Algerian Democratic and Popular Republic
Ministry of Higher Education and Scientific Research
University of Abderrahmane Mira-BEJAIA
Faculty of Exact Sciences
Doctoral School of Cloud Computing

Dissertation

Presented by

HADDADI Mohamed

For obtaining the Diploma of Magister

Speciality:Computer Science

Option: Cloud Computing

Theme

Securing Cloud Computing

Supported the :14/01/2016 In front of the Jury made up of :

Name and First name	Grade	University of origin	
Mr.GHEDAMSI Kaci	M.C.A	Univ of Bejaia	**President**
Mr.BEGHDAD Rachid	M.C.A	Univof Bejaia	**Supervisor**
Mr.BATOUCHE Mohamed Chawki	Professor	Univof Constantine3	**Examiner**
Mr.IDOUGHI Djilali	M.C.A	Univ of Bejaia	**Examiner**

University Year:2015-2016

Acknowledgements

I am deeply indebted to my research supervisor, ***Dr Rachid BEGHDAD*** *for presenting me such an interesting thesis topic. Each meeting with him added invaluable aspects to the implementation and broadened my perspective. He has guided me with his invaluable suggestions, lightened up the way in my darkest times and encouraged me a lot in the academic life. From him I have learned to think critically, to select problems, to solve them and to present their solutions.*

I will be failing in my duty if I don't acknowledge some of my friends in the campus with whom I have shared my research experiences since it was a joy and enlightenment to me.

I would like to address special thanks to the unknown reviewers of my thesis, for accepting to read and review this thesis.

DEDICATION

DEDICATED WITH EXTREME AFFECTION AND GRATITUDE TO

my parents

my brothers and my sisters

my research supervisor ***Dr Rachid BEGHDAD***

my friends specially Amar Lamraoui ,Mohamed Chekrid and his friend who called Fateh.

"All truths are easy to understand

once they are discovered;

the point is to discover them."

Galileo Galilei (1564 – 1642)

الملخص

الحوسبة السحابية تتضمن العديد من المفاهيم الإفتراضية ،والتطبيقات الموزعة ،وغيرها لنشر وتوزيع مختلف الخدمات والتطبيقات بشكل سهل ومرن ،وذلك عن طريق الأنترنت .

الحوسبة السحابية لديها القدرة على تغير وتطوير تكنولوجيا المعلوماتية من جلال توفير مختلف البرمجيات والمنصات وخدمة البيانات التحتية،حيث أنها أحدثت ثورة على مستوى الوسائل والطرق المستعملة خاصة في المؤسسات الإقتصادية ولكن في السنوات الأخيرة تعرضت إلى العديد من الهجمات مما أثر سلبا على عملها الإجابي وخاصة هجمات DDoS/DoS أو بما تعرف بهجمات الفيضانات التي تعتبر من الهجمات الصعبة جدا لستعملها IP spoofing في أغلب الحالات لعدم معرفة مصدر هجومها والتي تمنع الدخول للمستعمل العادي إلى الحوسبة السحابية . في هذا الصدد نقترح خوارزمية مشتقة (HCFm) من الخوارزمية الأصلية المسماة "التصفية بعدد القفزات(HCF)" بتغير حالة التنبه إلى تلك الحالة التي تشمل كل القيم الممكنة لعدد القفزات (HC) وكذلك التحليل الإحصائي لحساب العدد الكلي لكل حزمةخلال مجال زمني معين لمعرفة هجمات الفيضانات التي لا تستعمل IP spoofing لإدراجه في القائمة السواء إذا كان عدده يفوق عتبة معينة مدروسة من قبل بحيث يجب تحرير حزمة من القائمة السوداء إذا لم تستعمل كمصدر هجوم خلال كمية معينة من الوقت .

بالمقارنة مع الخوارزمية الأصلية HCF يمكن أن نقول أن الخوارزمية HCFm متينة من حيث معدلات الكشف بحيث قللة جدا من معدل إيجابية كـــاذبة (FP) ورفعة من نسبة دقـــــة الكشف إلى 92% وكذلك قادرة على معرفة هجمات الفيضانات التي لا تستعمل IP spoofing.

الكلمات المفتاحية : الحوسبة السحابية ، IP spoofing، HCFm ،القائمة السوداء ،HCF، هندسة VMware ،حماية الحوسبة السحابية ، هجمات الفيضانات.

Résumé

Cloud computing est la convergence et l'évolution de plusieurs concepts de virtualisation, des applications distribuées, grille computing et d'autres pour permettre une approche plus souple pour le déploiement et la mise à l'échelle des applications et des services délivrés via l'Internet.Le Cloud Computing, comme un service public, a le potentiel de transformer l'IT de l'industrie par la fournissant des softwares, des plates-formes et des infrastructures comme un service, et en révolutionnant les moyens et les manières traditionnelles de gestion d'IT de l'entreprise. Dans les dernières années, le cloud computing était exposé aux attaques les plus dangereux qui sont les attaques par inondation (flooding) ,et qui sont une nouvelle forme d'attaque en utilisant l'IP spoofing dans la majorité des cas .Dans ce papier ,nous proposons un algorithme (HCFm) ,inspiré de la technique de filtrage par nombre de sauts (HCF :Hop Count Filtering) qui change l'état d'alerte de HCF pour indiquer toutes les possibilités des valeurs de nombres de sauts (Hc) et les adresses IP sources par extraction à partir des en-têtes TCP et IP ,UDP et IP ,et ICMP et IP, et le seuil pour vérifier le nombre de chaque paquet pendant une tranche de temps (slot time) pour connaitre les attaques par inondation qui n'utilisent pas l'IP spoofing et ajouter l'adresse IP source de l'attaquant dans la table de la liste noire s'il dépasse le seuil pour minimiser le temps de traitement de chaque paquet d'attaque durant les attaques par inondation (flooding),mais il libère une adresse IP qui n'est pas utilisée durant X quantité de temps bien déterminée à l'avance .Par comparaison avec l'algorithme original (HCF),notre algorithme ,qui s'appelle HCFm , apparaît très efficace en terme de taux de faux positifs (FP) et le pourcentage de détection qui atteint 92% et aussi ,il peut connaître les attaques par inondation(flooding) qui n'utilisent pas l'IP spoofing.

Mots clés : Cloud computing , architecture VMware , HCF,IP spoofing , liste noire ,sécurité de cloud computing , HCFm , attaques par inondation .

Abstract

Cloud Computing is one of today's most exciting technologies, because it can reduce the cost and complexity of applications. On the other hand, such complex and distributed architectures become an attractive target for attacks .Flooding attacks based DoS/DDoS represent a serious danger which can deny the legitimate users access to the service delivered by cloud by using IP spoofing in the majority of cases. In this paper, we propose an algorithm, inspired by the Hop Count Filtering(HCF) technique that changes the alert state of HCF to include all the possible available Hop Count values ,and statistical analysis such as threshold for detecting every IP packet of attackers that don't use IP spoofing and classifying it in the black list when its number is greater than the threshold during a slot-time ,but we must remove an IP address from the blacklist if it has not sent a spoofed packet in X amount of time. Compared to the original HCF method and its variants, our proposed method performs better than them and which achieves high detection accuracy (92%) with fewer false alarms and also, it can detect attackers that don't use IP spoofing.

Keywords : cloud computing ,cloud security, VMware architecture, Flooding attacks, HCFm , Blacklist ,HCF ,IP spoofing.

Abbreviations

A

AVG : Anti-Virus Guard
ACK: Acknowledgement
API : Application Programming Interface
ARP : Address Resolution Protocol

B

BEP: Business Executive Program
BBC : British Broadcasting Corporation

C

CBF: Confidence Based Filtering
CDN : Content Delivery Network.
CPU :Central Processing Unit
C&C : Command and Control

D

DDoS: Distributed Denial of Service.
DNS:Domain Name System
DoS : Denial of Service
DDNS: Dynamic Domain Name System

E

EBS : Elastic **B**lock **S**torage.
EC2 : **E**lastic **C**ompute **C**loud

H

HC: Hop Count .
HCF: Hop Count Filtering .
HTML:Hyper-Text Message Language

I

IaaS :Infrastructure as a Service.
ICMP : Internet Control Message Protocol
IFrame : Inline Frame
IP:Internet Protocol.
ISO: International Standardisation Organisation
IT : Information Technology
IGMP : Internet Group Management Protocol

J

JPCap: Java Packet Capture

N

NAT :Network Address Translator
NIST :National Institute of standards and Technology
NUI: Natural User Interface

O

OS :Operating System.

P

PaaS : Platform as a Service.
PC : personnel computer
P2P: Peer to Peer

Q

QoS : Quality of Service.

R

R2L: Remote to Local access.
RAM : Random Access Memory .
RDS : Relational **D**atabase **S**ervice.
RR : Resource Record
RARP : Reverse Address Resolution Protocol

S

S3 : Simple **S**torage **S**ervice.
SaaS : Software as a Service**.**
SLA : Service Level Agreement. .
SOAP : *Simple Object Access Protocol*
SQS : Simple **Q**ueue **S**ervice.
SSL : Secure Sockets Layer. .
SYN: Synchronization.

T

TCP :Transport Control Protocol .
TLS : Transport Layer Service

U

U2R : User to Root attacks.
UDP: User Data Protocol
UGNazi : **Underground Nazi Hacktivist Group**

W

WinPcap : Windows Packet Capture

X

XML :eXtensibleMarkupLanguage

Table of Contents

Chapter II :State of the Art of Flooding attacks based DoS/DDoS

Chapter III :Proposed Algorithm

Chapter IV: Simulation and performance evaluation

Liste of Figures

Chapter I :Cloud Computing Security and Flooding Attacks based DoS/DDoS

Chapter II :State of the Art of Flooding attacks based DoS/DDoS

Chapter III :Proposed Algorithm

Chapter IV: Simulation and performance evaluation

List of Tables

Chapter I :Cloud Computing Security and Flooding Attacks based DoS/DDoS

Chapter II :State of the Art of Flooding attacks based DoS/DDoS

Chapter III :Proposed Algorithm

Chapter IV: Simulation and performance evaluation

General Introduction

Cloud computing has become the newest rave in the computing industry. Its ability to save business's cost by eliminating the need to purchase huge amounts of software and/or software licenses for every employee, reducing the need for advanced hardware, eliminating the need for companies to rent physical space to store servers and databases, and shifting the workload from local computers that has appealed to cloud computing providers such as Amazon, Google, IBM , Yahoo, Microsoft, etc. [35, 36].That means : Instead of purchasing actual physical devices servers , storage, or any networking equipment, clients lease these resources from a cloud provider as an outsourced service.

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [22].

Cloud computing is composed of five essential characteristics, three service models, and four deployment models as shown in the figure bellow .

• Essential Characteristics: (On-demand self-service, broad network access, resource pooling, rapid elasticity (a user can have as much or as little of a service as needed at any given time) , measured service (typically by the minute or hour)).

• Service Models: These services are categorized as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [35]. Infrastructure as a Service provides low-level services which can be booted with a user-defined hard disk image such as Amazon EC2. In Platform as a Service, the cloud provider offers an API which can be used by an application developer to create applications on the provider's platform. Examples of PaaS include Force.com, GoogleApps, etc. With Software as a Service, the vendor supplies the software

product and interacts with users through a front-end portal; web-based office applications like Google Docs or Calendar are examples of SaaS [36].

• Deployment Models: (Private cloud, Community cloud, Public cloud, Hybrid cloud).

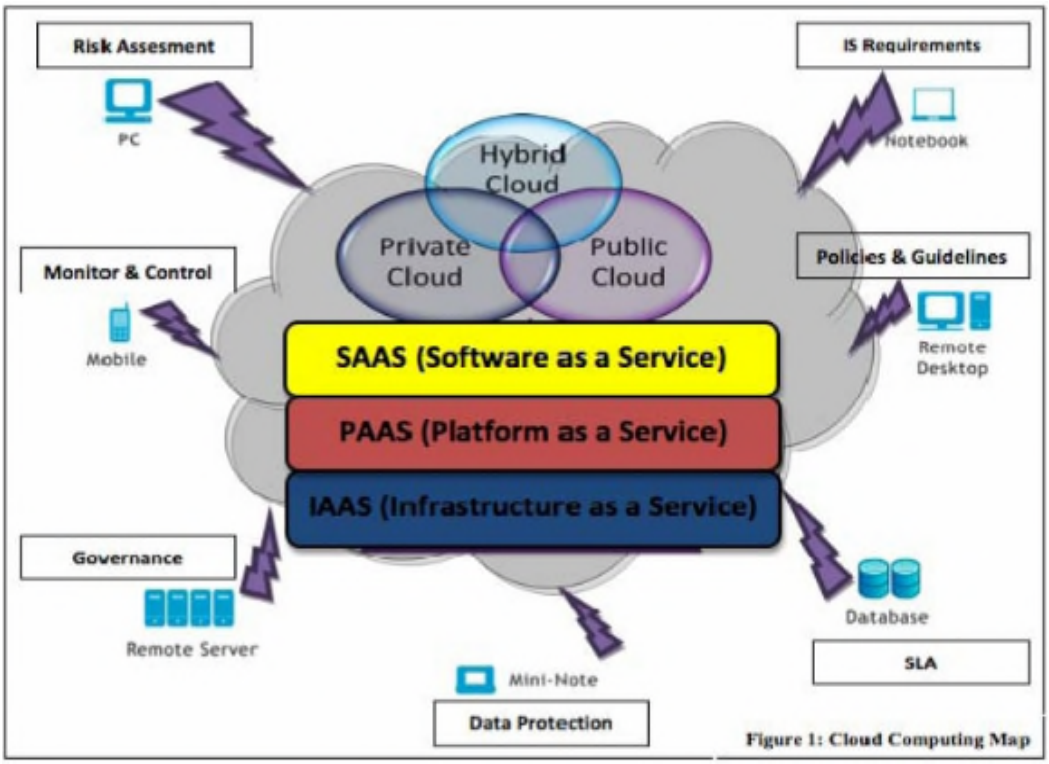


Figure 1: Cloud computing Architecture [37]

Which are provided by cloud providers that use virtualization technologies combined with self-service abilities for computing resources via network infrastructures, especially the Internet and multiple virtual machines are hosted on the same physical server.

The cloud computing security is one of the biggest obstacles because while the Cloud computing offers numerous advantages, therefore hackers (attackers) are also interested in it. Recently flooding attacks based DoS/DDoS (Distributed Denial of Service) attacks, which use IP spoofing in the majority of cases on clouds, have become one of the serious threats to this buzzing technology. They are a significant problem because they are very hard to detect, there is no comprehensive solution and they can shut an organization off from the Internet. The primary goal of an attack is to deny the victim's access to a particular resource.

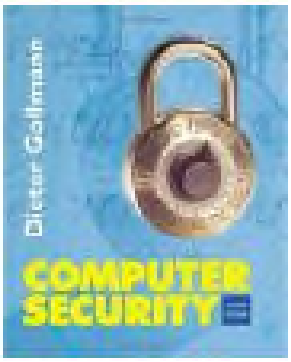
We have examined cloud computing security terms and some types of flooding attacks based DoS/DDoS in the first chapter . The chapter II gives the state of art of cloud computing flooding attacks approaches (algorithms). The proposed flooding attack algorithm will be presented in chapter III in order to secure cloud computing physical servers .Simulation and performance evaluation will be discussed in chapter IV in order to study the performance of our proposed algorithm in term of detection accuracy and false positive (FP) and false negative(FN) . The conclusion and future work will be the end of this work.

Chapter I

Cloud Computing Security

and

Flooding Attacks based DoS/DDoS



1-Introduction

At present, a major concern in cloud adoption is its security. Attacks prospect within cloud environment are many and with high gains. Security issues are of more concern to cloud service providers who are actually hosting the services. In most cases, the provider must guarantee that their infrastructure is secure and clients' data and applications are safe by implementing security policies and mechanisms. While the cloud customer must ensure that provider has taken proper security measures to protect their information.

2-A survey of cloud computing security

In this subsection ,we are going to give a survey on state of the art of cloud computing security, we'll explore exactly what are the security principles and terms, because one of the most confusing things about security is security terms such as vulnerabilities , threats , and risks, and in some cases those are used interchangeably [1].

2.1- Vulnerabilities

According to the Open Group's risk taxonomy [2], Vulnerability is "the probability that an asset will be unable to resist the actions of a threat agent.

2.1.1- Essential Characteristic Vulnerabilities:

NIST describes five essential cloud characteristics. However, in some of these characteristics, there may underlie a basis for a vulnerability; for example, Unauthorized access to management interface which is caused by cloud characteristic on-demand self-service, Data recovery vulnerability which is a result of cloud characteristics elasticity and resource pooling (there is the possibility of recovering data written by previous user), and Metering and billing evasion is finally a resulting vulnerability of the measured service characteristic; the data able to achieve this characteristic can be manipulated [1].

2.1.2- **Cloud services vulnerabilities**

As cloud computing mainly provides three types of services so in each layer have some soft corners which invite attackers to attack [3]. Some of these soft corners are mentioned as follows:

2.1.2.1-**SaaS vulnerability**

a-Insecure Application Programming Interface (API)

b-Account or Service hacking

c-Attack on cloud firewall / Attack on public firewall

d-Attack on consumer browser

e-Integrity, Confidentiality and Availability

2.1.2.2-**PaaS vulnerability**

a-Insecure Application Programming Interface (API)

b-Unknown risk profile (Heartland Data Breach)

c-Integrity, Confidentiality and Availability

2.1.2.3-**IaaS vulnerability**

a- Data leakage in Virtual Machine

b- Shared technology issues

c-Integrity, Confidentiality and Availability

2.2 Threats

According to the Open Group's risk taxonomy [2], a threat is "Anything that is capable of acting in a manner resulting in harm to an asset and/or organization; for example, acts of God (weather, geological events, etc.); malicious actors; errors; failures." A threat can be either intentional or accidental, and is a possible danger that can exploit a vulnerability with the potential to adversely impact systematic operations.

2.3 Risks

According to ISO 27005, Information Security Risk Management guideline, risk is "the potential that a given threat will exploit Vulnerability of an asset or group of assets and there by cause harm to the organization [4]." Risk is the probable frequency and probable magnitude of future loss.

3-Guest and Provider Sides of Cloud Computing

Cloud computing consists of guest and provider sides. The guest side is the end users who use the cloud such as laptops, tablets, cell phones, various computers and enterprise centers. It provides the end users with the ability to choose cloud services and environment. It is the interface that clients see after they enter credentials and have the ability to use the services provided by the cloud. The provider side of cloud computing is the service providers which consists of application servers, service platforms, runtime environment, and datacenters etc. Figure 1.1 is an example that shows the basic layout of the guest side and provider side of cloud computing [5].

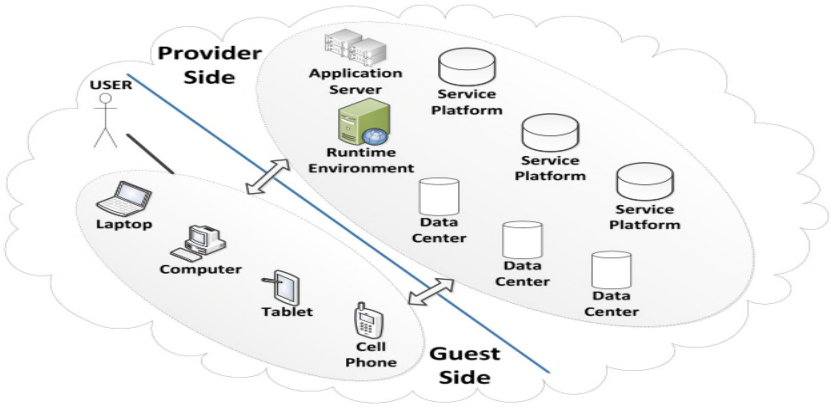


Figure 1.1: Guest and Provider Sides of Cloud Computing [6]

4-Cloud Computing Security Attacks and real-world cases

Cloud computing offers great potential to improve productivity and reduce costs, but at the same time it possesses many new security risks .We identify the possible security attacks in the clouds including: XML Signature Wrapping Attack, Malware Injection ,Social Engineering Attack, Account Hijacking ,Traffic Flooding and finally Wireless Local Area Network Attack .Multiple real-world cases where cloud computing were compromised and the ways the company mitigated the incident will be discussed. For each case the attack type will be briefly described, the details of the case will be presented.

4.1-XML Signature Wrapping Attack:

When a user makes a request from his VM through the browser, the request is first directed to the web server. In this server, a SOAP message is generated. This message contains the structural information that will be exchanged between the browser and server during the message passing. Before message passing occurs, the XML document needs to be signed and canonicalization has to be done. Also, the signature values should be appended with the document. Finally, the SOAP header should contain all the necessary information for the destination after computation is done. For a wrapping attack, the adversary does its deception during the translation of the SOAP message in the TLS (Transport Layer Service) layer. The body of the message is duplicated and sent to the server as a legitimate user. The server checks the authentication by the Signature Value (which is also duplicated) and integrity checking for the message is done. As a result, the adversary is able to intrude in the cloud and can run malicious code to interrupt the usual functioning of the cloud servers[6].

real-world cases:

In 2011, researchers lead by Dr. JorgSchwenk from Ruhr-University Bochum found a cryptographic hole in Amazon’s EC2 and S3 services. The flaw was located in the web

services security protocol and enabled attackers to trick servers into authorizing digitally signed SOAP messages that have been altered. The attackers hijacked control interfaces used to manage cloud computing resources, which would allow attackers to create, modify, and delete machine images, and change administrative passwords and settings [7].

4.2-Malware Injection:

In a malware-injection attack an adversary attempts to inject malicious code into a system. This attack can appear in the form of code, scripts, active content, and/or other software. When an instance of a legitimate user is ready to run in the cloud server, the respective service accepts the instance for computation in the cloud. The only checking done is to determine if the instance matches a legitimate existing service. However, the integrity of the instance is not checked. By penetrating the instance and duplicating it as if it is a valid service, the malware activity succeeds in the cloud [6].

real-world cases:

a-Case one occurred in May 2009. The United States Treasury Department moved four public websites offline for the Bureau of Engraving and Printing after discovering malicious code was added to the parent side [8]. The third-party cloud service provider hosting the company's website was victim to an intrusion attack. As a result numerous websites (BEP and non-BEP) were affected. *Roger Thompson*, chief research officer for Anti-Virus Guard (AVG) Technologies, discovered malicious code was injected into the affected pages. Hackers added a tiny snippet of a virtually undetectable iFrame HTML code that redirected visitors to a Ukrainian website. IFrame (Inline Frame) is an HTML document embedded inside another HTML document on a website. From there, a variety of web-based attacks were launched using an easy-to-purchase malicious toolkit called the Eleonore Exploit Pack [9].

b-Case two occurred in June 2011. The cyber criminals from Brazil who first launched their attacks as spam/phishing campaigns, where users were sent spoofed emails with links that took them to one of the malicious domains, created some major problems in Amazon Web Services [9]. The attackers installed a variety of malicious files on the victims' machines. One component acted as a rootkit (a type of malicious software that is activated each time a user's system boots up) and attempted to disable installed anti-malware applications. Additional components that were downloaded during the attack attempted to retrieve login information from a list of nine Brazilian banks and two other international banks, steal digital certificates from eTokens stored on the machine, and collect unique data about the PC itself that is used by some banks as part of an authentication routine [9].

4.3-Social Engineering Attack.

A social engineering attack is an intrusion that relies heavily on human interaction and often tricking other people to break normal security procedures [10]. It can happen in cloud computing.

real-world cases:

In August 2012, hackers used a social engineering attack to completely destroy technical writer Mat Honan's digital life by remotely deleting the information from his iPad, MacBook, and iPod [11]. The heart of the story revealed the dangerous blind spot between the identity verification systems used by Amazon and Apple. The hackers found the victim's @me.com address online which informed them that there was an associated AppleID account [11]. The hacker called Amazon customer service wanting to add a credit card number to the victim's account. The representative asked the hacker for the name, billing address, and an associated email address (all information the hacker found on the internet) on the victim's account. Once the hacker answered these questions successfully the representative added the new credit card onto the account. Once ending the call, the hacker called Amazon customer service back and explained to the representative that he had lost access to his account. The Amazon representative asked the hacker for his billing address and a credit card associated with the account; the hacker used the new credit card information he provided from the previous phone call. Once the hacker gave the representative the information they added a new email address to the victim's account. Upon logging onto Amazon's website the hacker requested a password reset the email address he just created. The hacker now had access to the victim's Amazon account and credit card information on file. The hacker then called Apple technical support and requested a password reset on the victim's @me.com email account. The hacker could not answer any of the victim's account security questions, but Apple offered him another option. The Apple representative only needed a billing address and the last four digits of the victim's credit card and issued the hacker a temporary password. Once the hacker had access to the victim's Apple iCloud account all the information from the victim's iPad, MacBook, and iPod account was remotely erased [11].

4.4-Account Hijacking:

Account hijacking is usually carried out with stolen credentials. Using the stolen credentials, attackers can access sensitive information and compromise the confidentiality, integrity, and availability of the services offered [12]. Examples of such attacks include: eavesdropping on

transactions/sensitive activities, manipulation of data, returning falsified information, and redirection to illegitimate sites [12].

real-world cases:

In July 2012, the hacker group, UGNazi, exploited a major flaw in Google's gmail password recovery process and AT&T's voicemail system which in turned allowed the group to access the CEO of CloudFare's personal gmail account [13]. The hacker deceived AT&T'S system into redirecting the victim's cell phone to a fraudulent voicemail box. The hacker visited gmail and initiated the account recovery feature for the victim's personal email address. A voicemail message was recorded on the compromised voicemail box to sound like someone was answering the phone. A call was placed to the victim from Google, but the victim did not recognize the number and let the call go to voicemail. Google's system was tricked by the fraudulent voicemail and a temporary PIN was left (which allowed the password to be reset) in the voicemail. The hacker logged into the victim's gmail account and added his email address to the 'account recovery control' feature.The victim's linked Cloudfare account received an email informing him that the recent password was changed. The victim initiated the account recovery process and changed the password back. An email is sent to the hacker informing him that the victim changed passwords, but immediately the hacker changed the password. Both users continue going back and forth to get control over the account. Soon, the hacker is able to remove the victim's mobile phone and email addresses authorized for account recovery preventing the victim from resetting the gmail password. The team at CloudFare is called to investigate the situation [13].

A flaw in Google's account recovery system allowed two-factor authentication setup on the victim's Cloudfare account to be bypassed and the hacker now had access to the account.

The victim's administrative privileges were used by the hacker to change passwords on other administrative accounts. Cloudfare's operations team suspended the victim's account, reset all CloudFare employee email passwords, and cleared all web mail sessions, which terminated the hacker's access to the email system [13]. Google fixed the flaw in the Google Enterprise Application account recovery process by no longer allowing a user to get around two-factor authentication. CloudFlare has stopped emailing blind copies of password resets and other transactional messages to administrative accounts [14]. Another case occurred in July 2012. Dropbox, the cloud storage service, confirmed that hackers used usernames and passwords stolen from third-party sites to access Dropbox users' accounts. It was altered after users complained about Spam they were receiving to email address used only for the Dropbox

accounts. One stolen password was used to access an employee account that contains a file that included user email addressed. The company believed users who use the same password on multiple websites make it easier for hackers to access their accounts on other websites [15].

4.5-Traffic Flooding:

Traffic flooding attacks bring a network or service down by flooding it with large amounts of traffic. Traffic flooding attacks occur when a network or service becomes so weighed down with packets initiating incomplete connection requests it cannot process genuine connection requests. Eventually, the host's memory buffer becomes full and no further connections can be made, and the result is a Denial of Service.

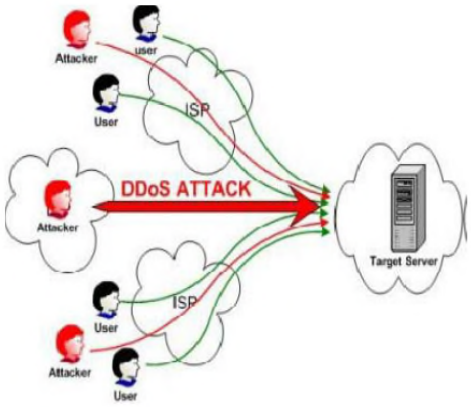


Figure1.2:DDoS Attack [16]

real-world cases:

In May 2011, LastPass, a cloud-based password storage and management company, announced a possible successful hack against its servers. There were no reports of any data leakage, but the company insisted that customer's take a few measures to ensure that their information is safe. Security experts discovered unusual behavior in the database servers that had more traffic going out compared to incoming data.

The company presumed this was hacking activity related to siphoning stored login credentials and other sensitive user data. Master passwords (passwords that protect lists of passwords to access other websites and online services in the cloud) were immediately changed to protect customers from possible data leakage [17].

4.6-Wireless Local Area Network Attack.

In a wireless local area network attack a hacker breaks into an authorized user's wireless local area network to perform attacks such as man-in-the-middle, accidental association, identify theft, denial of service, network injection attacks, etc.

real-world cases:

In January 2011, German security researcher Thomas Roth used cloud computing to crack wireless networks that relied on pre-shared passphrases, such as those found in homes and small businesses. The results of the attack revealed that wireless computing that relies on the pre-shared key system for protection is fundamentally insecure. Roth's program was run on Amazon's Elastic Cloud Computing (EC2) system. Using the massive power of Amazon's cloud the program was able to run through 400,000 possible passwords per second. It would typically cost tens of thousands of dollars to purchase the computers to run the program, but Roth claims that a typical password can be guessed by EC2 and his software in about six minutes [18]. The type of EC2 computers used in the attack costs $.28 cents per minute, so $1.68 is all it took to hack into a wireless network [6].

5-Flooding attacks based DoS/DDoS

A Denial of Service (DoS) attack involves, using one computer or internet connection to flood a server with packets (TCP/UDP). The objective of this attack is to 'overload' the server's bandwidth, and other resources, so that anyone who may be trying to get access to the server is not served, hence the term "denial of service"[19].See the figure bellow which shows general architecture of DDoS attacks.

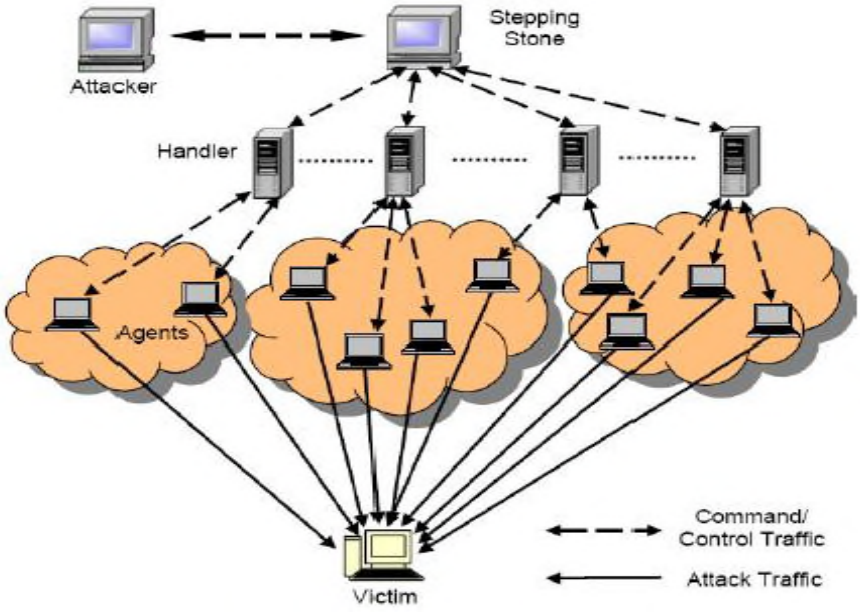


Figure 1.3: General architecture of DDoS attacks [19].

5.1 -Characteristics of DDoS Attack

Following are the different ways to characterize the distributed denial of service attack [20]:

1- Disruptive/Degrade Impact

After being a part of attack, the victim either to stop providing services to the client or the services are degraded that means some of the services are still being provided to the client even the victim's system is under the attack.

2-Exploiting Vulnerability

Network of machines which follows the instructions of master attacker to send request for a service on a victim's machine to consume its all the resources.

3- Dynamic Attack Rate

Sometime attacker make down the websites very quickly by sending large number of request more than its capacity, is known as constant attack rate. While sometimes attacker takes time to make it down by sending packets in variable length of request that is not constant, known as variable attack rate.

4- Automated Tools

Attackers can be classified by automated tools also and their skills. Attack can be performed manually; semi automated or fully automated tools

5.2 - DDoS Attacks Components

Figure 1.4 describes the component of DDoS attack[20],who initiates the attack by selecting vulnerable system as agents and further the agents use botnet to exhaust the victim's system.

1- Master Mind/Planner: The Original Attacker, who creates reasons and answers for, why, when, how and by whom the attack will be performed.

2- Controller/Handler: Coordinator of original attacker, who may be one or more than one machine, is used to exploit other machines to process DDoS attack .

3- Agents/Zombies/Botnets: Agents, also known as slaves or attack daemons, sub ordinates are programs that actually conduct the attack on the victim. These programs are usually deployed on host computers. These daemons influence both the machines: target and the host computers. It facilitates the attacker to gain access and infiltrate the host computers.

4- Victim/Target: A victim is a target host that has been selected to receive the impact of the attack.

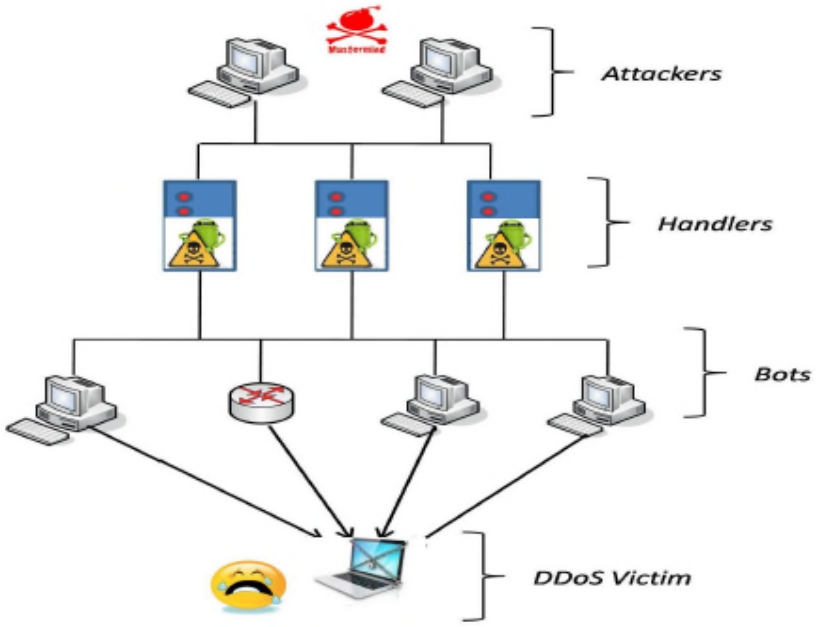


Figure 1.4 : Components of DDoS Attack [21]

5.3-Major form of flooding attack based DoS/DDoS

Distributed denial-of-service (DDoS) attacks pose a serious threat to network security. There have been a lot of methodologies and tools devised to detect DDoS attacks and reduce the damage they cause. Still, most of the methods cannot simultaneously achieve (1) efficient detection with a small number of false alarms and (2) real-time transfer of packets. The DDoS attacks can be classified into following three main categories:

-**Bandwidth Attacks** are intended to overflow and consume resources available to the victim (i.e., network bandwidth and equipment throughput). Examples of Bandwidth DDoS attacks are TCP SYN Flood, ICMP Flood and UDP Flood[19] which use IP spoofing (see figure bellow).

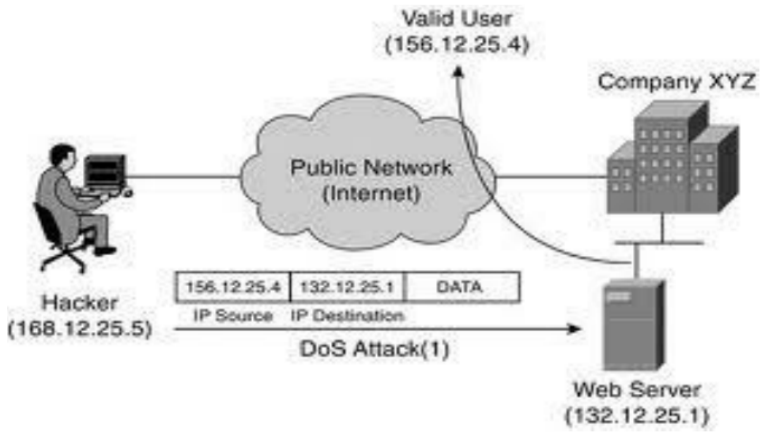


Figure 1.5 : DoS using IP Spoofing [23]

-**Protocol Attacks** take advantage of protocol inherent design (i.e., SMURF and DNS).

-Software Vulnerability Attacks attempt to exploit a software program design flaw (i.e., Land attack, Ping of Death, and Fragmentation).

Jelena Mirkovic, Janice Martin and Peter Reiher [24] have discussed detailed classification of DDoS attacks based on the Degree of Automation, Exploited Vulnerability, Attack Rate Dynamics and Impact. Some of the common DDoS attacks are discussed below:

5.3.1- SYN Flood Attack

A SYN flood occurs when a host sends a flood of TCP/SYN packets, often with a fake sender address. Each of these packets is handled like a connection request, causing the server to spawn a half-open connection, by sending back a TCP/SYN-ACK packet (Acknowledge), and waiting for a packet in response from the sender address (response to the ACK Packet). However, because the sender address is fake and the responses never come. These half-open connections saturate the number of available connections that the server is able to make, keeping it from responding to legitimate requests until after the attack ends [19].

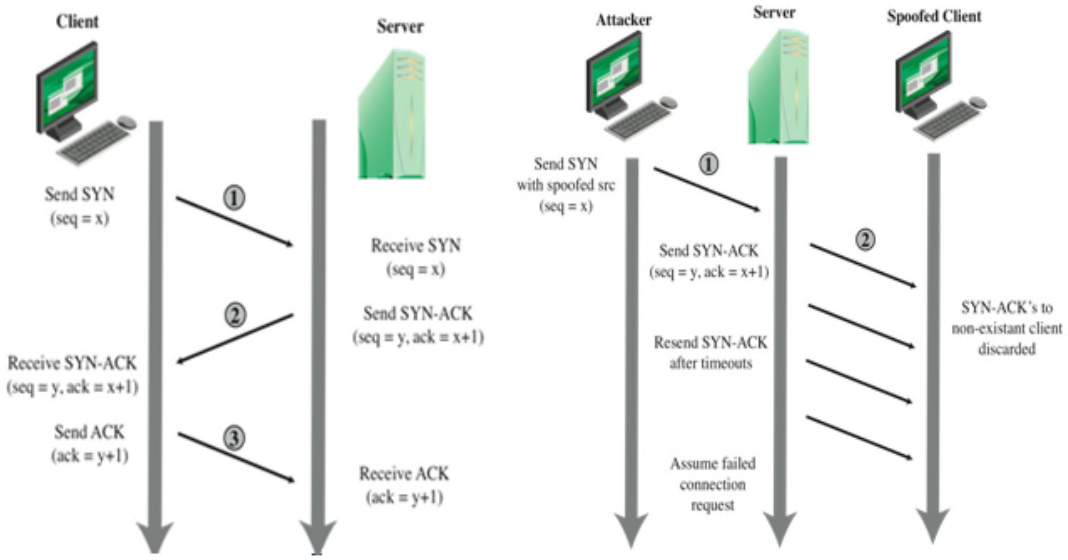


Figure 1.6 : (a),Package flow in three-way handshake , (b) TCP SYN attack [23]

5.3.2 - Smurf Attack

A smurf attack is one particular variant of a flooding DoS attack on the public Internet. It relies on erratically configured network devices that allow packets to be sent to all computer hosts on a particular network via the broadcast address of the network, rather than a specific machine. The network then serves as a smurf amplifier. In such an attack, the perpetrators will send large numbers of IP packets with the source address faked to appear to be the address of

the victim. The network's bandwidth is quickly used up, preventing legitimate packets from getting through to their destination [25].

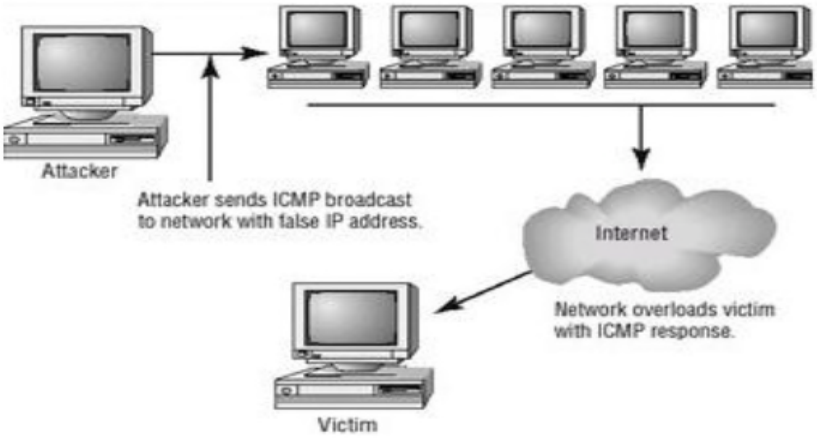


Figure 1.7 : ICMP Flood[19]

Like the other flooding attacks, this one is accomplished by broadcasting a bunch of ICMP packets, usually the ping packets. The idea is to send large amount of data to the system, so that it slows down so much and gets disconnected due to timeouts. Particularly, Ping flood attacks attempt to saturate a network by sending a continuous series of ICMP echo requests over a high-bandwidth connection to a target host on a lower bandwidth connection. The receiver must send back an ICMP echo reply for each request.

5.3.3 - Ping of Death

A ping of death involves sending a malformed or otherwise malicious ping to a computer. A ping is normally 32 bytes in size. Ping of death attack is caused by an attacker deliberately sending an IP packet larger than the 65,536 bytes allowed by the IP protocol. Many operating systems don’t know what to do when they receive an oversized packet, so they freeze, crash or reboot. Ping of death attacks were particularly nasty because the identity of the attacker sending the oversized packet could be easily spoofed and because the attacker didn't need to know anything about the machine they were attacking except for its IP address. By the end of 1997, operating system vendors had made patches available to avoid the ping of death. Many new variants of ping of death include jolt, sPING, ICMP bug, Ice Newk, Ping of Death[26]. However most modern day firewalls are capable of filtering such oversized packets[19].

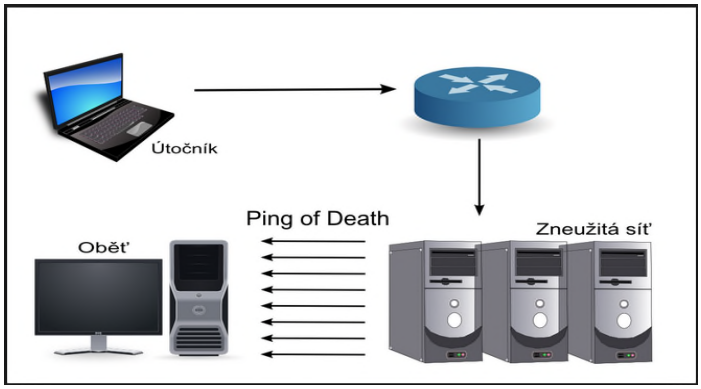


Figure 1.8 : Ping of Death

5.3.4 - Land Attack

A LAND attack consists of a stream of TCP SYN packets that have the source IP address and TCP port number set to the same value as the destination address and port number (i.e., that of the attacked host). Some implementations of TCP/IP cannot handle this theoretically impossible condition, causing the operating system to go into a loop as it tries to resolve repeated connections to itself. Service providers can block LAND attacks that originate behind aggregation points by installing filters on the ingress ports of their edge routers to check the source IP addresses of all incoming packets. If the address is within the range of advertised prefixes, the packet is forwarded; otherwise it is dropped [19].

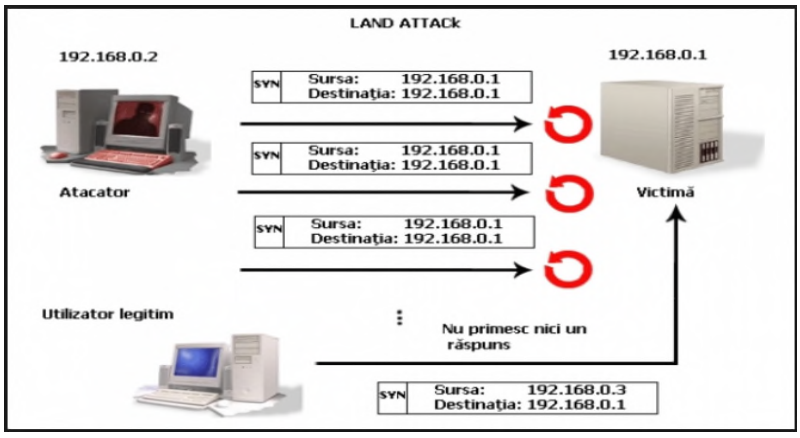


Figure 1.9 : Land Attack

5.3.5 - Teardrop

The Teardrop, though, is an old attack that relies on poor TCP/IP implementation that is still around. It works by interfering with how stacks reassemble IP packet fragments. The trick here is that as IP packets are sometimes broken up into smaller chunks, each fragment still has the original IP packet's header, and field that tell the TCP/IP stack what bytes it contains. When it works right, this information is used to put the packet back together again. What happens with Teardrop though is that your stack is buried with IP fragments that have

overlapping fields. When the stack tries to reassemble them, it cannot do it, and if it does not know to toss these trash packet fragments out, it can quickly fail. Most systems know how to deal with Teardrops and a firewall can block Teardrop packets in return for a bit more latency on network connections since this makes it disregard all broken packets. Of course, if you throw a ton of Teardrop busted packets at a system, it can still crash. Many other variants, such as Targa , SynDrop, Boink, Nestea Bonk, TearDrop2 and New Tear are available to accomplish this kind of attack[19].

5.3.6- Ping flood

A ping flood is the most basic form of DOS. The attacker simply sends a huge number of ping packets to the target. If the target sends replies, the effect is amplified [27].

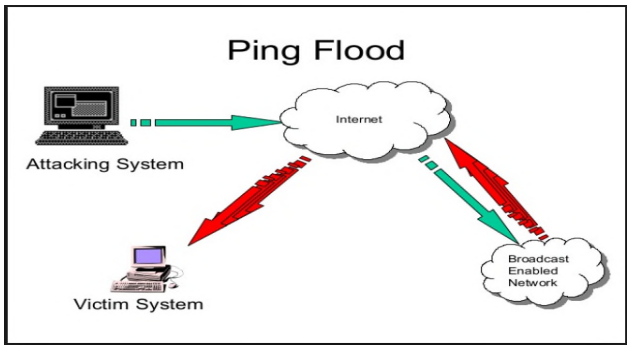


Figure 1.10 : Ping Flood

5.3.7- UDP flood

In a UDP flood attack, the attacker sends a large number of UDP packets to random ports on the target. As the UDP does not have a congestion control system, the attacker can potentially send a very large number of packets. This attack is generally used with IP address spoofing, so that the attacker can stay away from detection [27].

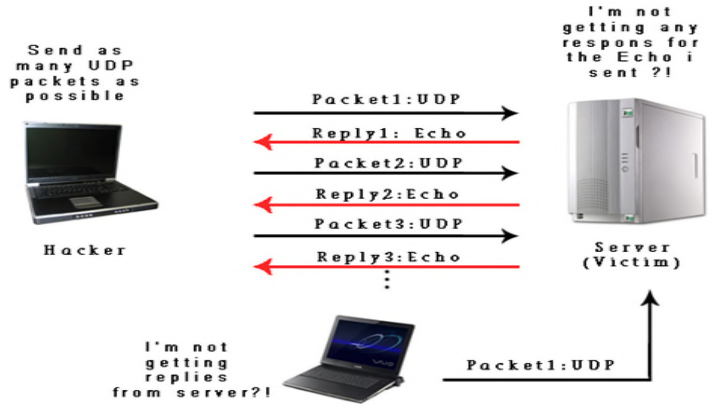


Figure 1.11: UDP Flooding Attacks

5.3.8- Application-level attack

The kinds of attack argued thus far all exploit network protocols or services. DOS attacks can also be accepted at the application level. For example, the attacker can command zombies to send HTTP requests to a web server to download a large file or execute expensive database operations. This will consume CPU and network resources at the server, limiting its availability to other legitimate clients [27].

Figure 1.12 : HTTP Get Flood

5.3.9- DNS amplification attack.

DNS amplification attack uses DNS queries . The size of the reply to a DNS query can be much larger than the DNS query. The attacker creates a reliable domain name server, “chance.com”, and registers a garbage text of large size, for example 5000 bytes, as the text Resource Record (RR) of chance .com. Next, the attacker commands zombies to send queries to their domain name servers for the text RR of chance.com, with the zombies’ IP address which is spoofed to be the victim’s IP address. When the domain name servers that receive queries allow recursion, they recursively query the reliable name server of chance .com for its text RR and get the reply to the source IP address, which is the address of the victim [27].

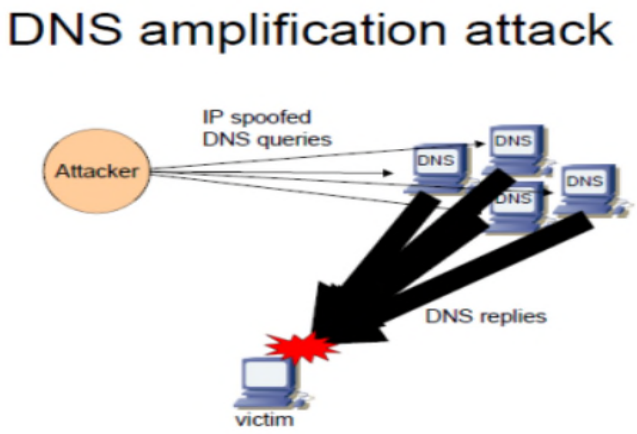


Figure 1.13 : DNS Amplification Attack

5.3.10 - Peer-to-peer attack.

Conventional DDOS attacks use zombie computers to send a large number of requests to the victim.P2P attacks use clients linked to P2P file sharing hubs[27].

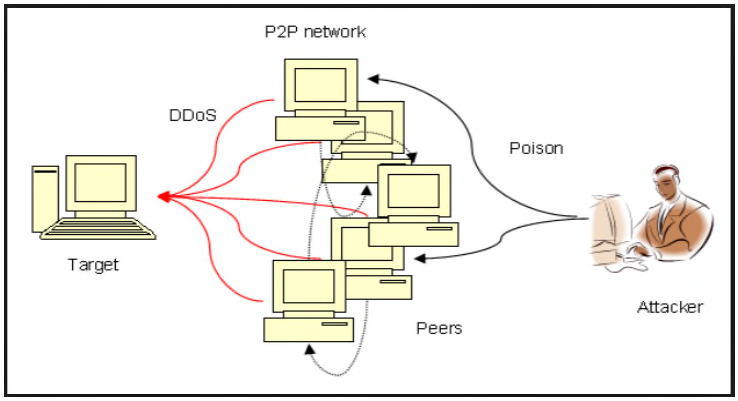


Figure 1.14 : Peer-to-Peer Attack

5.3.11- Mail bomb attack.

In a mail bomb attack, the attacker sends a large amount of e-mails to a target e-mail address to overflow the victim's mailbox or slow down the mail server. The attacker may command zombies to send e-mails to the victim e-mail address simultaneously. Attacker may create each e-mail with a different message to pass the spam filters [27].

5.3.12- Variable-rate and low-rate attacks

Although it is difficult to trace back the attack, in general it is simple to know when the attack actually takes place, because the server becomes unavailable or drastically slows down. The monitoring system at the target system raises an alarm when there is an unusually large volume of traffic at a constant rate. However, the attacker may send variable-rate and low-rate traffic to the victim, making it complex for the victim to understand that actually an attack is taking place. And if the attack is not detected, the administrators may incorrectly conclude that legitimate traffic has increased and increase investment in network bandwidth [27].

5.4-History of DDoS Attacks

5.4.1-Analytical Study of DDoS Attacks

A long run way which has no end point of attacks can be seen even in advanced technical society. To develop defense mechanism, behavior of attack can be analyzed, which leads to the categorization of DDoS attack.

Practical Unix and Internet Security [28], the "bible" for many system administrators of the early commercial web, offers a chapter on denial of service attacks. Carne- gie Mellon's Computer Emergency Response Team.

2013	The Czech financial sector was targeted in cyber attacks on Wednesday, at the same time on the national bank and stock exchange websites which get disrupted by dedicated denial of service (DDoS) attacks—London, 8 March, 2013.
2012	US and UK Government Sites Knocked Down by Anonymous—April 16, 2012. DDoS Attack Impacts Canadian Political Party Elections—March 24, 2012.
2011	A DDoS attack on Sony was used—April 16-20 2011.
2010	PayPal Transaction is suspended over WikiLeaks website after attacked by DDoS—December 3-5, 2010.
2009	The Mydoom virus code was re-used to launch DDoS flooding attacks against major government news media and financial websites in South Korea and the United States in July 2009 [29].
2008	BBC hit by DDoS Attack, two DDoS attacks on Amazon.com and eBuy.
2007	Estonia Cyber Attack [30].
2006	US Banks have been targeted for financial gain.
2004	SCO Group website inaccessible to legitimate users.
2003	Mydoom defiled thousands of victims to attack SCO and Microsoft [31].
2002	13 root servers that provide the Domain Name System (DNS) service to Internet users around the world shut down for an hour because of a DDoS flooding attack [32].
2001	First major attack involving DNS servers as reflectors. The target was Register.com. The Irish Government's Department of Finance server was hit by a denial of service attack carried out as part of a student campaign from NUI Maynooth.
2000	Yahoo! Experienced one of the first major DDoS flooding attacks that kept the company's services off the Internet for about 2 hours incurring a significant loss in advertising revenue [33].

Table 1.1 : DDoS attack statistics [20].

5.5-Cloud Computing Attacks in virtualization

In cloud computing where infrastructure is shared by large number of VM clients and if cloud has not sufficient resource to provide services to its VMs then maybe cause undesirable DDoS attacks such as *ARP* spoofing at the network layer. See figure bellow which means Client to client attacks in the cloud environment that are major security risk [34].

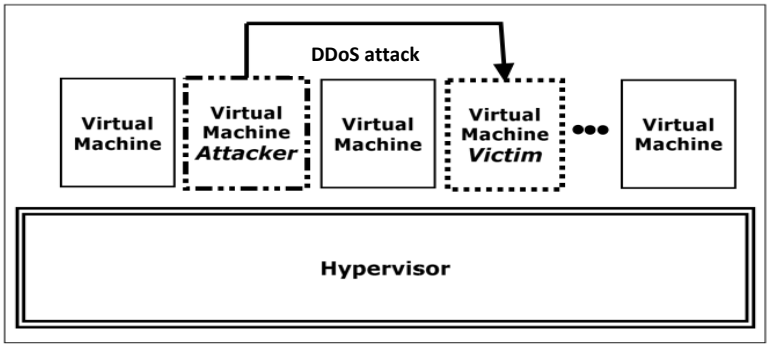


Figure 1.15 : Attack scenario within cloud [34].

5-Conclusion

Cloud computing security involves different areas and issues. Various security terms ,in cloud computing systems ,have been studied . Several real world cases where companies' clouds were infiltrated by attacks are presented .Social engineering attack, XML signature wrapping attack, malware injection, data manipulation , account hijacking, SYN flood, and wireless local area network attack are discussed. Major form of flooding attack based DoS/DDoS are presented which is the greatest and the dangerous attack in the cloud computing systems .

We are going to present some solutions (approaches) for the precedent attacks (flooding attacks based DoS /DDoS) in the next chapter of state of the art.

Chapter II

State of the Art of Flooding Attacks based DoS/DDoS

1-Intrduction

The flooding attacks based DoS/DDoS are considered such as a new form of attack in the cloud environment .They can shutdown the cloud servers and deny the legitimate user access to the cloud services .Those attacks are not detected by intrusion detection systems (IDS) because they generate a great quantity of packets by employing IP spoofing in the majority of cases.The detection schemes for those attacks have been classified broadly into three categories – *detection schemes based on the router data structure, detection schemes based on statistical analysis* of the packet flow and *detection schemes based on artificial intelligence* such as neural network (as shown in the following figure) .

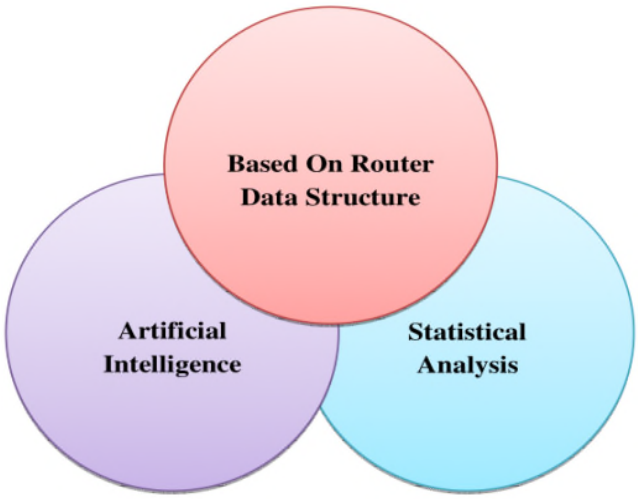


Figure 2.1: DoS Attacks classification schemes [47]

Several works have being applied for detecting the flooding attacks based DoS/DDoS by using statistical analysis ,but they can't find those attacks within high detection accuracy ,and low false positive and false negative rates.

We are going in the following section to present few algorithms (approaches) for detecting the flooding attacks based DoS/DDoS .

2-State of the art about flooding attacks based DoS/DDoS

Several studies are interested in the problem of flooding attacks such as TCP/SYN flooding ,UDP flooding or ICMP flooding which are employing IP spoofing in the majority of cases.

In this chapter ,we are going to present some algorithms for detecting flooding attacks by using *detection schemes based on statistical* analysis of the packet flow and ending by giving a comparative table in term of percentage accuracy and false positive (FP) , and false negative (FN) rates ,if they exist.

2.1: Hop-Count Filtering : An Effective Defense Against Spoofed DDoS Traffic

In this victim-based approach[38] the author proposed an effective defense against spoofed DDoS attack that limit and block legitimate users' access by exhausting victim servers' resources and saturating stub networks' access links to the Internet by using IP spoofing to conceal flooding sources and localities in flooding traffic (see the following figure). Attackers often spoof IP addresses by randomizing the 32-bit source-address field in the IP header.

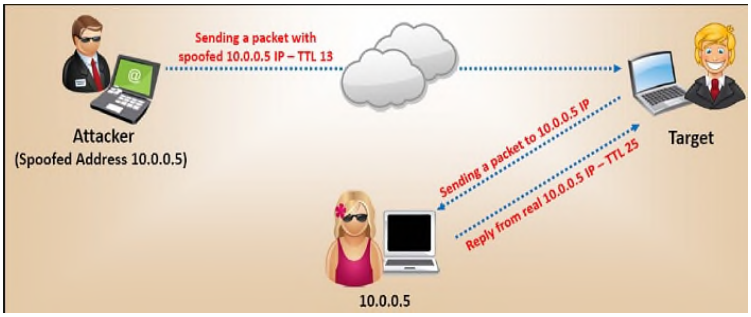


Figure 2.2: IP spoofing

For this ,the author has utilized Hop-Count information ,which is the difference between the initial TTL at the source and the final TTL value at the destination (as shown in the figure 2.4 below) .

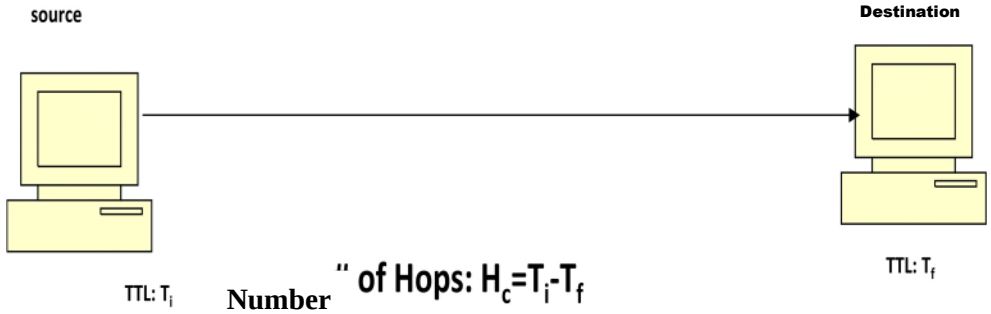


Figure 2.3: Hop-Count Computation

Because an attacker can forge any field in the IP header, but cannot falsify the number of hops (as shown in the figure 2.3 below) an IP packet takes to reach its destination or cannot sabotage routers to alter TTL values of IP packets that traverse them.

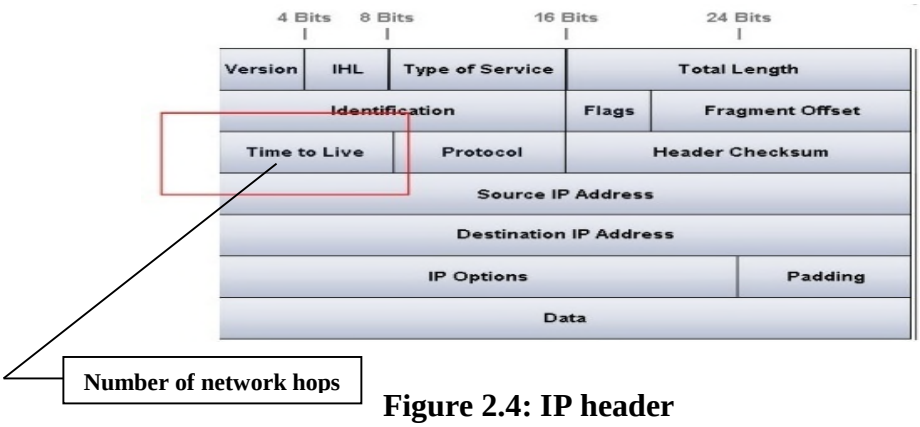


Figure 2.4: IP header

1-An example of Hop-Count computation

Most modern operating systems use only a few selected initial TTL values such as 30, 32, 60, 64, 128 and 255[40] .If the final value is 112, the initial TTL value is 128 , then the hop-count is 16 = 128 – 112.

In addition , Hop-Count Filtering (HCF) builds an accurate IP-to-hop-count (IP2HC) mapping table to capture hop-count changes.

2-Basic Organization of Hop-Count Filter

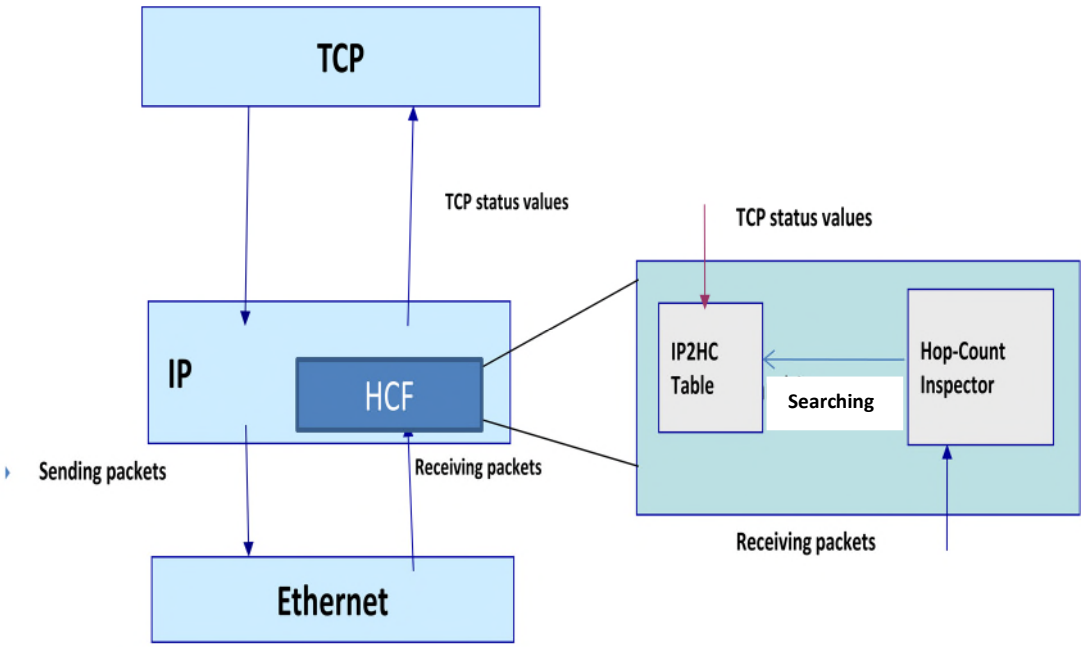


Figure 2.5: Basic Organization of Hop-Count Filter

By using Hop-Count Inspection Algorithm which extracts the source IP address and the final TTL value from each IP packet, then infers the initial TTL value and subtracts the final TTL value from it to obtain the hop-count. The source IP address serves as the index into the table to retrieve the correct hop-count for this IP address .The both are implemented in the IP layer as shown in the figure 2.5 above.

3-Diagram of Inspection Algorithm Hop-Count

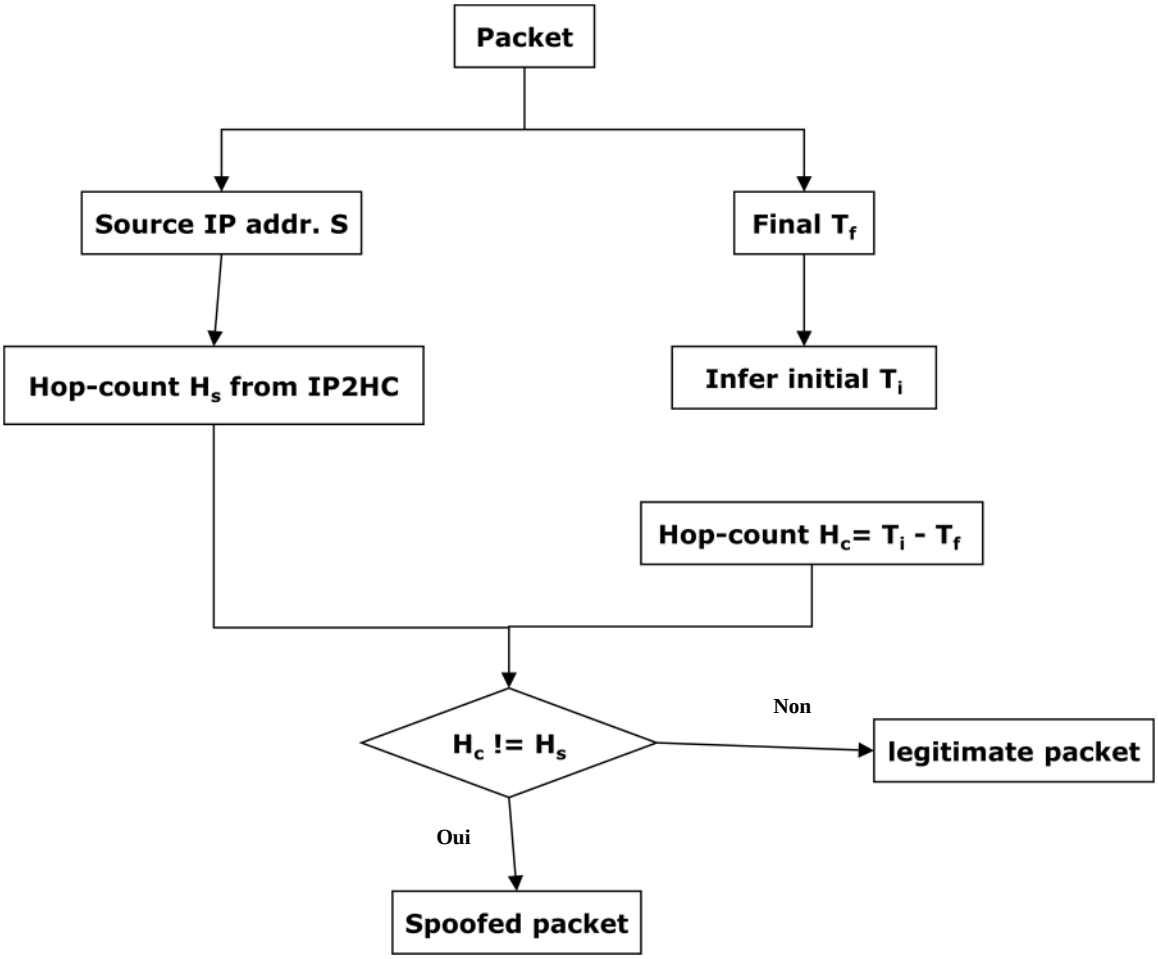


Figure 2.6: Diagram of Inspection Algorithm Hop-Count

The precedent diagram is illustrated by the following algorithm

```
for each packet:
  extract the final TTL T_f and the source IP address S;
  infer the initial TTL T_i;
  compute the hop-count H_c = T_i − T_f;
  index S to get the stored hop-count H_s;
  if   (H_c ≠ H_s)
       the packet is spoofed;
  else
       the packet is legitimate;
```

Figure 2.7: Hop-Count Inspection Algorithm

4-Approach Mechanism

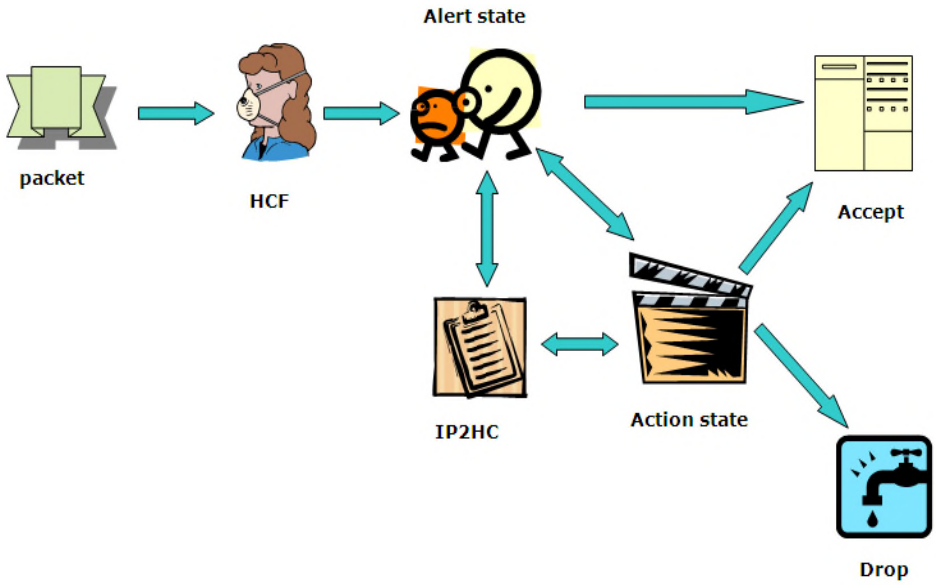


Figure 2.8 : Hop Count Filtering Mechanism

This mechanism contains tow running states, alert and action :In the first the HCF detects the presence of spoofed IP (See the following figure)by applying the following operations :

1-Sample incoming packets for hop-count inspection.

2- Calculate the spoofed packet counter

3- Update the IP2HC mapping table in case of legitimate hop-count changes.

```
In alert state:
  for each sampled packet p:
     spoof = IP2HC_Inspect(p);
     t = Average(spoof);
     if ( spoof )
                if ( t > T1 )
                           Switch_Action();
     Accept(p);

  for the k-th TCP control block tcb:
     Update_Table(tcb);
```

Figure 2.9:Alert state

,but in the second state ,it discards spoofed packets that are mismatching hop-counters by applying the following operations :

1- Performs per-packet hop-count inspection and discards spoofed packets, if any

2- Examine every packet

3- Discards spoofed packets

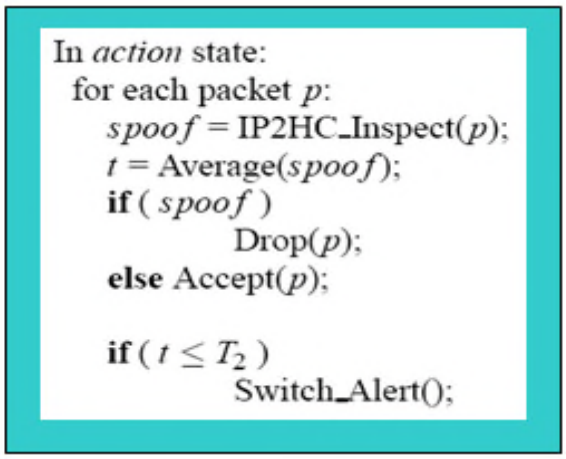


Figure 2.10: Action state

This algorithm detects flooding attacks within a percentage of 90% ,and false positive(FP) which equals to 10% and false negative (FN) which equals to 5%.

5-Critical Section :

1-This victim-based approach cannot recognize forged packets whose source IP addresses have the same hop-count value as that of a zombie i.e it can't recognize forged packets whose source IP addresses have the same hop-count. For this issue ,we must add a count for computing the number of packets within the same source IP address during a slot time.

2-In this victim-based approach , the author did not study the case where there are multiple routes between the sender and the destination ,which results in multiple allowed HC values : It means the author must add these filtering types :strict filtering, +1 filtering and +2 filtering. Strict filtering drops a packet when the HC differs from the HC profile.The +1 filtering drops a packet when the HC differs by more than one from the HC profile. Finally, +2 filtering drops a packet when the HC differs by more than two from the HC profile. All this for solving the issue of existence of NAT (Network Address Translator) boxes, each of which may connect multiple stub networks, could make a single IP address appear to have multiple valid hop-counts at the same time.

3-The author did not study the issue of computation time and updates of IP to Hop count by using packets from established TCP connections (SYN flag) ensures that an attacker cannot slowly pollute a IP2HC table by spoofing source IP addresses.

4-In a case of massive legitimate network accessing the problem of huge network access resulted false positive alarms i.e DDoS is also detectable ,but really it is not a real DDoS attack. For solving this issue ,the author must classify every IP packet like spoofed when its number is greater than a predefined value during a slot-time.

2.2:Defense Against Spoofed IP Traffic Using Hop-Count Filtering

In this victim-based approach [39] the author proposed a technique which matches the precedent technique [38] within the same critical technique .

2.3: Packet Monitoring Approach to Prevent DDoS Attack in Cloud Computing

In this approach [41]the author has added another feature which is called SYN flag which is systematically used in the three-way handshake TCP connection. It can solve the issue of computation time and updates of IP to Hop-count that can immediately saturate the IP2HC table.

Algorithm :

Considering the following notations :

synflag = Syn bit of TCP packet.

mcount =malicious packet counter ,

T_f= final value of TTL.

T_i=initial value of TTL.

```
Initialize mcount=0;
 For each packet
 Set TTL = ExtractFinalValueOfTTL( );
       //get time-to-leave field of IP packet
 Set srcIp = ExtractSourceIP( );
       //get source IP address from IP packet
Set synflag = ExtractSynBit( );
       //get Syn flag value from TCP packet
If (synflag is set)
 {
   If (establish_tcp_connection( ))
         //true when connection established
  {
      If ( srcIp is exist in IP2HC table )
     {

        ComputePacket ( srcIp , TTL , synflag);
          // function call which filter the spoofed
packet
      }
      else //new connection packet
      {
        Hc=ComputeHopCount( TTL );
          //get hop-count value
        NewEntryInTable(srcIp,Hc);
          //Add entry into IP2HC table
      }
   }
   else
   {
        // ignore packet
    }
 }
 else //synflag is not set
 {

    If ( srcIp exist in IP2HC Table)
   {
      ComputePacket ( srcIp , TTL, synflag );
// function call which filter the spoofed packet
    }
    else
    {
      'drop the packet' //Packet is spoofed
mcount++;  // increment in malicious  packet by 1
else
    {
       'drop the packet' //Packet is spoofed
mcount++;  // increment in malicious  packet by1


}
 }
 ComputePacket ( string srcIp , int Tf , boolean
synflag)
 {
    Hc=ComputeHopCount( Tf ); //get hop-count    value

    Hs=RetreiveStoredHopCount(srcIp);
       //get stored hop-count value
    If ( Hc != Hs )
    {
       if( synflag is set)
       {
 UpdateTable ( srcIp , Hc);
   //update hop-count value in IP2HC

     table
       }
       else
       {
 'drop the packet' //Packet is spoofed
  mcount++;
   // increment in malicious packet by 1

        }

    }

    else

    {

       'allow the packet' // packet is legitimate
     }
 }
 int ComputeHopCount( int Tf )
 {
    Set Ti= InvestigateInitialTTL(Tf);
    return Ti - Tf; //return hop-count value
}
```

Critical Section :

In this approach ,only the third issue has been solved ,but the others stay like they are in the precedent approach [38] .

2.4:Hop Count Based Packet Processing Approach to Counter DDoS Attacks

In this approach [42] the author proposed a technique to detect IP spoofing by checking both the HC and the path identification (PID) at every router. The PID is inserted in each IP packet in the identification field (See the figure bellow).

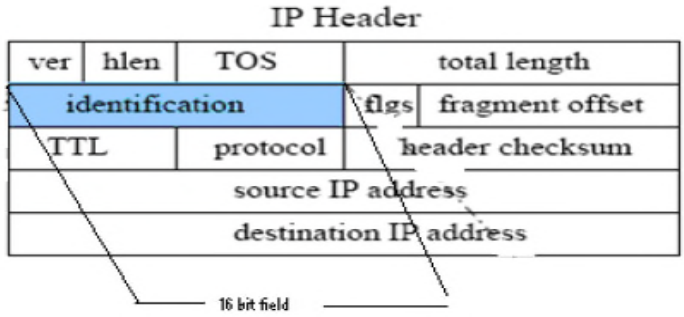


Figure 2.11 : IPV4 header with identification field

The author algorithm consists of four modules to counter DDoS attacks as follows :

1-Packet Marking 2-PacketAnalysing 3-Attack path construction 4- Packet Filtering

1. Packet Marking Algorithm

Each packet traveling along the router path is marked with some path Identification (PID).The Hop-count is computed based on the 8-bit TTL filed of IP header. The packets from the systems at the same hop count are marked with the same identification number (PID) which is derived from the concatenation of the hash value of the 32 bits of the IP address of the router path and the encrypted value of the hop count.

2. Packet Analyzing Algorithm

```
for each packet
        extract the final TTL (Time To Live) Tf
        and source IPaddress S;
    Let initial TTL be Ti;
        Compute the HC value as Ti-Tf
        Stored Hop count be Hs
        Stored PID be PIDs
        If (Hc==Hs)
            If (PID==PIDs)
            Packet is legitimate
else
        Packet is spoofed
        Router starts attack detection process
        Sends request to all router interfaces
        Attack path is constructed
        All packets from attack source dropped
end
```

Figure 2.12 : Packet Analyzing Algorithm

3. Attack Path Construction

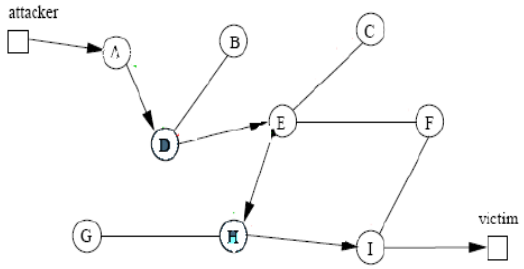


Figure 2.13 : Attack Path Construction

The source of attack packets can be easily identified with this technique. Packets are filtered nearer to attack sources. The attack path is also constructed to protect the victim before an actual attack occurs.

4. Packet filtering

Packet filtering is executed by each router and filtering is done close to attack sources. It is possible with the help of our packet identification approach to filter the packets just after receiving a single packet. The filtering process is executed with the help of the IP to hop count value table maintained at the router, if both the HC and PID match, then the packet is considered legitimate. Otherwise, the routers start an attack-detection process. The algorithm requires a shared key between every pair of adjacent routers.

At the receiving side of the router interface the hop count value of the incoming packet is checked with the already stored value in the IP2HC.If the values are equal , the packet is legitimate , otherwise , the routers start the attack detection process.

This technique provides an advantage of immediately filtering the traffic after receiving just one attack packet and it does not require any change in the existing protocols. Thus this technique has a significant potential in reducing the threats caused by the DDoS attacks.

Critical Section :

1-In this approach , the author did not study the case where there are multiple routes between the sender and destination which results in multiple allowed HC values : It means the author must add these filtering types :strict filtering, +1 filtering and +2 filtering. Strict filtering drops a packet when the HC differs from the HC profile. The +1 filtering drops a packet when the HC differs by more than one from the HC profile. Finally, +2 filtering drops a packet when the HC differs by more than two from the HC profile. All this for solving the issue of existence of NAT (Network Address Translator) boxes, each of which may connect multiple stub networks, could make a single IP address appear to have multiple valid hop-counts at the same time.

2- The author did not study the issue of computation time and updates of IP to Hop count by using packets from established TCP connections (SYN flag) ensures that an attacker cannot slowly pollute a IP2HC table by spoofing source IP addresses.

2.5:Mitigating DDoS attack and Saving Computational Time using a probabilistic approach and HCF method

In this approach [43] the author presented a probabilistic theoretical model based on HCF(see the figure bellow).

Algorithm 1: Boolean *HCF(packet)*

```
for each packet:
    extract the final TTL T and IP address S;
    infer the initial TTL T0;
    compute the hop count Hc=T-T0;
    index S to get the stored hop count Hs;
    if(Hc ≠ Hs)
        packet is spoofed;
    else
        packet is legitimate;
```

Figure 2.14 :Hop Count Filtering Algorithm (Algorithm 1)

So another new probabilistic model is necessary for detecting uncertain DDoS attacks or the number of malicious packets. Suppose each packet arrives at the server being malicious with probability 'p' or non-malicious with probability '1-p'.So the joint probability of 'n' packets among the total traffic are malicious and 'm' packets among the traffic are non-malicious is as follows : Let N1 denote the number of malicious packets, N2 denote the number of non-malicious packets. Also let N=N1 +N2 be the total number of traffic arrive at the server .Now conditioning on N gives

$$P\{N1=n,N2=m\}=\sum_{i=0}^{\infty}P\{N1=n,N2=m|N=i\}P\{N=i\}$$

Because $P\{N1=n,N2=m|N=i\}=0$ when $i=m+n$ the preceding equation yields that,

$$P\{N1=n,N2=m\} = P\{N1=n,N2=m|N=n+m\}(\lambda^{n+m}/n+m)$$

Probability of 'n' packets to be malicious is,

$$P\{N1=n,N2=m\} = ({}^{n+m}C_n\, p^n(1-p)^m)e^{-\lambda}(\lambda^{n+m}/n+m)$$

$$=((m+n)!/m!n!)\, p^n(1-p)^m e^{-\lambda}(\lambda^{n+m}/n+m)$$

$$=((m+n)!/m!n!)\, p^n(1-p)^m e^{-\lambda p}e^{-\lambda(1-p)}(\lambda^{n+m}/n+m)$$

$$=e^{-\lambda p}(\lambda p^n/n!)\, e^{-\lambda(1-p)}(\lambda(1-p)^m/m!)$$

Because the precedent joint probability mass function factors in two products ,one of which depends upon ‘n’ So

$$P\{N1=n\}=\sum_{m=0}^{\infty} P\{N1=n,N2=m\}$$
$$=e^{-\lambda p}(\lambda p^{n}/n!)$$

This derivation shows the probabilities of 'n' packets among the total packets with poisson's distribution of arrival rate 'X' are malicious.

After calculating the probability of some number of packets being malicious we can filter out that many number of packets from the given number of packets. After checking each packet when we will reach at the desired number of packets, we will let other packets to enter the server. The checking operation can be done using Hop Count method. By application of this method ,we can say that the execution time and the memory will been diminished .See the figure bellow .

Algorithm 2: ***proposed_algorithm***

```
for given ‘λ’, ‘p’, ‘m+n’
calculate ‘n’ such that P(n)=1;
integer count=0;
for each value of count upto ‘n’
  for each packet ‘i’
     if (count ≠ n)
        status= HCF(i);
      if status is ‘spoofed’
            count++;
             drop the packet;
     else
           allow the packet;
     end if;
   end if;
  allow all the rest packets upto m+n
```

Figure 2.15: Proposed algorithm for mitigation of DDoS (Algorithm 2)

The author algorithm says, for the given values of poisson's arrival rate of packets 'λ ',error probability of each packet 'p' and a definite value of 'm+n' , we can calculate the value of 'n' by specifying the probability of 'n' to 1. This value of 'n' can be calculated from the developed probabilistic equation .we examine each and every packet reaching to the server to calculate the hop count and to identify the maliciousness. Whenever we find one malicious packet we are increasing the count value so that the value of count will go maximum upto 'n' whenever the value of count will reach 'n', we will simply allow other packets upto 'm+n' without checking them.

But ,combined approaches which is installed in server level (target server)for weeding out within detection accuracy between **80% et 85 %.**

Critical Section :

1-In this approach , the author did not study the case where there are multiple routes between the sender and the destination which results in multiple allowed HC values : It means the author must add these filtering types :strict filtering, +1 filtering and +2 filtering. Strict filtering drops a packet when the HC differs from the HC profile. The +1 filtering drops a packet when the HC differs by more than one from the HC profile. Finally, +2 filtering drops a packet when the HC differs by more than two from the HC profile. All this for solving the issue of existence of NAT (Network Address Translator) boxes, each of which may connect multiple stub networks, could make a single IP address appear to have multiple valid hop-count sat the same time.

2- The author did not study the issue of computation time and updates of IP to Hop count by using packets from established TCP connections (SYN flag) ensures that an attacker cannot slowly pollute a IP2HC table by spoofing source IP addresses.

3-The value of count for counting the number of packets for the same source IP address must be during a slot time.

2.6 : Detection and Defense Against DDoS Attack with IP Spoofing

This approach [44] contains two execution steps: The first ,it's a learning state and the second it's the filtering state.In the learning state , the packets aren't weed out ,but it inspects the spoofed packets and detect the attack. If the attack is detected by the learning state , so the HCF goes to Filtering state where all spoofed packets will be weed out.

For changing from step to another step , a corrected threshold must be decided .While we calculate the spoofed packet mean during a slot time **t** ,if it's greater than a threshold value TL (Thresold of Learning) which depends on server's workload, so the HCF changes to filtering state and if the spoofed packet mean is under the threshold value TF(Thresold of Filtering) then the HCF come back to the learning state (See the figure bellow).

This combined approach can weed out the spoofed packets within **0 %** of false positive and **great rate** of false negative.

*L*earning State:
f or number of IP packets;
spoof=Inspect IP2HC(IP addr);
spoof count =Average(spoof);
*i*f (spoof pkts > threshold T_L)
*t*hen switch to filtering state
*e*lse learning state

*F*iltering State:
f or N number of IP packets;
spoof=Inspect IP2HC(IP addr);
spoof count =Average(spoof);
*i*f (spoof pkts > threshold T_F)

Figure 2.16 :Algorithm Steps

See bellow , Hop count values of different IP addresses ,

```
3) ip->saddr = 62.216.128.141   final_ttl = 239 hop_count = 17
4) ip->saddr = 4.68.124.246     final_ttl = 244 hop_count = 12
5) ip->saddr = 64.159.1.34      final_ttl = 245 hop_count = 11
6) ip->saddr = 63.210.253.146   final_ttl = 243 hop_count = 13
7) ip->saddr = 64.159.0.161     final_ttl = 244 hop_count = 12
8) ip->saddr = 220.224.140.29   final_ttl = 248 hop_count = 8
9) ip->saddr = 62.216.128.141   final_ttl = 239 hop_count = 17
10) ip->saddr = 212.187.151.177 final_ttl = 240 hop_count = 16
11) ip->saddr = 4.68.116.1      final_ttl = 243 hop_count = 13
12) ip->saddr = 64.159.1.34     final_ttl = 245 hop_count = 11
13) ip->saddr = 4.68.124.246    final_ttl = 244 hop_count = 12
14) ip->saddr = 203.199.141.99  final_ttl = 255 hop_count = 1
15) ip->saddr = 209.85.139.18   final_ttl = 64  hop_count = 0
16) ip->saddr = 220.224.140.29  final_ttl = 248 hop_count = 8
17) ip->saddr = 63.210.253.146  final_ttl = 243 hop_count = 13
18) ip->saddr = 64.159.0.161    final_ttl = 244 hop_count = 12
19) ip->saddr = 62.216.128.141  final_ttl = 239 hop_count = 17
20) ip->saddr = 212.187.151.177 final_ttl = 240 hop_count = 16
21) ip->saddr = 4.68.124.246    final_ttl = 244 hop_count = 12
22) ip->saddr = 203.199.141.99  final_ttl = 255 hop_count = 1
23) ip->saddr = 4.68.116.1      final_ttl = 243 hop_count = 13
24) ip->saddr = 64.159.0.161    final_ttl = 244 hop_count = 12
25) ip->saddr = 64.159.1.34     final_ttl = 245 hop_count = 11
26) ip->saddr = 63.210.253.146  final_ttl = 243 hop_count = 13
27) ip->saddr = 220.224.140.29  final_ttl = 248 hop_count = 8
28) ip->saddr = 62.216.128.141  final_ttl = 239 hop_count = 17
29) ip->saddr = 4.68.124.246    final_ttl = 244 hop_count = 12
30) ip->saddr = 212.187.151.177 final_ttl = 240 hop_count = 16
31) ip->saddr = 203.199.141.99  final_ttl = 255 hop_count = 1
32) ip->saddr = 64.159.1.34     final_ttl = 245 hop_count = 11
33) ip->saddr = 4.68.116.1      final_ttl = 243 hop_count = 13
34) ip->saddr = 63.210.253.146  final_ttl = 243 hop_count = 13
35) ip->saddr = 220.224.140.29  final_ttl = 248 hop_count = 8
36) ip->saddr = 62.216.128.141  final_ttl = 239 hop_count = 17
37) ip->saddr = 212.187.151.177 final_ttl = 240 hop_count = 16
38) ip->saddr = 64.159.1.34     final_ttl = 245 hop_count = 11
39) ip->saddr = 64.159.0.161    final_ttl = 244 hop_count = 12
40) ip->saddr = 4.68.124.246    final_ttl = 244 hop_count = 12
41) ip->saddr = 63.210.253.146  final_ttl = 243 hop_count = 13
```

Figure 2.17 : Hop count values of different IP addresses

Critical Section :

1-In this victim-based approach , the author did not study the case where there are multiple routes between the sender and the destination which results in multiple allowed HC values : It means the author must add these filtering types :strict filtering, +1 filtering and +2 filtering. Strict filtering drops a packet when the HC differs from the HC profile. The +1 filtering drops a packet when the HC differs by more than one from the HC profile. Finally, +2 filtering drops a packet when the HC differs by more than two from the HC profile. All this for solving

the issue of existence of NAT (Network Address Translator) boxes, each of which may connect multiple stub networks, could make a single IP address appear to have multiple valid hop-counts at the same time.

2- The author did not study the issue of computation time and updates of IP to Hop count by using packets from established TCP connections (SYN flag) ensures that an attacker cannot slowly pollute a IP2HC table by spoofing source IP addresses.

2.7 : A Three Layer Defence Mechanism based web Servers against Distributed Denial of Service Attacks.

This mechanism [45] is based on three layers ; network ,transport and application (see the following figure).

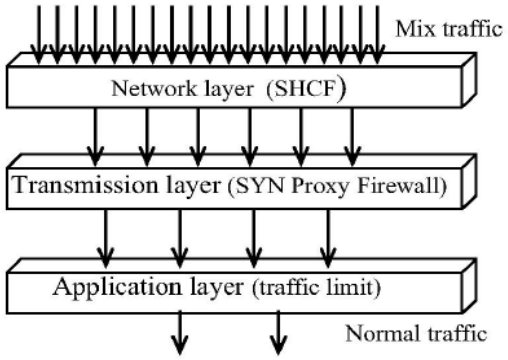


Figure 2.18 : the three-layer defense mechanism

Thus the majority of spoofed traffic (spoofed packets : if the difference in term of hop count is greater than 2) is weed out per SHCF (Simplified Hop Count Filtering) algorithm in the network layer (see the following figure).

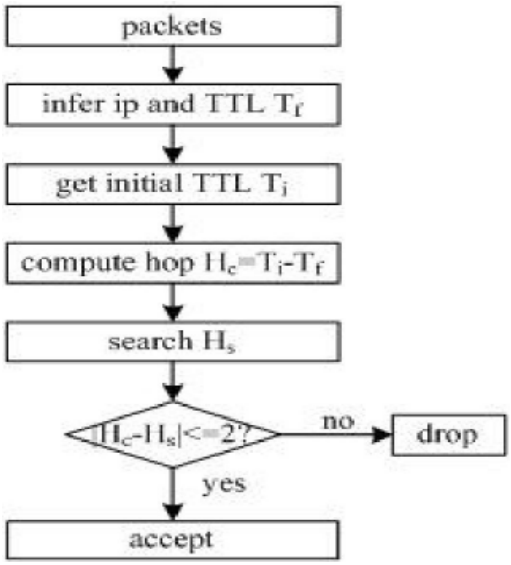


Figure 2.19 : Hop-Count Inspection

The rest of illegitimate traffic is weed out by SYN Proxy Firewall algorithm in the transport layer in the case that the spoofed packet which has the same hop count (HC) like the legitimate packet (see the figure bellow).

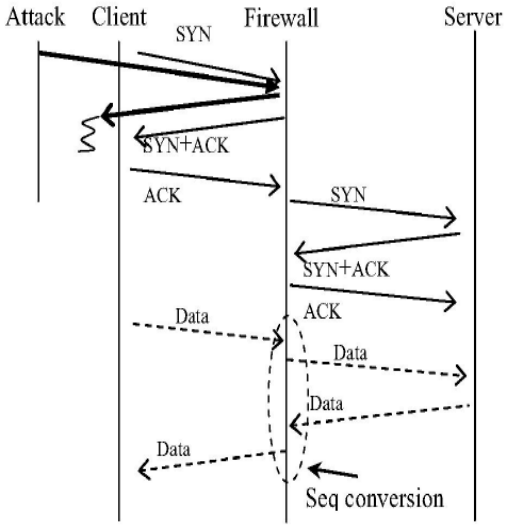


Figure 2.20 : SYN Proxy Firewall

And the traffic limit is used for preventing DDoS attacks which use legitimate (real) IP addresses in the application layer i.e we can take a quantity Q such as the number of sent packets per a legitimate user which corresponds to a probability 1/10 of its part (see the following figure). A probability of legitimate transaction which send more than Q packets ,it's very small .

This mechanism can weed out the spoofed traffic within a pourcentage between 98,73% and 98,93% and a rate of (≈ 4 %) of false negative.

Critical Section :

1- The author did not study the issue of computation time and updates of IP to Hop count in the network layer .By using packets from established TCP connections (SYN flag) ensures that an attacker cannot slowly pollute a IP2HC table by spoofing source IP addresses.

3-The value of count for counting the number of packets (limit traffic) for the same source IP address must be during a slot time.

2.8 : A Scheme of Distributed Hop-Count Filtering of Traffic

This approach [46] introduces DHCF (Distributed Hop Count Filtering) based HCF which is used into intermediate systems for firstly prevent the server and network bandwidth ,if we install the filter into the router R4 ,the victim is protected ,but the workload between R2 et R3 is not protected (see the following figure).

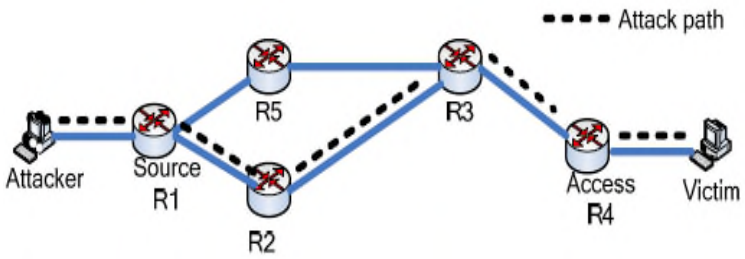


Figure 2.22: DDOS attack topology

And secondly, we only need to save few Hop Count values (Hc) for minimizing the space of memory.The compenents of DHCF schemas are :the DHCF module and and the other is IP-List creation module (see the following figure). At first, we must initialize the IP-List in IP-List creation module in processing machine when the network is running in normal conditions. Once one touches the filter by some flow values or something else ,DHCF module will start.

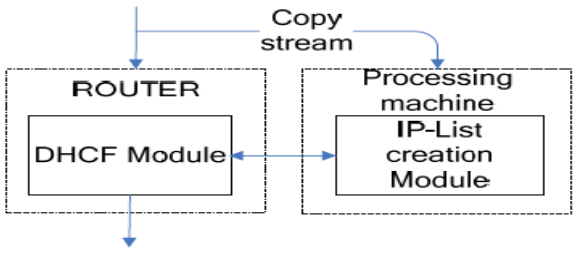


Figure 2.23 : DHCF module

Just after the reformulation of HCF[38] ,we extract TTL value and source IP address ,and after we calculate hop count ,if the HC = Hr ,le DHCF search the liste-IP for source IP address S , if S is found ,so the packet within S is legitimate .otherwise ,this packet will be sent to the processing machine for detection if it means an attack (see the following figure) suppose Hr=8 for filtering the spoofed packets in every router level within the hop count which equals to 8 (Hc=8).

```
DHCF computation:
 for each packet:
  set the threshold Hr = 8;
  extract the final TTL T and IP address S;
  infer the initial TTL T0;
  compute the hop-count Hc= T0-T;
  if ( Hc == Hr )
       {
       Research IP-List for S;
       if (no S item)
            Discard the data packet to Processing
          machine;
       }
```

Figure 2.24 : DHCF computation

Hr : hop count threshold .

This mechanism can weed out the spoofed traffic within a percentage between 93% and 100%

Critical Section :

1- The author did not study the issue of computation time and updates of IP to Hop count in the network layer. By using packets from established TCP connections (SYN flag) ensures that an attacker cannot slowly pollute a IP2HC table by spoofing source IP addresses.

3-Approaches Comparative Table

Approaches	Detection accuracy	False positive rate (FP)	False negative rate (FN)
HCF[38]	90%	10%	5%
HCF[39]	90%	10%	5%
HCF[41]	90%	/	/
HCP[42]	/	/	/
PHCF[43]	80-85%	/	/
[44]	/	0%	/
[45]	98,73%-98.93%	/	≈4%
[46]	93-100%	/	/

Table 2.1:Approaches Comparative Table

4-Discussion

All approaches [38,39,41] based on *statistical analysis* of the packet flow that are used for detecting flooding attacks based DoS/DDoS within 90% of detection accuracy ,and 10% of false positive rate and 5 % of false negative rate.

The rest of approaches have a detection accuracy which vary between 80% and 100% without neither false positive rate nor false negative rate except the approaches[44,45] with a false negative rate which equals to 4% ,but without false positive rate for [45] and with a false positive rate which equals to 0 %, but without false negative rate for [44].

5- Conclusion

We have presented in this chapter which is called state of the art of detection mechanisms of flooding attacks based DoS/DDoS based on statistical analysis of the packet flow that are based on different filtering methods of hop count such as SHCF(Simplified Hop-Count Filtering), DHCF (Distributed Hop-Count Filtering), Hop-Count Filtering and probabilistic approach….etc.

Within combined or simple algorithms of every method with detection rate, false positive and false negative rates if they exist.

The following chapter will take into account our proposed/ameliorated algorithm description.

Chapter III

Proposed Algorithm

1-Introduction

We have summarized several studies (algorithms/ approaches) which are using Hop Count for detecting flooding attacks based DoS/DDoS ,which use IP spoofing in the majority of cases ,and have given their detection accuracy ,and false positive and false negative rates for ones ,but for others the authors didn't give them.

In this chapter which is called the proposed algorithm ,we are going to modify an algorithm (approach) [41] which is better than others that are summarized in the precedent chapter which is called the state of the art in term of computation time , detection accuracy ,and false positive and false negative rates.

2-Problematic and Motivation

In that algorithm [41] which is called 'Packet Monitoring Approach to Prevent DDoS Attack in Cloud Computing' and which can detect flooding attack based DoS/DDoS in cloud environment within a detection accuracy that equals to 90% and false positive rate (FP) that equals to 10 % and false negative rate that equals to 5% It seems for detecting attacks in the cloud environment when the packet seeks to reach its destination (see the figure bellow) .

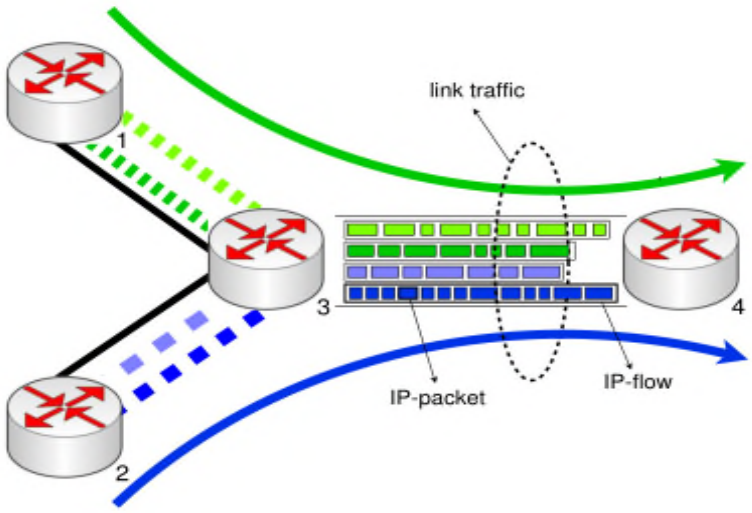


Figure 3.1: Access Link

For this reason, we are going to modify it for increasing the detection accuracy and Diminish false positive (FP) and false negative rates in this chapter.

3-Flooding Attacks based DoS/DDoS in Cloud Computing

Flooding is often called a Denial of Service (DoS) attack or DDoS (Distributed Denial of Service) in literature which is a type of DoS ,thus every flooding attack is a Denial of Service attack. There are Three main types of flooding attacks such as TCP/SYN flood attack, ICMP flood and UDP flood which use IP spoofing in the majority of cases (see the figure bellow) to :

1-Conceal flooding sources and localities in flooding traffic.

2- Coax legitimate hosts into becoming reflectors, redirecting and amplifying flooding traffic, so that the attacker can stay away from detection.

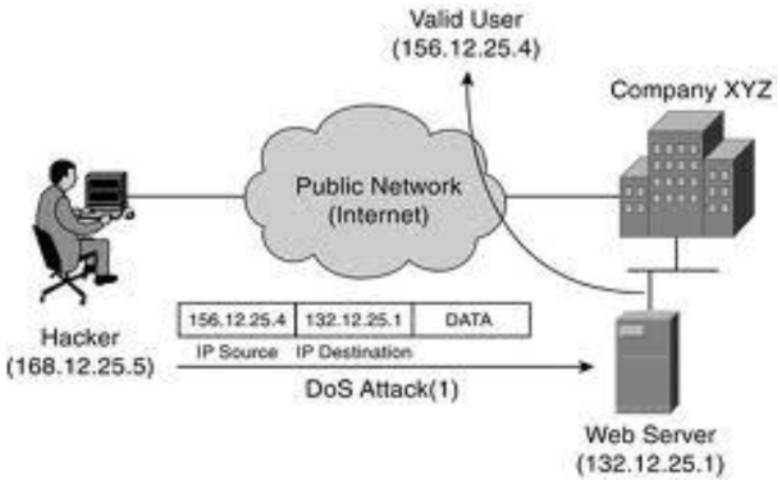


Figure 3.2: DoS Using IP Spoofing [23]

3.1-TCP/ SYN-Flood attacks

A SYN flood occurs when a host sends a flood of TCP/SYN packets, often with a fake sender address. Each of these packets is handled like a connection request, causing the server to spawn a half-open connection, by sending back a TCP/SYN-ACK packet (Acknowledge), and waiting for a packet in response from the sender address (response to the ACK Packet). However, because the sender address is fake and the responses never come. These half-open connections saturate the number of available connections that the server is able to make, keeping it from responding to legitimate requests until after the attack ends. See the figure bellow

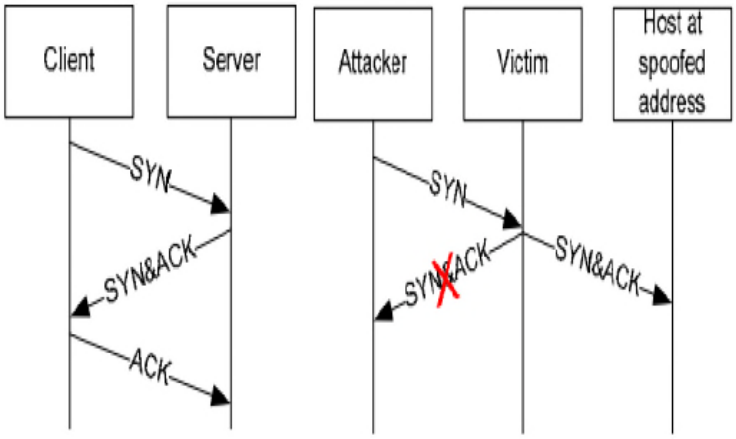


Figure 3.3: Package flow in three-way handshake (a) and TCP SYN attack (b)[19]

3.2-ICMP flood attacks (e.g: ping floods): Like the other flooding attacks, this one is accomplished by broadcasting a bunch of ICMP packets, usually the ping packets which attempt to saturate a network by sending a continuous series of ICMP echo requests over a high-bandwidth connection to a target host on a lower bandwidth connection. The receiver must send back an ICMP echo reply for each request.

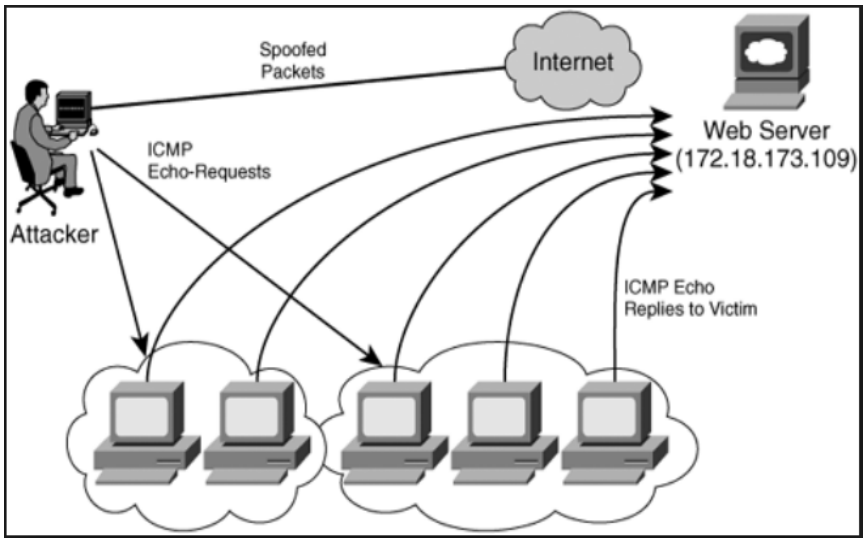


Figure 3.4: ICMP flood attack [19]

3.3-UDP flood attacks : A huge amount of UDP packets are sent to the victim host. As the UDP does not have a congestion control system, the attacker can potentially send a very large number of packets .

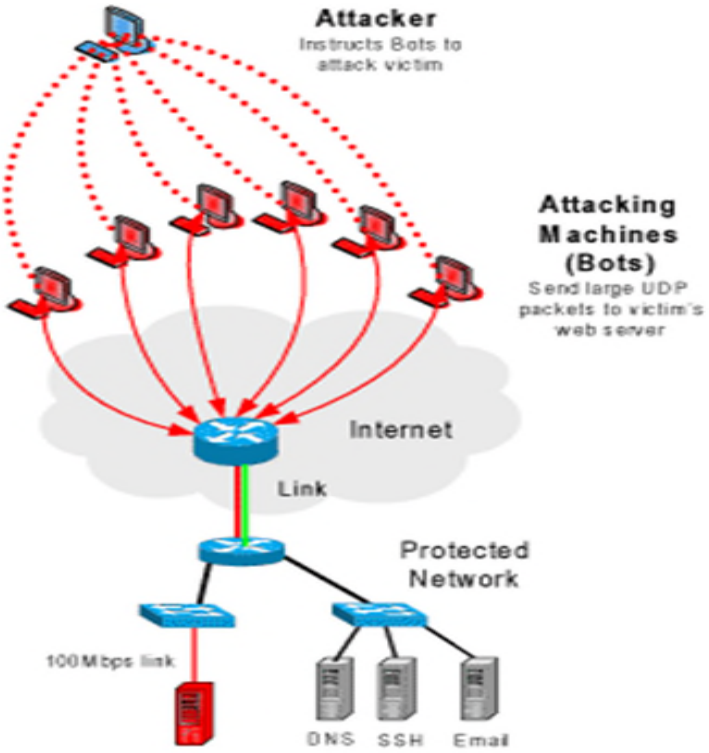


Figure 3.5: UDP flooding attacks [48]

The three precedent main types of Flooding Attacks based DoS/DDoS are draw as follow :

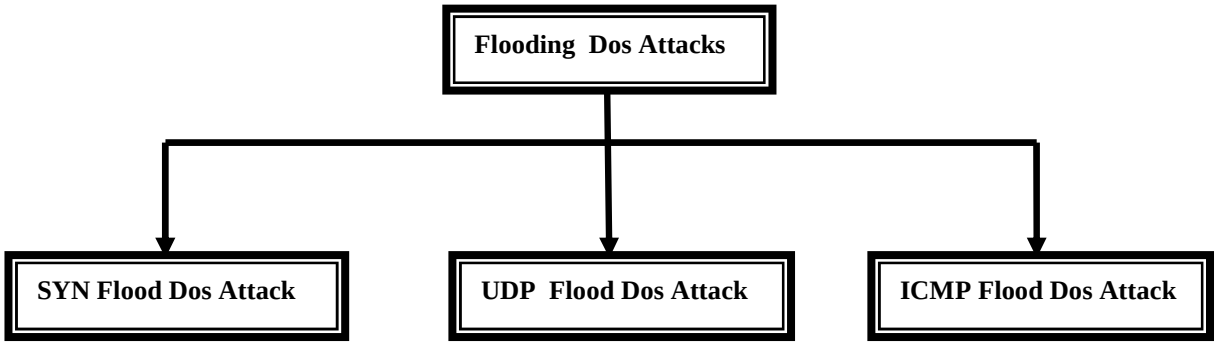


Figure 3.6 :The three main types of Flooding Attacks based DoS/DDoS

4- Botnet(compromised hosts):

The precedent main types of flooding attacks based DoS/DDoS can use Botnet to amplify the traffic flooding attacks. The botnet is defined with its life cycle [49,50] as follows :

4.1- Botnet is one of the biggest threats to the Internet, is the presence of large pools of compromised computers, also known as botnets, or zombie (drone) armies, sitting in homes, schools, businesses and governments around the world. Under the control of a single (or a small group of hacker, commonly known as a botmaster, botnets are often used to conduct various attacks, ranging from Distributed Denial-of-Service (DDoS) attacks to e-mail spamming, keylogging, click fraud, and spreading new malware.

4.2-Botnet Life Cycle :

The success of any process mainly lies in how well the sequence of steps is organized. The major reason of dramatic success and spread of botnets is their well organized and planned formation, generation and propagation. The life cycle of a botnet from its birth to disastrous spread undergoes the following phases:

1. Bot-herder configures initial bot parameters.
2. Registers a DDNS.
3. Register a static IP.
4. Bot-herder starts infecting victim machines either directly through network or indirectly through user interaction.
5. Bots spread.
6. Bot joins the Botnet through C&C server.
7. Bots are used for some activity (DDoS, Identity Theft etc.)
8. Bots are updated through their Botoperator which issues update commands.

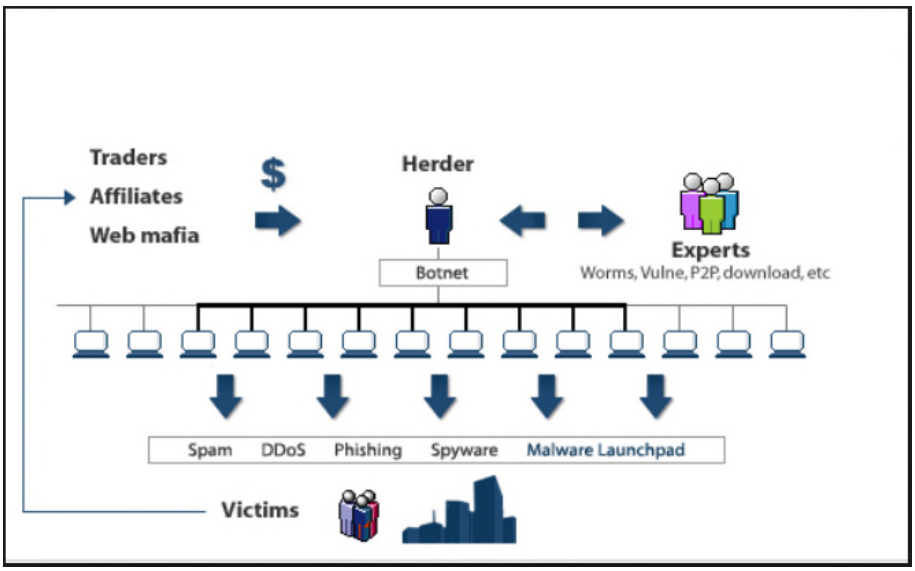


Figure 3.7 :Example of using Botnets

5-Features extracting from TCP / IP headers .

The author extracts the precedent features from TCP/IP headers as shown in the following figures .

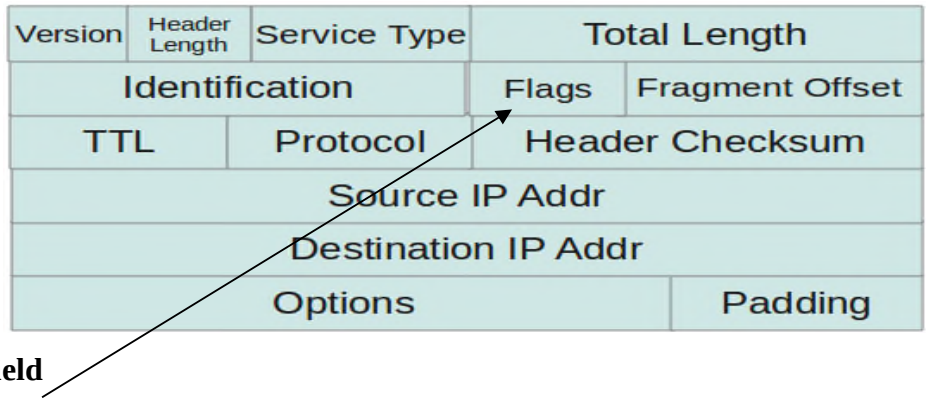


Figure 3.8 : IP header

from this control field ,the author algorithm extracts the SYN flag and the source IP address which is used like index for finding the stored Hop Count

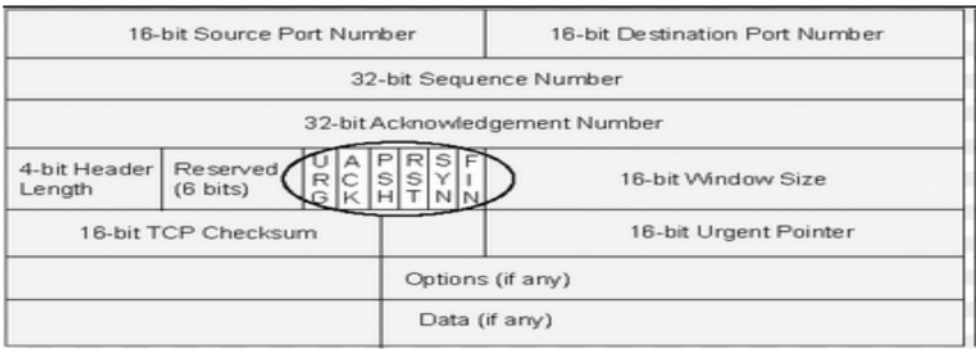


Figure 3.9 : TCP header

The control field above illustrates all flags of TCP header such as: URG ,ACK , PSH ,RST , SYN and FIN. Each flag occupies only 1 bit ,but the author algorithm extracts only the SYN flag .The following table gives us a brief description of each bit .

Flag	Description
URG	The value of urgent pointer field is valid
ACK	The value of acknowledgement field is valid
PSH	Push the data
RST	The connection must be reset
SYN	Synchronize sequence numbers during connection
FIN	Terminate the connection

Table 3.1:Description of Flags in the control Field of TCP Header

5.1-An example of computation hop count

If the final value is 112, the initial TTL value is 128,then the hop-count is 16 (= 128 – 112).

5.2- Initial TTL Values

The following table shows us some initial TTL values of few known operating systems (OSs).

Operating System	Initial TTL
Windows 95	32
Windows 98, XP, Vista	128
Mac OS X	64
NetBSD	255
RedHat 9	64
HP - UX	30

Table 3.2- List of possible initial TTL values

5.3- An example of a IP2HC table

Source IP address	Hop Count
Net 1.2.3	**5**
Net 3.5.7	**2**
Net 2.2.7	**10**

Table 3.3: IP2HC table

6-Author Algorithm

The auhor algorithm uses the hop count filtering mechanism, and provides a clear idea of implementation so that it can be used in Cloud environment to prevent DDoS/DoS attacks. The algorithm requires continuous monitoring of packets travelling over the network in the Cloud, and thus, we extract SYN flag (Syn), TTL_f value and source IP address (Src) from these monitored TCP/IP packets. The algorithm recognizes four cases for each captured packet in the whole operation.

1. If SYN flag is set and source IP address exists (Syn=1 and Src=1) in IP2HC table then calculate hop-count by using TTL value of IP packet. Now check if the hop-count matches with the stored hop-count, if not, then update source hop-count field of table for that source IP address.

2. If SYN flag is set and source IP address does not exists (Syn=1 and Src=0) in the IP2HC table then calculate hop-count and add a new entry for the Source IP address with the corresponding hop count in the IP2HC table.

3. If SYN flag is not set and source IP address exists (Syn=0 and Src=1) in IP2HC table then calculate hop-count and if this hop count does not matches the stored hop count entry in the IP2HC table for the corresponding source IP address, then packet is spoofed, else the packet is legitimate.

4. If SYN flag is not set and source IP address does not exists (Syn=0 and Src=0) in IP2HC table then it means that the packet is spoofed, because every legitimate IP address having a valid TCP connection will have its entry in the IP2HC table.

The inspection algorithm extracts the source IP address and the final TTL value from each IP packet [51]. The algorithm infers the initial TTL value and subtracts the final TTL value from it to obtain the hop-count. The source IP address serves as the index into the table to retrieve the correct hop-count for this IP address. If the computed hop-count matches the stored hop-count, the packet has been “authenticated” otherwise; the packet is likely spoofed [51].

7-The author algorithm implementation: In this following algorithm ,the author have given the source code of his algorithm which was installed in the victim (target server).

ALGORITHM

Consider the following notations:

synflag = Syn bit of TCP packet.

mcount =malicious packet counter

T_f= final value of TTL.

T_i=initial value of TTL.

Initialize mcount=0;

For each packet

```
Set TTL = ExtractFinalValueOfTTL( );                    //get time-to-leave field of IP packet
Set srcIp = ExtractSourceIP( );                         //get source IP address from IP packet
Set synflag = ExtractSynBit( );                         //get Syn flag value from TCP packet
If (synflag is set)
 {
   If (establish_tcp_connection( ))                     //true when connection established
   {
       If ( srcIp exists  in IP2HC table )
       {
         ComputePacket ( srcIp , TTL, synflag);//function call which filters the spoofed packet
       }
      else                                              //new connection packet
      {
         Hc=ComputeHopCount( TTL );                     //get hop-count value
         NewEntryInTable(srcIp,Hc);                     //Add new  entry into IP2HC table
      }
 }
 else
   {
     // ignore packet
    }
}
else  //synflag is not set
{
    If ( srcIp exists  in IP2HC table)
   {
       ComputePacket ( srcIp , TTL, synflag );//function call which filters the spoofed  packet
    }
    else
    {
        'drop the packet'                               //Packet is spoofed
         mcount++;                                      // increment in malicious  packet by 1
    }
}
```

```
ComputePacket ( string srcIp , int Tf , boolean  synflag)
 {
     Hc=ComputeHopCount( Tf );                    //get hop-count   value
     Hs=RetreiveStoredHopCount(srcIp);            //get stored hop-count value

 If ( Hc != Hs )
     {
         if( synflag is set)
         {
          UpdateTable ( srcIp , Hc);              //update hop-count value in IP2HC   table
         }
         else
         {
           'drop the packet'                      //Packet is spoofed
           mcount++;                              // increment in malicious packet by 1
          }
     }
     else
     {
          'allow the packet'                      // packet is legitimate
      }
 }
```

```
   int ComputeHopCount( int Tf )
{
     Set Ti= InvestigateInitialTTL(Tf);
     return Ti - Tf;                              //return hop-count value
}
```

8-Author Algorithm weeknesses

1-This victim-based approach cannot recognize forged packets whose source IP addresses have the same hop-count value as that of a zombie . For this issue , we must add a count for computing the number of packets within the same source IP address during a slot time.

2-In this victim-based approach , the author did not study the case where there are multiple routers between the sender and destination, which results in multiple allowed HC values : It means the author must add these filtering types : strict filtering, +1 filtering and +2 filtering. Strict filtering drops a packet when the HC differs from the HC profile. The +1 filtering drops a packet when the HC differs by more than one from the HC profile. Finally, +2 filtering drops a packet when the HC differs by more than two from the HC profile.All this for solving the issue of existence of NAT (Network Address Translator) boxes, each of which may

connect multiple stub networks, could make a single IP address appear to have multiple valid hop-counts at the same time.

3-In a case of massive legitimate network accessing the problem of huge network access resulted false positive alarms i.e DDoS is also detectable , but really it isn't a real DDoS attack .For solving this issue ,the author must classify every IP packet like spoofed when its number is greater than a predefined value during a slot-time.

4-The author has not studied Detection Accuracy in case of detection Low Rate Traffic.

9-Design of the Proposed Algorithm (HCFm)

Generally, if the algorithm (approach) does not work correctly or cannot protect totally the cloud system (victim/target server) where it's installed ,we will have more and more security problems within time, the cloud system becomes absolutely vulnerable to attacks such as flooding attacks based DoS/DDoS .Because of those author algorithm weeknesses , which is called HCF , and for solving the majority of them, if possible ,we have created (proposed) the following algorithm which is called HCFm and which is compatible within cloud network ,and which is installed in the victim (target server) .

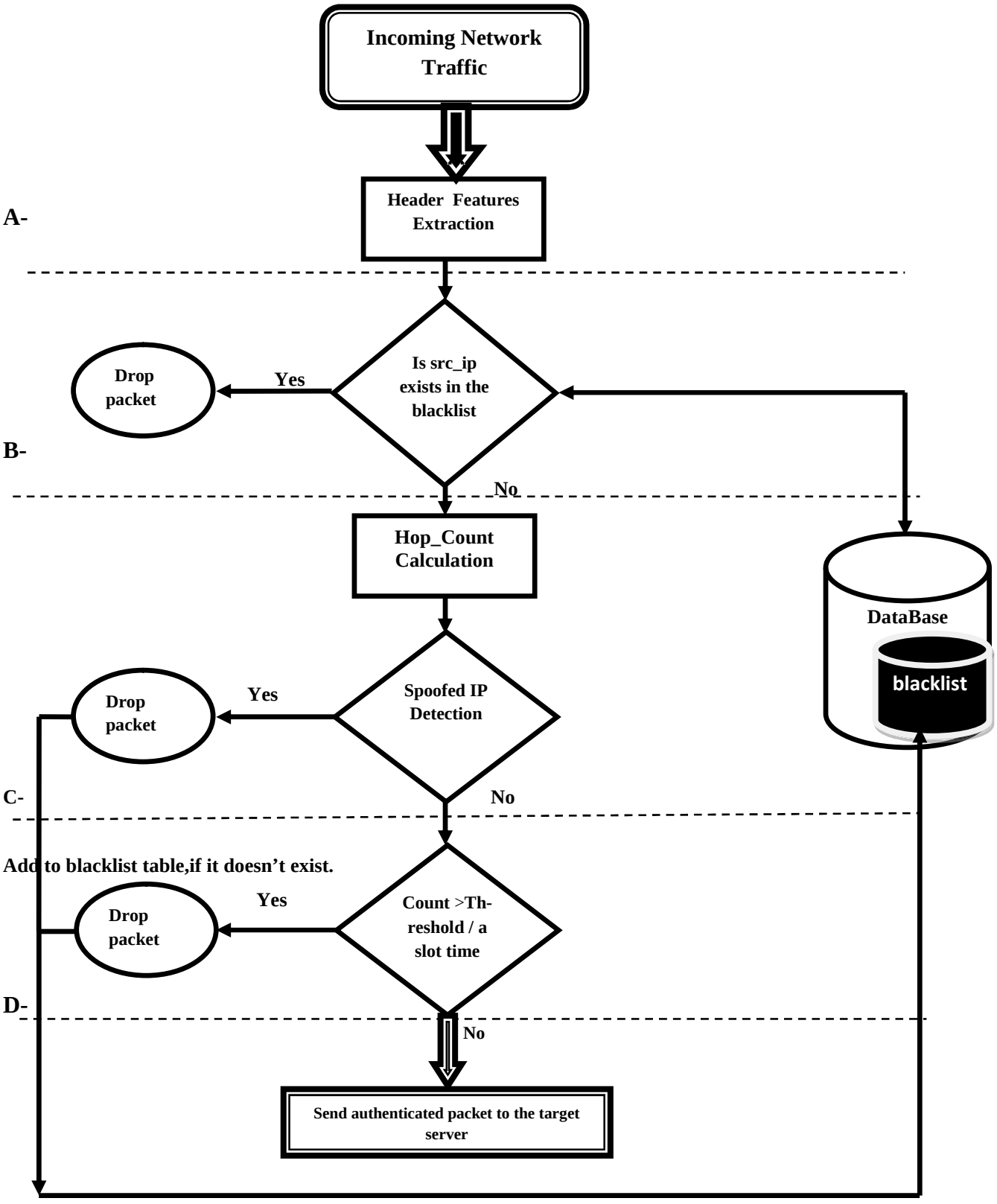


Figure 2.10: Architecture of the proposed Algorithm (HCFm) .

Level A: In this level, the algorithm extracts the three header features of each packet which are source IP address ,TTL_f value and synflag ,and saving them in the database.

Level B:In this level ,we use the precedent features such as source IP address to verify ,if it exists in the blacklist table ,if Yes the packet is immediately dropped without run other levels(steps) which can minimize the computation time of each packet and that of the system , if No the packet will be forwarded to the next level which is called **Level C .**

If (src_ip exists in the blacklist table)

The packet is spoofed so drop it;
Else

The packet is legitimate so Forward it;
End If

Level C : In this level **,**we use the Time to Live (TTL in short) value to compute the number of hops the packet has travelled (Hc). Attacker can spoof the packet header, but not able to manipulate the Hop count value. By comparing Hc's value with the stored hop count (Hs) in the IP2HC table which have the same source IP. If no accurate matches are found, then the packet is spoofed and the algorithm discards it directly ,and adds it, if it does not exist ,in the blacklist within it current system date (sys_date) , else the packet is forwarded to the next level which is called **Level D.**

Compute the hope-count $Hc=TTL_i - TTL_f$;

Get the stored hope_count (Hc) for the source IP address
If *(Hc ≠ Hs)*
The packet is spoofed so drop it;//**insert the (src_ip , sys_date) in the blacklist table**
Else
The packet is legitimate so Forward it;
End If

Level D: In this level which verifies the number of packets of the same source IP address during the same slot time in the IP2HCm table ,if the count > threshold ,the packet is spoofed so delete it from the precedent table and add it to the blacklist table ,if it does not exist , within its current system date, else we send the authenticated packet to the target server (victim).

If learning period
For each packet increases the count in the IP2HCm table, if it's legitimate ;
Else // slot time has been finished
For each source IP in the **database (IP2HC table)** ,extract the count
If (count > threshold)
The packet is spoofed so drop it, after
insert the (src_ip , sys_date) in the blacklist table///sys_date :current-system-date
Else count =0 ; initialize the count by zero (0);
EndIf

In the end of X amount of time ,we must verify the black list for removing that the IP address which was already blocked , if it didn't send a spoofed IP by subtracting the current system date from sys_date which is into the precedent blacklist table.

10 - HCFm items utilization

HCFm items	**Utilizations**
Profile Hop Count values	- To include all the possible available Hop Count values. -To extract the threshold by trying several times in the coud lab or in the cloud environment.
Threshold	-Calculated before starting attacks and attack detection ; it means in the case of legitimate users. -It have been used for computing the number of packets during a slot time for example 1000 packets /second .
Counter	**-Generally , it have used for detecting flooding attacks based DoS/DDoS that don't use IP spoofing .** -To compute the number of packet IP during a slot time (an interval of time of 10 seconds for example). - We initialize all their counters ,in IP2HCm table by zero (count = 0) just after the end of slot time and just after the attack detection into the **IP2HCm** table.
Blacklist	-Contains spoofed IP addresses for minimizing the compution time and updates in IP2HCm table. -Contains every legitimate IP address when its count greater than the threshold during a slot time (attackers that they don't use IP spoofing). -We remove from it every IP address that is not used by an attacker during X amount of time.

Table 3.4: HCFm items and their utilizations

11-Global proposed HCFm algorithm

Consider the following notations:
synflag = Syn bit of TCP packet.
Blacklist =Empty // list of attacker IP which are detected when the number of the packet IP (**count**) is greater than the **threshold** **(eq:100 packets /s)**or when it's a spoofed packet and doesn't exist.
count =number of packet IP during a slot time (e.g:slot-time=10 seconds).
mcount =malicious packet counter
TTL_f= final value of TTL.
TTL_i=initial value of TTL.
mcount =0 // Initialize mcount.
X_time =24 hours for example//amount of time for removing IP addresses from blacklist table if possible

```
If learning period                    //the slot time doesn't finish.
{ For each packet
 Set TTLf = ExtractFinalValueOfTTL( );          //get time-to-leave field of IP packet
 Set srcIp = ExtractSourceIP( );                //get source IP address from IP packet
 Set synflag = ExtractSynBit( );                //get Syn flag value from TCP packet
 If ( srcIp exists in the blacklist )
   {
      // ignore packet.
   }
 else
 {
    If (synflag is set)
     {
       If (establish_tcp_connection( ))          //true when connection established
       {
          If ( srcIp exists in IP2HC table )
           {
            ComputePacket ( srcIp , TTLf , synflag );//function call which filters the spoofed packet
            }
           else  //new connection packet
            {
            Hc=ComputeHopCount( TTL );            //get hop-count value
            NewEntryInTable (srcIp,Hc ,1 );       //Add entry into IP2HC table
             }
       } //end establishment_tcp_connection( )
       else
       {  // ignore packet
        }
     }
     else  //synflag is not set
     {
        If ( srcIp exists in IP2HC Table)
        {
         ComputePacket ( srcIp , TTLf, synflag  ); // function call which filters the spoofed packet
        }
       else
        {  'drop the packet'                      //Packet is spoofed
       //insert the (src_ip , sys_date) in the blacklist table
            mcount++;                             // increment in malicious packet by 1
        }
     }
 } //end test packets in the blacklist
}
```

```
else  // the slot time has  been finished
{
   count =RetreiveStored countIP (count);    //get stored count IP value from IP2HCm table
 for  each src IP  in IP2HCm  table
  {
    if (count >  threshold )
    {
      //add to blacklist table // this blacklist contains all attackers IP which have the  number of
packets IP  greater than threshold during a slot time and IP spoofed if doesn't exist.
          //delete ( srcIp , Hc  , count) from IP2HCm table.
          //insert the (src_ip ,  sys_ date) in the blacklist table
    } else {count =0;} //initialize the count for restarting the counter for each arriving packet
  }  // end for
}  //end else.
// We do this operation in the  end of X amount of time for example every day )
// for removing IP address from the blacklist table
If   X_time has not been  finished     // for verifying if the time has been finished.
 For  each src_ip  in blacklist  table
  {  If  (current system_date  -  sys_date >=  X_time )    // - :it means the minus
    {  // delete ( src_ip , sys_date) from blacklist  table.
       //  Which removes the IP address from blacklist table
    }  endif
 }  end for
}endif
  }
}
```

NB:the **current system date** means the date of the system when the packet in the treatment for example 12/01/2015 ,but the **sys_date** means the date of system when the packet was spoofed for example 11/01/2015

```
ComputePacket ( string srcIp , int Tf , boolean  synflag, int count )
{
    Hc=ComputeHopCount( Tf );                    //get hop-count  value
    Hs=RetreiveStoredHopCount(srcIp);            //get stored hop-count value
    If ( Hc != Hs )
    {
        if( synflag is set)
        { count ++ ;//increment the number of this packet IP and store it in the  IP2HCm  table.
         UpdateTable ( srcIp , Hc);                  //update hop-count value in IP2HCm   table
        }
        else
        {
          'drop the packet'                          //Packet is spoofed
          //insert the (src_ip , sys_date) in the blacklist table
          mcount++;                                  // increment in malicious packet by 1
         }
    }
    else
    {  count ++ ;//increment the number of this packet IP and store it in the  IP2HCm  table.
        'allow the packet'                           // packet is legitimate
     }
}
```

```
int ComputeHopCount( int Tf )
 {    Set Ti= InvestigateInitialTTL(Tf);
      return Ti - Tf;        //return hop-count value
 }
```

12-HCFm Algorithm Weaknesses

1-It is conceivable that a specific IP or network address could legitimately appear to come from any number of physical hosts located across the globe and have a widely varied hop count. In those scenarios there is a high likelihood of incorrectly blocking legitimate traffic.

2-Our proposed /ameliorated algorithm can't detect others cloud attackers such as probing ,U2R (User to Root attacks) ,R2L (Remote to Local access)……..

13-Conclusion

we have proposed/ameliorated algorithm inspired from [41] by changing the alert state of HCF to include all the possible available Hop Count values and by adding to it a counter to compute the number of packet IP during a slot time , by adding to the IP2HC table another column which can compute the number of the same packet during the same slot time for detecting attackers that they don't use IP spoofing ,and by using a blacklist table where we make all packet IP which are detected like legitimate ,but they have been forwarded to the target server/victim ,but the number of its packets (count) is greater than the threshold (e.g :1000 packets /second) during a slot time , and IP packets that are spoofed must be added to precedent blacklist table .When we have found an attacker IP ,we must add it to the blacklist table and delete it from the modified IP2HC (IP2HCm) for facilitating the search of source IP address during the running levels of our ameliorated /proposed algorithm .At the same time ; we must remove an IP address from the blacklist if it has not sent a spoofed packet in X amount of time because in this case ,it can be used by an attacker during a precedent attack without knowing ,in this context ,it's become a victim of attack .For others ,we must initialize all their counters by zero(count=0) for restarting the search in the end of slot time.The purpose of our proposed/ameliorated algorithm is generally to prevent cloud computing against flooding attacks based DoS/DDoS such as IP spoofing which is used by Syn flood attack, specially for increasing the detection accuracy over 90% and reducing the false negatives rate and false positives rate by using the precedent modified / proposed algorithm which is already discussed and which is called HCFm and also ,it can detect attackers that they don't use IP spoofing by using the counter.

The simulation and the performance evaluation of our proposed/ameliorated algorithm which is called HCFm well be discussed in the next chapter.

Chapter IV

Simulation and performance evaluation

1- INTRODUCTION

We are specially focused on detection of flooding attacks based DoS/DDoS in the cloud environment and which use IP spoofing in the majority of cases .We are going to implement our proposed/ameliorated algorithm which is called HCFm in the hypervisor (target server/victim). Firstly, we have included all the possible available Hop Count values and source IP addresses by extracting them from TCP and IP ,UDP and IP ,and ICMP and IP headers. We capture all the packets within the network for analysis. The packets are captured with the help of two packages namely WINPCAP and JPCAP(see the figure 4.1below). WINPCAP interacts with the OS and NIC to capture the packets. While JPCAP is a java package which gets the captured packets from the WINPCAP to the java program .Add the IP address to the blacklist table when it's a spoofed packet, and the threshold to verify the number of each packets during a slot time for detecting attackers that don't use IP spoofing ,if the number of each packet greater than the threshold ,it adds the IP attacker in the black list table for minimizing the computation time of each attack packet during flooding attacks.

2-HCFm ALGORITHM ARCHITECTURE:

Our proposed /ameliorated HCFm algorithm must firstly uses JPcap and WinPcap that are defined as follows :

JPcap: [52] which is a java library for capturing and sending network packets. Using JPCAP, we can develop applications to capture packets from a network interface and visualize/ analyze them in java. We can also develop Java applications to send arbitrary packets through a network interface. JPCAP can capture Ethernet, IPv4, IPv6, ARP/RARP, TCP, UDP, IGMP and ICMPv4 packets [53].

WinPcap: [54] which consists of a driver that extends the operating system to provide low-level network access, and a library that is used to easily access the low-level network layers.

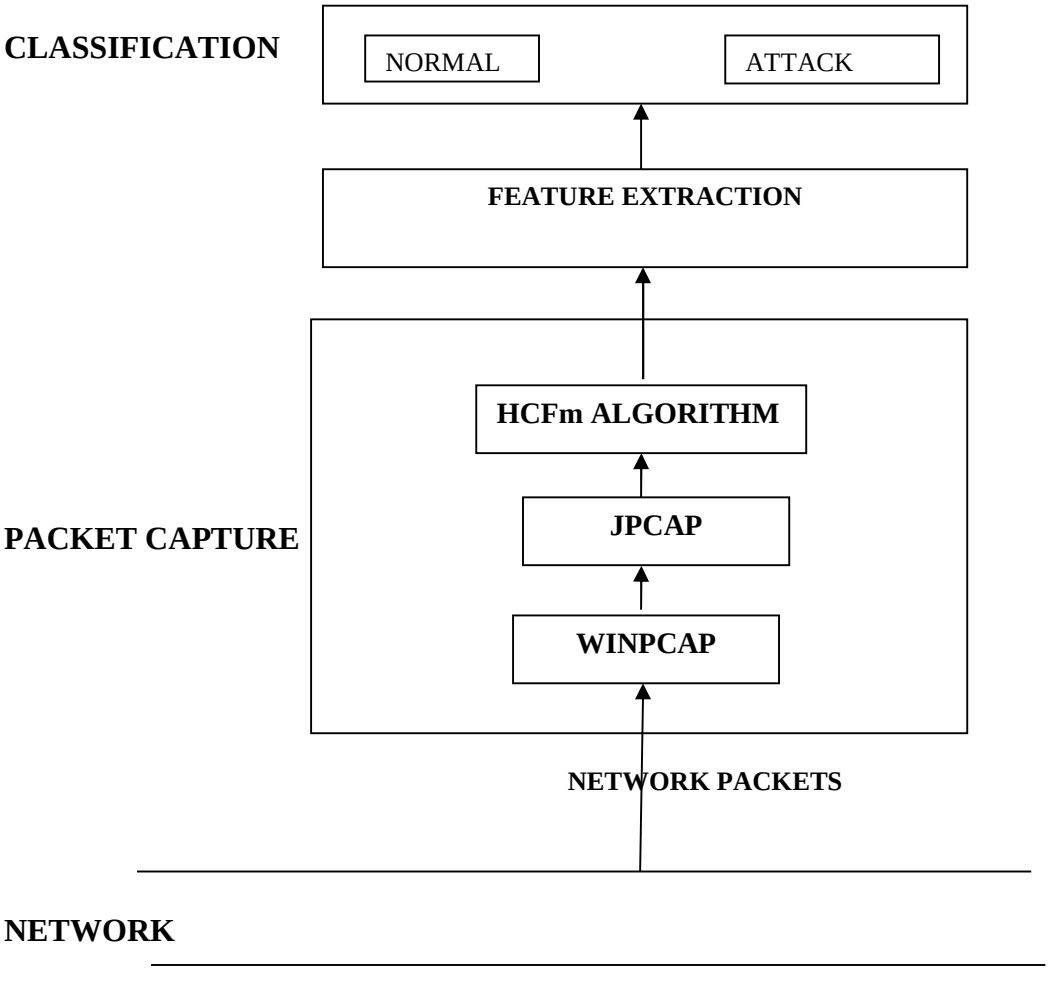


Figure 4.1: HCFm ALGORITHM ARCHITECTURE

3-PERFORMANCE EVALUATION

3-1:Expirmental Test Bed :

The following test bed includes local network infrastructure connected with the internet which includes two legitimate users , an attacker , and the other is a physical server (**hypervisor**)which is a victim where we have installed our proposed/ameliorated algorithm (HCFm). The physical server forwards really the authenticated packets towards a virtual machine that has the windows 7 like OS,but it (physical server) drops the spoofed IP packets.

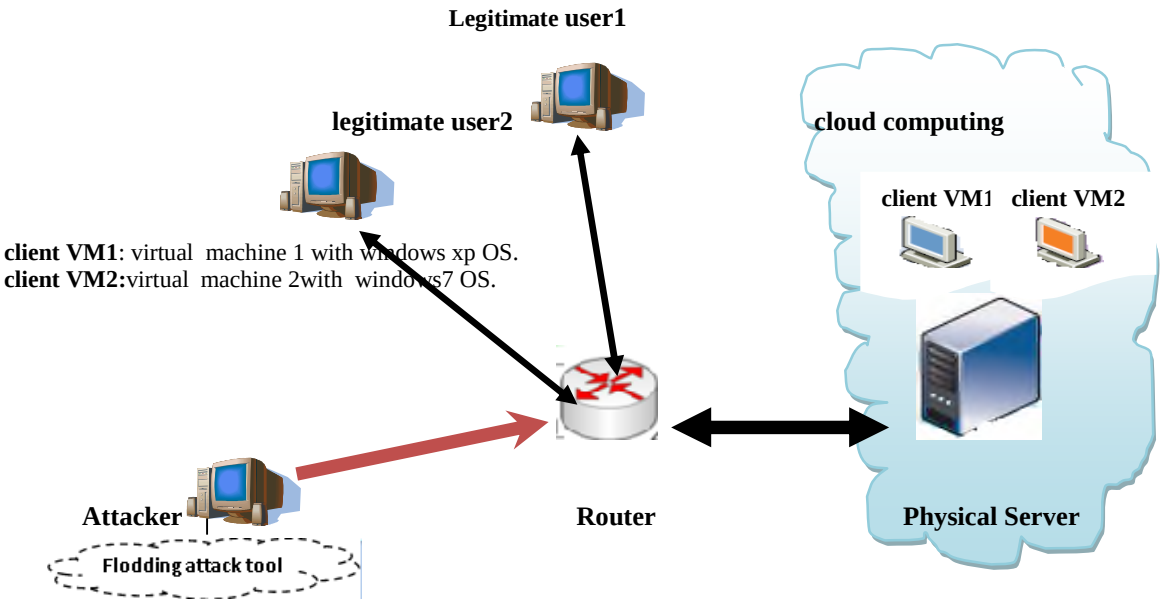


Figure 4.2:Architecture of our experimentation with VMware.

3-2: RESULTS DISCUSSION

This ameliorated/proposed algorithm is evaluated with respect to implementation. For generating network traffic and flooding attacks based DoS/DDoS , we have created a cloud environment . We have used **VMware** as virtual machine manager (VMM) to make placement within the cloud network compatibility ,and windows 7 as guest operating system installed in the physical server which has the following features: Intel core i3 (2.20 Ghz), 8GB RAM, and send authenticated packets to virtual machines(VMwin7 OS or VMwinxp OS).We have also installed 2 hosts with the following operating systems:The first is windows

OS :operating System

xp (32 bit) and the second is the windows 7 (32 bit).

For attacking this server (physical server or victim),we have used two users which are generating normal traffic consist of FTP access ,Web page access ,e-mail access or UDP traffic ,but the attacker is generating attacks such as TCP/SYN flood attack with spoofed IP addresses by using DoS/SynAttack tool (see the figure bellow).

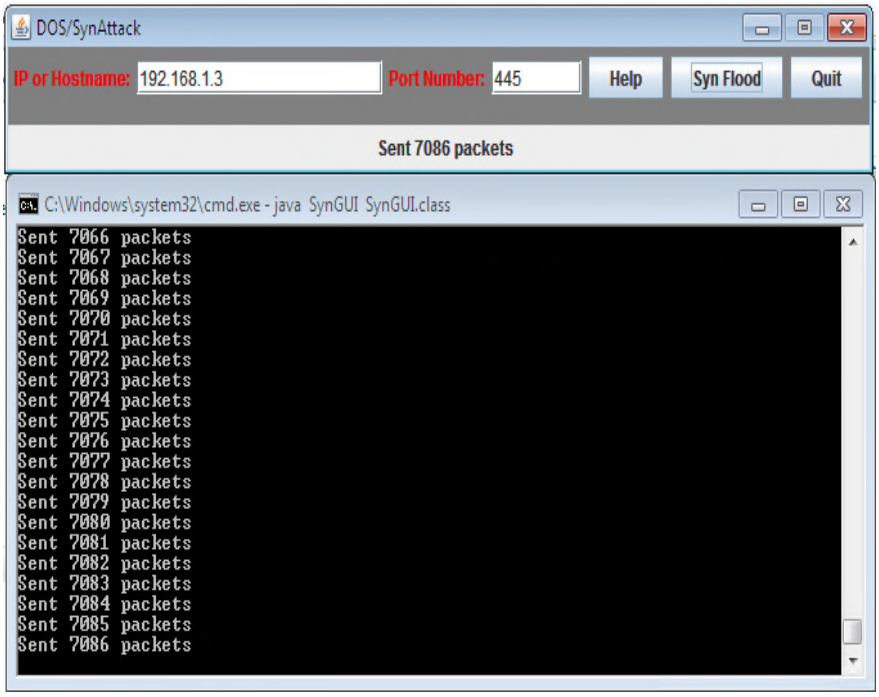


Figure 4.3: DoS/Syn Attack tool

In our ameliorated (modified) algorithm (**HCFm**), only the TTL_f value, synFlag and source IP address of packet are extracted. To capture packets and access all its header information, a packet capturing tool **JPCap** (Java Packet Capture) is used. JPCap is an open source java Library for transferring and capturing network packets. By using **JPCap ,** we can establish applications to capture packets from a network interface and explore them in java .Figure 4.5 shows CPU performance plots of our server during TCP/SYN flood attacks. In the beginning, all non spoofed packets are allowed in the learning period and all their headers (it means : source IP address , TTL_f value and synflag) are extracted, and once profiled are learned to the number of packets are allowed for each source IP address during a slot time (an interval of time) to recognize deviations by applying the deviation rule (e.g: number of packets/second >

threshold) i.e: the frequency count of each packet is checked ,if it exceeds the threshold value for a particular IP. Selecting inaccurate value may raise false alarm .If the value is too low or if it is too high , it can cause the legitimate traffic being considered as malicious traffic. The other rule of hop count , all packets that don't respect the precedent rules ,are discarded in order to stop similar packets to infiltrate the system .This might decrease the amount of **false alarm** to small extent and increase the **detection rate** (over 90%)which means the ability of the system to detect attacks over the total amount of attacks. The **false alarm** is the number of data incorrectly predicted as attack traffic.

The performance of server in case of normal traffic is shown in Figure 4.4 bellow.

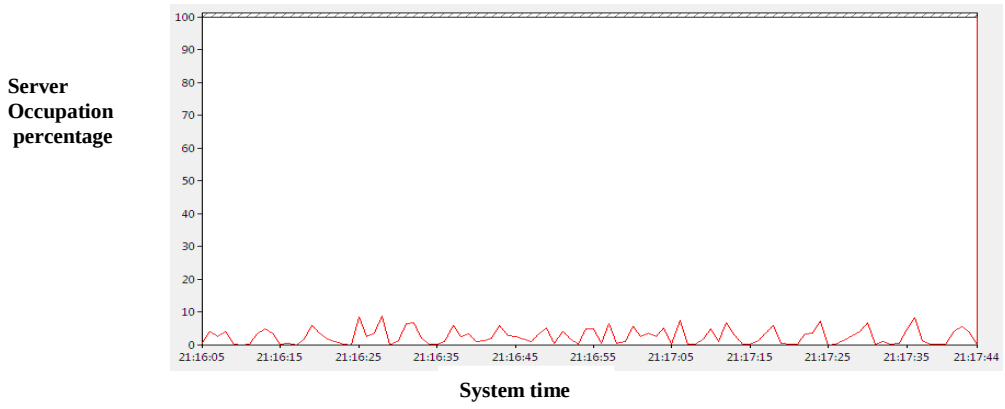


Figure 4.4 :server performance in normal trafic

And under TCP synflood attack in the figure 4.5 bellow.

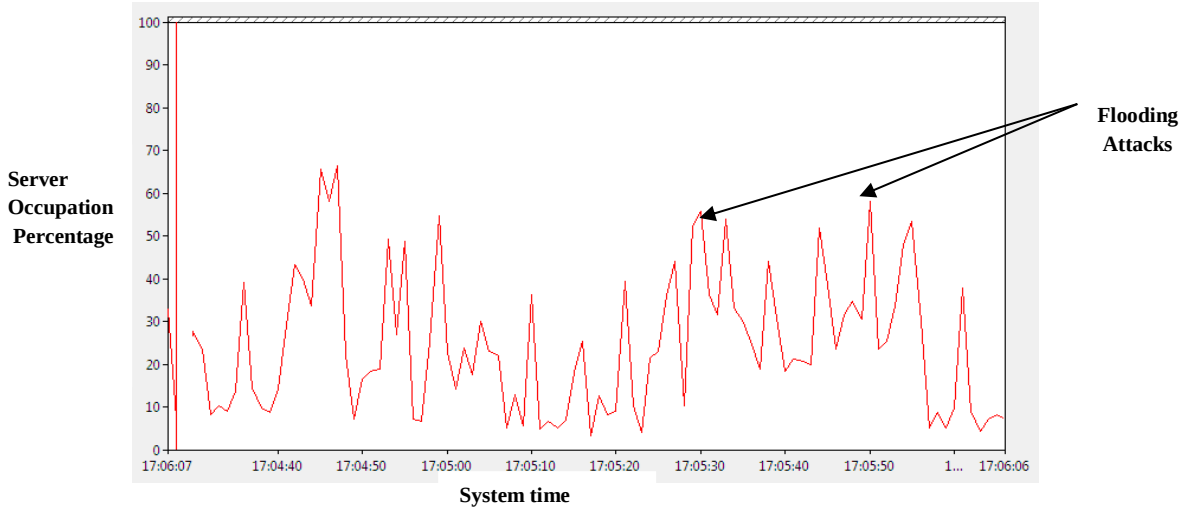
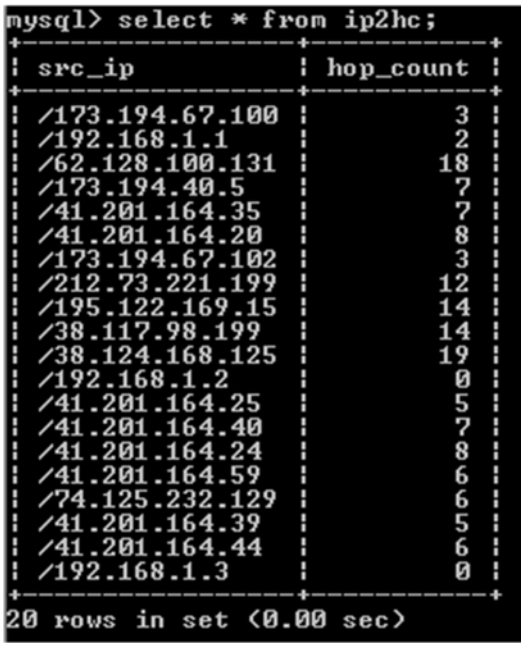


Figure 4.5 :server performance under TCP syn_flood attack.

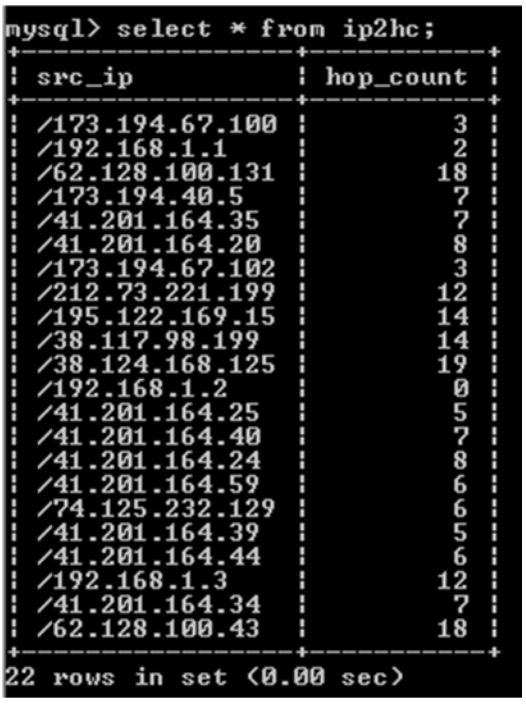
Before the learning period, the IP2HC table was empty, but just after ,it's like shown in the following figure.



```
mysql> select * from ip2hc;
+-----------------+-----------+
| src_ip          | hop_count |
+-----------------+-----------+
| /173.194.67.100 |         3 |
| /192.168.1.1    |         2 |
| /62.128.100.131 |        18 |
| /173.194.40.5   |         7 |
| /41.201.164.35  |         7 |
| /41.201.164.20  |         8 |
| /173.194.67.102 |         3 |
| /212.73.221.199 |        12 |
| /195.122.169.15 |        14 |
| /38.117.98.199  |        14 |
| /38.124.168.125 |        19 |
| /192.168.1.2    |         0 |
| /41.201.164.25  |         5 |
| /41.201.164.40  |         7 |
| /41.201.164.24  |         8 |
| /41.201.164.59  |         6 |
| /74.125.232.129 |         6 |
| /41.201.164.39  |         5 |
| /41.201.164.44  |         6 |
| /192.168.1.3    |         0 |
+-----------------+-----------+
20 rows in set (0.00 sec)
```

Figure 4.6 :IP2HC table after alert_state of HCF algorithm (author algorithm)

In the action state ,the precedent IP2HC table becomes as follows :



```
mysql> select * from ip2hc;
+-----------------+-----------+
| src_ip          | hop_count |
+-----------------+-----------+
| /173.194.67.100 |         3 |
| /192.168.1.1    |         2 |
| /62.128.100.131 |        18 |
| /173.194.40.5   |         7 |
| /41.201.164.35  |         7 |
| /41.201.164.20  |         8 |
| /173.194.67.102 |         3 |
| /212.73.221.199 |        12 |
| /195.122.169.15 |        14 |
| /38.117.98.199  |        14 |
| /38.124.168.125 |        19 |
| /192.168.1.2    |         0 |
| /41.201.164.25  |         5 |
| /41.201.164.40  |         7 |
| /41.201.164.24  |         8 |
| /41.201.164.59  |         6 |
| /74.125.232.129 |         6 |
| /41.201.164.39  |         5 |
| /41.201.164.44  |         6 |
| /192.168.1.3    |        12 |
| /41.201.164.34  |         7 |
| /62.128.100.43  |        18 |
+-----------------+-----------+
22 rows in set (0.00 sec)
```

Figure 4.7: IP2HC table after action state of HCF algorithm (author algorithme)

In the following table ,we have executed the author algorithm which give us the number of malicious packets and the number of allowed packets where we have used one legitimate user and an attacker which has used TCP synflood attack .

	malicious packets	allowed packets
10sec	281	18
20s	564	41
30s	1119	226
40s	1439	226
50s	1790	226

Table 4.1:Traced Data

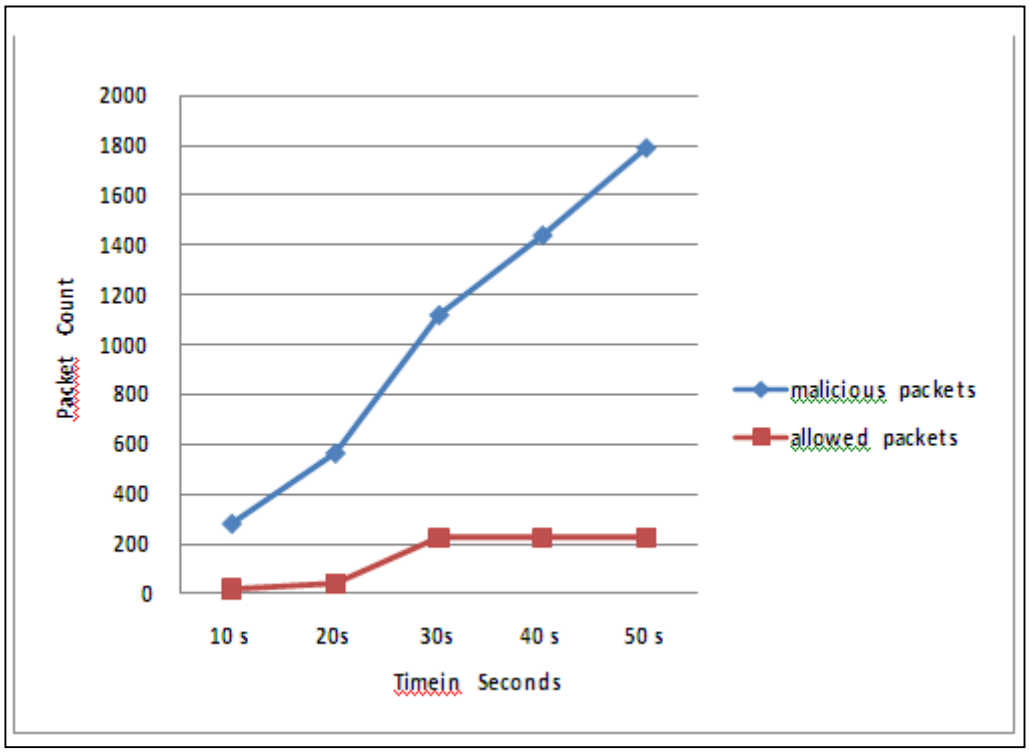


Figure 4.8 :Effect of TCP syn flood attack .

In the HCFm (our proposed algorithm) ,the IP2HCm table and the blacklist were empty ,but the alert state ,the IP2HCm table become like this (see the figure bellow).

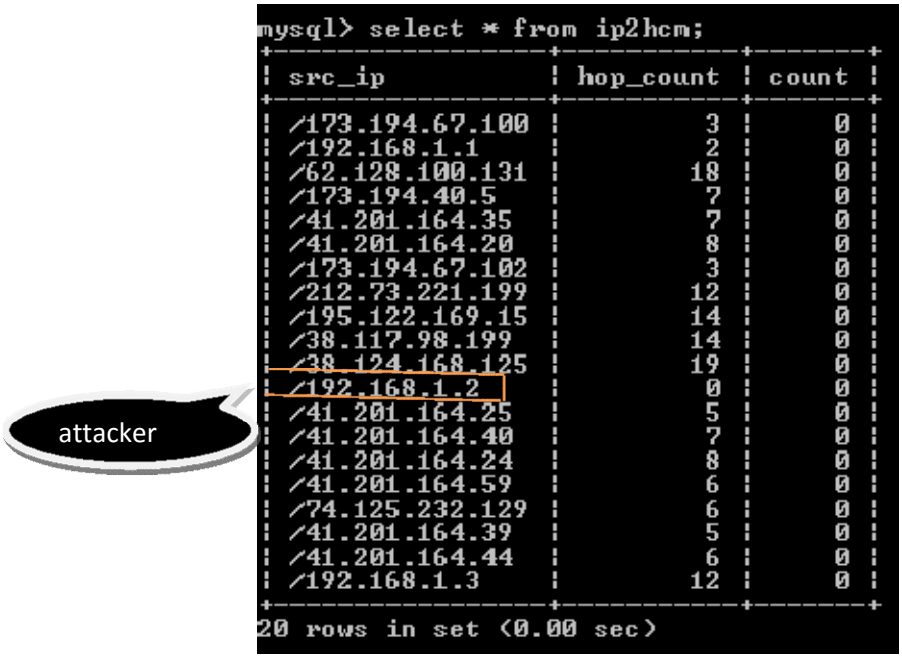


Figure 4.9:IP2HCm table after alert_state of HCFm algorithm(our algorithm)

But the blacklist rested such it was, so it's also empty.

In the action state during 50 seconds (near a minute), the IP2HCm table become like this (see the figure bellow) .

.

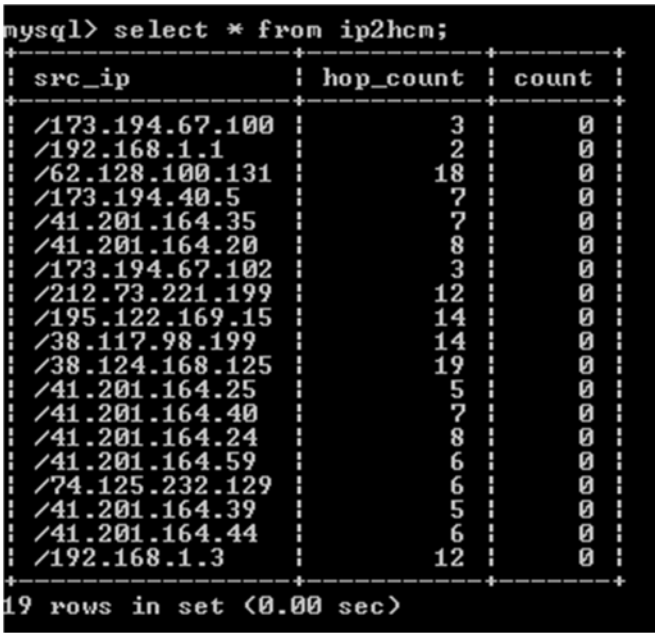


Figure 4.10 :IP2HCm table after action_state of HCFm algorithm(our algorithm)

If we regard in the above table ,we can see an attacker that it was deleted in action state as shown in IP2HCm table and was entered in the black list as shown in the following figure just after action state and after an interval of 50 seconds.

```
mysql> select *  from blacklist;
+---------------+------------+
| src_ip        | sys_date   |
+---------------+------------+
| /192.168.1.2  | 2014-12-30 |
+---------------+------------+
1 row in set (0.00 sec)
```

Figure 4.11: Blacklist after action_state of HCFm algorithm(our algorithm).

In the following table ,we have calculated the number of malicious and allowed packets in the action state during 50 seconds (an attacker (TCP syn flood attack) and one legitimate user).

	malicious packets	allowed packets
10seconds	113	44
20 s	400	44
30 s	697	44
40 s	996	44
50 s	1326	44

Table 4.2 :Traced Data

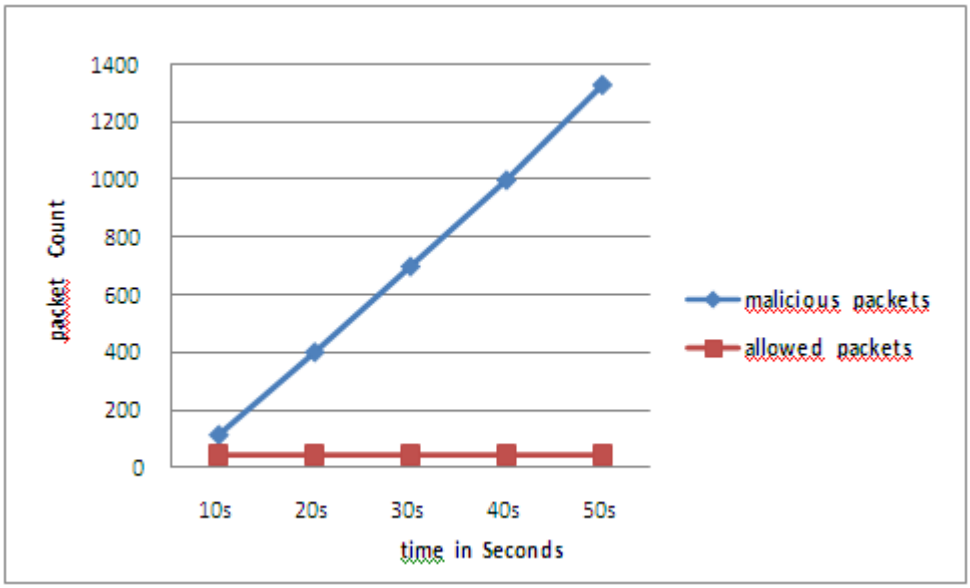


Figure 4.12: Effect of TCP syn flood attack.

3-2-1: Computation Time Comparison

For computation time comparison simulation between the author algorithm(HCF) and the modified algorithm(HCFm) ,the sample inputs are taken as arrival rate in seconds, various results has been analyzed and presented in the Table 4.3.

samples	Sample input (Arrival rate in packets/sec)	Computation time in mseconds (HCFm)	computation time in mseconds (HCF)
1	1000	22	14
2	2000	41	23
3	3000	64	47
4	4000	83	44
5	5000	101	33
6	6000	110	114
7	7000	222	250
8	8000	230	275

Table 4.3 : sample inputs.

The graph in the figure bellow shows how our proposed algorithm (**HCFm**)is very good in the case of high rate ,but it's totally the contrary in the case of low rate .In case of samples 3, 4 and 5 there is a very good performance in HCF, it means sample 3 needs more time then sample 4 and 5 because it depends on receiving field of packets ,but in HCFm the computation time rests in continuous increasing because of several verifications (blacklist table and threshold for example).Computation time is relevant factor for performance measurement of cloud network and it improves processing power of cloud server and minimizes loss of available resources.

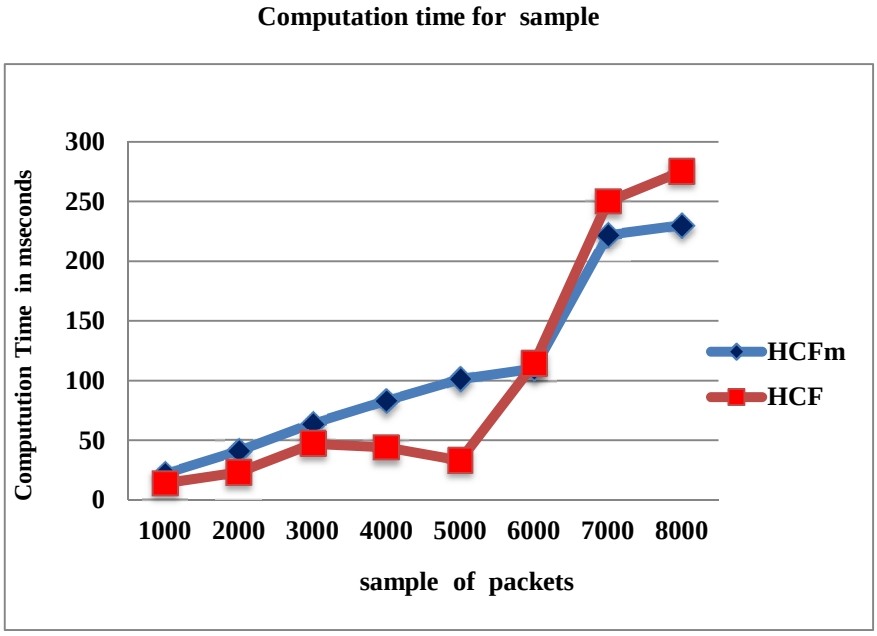


Figure 4.13: Graph showing computation time comparison

3-2-2: Accuracy Detection Comparison :

In the following table, we have done comparison between two algorithms (author algorithm and our proposed algorithm (modified algorithm :**HCFm**).

Algorithms / Slot-times	malicious packets HCF	malicious packets HCFm
10s	0,93	0,85
20s	0,78	0,9
30s	0,83	0,94
40s	0,86	0,95
50s	0,88	0,96

Table 4.4:Traced data.

For example The value of 0,85 (detection rate) refers to the ratio of malicious packets to the total packets .

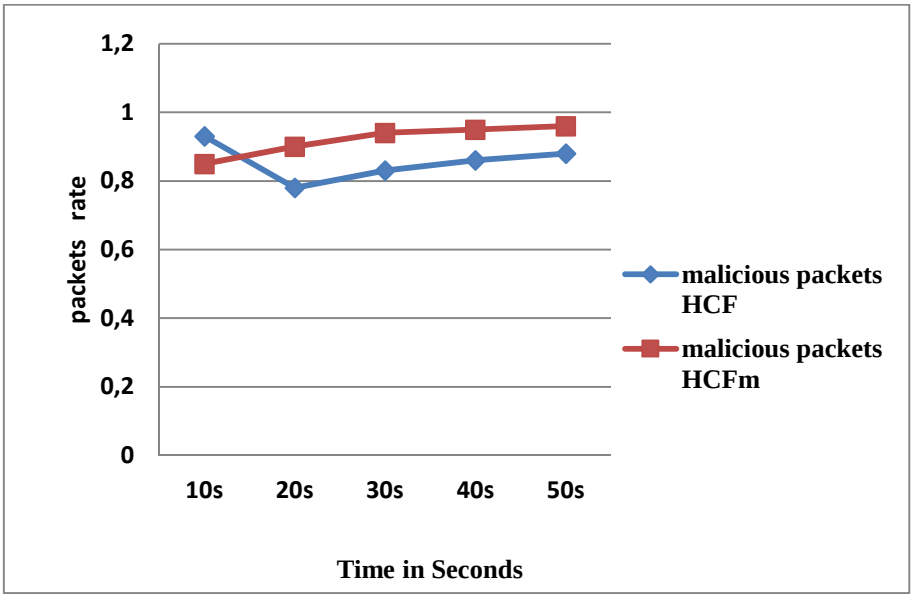


Figure 4.14 : Effect of TCP syn flood attack in the both algorithms(HCF and HCFm).

The detection rate of the both algorithms are mentioned as follows:

The first algorithm (HCF) was approximately (≈86%) : it means

[(0,93+0,78+0,83 +0,86+0,88)/5]*100 ≈ **86% ,**

But the second algorithm (HCFm or our algorithm) was approximately 0,92 it means

[(0,85+0,90+0,94 +0,95+0,96)/5]*100 ≈ **92%**

1-False positives are those legitimate client IP addresses that are incorrectly identified as spoofed. In our algorithm ,all packets are nearly identified correctly ,so our false positive rate approximately equals to zero (≈**0**) ,but in the author algorithm (HCF)it approximately equals to 10%.

2-*False negatives* are spoofed packets that go undetected by **HCFm**.In our algorithm ,we have tried to send a limit number of attack packets but, we haven't detected only **8,9** % ,so this is our false negative rate , but in the author algorithm it equals to 5%.

As mentioned in above paragraph the detection accuracy in HCFm is above 92% but the false positive rate is 0% .HCF has 86% of detection rate with more than 10% of false positive.

By this comparison ,we the observe that the HCFm algorithm ,in detection accuracy and false positive rate, has better performance in comparison to the author algorithm. But in false

negative rate, the author algorithm has better performance in comparison to our algorithm (HCFm).The results show that our proposed algorithm has problem in false negative rate of attack traffic. In the contrary, in the beginning of action state , the author algorithm has better performance in the detection accuracy (**0,93**(HCF) against **0,85**(HCFm)).See figure 4.14 above.

2-3: Performance measures

To solve the problem of traffic flooding attacks based DoS/DDoS which uses the technique of IP spoofing, previous researches adopted various mechanisms to detect this type of attack such as Hop-count Filtering [41] and Hop-count Filtering [38] which have their respective advantages and weaknesses . Nevertheless, the detection of the precedent attack is still complex. When a user sends many packets with a spoofed IP addresses, it will be difficult to model the traffic flow according to the mechanisms above as the packet flow will be in bursty (non-linear) during the attacking period. On the other hand, some users send this attack under the normal flow rate which causes difficulty in accurately diagnosing the traffic flow and detecting the attack due to *false positive* and *false negative* values. As a result, a large number of pointless requests is sent, leading to the shutdown of available services .The performance of flooding attacks based DoS/DDoS detection schemes may be measured in terms of false positive and false negative detection and accuracy of detection at high rates and low rates. The performance comparison is summarized in Table 4.5.

Agorithms	False Positive (FP)	False Negative (FN)	Accuracy Detection	
			Detection High Rate Traffic	Detection Low Rate Traffic
HCF[41]	**10%**	**5%**	**86%**	**93%**
HCFm Proposed-algorithm	**≈0%**	**8,9%**	**≈92%**	**85%**
HCF[38]	**10%**	**5%**	**90%**	**/**

Table 4.5 : Performance Measures Comparison.

4-Attackers Implementation

To implement Attackers ,we use a small java program to simulate such those attacks which are built by using a combination of libraries like Jpcap , WinPcap to send spoofed packets to the target server (victim).We have implemented three types of attackers as follows:

4.1-Syn flood based Dos

```
// tcp header fields
int source = 5001 ;  // source port
int dest = 80 ; // destination port
int seq =1401;  //sequence number
int ack_seq = 1501;  // ACKnowledged number
boolean   offset_res =false;
//tcp flags
boolean fin = false; // fin flag
boolean syn = true;  // syn flag
boolean rst = false;  // rst flag
boolean psh =false;  // psh flag
boolean ack = false;  // ack flag
boolean urg = false;  // urg flag
boolean urg_ptr=false;
int window = 10; //  maximum allowed window size
int checksum = 10; //checksum
//create a TCP packet with specified port numbers, flags, and other parameters
TCPPacket p=new TCPPacket(source,dest,seq,ack_seq,urg,ack,psh,rst,syn,fin,offset_res,urg_ptr,window,checksum);
// ip header fields
byte version = 4; // version IPv4
int ihl = 5; //ip header length
int priority=0;  //priority
boolean d_flag=false; // IP flag bit: [D]elay
boolean t_flag=false; //IP flag bit: [T]hrough
boolean r_flag=false; //IP flag bit: [R]eliability
int rsv_tos=0;  //Type of Service (TOS)
boolean rsv_frag=false; //Fragmentation Reservation flag
boolean dont_frag=false; //Don't fragment flag
boolean more_frag=false;  //More fragment flag
int  offset=0;
int ident = 54321;  //Id of this packet
short TTL = 255; // time to live
short protocol = IPPacket.IPPROTO_TCP ; //This value is ignored when this packets inherits a higher layer protocol(e.g. TCPPacket)
short check = 10; //checksum
InetAddress saddr = InetAddress.getByName("192.168.1.2");  //Spoof the source ip address if you want to //Source  IP address
InetAddress daddr = InetAddress.getByName( "192.168.1.3" );//Destination IP address
```

Figure 4.15 : Excerpt from the source code of Syn flood Dos Attacker.

4.2-ICMP flood based Dos

```
//create ICMP header
short protocol =IPPacket.IPPROTO_ICMP;
InetAddress saddr = InetAddress.getByName("192.168.1.2");  //Spoof the source ip address if you want to //Source  IP address
InetAddress daddr = InetAddress.getByName( "192.168.1.3" );//Destination IP address
p.setIPv4Parameter(0,false,false,false,0,false,false,false,0,1010101,100,protocol,saddr,daddr);
//set the data field of the packet
p.data="data".getBytes();
//create an Ethernet packet (frame)
EthernetPacket ether=new EthernetPacket();
ether.frametype=EthernetPacket.ETHERTYPE_IP;
//set source and destination MAC addresses
String strdst = new String("00:21:9B:D7:98:54");
ether.dst_mac = strdst.getBytes();
String strsrc = new String("00:0C:F1:DE:00:D0");
ether.src_mac = strsrc.getBytes();
//set the datalink frame of the packet p as ether
p.datalink=ether;
//send the packet p
```

Figure 4.16 : Excerpt from the source code of ICMP flood Dos Attacker.

4.3-UDP flood based Dos

```
// UDP  header fields
short protocol = IPPacket.IPPROTO_UDP ;
InetAddress saddr = InetAddress.getByName("192.168.1.2"); //Source  IP address .......Spoof the source ip address if you want to
InetAddress daddr = InetAddress.getByName( "192.168.1.3" );//Destination IP address
UDPPacket p=new UDPPacket(12345,54321);
p.setIPv4Parameter(0,false,false,false,0,false,false,false,0,1010101,100,protocol,saddr,daddr);
p.data="data".getBytes();
EthernetPacket ether=new EthernetPacket();
ether.frametype=EthernetPacket.ETHERTYPE_IP;
//set source and destination MAC addresses
String strdst = new String("00:21:9B:D7:98:54");
ether.dst_mac = strdst.getBytes();
String strsrc = new String("00:0C:F1:DE:00:D0");
ether.src_mac = strsrc.getBytes();
//set the datalink frame of the packet p as ether
p.datalink=ether;
//send the packet UDP
```

Figure 4.17: Excerpt from the source code of UDP flood Dos Attacker.

5- SYN Flood Dos packet with SYN flag set to 1:

The following figure illustrates how the precedent Syn flood Dos Attacker implementation works correctly or non when we initialize the three-way handshack TCP connection (SYN flag = 1) or (SYN flag =0 in case of non three-way handshack TCP connection).

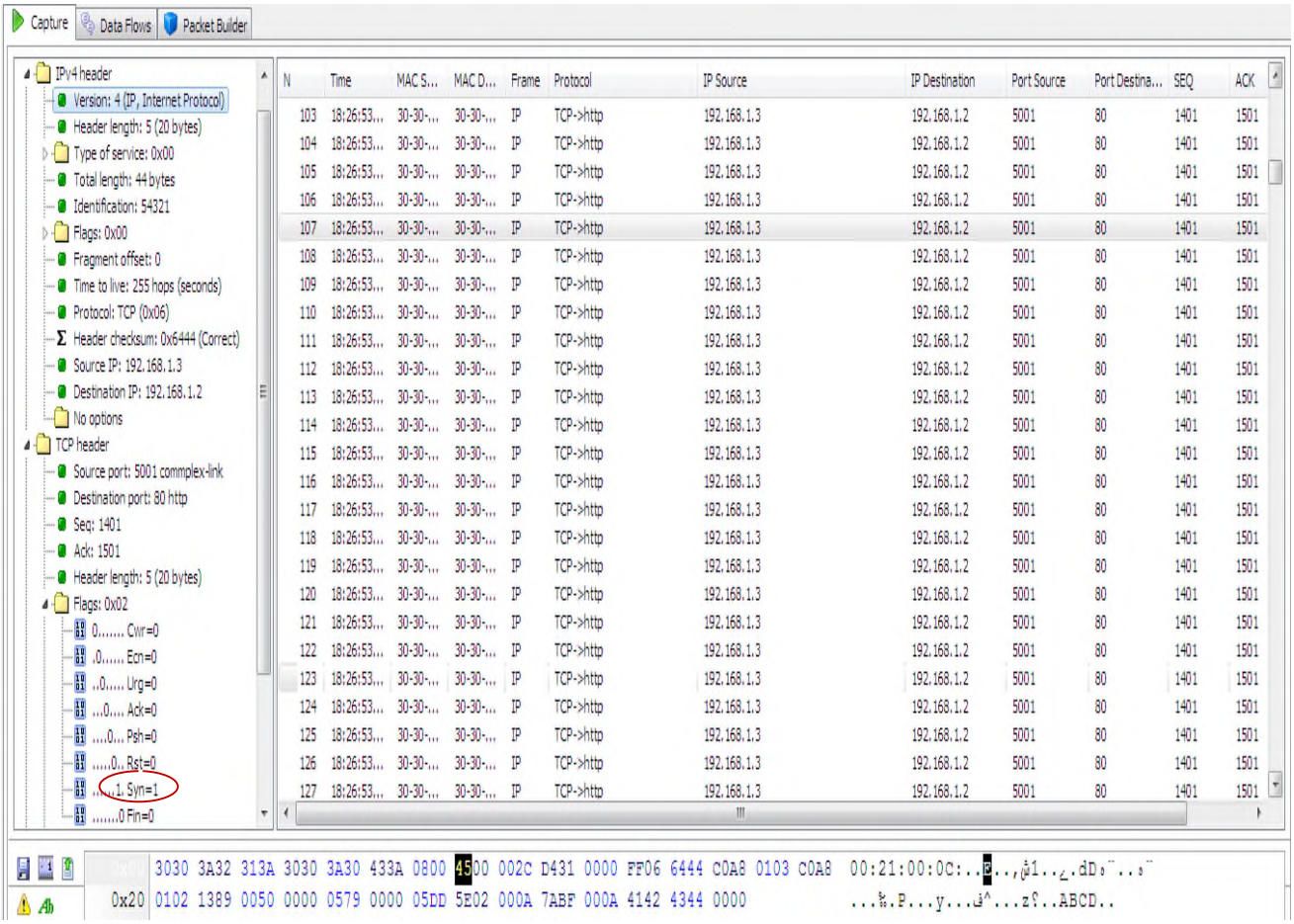


Figure 4.18 :SYN Flood Dos packet with SYN flag set to 1

6-Few excerpts from the source code of HCFm algorithm

6.1-NewEntryTable

The following source code which was extracted from the source code HCFm algorithm for illustrating how to add a new entry in IP2HCm table when SYN flag equals to 1 and the source IP address doesn't exist also in the IP2HCm table (see the figure bellow).

```
//add new entry in the IP2HC table.
public  static void NewEntryTable( InetAddress srcIP,int hc)
 {
    Connection connection = null;
    Statement stmt = null;
    ResultSet res =null;
    try {
     Class.forName("com.mysql.jdbc.Driver");
     connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/ddos", "root", "haddadi");
     stmt = connection.createStatement();
     String query="INSERT INTO ip2hcm(src_ip,hop_count,count) VALUES ('"+srcIP+"','"+hc+"',1)";
     stmt.executeUpdate(query);
       } catch (Exception e) {e.printStackTrace();}
       finally {//finally block used to close resources
           try { if (res != null) res.close(); } catch (SQLException e) { e.printStackTrace(); }
           try { if (stmt != null) stmt.close(); } catch (SQLException e) { e.printStackTrace(); }
           try { if (connection != null) connection.close(); } catch (SQLException e) { e.printStackTrace(); }
          }
 }
```

Figure 4.19 : Excerpt from the source code of HCFm Algorithm.

6.2-Black list database

The following source code which was extracted from the source code HCFm algorithm in the end of slot time (an interval of time) for illustrating how our HCFm algorithm delete the source IP address from IP2HCm table when its frequency count greater than the threshold value.

```
//testing in the database"ddos" ,if there is count >=threashold during the precedent slottime
    public   void  Blacklist() { // IP packet black list
        Connection connection = null;
        Statement stmt = null,stmte=null,stmt1=null;
        ResultSet res =null;
        try {
         Class.forName("com.mysql.jdbc.Driver");
         connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/ddos", "root", "haddadi");
         stmt = connection.createStatement();
         stmte = connection.createStatement();
         stmt1 = connection.createStatement();
         String query="SELECT src_ip FROM ip2hcm WHERE count >= 20";
         res =stmt.executeQuery(query);
           while( res.next()){// Move the cursor to the next row
           String sip = res.getString("src_ip");
           System.out.println(sip);
           String req="INSERT INTO blacklist (src_ip) VALUES ('"+sip+"')";
           stmte.executeUpdate(req);
           String req2="delete from ip2hcm where src_ip='"+sip+"'";//delete src_ip which existe in the black list from IP2HCm
           stmt1.executeUpdate(req2);
           }
        stmte = connection.createStatement();
         String requete="UPDATE  ip2hcm  SET count =0 ";
         stmte.executeUpdate(requete);
           } catch (Exception e) {e.printStackTrace();}
            finally {//finally block used to close resources
                  try { if (res != null) res.close(); } catch (SQLException e) { e.printStackTrace(); }
                 try { if (stmt != null) stmt.close(); } catch (SQLException e) { e.printStackTrace(); }
                 try { if (connection != null) connection.close(); } catch (SQLException e) { e.printStackTrace(); }
              }
       }
```

Figure 4.20: Excerpt from the source code of HCFm algorithm

7-Conclusion

In this proposed algorithm, We have simulated the proposed algorithm inspired from HCF [41] in the cloud lab that is closest to cloud environment .The purpose of our proposed algorithm is generally to prevent cloud computing against flooding attacks based DoS/DDoS such as IP spoofing which is used by Syn flood attack.

We have found our proposed algorithm which is called HCFm better than that which is already discussed in the chapter 3 [41] in term of computation time , updates of IP2HCm table , specially the detection accuracy which is approximately equals to 92% and reduces the false negatives rate to 0% and a false positives rate to 8,9 and also it (HCFm)detects attackers that they don’t use IP spoofing.

General Conclusion

Cloud computing is constantly evolving; which means that as the technology matures, new types of security issues will arise.From a security point of view, cloud computing contains numerous vulnerabilities , threats and risks. This work presents a survey of cloud computing security ,the major form of flooding attacks based DoS/DDoS ,state of the art about flooding attack solutions that need to be applied in order to reach an objective for securing cloud computing ,but those algorithms are not capable to solve the security problems with high detection accuracy and lower false alarms.

We have ameliorated an algorithm that should be applied and implemented in the cloud lab which is closest to the cloud environment .This algorithm inspired by the Hop Count Filtering (HCF) technique[41] that works in the IaaS (Infrastructure as a service) layer of the cloud stack and changes the alert state of HCF to include all the possible available Hop Count values ,and statistical analysis such as threshold for classifying every IP packet in the blacklist when its number is greater than a predefined value which is called the threshold during a slot time(an interval of time) and when it’s a spoofed packet, but we must remove an IP from the blacklist if it has not sent a spoofed packet in X amount of time. Compared to the original HCF technique and its variants, our proposed algorithm decreases the false alarm rate and consequently increases the detection accuracy of flooding attacks to 90%. Our proposed method performs in general better than HCF algorithm and its variants.

Like future work for our algorithm :

While our algorithm certainly works in a cloud lab environment , it would be better to test using environments closer to reality that are mentioned as follows:

- Try to create virtual machines (VMs) in two different cloud environments, and test between them in a way that simulates real world flooding attacks based DoS/DDoS at a lower scale.

- Get permission from the cloud service providers to test our algorithm by explaining our research goal and algorithm to them.

-Combine two algorithms that are already discussed in the state of the art for may be finding good results.

References

[1] M. LEMOUDDEN, N. BEN BOUAZZA, B. EL OUAHIDI, D. BOURGET, " A SURVEY OF CLOUD COMPUTING SECURITY OVERVIEW OF ATTACK VECTORS AND DEFENSE MECHANISMS ", Journal of Theoretical and Applied Information Technology , pp.325-330, Vol. 54 No.2, 20th August 2013.

[2] The Open Group, "Risk taxonomy", 2009.

[3] Tarun Karnwal , T.Sivakuma ,G Aghila ,"A Comber Approach to Protect Cloud Computing against XML DDoS and HTTP DDoS attack",IEEE Students' Conference on Electrical, Electronics and Computer Science,pp.1-5, 2012.

[4] ISO 27005, Information technology - Security techniques, "Information security risk management guideline", 2011.

[5] K.Decker,"What Joni Mitchell might say about cloud computing",2010.Available http://decker.com/blog/2010/05/what-joni-mitchell-might-sayabout-cloud-computing/

[6] Chimere Barron, Huiming Yu and Justin Zhan , "Cloud Computing Security Case Studies and Research", Proceedings of the World Congress on Engineering, London , U,K ,pp.17-21, July 3 - 5, 2013.

[7] A. Hickey, "Researchers uncover 'massive security flaws' in Amazon cloud" , Available:http://www.crn.com/news/cloud/231901911/researchers-uncovermassive-security-flaws-in-amazon-cloud.htm

[8] M. Kronfield, "Treasury Dept. has cloud hacked", Available: http://www.gsnmagazine.com/article/20691/treasury_dept_has_cloud_hacked

[9] D. Fisher, "Attackers using Amazon cloud to host malware", Available: http://threatpost.com/en_us/blogs/attackers-using-amazon-cloud-hostmalware-060611

[10] I. Kotenko, M. Stepashkin, and E. Doynikova, "Security analysis of information systems taking into account social engineering attacks", *IEEE 19th International Eurimicro Conference on Parallel, Distributed,and Network-Based Processing*, 2011.

[11] J. Pepitone, "Hack attack exposes major gap in Amazon and Apple security", Available: http://money.cnn.com/2012/08/07/technology/mathonan-hacked/index.htm

[12] Cloud Security Alliance, " Top threats to cloud computing ", *Cloud Security Alliance*, March 2010.

[13] M. Prince, "The four critical security flaws that resulted in last Friday's hack", Available:http://blog.cloudflare.com/the-four-critical-securityflaws-that-resulte

[14] L.Tung,"CloudFare boss's Gmail hacked in redirect attack on 4Chan",Available: http://www.cso.com.au/article/426515/cloudflare_boss_gmail_hacked_redirect_attack

_4chan/

[15] D. Kerr, “Dropbox confirms it was hackers, offers users help”, Available: http://news.cnet.com/8301-1009_3-57483998-83/dropbox-confirms-itwas-hacked-offers-users-help/

[16] Neha Jain and Gurpreet Kaur ‘Implementing DES Algorithm in Cloud for Data Security” VSRD International Journal of CS & IT Vol. 2 Issue 4, 2012, pp. 316-321.

[17] Kiril, “LassPass possibly hacked, cloud security concerns on the rise”,Available: http://www.cloudtweaks.com/2011/05/lastpass-possiblyhacked-cloud-security-concerns-on-the-rise/

[18] PC World Staff,“Cloud computing used to hack wireless passwords”,Available: www.pcworld.com/article/216434/cloud_computing_used_to_hack_wireless_passwords.html

[19] Niraj Suresh Katkamwar, Atharva Girish Puranik and Purva Deshpande,” Securing Cloud Servers against Flooding Based DDoS Attacks”, International Journal of Application or Innovation in Engineering & Management (IJAIEM),,Volume 1,pp.50-55, November 2012.

[20] Shweta Tripathi, Brij Gupta*, Ammar Almomani, Anupama Mishra, Suresh Veluru, “Hadoop Based Defense Solution to Handle Distributed Denial of Service (DDoS) Attacks ”,*Journal of Information Security*, pp.150-164, July 2013.

[21] E. Alomari, S. Manickam, B. B. Gupta, S. Karuppayah and R. Alfaris, “Botnet-Based Distributed Denial of Ser-vice (DDoS) Attacks on Web Servers: Classification and Art,”*International Journal of Computer Applications*,Vol.49,No.7,2012,pp.24-32.

[22] Peter Mell and Timothy Grance. The NIST Definition of Cloud Computing. Technical Report SP 800-145 Draft, National Institute of Standards and Technology, Information Technology Laboratory, January 2011.

[23] Nisha H. Bhandari , “Survey on DDoS Attacks and its Detection & Defence Approaches”,International Journal of Science and Modern Engineering (IJISME), Volume-1, pp.67-71, February 2013.

[24] JelenaMirkovic, Janice Martin and Peter Reiher, "A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms", Computer Science Department,University of California, Los Angeles.

[25] Zhou,R.and K. Hwang,"Trust-Preserving Overlay Networks for Global Reputation Aggregation in Scalable P2P Systems ", IEEE transaction on Parallel and Distributed Systems, (TPDS), revised March 2006.

[26] Houle, K., G. Weaver, N. Long, and R. Thomas, "Trends in Denial of Service Attack Technology", CERT Coordination Center Document, 2001, www.cert.org/archive/pdf/

[27] Kim, W., Jeong, Ok-Ran, Kim, C., So, J. The dark side of the Internet: Attacks, costs and responses. Information Systems, *36(3), (2011), pages: 675–705*.

[28] S. Garfinkel and G. Spafford, "Practical Internet and UNIX Security," O'Reilly Media, 1996

[29] K. Zetter, "Lazy Hacker and Little Worm Set off Cyber- war Frenzy," 2009. http://www.wired.com/threatlevel/2009/07/mydoom/

[30] L. Greenemeier, "Estonian Attacks Raise Concern over Cyber "Nuclear Winter"," Information Week, 2007. http://www.informationweek.com/estonian-attacks-raise-concern-over-cybe/199701774

[31] J. Vijayan, "Mydoom Lesson: Take Proactive Steps to Prevent DDoS Attacks," 2004.
http://www.computerworld.com/s/article/89932/Mydoom_lesson_Take_proactive_steps_to_prevent_DDoS_attacks?%20taxonomyId=017

[32] "Powerful Attack Cripples Internet," 2002. http://www.greenspun.com/bboard/q-and-a-fetch-msg.tcl msgid=00A7G7 .

[33]Yahoo on Trail of Site Hackers," Wired.com, 2000. http://www.wired.com/techbiz/media/news/2000/02/34221

[34] Farzad Sabahi ,*Member, IEEE* , " Secure Virtualization for Cloud Environment Using Hypervisor-based Technology ",*International Journal of Machine Learning and Computing, Vol. 2, No. 1, pp.39-45 , February 2012.*

[35] F. Sabahi, "Cloud computing security management",2nd International conference on Engineering Systems Management and Its Applications
(ICESMA), March 2010.

[36] F. Shaihk and S. Haider, "Security threats in cloud computing", IEEE 6th International Conference on Internet Technology and Secured Transactions
, December 2011.

[37] Poonam Yadav and Sujata ," Security Issues in Cloud Computing Solution of DDOS and Introducing Two-Tier CAPTCHA ",International Journal on Cloud Computing: Services and Architecture (IJCCSA) ,pp.25-40,June 2013

[38] Cheng Jin, Haining Wang, Kang G. Shin," Hop-Count Filtering: An Effective Defense Against Spoofed DDoS Traffic ",CCS'03, Washington,pp-30-41, October 27–31, 2003.

[39] Wang ,H., Jin, C., and Shang, K.2007.Defense Against Spoofed IP Traffic Using Hop-Count Filtering., IEEE/ACM Trans. Networking, vol. 15,no. 1, 40-53.

[40] The Swiss Education and Research Network. Default TTL values in TCP/IP, 2002. Available: http://secfr.nerim.net/docs/fingerprint/en/ttl default.html.

[41] Vikas Chouhan & Sateesh Kumar Peddoju, "Packet Monitoring Approach to Prevent DDoS Attack in Cloud Computing ", IJCSEE,pp.38-42,2012.

[42] KrishnaKumar, B., Kumar, P.K., Sukanesh, R. , "Hop Count Based Packet Processing Approach to Counter DDoS Attacks", International Conference on Recent Trends in Information, Telecommunication and Computing (ITC), pp.271-273, 2010.

[43] Swain, B.R.; Sahoo, B., "Mitigating DDoS attack and Saving Computational Time using a Probabilistic approach and HCF method" in Advance Computing Conference, 2009. IACC 2009. IEEE International , pp.1170-1172, 2009.

[44] Mopari, I.B., Pukale, S.G., Dhore, M.L., "Detection and defense against DDoS attack with IP spoofing", International Conference on Computing,Communication and Networking, 2008.

[45] Zhijun W., Zhifeng C., "A Three-Layer Defense Mechanism Based on WEB Servers Against Distributed Denial of Service Attacks", First International Conference on Communications and Networking in China,pp.1-5, 2006.

[46] Wang, Xia, Li, Ming, Li, Muhai, "A scheme of distributed hop-count filtering of traffic," IET International Communication Conference on Wireless Mobile and Computing (CCWMC), pp.516-521, 7-9 , 2009.

[47] Mehdi Ebady Manna and Angela Amphawan, "REVIEW OF SYN-FLOODING ATTACK DETECTION MECHANISM " International Journal of Distributed and Parallel Systems (IJDPS),pp.99-117, January 2012.

[48] Charalampos Patrikakis, Michalis Masikos, and Olga Zouraraki ,"Distributed Denial of Service Attacks.

[49] Simona RAMANAUSKAITĖ “Modeling and research of Distributed Dienal of service attack “Available:http://vddb.laba.lt/fedora/get/LT-eLABa-01:E.02~2012~D_ 201207 23_10503170003/DS.005.1.01.ETD

[50]- Common DDoS Attacks :www.nsfocus.com

[51] W.Haining ,et al .,”Defence Against Spoofed IP Traffic Using Hop-Count Filtering ,”Networking ,IEEE / ACM Transactions on ,vol .15 ,pp. 40-53 ,2007.

[52] netresearch .ics.uci.edu/kfujii/Jpcap/doc/tutorial/index.html

[53] Gnanambal Chithambaram,Sandeep Dubey Smrithi Bamenkula,Subraja Krishnamurthy, Sucheta PKodali, “ IP monitoring and filtering”

[54] www.winpcap.org/

Résumé :

Cloud computing est la convergence et l'évolution de plusieurs concepts de virtualisation, des applications distribuées, grille computing et d'autres pour permettre une approche plus souple pour le déploiement et la mise à l'échelle des applications et des services délivrer via l'Internet. Le Cloud Computing , comme un service public, a le potentiel de transformer l'IT de l'industrie par la fournissant des softwares, des plates-formes et des infrastructures comme un service, et en révolutionnant les moyens et les manières traditionnelles de gestion d'IT de l'entreprise .Dans les dernières années le cloud computing était exposé aux attaques les plus dangereux qui sont les attaques floodings ,et qui sont une nouvelle forme d'attaques en utilisant IP spoofing. Dans ce papier ,nous proposons un algorithme (HCFm) ,inspiré de la technique de filtrage par nombre de sauts (HCF :Hop Count Filtering) qui change l'état d'alerte de HCF pour indiquer toutes les possibilités des valeurs de nombres de sauts (HC) et les adresses IP sources par extraction à partir des en-têtes TCP et IP ,UDP et IP ,et ICMP et IP, et le seuil pour vérifier le nombre de chaque parquet pendant une tranche de temps (slot time) et ajouter l'adresse IP source de l'attaquant dans la table de la liste noire s'il dépasse le seuil pour minimiser le temps de traitement de chaque paquet d'attaque durant les attaques par inondation(flooding), mais il libérer une adresse IP ,si n'est pas utilisée durant une quantité de temps bien déterminée .Par comparaison avec l'algorithme original (HCF),notre algorithme ,qui s'appelle HCFm, apparait très efficace en terme de taux de faux positifs (FP) et le pourcentage de détection avec un taux de 92%.

.

Mots clés : Cloud computing , architecture VMware,HCF , IP spoofing , liste noire , sécurité de cloud computing , HCFm , attaques floodings.

ABSTRACT

Cloud Computing is one of today's most exciting technologies ,because it can reduce the cost and complexity of applications. On the other hand, such complex and distributed architectures become an attractive target for attacks .Flooding attacks based DoS represent a serious danger which can deny the legitimate users access to the service delivered by cloud. In this paper,we propose an algorithm, inspired by the Hop Count Filtering(HCF) technique that changes the alert state of HCF to include all the possible available Hop Count values ,and statistical analysis such as threshold for classifying every IP packet in the black list when its number is greater than the threshold during a slot-time, but we must removing an IP from the blacklist if it has not sent a spoofed packet in X amount of time. Compared to the original HCF method and its variants, our proposed method performs better than HCF algorithm and its variants,which achieves high detection accuracy (92%) with fewer false alarms.

Keywords :cloud computing ,cloud security, VMware architecture, Flooding attacks, HCFm , Blacklist ,HCF , IP spoofing.