cloud-10-benefits-of-cloud-computing.html. Accessed 9 Jan 2019.
9. DDoS top 6: Why hackers attack. https://www.pentasecurity.com/blog/ddos-top-6-hackers-attack/. Accessed 9 Jan 2019.
10. Widespread cyberattack takes down sites worldwide. http://money.cnn.com/2016/10/21/technology/ddos-attack-popular-sites/index.html. Accessed 9 Jan 2019.
11. The 5 most significant DDoS Attacks of 2016. https://www.tripwire.com/state-of-security/security-data-protection/cyber-security/5-significant-ddos-attacks-2016/. Accessed 9 Jan 2019.
12. Application attacks against clouds up 45%. https://www.csoonline.com/article/2991409/cloud-security/application-attacks-against-clouds-up-45.html. Accessed 9 Jan 2019.
13. Denial-of-Service attacks meet the cloud: 4 Lessons. https://www.cio.com/article/2413818/cloud-computing/denial-of-service-attacks-meet-the-cloud–4-lessons.html. Accessed 9 Jan 2019.
14. Cloud delivery models. http://whatiscloud.com/cloud_delivery_models/index. Accessed 9 Jan 2019.
15. Cloud deployment models. http://whatiscloud.com/cloud_deployment_models/index. Accessed 9 Jan 2019.
16. Dhanapal, A., et al. (2018) A review of cloud computing adoption issues and challenges. Recent Patents on Computer Science. https://doi.org/10.2174/2213275911666181114142428.
17. World Cup Web Site Access Logs. http://ita.ee.lbl.gov/html/contrib/WorldCup.html.
18. Dhanapal, A., et al. (2017) An effective mechanism to regenerate HTTP flooding DDoS attack using real time data set. In ICICICT (pp. 570–575).
19. Smirnov, A.V., et al. (2016) Network traffic processing module for infrastructure attacks detection in cloud computing platforms. In IEEE (pp. 199–202).
20. Kobayashi, R., et al. (2016). Defense method of HTTP GET flood attack by adaptively controlling server resources depending on different attack intensity. *Journal of Information Processing, 24*(5), 802–815.
21. Shruthi, B. T., et al. (2016). X-DoS (XML Denial of Service) attack strategy on cloud computing. *Imperial Journal of Interdisciplinary Research, 2*(12), 1665–1669.
22. Shahanaz Begum, I., et al. (2016). DDoS attack detection and prevention in private cloud environment. *International Journal of Innovations in Engineering and Technology, 7*(3), 527–531.
23. Kiruthika Devi, B. S., et al. (2016). Comparative analysis of security methods for DDoS attacks in the cloud computing environment. *Indian Journal of Science and Technology, 9*(34), 1–7.
24. Karnwal, T., et al. (2012) A comber approach to protect cloud computing against XML DDoS and HTTP DDoS attack. In IEEE Students' Conference on Electrical, Electronics and Computer Science (pp. 1–5).
25. Osanaiye, O., et al. (2016) Change-point cloud DDoS detection using packet inter-arrival time. In 8th Computer Science and Electronic Engineering Conference (CEEC) (pp. 204–209).

A. Dhanapal received his B.E., in Computer Science and Engineering from Bharathiar University, India. He received his M.E., in Computer Science and Engineering from Sathyabama University, India and currently pursuing Ph.D. in the field of Cloud Computing Security in VIT University, India. He has more than 10 years of industry experience and worked in various areas like high-performance server computing, storage technologies and networking for companies like NetApp, Brocade, etc. He is passionate about open source, and the area of interests includes Cloud Computing, Cloud Security, Networking, Virtualization, etc. He is an active reviewer in the prestigious journals like Springer—The Journal of Super Computing, Elsevier—The Journal of Network and Computer Applications. He is a co-author of the book "Network Theory and Analysis" of Nova Science Publishers, Inc. He is a member of IEEE student community and a member of National Cyber Safety and Security Standards (NCSSS) of India.

P. Nithyanandam received his B.E., and M.E., Degree in Computer Science and Engineering. He received his Ph.D. Degree from Anna University, India. He is currently working as Professor in VIT University, Chennai, India. His area of interest includes Cryptography, Image Processing, Information Hiding and Cloud Computing. He is a co-author of the book "Network Theory and Analysis" of Nova Science Publishers, Inc.