A Game Theoretic Model for Security in Cloud Environment

A Thesis

Submitted for the award of degree of

DOCTOR OF PHILOSOPHY

Submitted By

Komal Singh Gill

(901603008)

Under the supervision of

Dr. Sharad Saxena

Associate Professor, CSED

TIET, Patiala

Dr. Anju Sharma

Assistant Professor

MRSPTU, Bathinda

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology, Patiala

(Deemed to be University)

August 2023

Candidate Declaration

I hereby certify that the work, which is being presented in the thesis, entitled **A Game Theoretic Model for Security in Cloud Environment**, in partial fulfillment of the requirements for the award of the degree of **Doctor of Philosophy** and submitted in Computer Science and Engineering Department, Thapar Institute of Engineering and Technology (Deemed University), Patiala, India is an authentic record of my own work carried out under the supervision of **Dr. Sharad Saxena** and **Dr. Anju Sharma**. I have also cited the reference about the text(s)/figure(s)/table(s) from where they have been taken.

The matter presented in this thesis has not been submitted elsewhere for the award of any other degree or diploma from any institution.

Date: **Komal Singh Gill**
Regd. No: 901603008

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

Dr. Sharad Saxena
Associate Professor
Computer Science and Engineering Department
Thapar Institute of Engineering and Technology
Patiala

Dr. Anju Sharma
Assistant Professor
Department of Computational Sciences
Maharaja Ranjit Singh Punjab Technical University
Bathinda

To the farms, where my roots are deeply spread,
allowing me to reach great heights in life.

Acknowledgement

I am grateful to the Almighty for granting countless blessings, opportunities, knowledge, and strength, without which I could not have accomplished this work.

I am indebted to my supervisors, Dr. Sharad Saxena (Associate Professor, TIET) and Dr. Anju Sharma (Assistant Professor, MRSPTU), for their continuous guidance and support. I have benefitted greatly from their wealth of knowledge and meticulous editing. The time and energy they devoted to this work have immensely improved the quality of this research. Their faith in me and encouragement have helped me stay focused over the years. Their honest approach towards research and life is a source of inspiration, which I hope to carry throughout my career.

I offer my sincere gratitude to our director Prof. Padmakumar Nair, Dr. N. Tejo Prakash (Dean, Research Sponsored Projects) and Dr. Shalini Batra (Professor and Head, CSED) for providing the necessary academic and administrative assistance in the completion of this work. I am thankful to my Doctoral committee members- Dr. Inderveer Chana (Professor, CSED), Dr. Neeraj Kumar (Professor, CSED), and Dr. Mukesh Singh (Professor, EIED) for ensuring the progress of my research work. I am also thankful to Ph.D. Coordinator Dr. Sushma Jain (Associate Professor, CSED) and all the faculty and staff members of CSED for always helping me. My special appreciation to Dr. Prashant Singh Rana (Associate Professor, CSED) for his help and valuable suggestions.

It would not have been possible without my mother's unconditional love and belief in me. My deepest thanks to my dear father, mother, and sister for encouraging me to start this journey and providing me with every possible support. The love showered by my nephew, niece, brother-in-law, and cousins have also been a great motivation in this journey. I will always be grateful to them for their patience and compassion in every sphere of my life.

I wish to extend my thanks to all the lab mates and peers for their constant support and for making this journey memorable. I treasure Arun Rana, Arwinder Dhillon, Mexson Fernandes, Rajesh Kondabala, and Parampreet Kaur for their support and encouragement. I gratefully acknowledge their cooperation and blessings, which motivated me throughout this degree.

Komal Singh Gill
August, 2023

Abstract

Cloud computing has become an increasingly popular technology for businesses and individuals. With the ability to store and access data and applications from any location with an internet connection, the benefits of cloud computing are numerous. However, security remains a significant concern for many organizations utilizing cloud services. The primary security risks associated with cloud computing include external and internal attacks. External attacks occur due to unauthenticated access to sensitive data, insecure Application Programming Interfaces, and misconfiguration of security settings and can lead to loss of control over data and systems. Another critical component of cloud security is managing access to data and applications, which can cause internal attacks. This involves controlling who has access to what data and what actions they are allowed to take with that data. These can lead to regular attacks like Denial of Services, Brute force, malware injection, ransomware *etc.* New and sophisticated attacks can be performed, which makes cloud providers very difficult to secure cloud resources.

To mitigate these risks, cloud service providers implement robust security measures such as encryption, authentication, and access controls and maintain high vigilance against potential threats. Also, Intrusion Detection System (IDS) is widely used to detect security attacks. The techniques like Machine Learning (ML), Deep Learning (DL) are used to enhance the accuracy of the IDS. But maintaining security in wide networks such as the cloud is very tedious. So, to enhance the security of the cloud further, the behavior of the attacker and defender can be analyzed. To model the interactions between the attacker and defender, Game Theory can be used. This mathematical framework allows researchers to analyze and address the complex security challenges, and the optimal strategies for each side in a given security scenario can be delineated. Taking into account factors such as the costs of launching an attack, the likelihood of success, and the consequences of a successful attack can provide valuable insights into the strengths and weaknesses of different security measures and inform the development of effective security policies and technologies.

To overcome the above-mentioned challenges, in this thesis, four different game-theoretic models are proposed and developed, namely Game Theoretic Model for Cloud Security *(GTM-CSec)*, Bayesian Optimized Game Theoretic Approach *(BOGTA)*, Game Theoretic Approach to enhance IDS detection*(GTA-IDS)*, and Non-Cooperative Game Theoretic Model *(NCGTM)*. (*GTM-CSec*) is developed to

defend the external attacks. This is the first model and it is modeled between the attacker and the defender competing against each other to gain maximum payoff. Different strategies for both players are delineated, and a mixed strategy Nash Equilibrium is attained to conclude the game. The payoffs of the attacker and defender are analyzed and compared. Out of the different strategies available for defenders, it chooses the best one to defend against the attack.

To increase the efficiency of *GTM-CSec* Model and to increase the accuracy of the IDS, a second model is devised namely *BOGTA*. It covers every possibility of working on three modules (signature, anomaly, or honeypot) of the defender system in a single cycle with the graphical Game Theory method that reduces the model's time complexity. To optimize the IDS to a greater extent, a Bayesian Optimised DNN is used, and hyperparameter tuning is also done. Three different datasets for different detection modules are considered to train and test the IDS. The results show that *BOGTA* has taken significantly less time to decide than *GTM-CSec* and performed well with an improvement of 9.66%, 3.75%, and 4.16% in detection rates of the signature, anomaly, and honeypot modules, respectively. The accuracy is increased by 10.29%, 2.1%, and 3.41%, and the False Positive Rate (FPR) is reduced to 0.01%, 0.026%, and 0.138% for the three modules. The higher values of the detection rate and lower values of the FPR depict the adequate performance of *BOGTA*.

The third model, *GTA-IDS* is devised to detect internal attacks, in which information theory is used along with the game theory. In this model, the information theory tracks the abnormal traffic flow and is sent to the GTA-IDS for further analysis. A game is modeled between the malicious node and the defender system. Different strategies for both players are also delineated, and a mixed strategy Nash Equilibrium is attained to conclude the game. The model is implemented on a benchmark real-time NSL-KDD dataset to check the detection rate and FPR of the defender system. With the addition of the *GTA-IDS* module, the detection rate of the IDS comes to be 99.5%, and FPR comes to be 0.07% which is better than the existing models like *KNN, CNN etc.*.

The fourth model namely *NCGTM* is devised to enhance the decision-making process of the hybrid IDS which uses signature-based, anomaly-based, and a hybrid of signature and anomaly-based modules. The Attacker's strategies are analyzed against the strategies of the IDS to make better predictions. Different machine learning models train the IDS to increase its detection rate. The *NCGTM* is tested on a real-time dataset and compared with the existing models. The overall performance of the IDS improves with a higher detection rate of 99% and a lower

FPR of 0.01.

The proposed models incorporate the interactions between the attacker and the defender, who have conflicting goals of maximizing their profits. The models analyze the optimal strategies for both players, the consequences of security breaches, and the incentives for cooperation and competition. The Machine Learning and Deep Learning techniques enhance the accuracy, detection rate and lower the FPR of the IDS. The results obtained from the four proposed models show improvement in the security of cloud services.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AD	Anomaly Detection
AES	Advanced Encryption Standard
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
APTs	Advanced Persistent Threats
ASCII	American Standard Code for Information Interchange
AWS	Amazon Web Services
BBA	Binary Bat Algorithm
BES	Bald Eagle Search Optimization
BNE	Bayesian Nash Equilibrium
BOGTA	Bayesian Optimized Game Theoretic Approach
CC	Command and Control
C2aaS	Confidentiality-based Classification-as-a-Service
CCS	Cloud Computing Service
CRDO	Chaotic Red Deer Optimization
CKMS	Cloud Key Management System
CNN	Convolutional Neural Network
CNS	Customized Network Security
CSA	Cloud Security Alliance
CSP	Cloud Service Provider
DAE	Deep Autoencoder
DARPA	Defence Advanced Research Projects Agency
DB	Data Base
DoS	Denial of Service
DDoS	Distributed Denial of Service
DKNN	Deep Kronecker Neural Network
DL	Deep Learning
DLM	Deep Learning Model
DNN	Deep Neural Network
DOS	Denial Of Service
DR	Detection Rate
DT	Decision Tree
ECC	Elliptic Curve Cryptography

ECSM-QKDP	Enhanced Cloud Security Model with Quantum Key Distribution Protocol
ExpSSOA	Exponential Shuffled Shepherded Optimization
EWMA	Exponential Weighted Moving Average
FEFS	Filter-based Ensemble Feature Selection
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GA	Genetic Algorithm
GBM	Gradient Boosting Machine
GCE	Google's Computing Engine
GQM	Goal Question Metric
GT	Game Theory
GTM-CSec	Game Theoretic Model for Cloud Security
GTA-IDS	Game Theoretic Approach for Intrusion Detection System
FP	False Positive
H-DEDU	Hybrid Encrypted Cloud Data Deduplication
HDFS	Hadoop Disributed File System
HHO	Harris Hawks Optimization
HIDS	Hybrid Intrusion Detection System
HLDNS	Hypervisor-level Distributed Network Security
HP	Hewlett Packard
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IDE	Integrated Development Environment
IDPS	Intrusion Detection and Prevention System
IDS	Intrusion Detection System
IHC	Improved Huffman Coding
IHO	Improved Heap Optimization
IoT	Internet of Things
IRS-AES	Improved Robust S-box-based Advanced Encryption Standard
ISAGA	Improved Self-Adaptive Genetic Algorithm
ISO	International Organisation for Standardisation
IT	Information Technology
JA	Jaya Algorithm
KDB	Knowledge Database
KNN	K-Neares Neighbor
LFSE	Lightweight Keyword Searchable Encryption

LR	Logistic Regression
LSTM	Long Short Term Memory
LU	Legitimate User
MAC	Media Access Control
MDP	Markov Decision Process
MECC	Modeified Elliptic Curve Cryptography
ML	Machine Learning
MLPAM	Machine Learning and Probablistic Analysis-based Model
MORE	Matrix Operation-based Randomization and Encipherment
NCGTM	Non-Cooperative Game Theoretic Model
NB	Naive Bayes
NE	Nash Equilibrium
NIDS	Network Intrusion Detection System
ODAC	Ontology Based Data Access Control
OS	Operating System
PaaS	Platform as a Service
PDCA	Plan Do Check Act
PHR	Patient Health Record
PoC	Proof-of-Cocept
PSO	Particle Swarm Optimization
QoS	Quality of Service
RBAC	Role Based Access Control
ReLU	Rectified Linear Unit
RF	Random Forest
RKO	Runge-Kutta Optimization
RNN	Recurrent Neural Network
RSA	Rivest-Shamir-Adleman
SaaS	Software as a Service
SAA	Self-Adaptive Genetic Algorithm
SaE-ELM	Self-adaptive Evolutionary Extreme Learning Machine
SA-ODAC	Security-aware mechanism and ontology-based Data Access Control
SARSA	State Action Reward State Action
SAT	Secure Awareness Technique
SD	Signature-based Detection
SDN	Software Defined Networks
SG	Smart Grid
SI	Swarm Intelligence

SLA	Service Level Agreement
SMBO	Sequential Model-Based Optimization
SPA	Stateful Protocol Analysis
S-SCAE	Sparse Autoencoder and Stacked Contractive Autoencoder
SSDLC	Secure Software Development Life Cycle
SSO	Shark Smell Optimization
SSOA	Shuffled Shepherded Optimization Algorithm
SVM	Support Vector Machine
TDO	Tasmanian Devil Optimization
TPA	Third Party Auditor
URL	Uniform Resource Locator
VM	Virtual Machine
VMM	Virtual Machine Manager
VMVM	Virtual Machine Vitality Measure
VNI-c	Virtual Network Interface for connection
VNI-p	Virtual Network Interface for port mirroring
WC-SM-AI	Wireless Communication-assisted Security Management System
XML	Extensible Markup Language

Chapter 1
Introduction

Cloud computing and its multifarious services are proven a boon for the setup of digital businesses. The end-users and organizations need not worry about acquiring and maintaining infrastructure, as the Cloud provides many services and resources. The increasing number of devices and users fabricates different challenges for the cloud computing paradigm. The primary security risks associated with cloud computing include unauthorized access to sensitive data, data breaches, and loss of control over data and systems. Also, internal attacks can be performed on the Cloud with social engineering. Misconfiguration of security settings and insecure Application Programming Interfaces (APIs) can be other causes of security hazards. As technology attracts more users, security attack risk is increasing daily. So, there is a need to build an intelligent defender system to monitor the network traffic and detect malicious activities on the network.

This Chapter overviews the cloud computing paradigm and its security issues. To tackle these security issues, the game theoretic approach is discussed, which is used to predict the attacker's strategies and thus help the defender system to take suitable action. It further uses modern techniques like Machine Learning (ML), Deep Learning (DL) to improve the accuracy of detecting security attacks. The Chapter concludes with the contribution of this research work and its organization.

1.1 Cloud Computing

Cloud computing is internet-based utility computing in which resources, software, and data are pooled and shared among many users [1]. IT services are delivered through the Internet and are paid for as they are used; this is known as a pay-per-use model. Due to the uplifting of technologies, cloud computing has become essential in our daily lives, like sports, medical sciences, education, business, agriculture, etc. Cloud computing's advantages can be leveraged in almost every field. Cloud computing provides many services like storage, processors, development platforms, *etc* [2]. Researchers define the term cloud computing in different ways as follows.

According to NIST (National Institute of Standards and Technology), cloud

computing is defined as *"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"* [3].

Buyya et al. state that *"A Cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers"* [4].

The history of cloud computing starts with Grid computing which was introduced in the mid-90s. Grid computing aims to solve significant problems with parallel computing [5]. A virtual pool of resources is provided by grid computing. From grid computing, the term utility computing was coined in the late 90s, offering computing services as a metered service. After that, Cloud raised the level of virtualization, and this model charged money by the value of the application to subscribers rather than the resources exhausted. In 2010, companies including Amazon Web Services, Microsoft, and OpenStack developed reasonably functional clouds. Concurrently, Apple introduced iCloud, which was designed to store personal data such as photos, videos, and music. By 2014, cloud services had grown in popularity and sophistication [6]. So, cloud computing is developed with the combined ideas of grid computing and utility computing.

Cloud services have altered the global landscape, and their significance cannot be understated. Personal computing has grown through three distinct phases since its inception. The first phase stored the data and program on the local PC. The application residing on a local server and desktop software, as well as the Internet's helpful information, can be termed the second phase. In the third phase, often known as cloud computing, most data and applications are hosted on virtual servers [7]. Cloud services can be leveraged by consumers through the Internet anywhere, and cloud providers charge the users based on the data storage size or the number of resources requested by the user. Some vendors charge the customers by considering the time the service is used. Cost, speed, global scalability, productivity, performance, dependability, and security are the primary drivers for businesses to use cloud computing over traditional techniques. The following Section explains that cloud computing has four deployment models to fulfill user requirements and three service models to provide services.

1.1.1 Conceptual Model of Cloud Computing

The conceptual reference model that includes different deployment models, service models, and characteristics of cloud computing is represented in Figure 1.1 and discussed in detail.

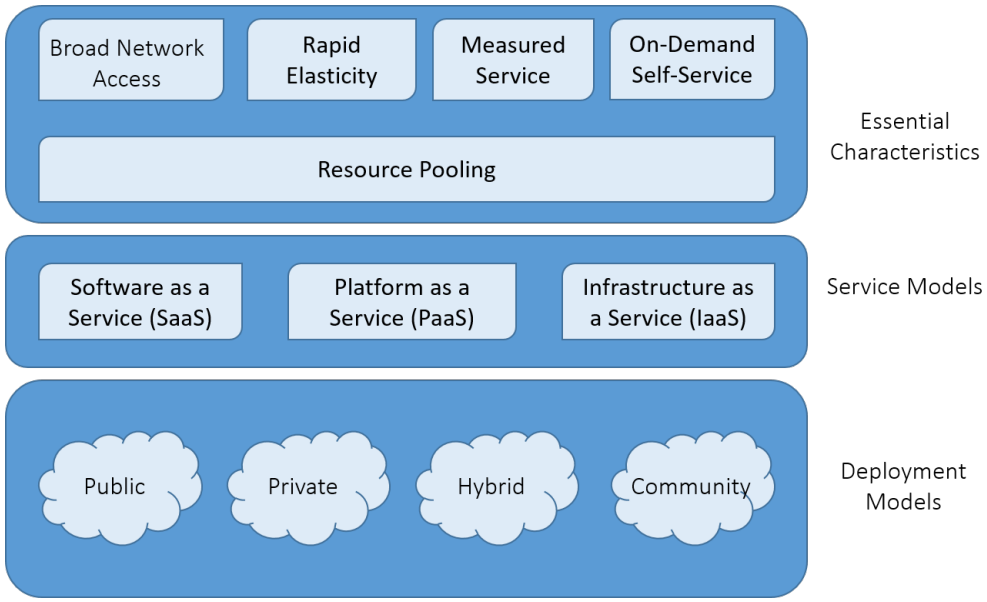


Figure 1.1: Conceptual reference model of cloud computing [8]

1.1.1.1 Deployment Models

Numerous services are provided to users through cloud resources and specific cloud deployment methods for delivering these services to users. The NIST has outlined four distinct deployment models: private cloud, public cloud, hybrid cloud, and community cloud [9]. Each cloud deployment model is based on an organization's diverse requirements and has distinct characteristics. An appropriate cloud deployment method is crucial for an organization to maximize its use of cloud resources. Figure 1.2 illustrates all four models' cloud deployment strategies. The four deployment models of the cloud are as follows:

Public Cloud: The cloud services are made available to public users. Anyone who has the necessary hardware and resources can operate the public cloud. The cloud service provider is responsible for efficiently operating the shared computing resources. Location independence, low price, and usability are common public cloud benefits. There are numerous public cloud service providers, with Amazon Web Services, IBM Cloud, Microsoft Azure, and Google Cloud Platforms being the most prominent.

Private Cloud: The private cloud is widely utilized by users. The private cloud

	Public Cloud	Private Cloud	Community Cloud	Hybrid Cloud
Setup and Use	Handle in house	Requires IT Professionals	Requires IT Professionals	Requires IT Professionals
Privacy and Security	Low	High	Medium	Varies from low to high
Control of Data	Low	High	Medium	Medium
Overall Reliability	Medium	High	Medium	Medium to high
Flexibility and Scalability	High	Medium	Medium	Very High
Cost	Lowest	Relatively High	Variable	Medium/ Variable
Hardware	Third-party	Variable	Variable	Medium/ Variable

Figure 1.2: Deployment models of the cloud computing [10]

resolves the primary problems associated with the public cloud. Private and public clouds are identical from a technical standpoint, except that private cloud services are not accessible to the general public. The most significant advantage of the private cloud over the public cloud is the ability to control the cloud's configuration and operation. In addition, a private cloud user can control data management and privacy and security policies. Hewlett Packard (HP) Enterprisers, VMware, Dell, Oracle, and Amazon Web Services are some of the largest providers of public cloud services.

Hybrid Cloud: A hybrid cloud comprises public and private clouds. Large corporations or organizations gain a great deal from the security features of the private cloud and the broad access to public cloud infrastructures. Hybrid clouds permit businesses to utilize both public and private clouds as needed. For instance, the company can use the public cloud for high-volume, low-security requirements and switch to a private cloud for sensitive data and information storage. Advantages of utilizing the hybrid cloud include data ownership and the ability to switch between public and private clouds.

Community Cloud: A public cloud is available to everyone, while a private cloud is reserved for a single organization. The community cloud, a semi-public cloud, is accessible to limited organizations. A community cloud is an option for organizations that find the public cloud risky and the private cloud costly. A community cloud is an arrangement between organizations with similar requirements.

The cost is shared between multiple organizations, making it less expensive than a single private cloud setup. Deployment models are designed and developed based on the client's need, and services will be given to them based on their needs, so the following Section discusses the various service models in detail.

1.1.1.2 Service Models

Cloud services are provided in terms of Software as a Service (SaaS), Infrastructure as a Service (IaaS), and Platform as a Service (PaaS) [11] to the customers or organizations. Organizations choose to use them quickly and affordably without bothering about external hardware and infrastructure [12]. The service providers are responsible for managing and securing the available services. To build these service models, the main layers of infrastructure components, such as networking, storage, servers, virtualization, and middle-ware application infrastructure components, such as the operating system and run-time, are used and represented in Figure 1.3.

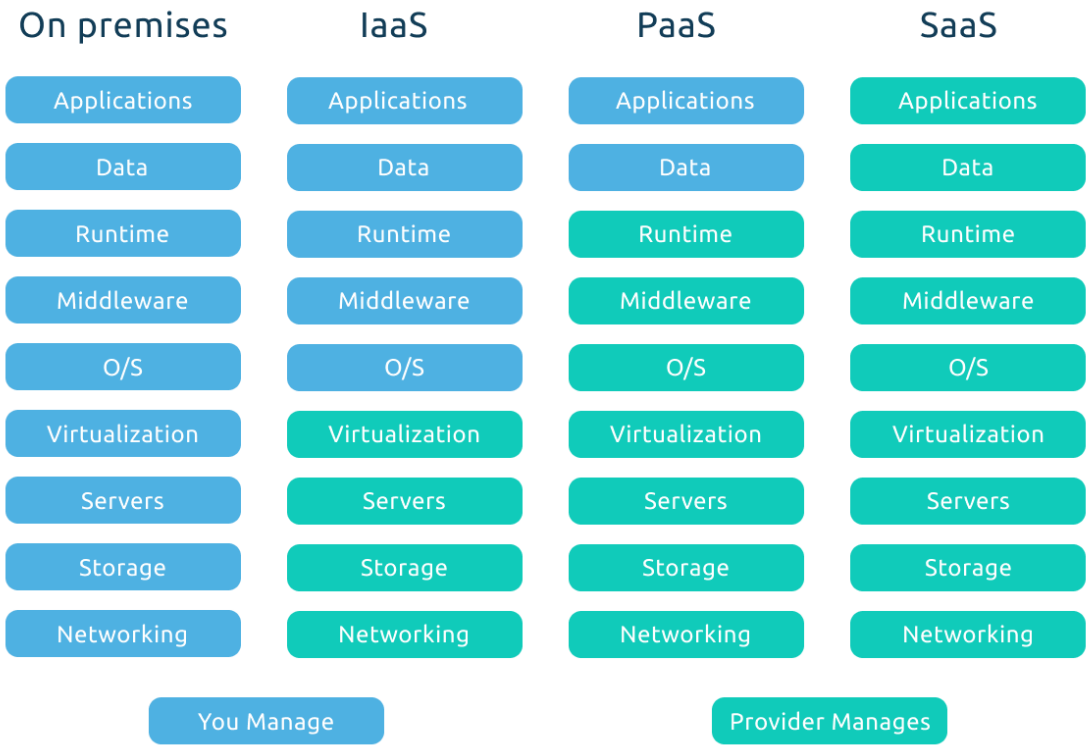


Figure 1.3: Service models of the cloud computing [13]

Software as a Service (SaaS): In the SaaS model, the cloud service provider delivers a fully functional and comprehensive software product to the user over the Internet. All software is installed on the service provider's server, where clients can access it after paying a subscription fee. Users can access this software from any

Internet-connected device anywhere in the world. Users need not worry about purchasing infrastructure, updating software or operating systems, or replacing outdated hardware or software. SaaS includes popular products such as Office365, Google Apps, SalesForce, Oracle, Microsoft, and Amazon Web Services.

Infrastructure as a Service (IaaS): It is a model in which the cloud service provider offers virtual machines and other resources to the user. The primary benefits of IaaS are that it is extremely flexible, scalable, and accessible to multiple users. Moreover, it is cost-effective. The user chooses an IaaS provider and immediately begins using the services. This pay-per-use service model is advantageous for businesses of all sizes, from the smallest to the largest. On IaaS platforms, many well-known SaaS services are hosted. Amazon EC2, IBM SoftLayer, and Google's Computing Engine (GCE) are the three most popular IaaS providers.

Platform as a Service (PaaS): It is a service model in which the cloud service provider leases software or applications to the user. The user is responsible for developing, deploying, and customizing the service, while PaaS provides access to the necessary resources. The user manages applications and data, while the service provider oversees other services. PaaS is less accountable than IaaS but more accountable than SaaS. This service is intended for developers, as it provides a platform for application development. It is ideal for small and medium-sized businesses. It is the most cost and time-efficient option for developers who wish to build applications. This service is provided by Google App Engine (GAE), DropBox, Microsoft's Azure, and Google Drive, among others. The service and deployment models of cloud computing offer many characteristics which help the user in various ways. These are discussed in the next subsection.

1.1.1.3 Characteristics of Cloud Computing

The various characteristics are discussed as follows [13]:

On-demand self-service: Based on the requirements, users can individually provision services, like database storage and bandwidth, without any middle expert between the cloud user and provider.

Broad network access: Cloud facilities are present on the network and can be obtained through cell phones and workstations.

Resource pooling: As the cloud supports multi-tenancy, the resources are accumulated so multiple users can access the services. The physical and virtual resources are accordingly assigned to the user's needs. The resources can be storage, processing, and bandwidth.

Rapid elasticity: Cloud resources are allocated and liberated according to the

user's need. The resources appear to be never-ending to the users and can be accessed at any number or anytime.

Measured service: Clouds can power and optimize the uses of resources by leveraging measured capability. To prevent any issues between the cloud provider and the consumer, the usage of resources can be checked and controlled.

Ready to use: The services offered by the cloud are always available if the user is connected to the Internet.

Ubiquitous access: Cloud services can be obtained anywhere in the world at anytime. So, there is no need to carry portable storage.

Improved collaboration: The dispersed people from different parts of the world can meet easily and share information in real time.

Environmental friendly: Cloud providers are using green computing strategies to reduce the bad environmental impact and make it more sustainable.

Pay-per-use method: The customers can measure the services they use and pay accordingly. The utilities provided by the cloud are usually at a meager cost.

Secure data: If the personal device on which the data is stored gets lost or corrupted, the data will also be lost. So storing data on the cloud makes the data secure.

Disaster recovery: The backups are stored in various data centers; thus, data can easily be recovered in any disaster.

The different characteristics of the cloud make it beneficial to the end users, and its services can be leveraged by many applications of the modern era, which are discussed in the next Section.

1.1.2 Applications of Cloud Computing in Current Era

These days cloud computing services are used by everyone as these services are easily manageable, scalable, and adaptable.

Processing Big Data: The end-users and technologies like IoT and VANETS generate large volumes of data which can be processed efficiently by cloud computing [14]. Also, the cloud can be used to perform big data analytics. Cloud computing is crucial in securing generated data, allowing devices to operate efficiently. Cloud computing application in processing big data has increased the productivity of routine tasks [15].

Game Streaming: The cloud enables users to play games stored on a powerful, Internet-connected remote server on a typical home computer without any technical fault. Cloud gaming's primary benefit to the gaming community is that it enables players to play high-end games on low-end consoles with excellent experi-

ence quality [16]. The failure of local devices results in the loss of game progress, which is not an issue with cloud gaming. Even if a device breaks, the user can continue progressing on another device. *NVIDIA GeForce Now*, *Google Stadia* are some of examples of cloud gaming services.

Data Storage: The term *"cloud storage"* immediately conjures up images of storing data (photos, videos, music, files) over the Internet, which can be accessed from any device. Consequently, the technical definition of cloud storage is a cloud computing model in which data is stored on a remote server that can be accessed whenever necessary via the Internet. A cloud storage service is designed to provide applications, services, and organizations with instant, flexible, and global access to offsite storage capacity [17]. Apple *(iCloud)*, Google *(Google Drive)*, Dropbox, Amazon *(Amazon web services)*, and Microsoft *(OneDrive)* are among the most popular cloud storage services providers currently available [18].

Social Media: Cloud computing and social media have a reliable relationship [19]. Social media websites have millions of users worldwide, and the traffic volume is always enormous. Social media services often experience fluctuating user activity, and cloud computing allows platforms to dynamically allocate resources based on demand, ensuring smooth user experiences during peak periods. By utilizing cloud servers efficiently, the traffic can be easily handled. In addition, services on the cloud can be hosted without purchasing and installing hardware and software, allowing businesses to save money.

Despite the discussed benefits and applications, many organizations and individuals fear losing their data on the cloud. The attacks on Apple and Sony cloud have also created fear in the mind of customers [20]. If cloud security is compromised, legitimate users' data will be in the wrong hands. The hacker can update or delete the data and add malicious data to the servers leading to mass damage to the cloud. So, security is the biggest concern in cloud computing [21]. Different security issues faced by cloud computing are discussed in the following Section.

1.2 Security Issues in Cloud Computing

As organizations increasingly rely on cloud-based services, the security of these services has become a critical concern. Security issues can result in sensitive data being accessed, modified, or deleted, which can cause significant harm to a cloud organization's reputation and financial stability. The security issues cloud computing faces can be broadly discussed as security threats and attacks. These are discussed as follows.

1.2.1 Security Threats

The prevalence of cloud computing has made it vulnerable. Many loose ends in security structures pose a huge threat to cloud servers. Cloud-based attacks account for 20% of all cyberattacks [22]. It is required to understand the threats to the cloud infrastructure and find a way to mitigate them with adequate security controls and practices like Intrusion Detection System (IDS). The Cloud Security Alliance (CSA), which works to improve cloud security best practices, recently surveyed industry professionals regarding the most pressing security threats. Significant risks include data breaches, unauthorized access to data due to insufficient identity and access management, insecure APIs, and service hijacking [23]. The major security threats are discussed as follows:

Poor Access Management: Poor access management poses a threat to cloud services. The failure to implement multifactor authentication for weak passwords contributes significantly to cloud attacks. Instead of attempting to breach complex and secure firewalls and IDS, modern attackers attempt to gain access by stealing employee credentials. The cyberattack on Deloitte cloud servers is a simple example of how ineffective access management can lead to a major security attack [24]. The hackers compromised the company's global email server by exploiting an *"administrator's account"* that required a simple password and gave them complete access.

Data Breach: Data breaches continue to be the greatest threat to cloud computing. A data breach is a security event in which data intended for internal cloud operations becomes accessible to unauthorized parties. Data breaches have numerous causes, including targeted attacks, individual negligence, poor security practices, and application vulnerabilities. In 2017, the most infamous recent data breach occurred at Equifax. It exposed the personal information of approximately 143 million users stored on the cloud. This is because Equifax did not update its software to address the identified flaw. This was exploited by hackers, resulting in the breach [25].

Insecure API: In an ideal world, APIs are designed to increase the efficiency of cloud computing processes. If left unprotected, APIs can provide communication channels that enable attackers to exploit private and sensitive data on cloud servers. There are statistics to support the reality of this threat. Cybercriminals are attracted to cloud APIs solely due to the growing reliance of businesses on APIs. In 2018, at least six high-profile data breaches were caused by insufficient API security. APIs are the most exploited vector in attacks against a company's data, according to Gartner [26].

Misconfigured Cloud Storage: Misconfiguration is also one of cloud computing's major threats. This typically occurs when cloud service users deviate from the default security and access management settings. When this occurs, sensitive information of cloud users can be made public, altered, or deleted. A misconfigured Amazon Simple Storage Service (Amazon S3) cloud storage bucket exposed the sensitive information of 123 million American households in 2017 [27].

These security threats reflect how vulnerable a threat can be to the cloud environment. Similarly, the cloud is susceptible to internal as well as external attacks. These are discussed in the next Section.

1.2.2 Security Attacks

With the existing threats in the cloud environment, it is simple for attackers to exploit them for various malicious activities, ranging from cyber attacks against large corporations to those against a single individual. Based on the various threats, attackers can perform security attacks on the cloud. Cloud security attacks refer to attempts by unauthorized individuals or organizations to gain access to cloud-based systems, applications, or data. These attacks can take various forms, including malware infections, phishing, denial-of-service (DoS) attacks, data breaches, and man-in-the-middle attacks. Cloud security attacks can target various aspects of cloud infrastructure, including the network, storage, and applications. These attacks can be categorized as either internal or external, depending on their origin [28].

Internal Attacks: Internal cloud attack typically refers to an employee compromising an organization's security. Typically, internal attacks are unintentional, but the organization cannot ignore the possibility that some attacks are deliberate. These attacks typically occur due to human error, which may result in data loss or leakage. Insider attacks occur for various reasons, including data theft, retaliation, and profiting from the sale of stolen data. If the attack occurs internally with the intent to cause harm, it can be difficult to detect even by IDS due to the nature of the attack [29]. Since the attacker is already inside the organization, they can conceal these attacks, and it is difficult to distinguish between malicious intent and simple error.

External Attacks: External attacks typically occur when an individual or group uses viruses or malware to gain unauthorized access to the cloud and steal sensitive data. These attacks are typically carried out by skilled hackers, making them more difficult to combat because no one knows where they are coming from or what will occur [30]. These attacks typically take the form of Spyware, Trojans,

and Rootkits. Because cloud environments are accessed via the Internet, they are especially susceptible to DoS and Distributed Denial of Service (DDoS) attacks, a typical external attack. Social engineering and phishing attacks are other types of external assault [31]. Once these login details or confidential information are exposed, the attacker can easily log into and exploit the system, as the cloud is accessible from anywhere. There is a need to address these challenges so people do not hesitate to access cloud services.

In the old era, to address security issues, basic practices are opting for solid passwords, multifactor authentication, and cloud cryptography [32]. Due to many security issues, the attackers find loopholes in cloud security and attack the cloud. In the modern era, it is essential to understand the behavior of the attacker and defender in terms of their interaction, how they compete, and their strategies for providing security at a more sophisticated level. Cloud security is the biggest challenge affecting the cloud environment's confidentiality, integrity, and availability. The attacker can be a person or a machine that sends malicious packets to compromise the cloud services. Cloud providers use defender systems like IDS to tackle these kinds of attacks. Game Theory provides plausible practice based on mathematical modeling that can be used to handle such types of problems. The mathematical model can be delineated with pre-defined parameters and does not require any datasets [33]. The decisions made by the game theory can assist the existing defender systems like Intrusion Detection Systems (IDS) in making better decisions. Machine Learning and Deep Learning can further optimize the working of IDS. The game theory in cloud security and optimization of IDS is discussed in detail in forthcoming Sections.

1.3 Game Theory in Cloud Security

The game theory examines competition and cooperation between players based on rational decision-making. Regardless of the game's circumstances and nature, a strategy exists for the players to win. These strategies are sometimes dependent on the opponent's choice. By applying the concept of game theory, rational decision-making can be maximized. Game theory can be used to explain the analysis of each player's strategy and the determination of the optimal strategy for each player to win the game when multiple players are involved [34]. The expertise of the game theory can be leveraged to improve cloud security.

In cloud security, game theory examines the nature of a cyber incident, where network defenders, attackers, and users interact to achieve an outcome. It de-

scribes each player's behavior and strategies and captures opponents' interaction. A single defensive error by the defender can be fatal. The cloud is a vast network, so defense mechanisms are expensive and must be 100 percent effective. On the other hand, attackers are continually modifying and establishing the rules. They have endless opportunities to try without consuming many resources, but only one successful attack can fail the cloud services. They can impersonate innocent users to evade detection. There is a significant need to develop a defender system capable of tackling every security attack on the cloud. Hence, game theory is employed.

Game theory in cloud security is broadly used for cloud-attack-defense analysis and cloud security assessment [35]. Cloud-attack-defense analysis can predict attackers' actions by modeling defense behaviors as games. In addition, it investigates the potential states of attack-defense equilibrium. Ideal counter-defense strategies can be determined based on the equilibrium state. The equilibrium state of cloud-attack defense can be analyzed, and the prognosis of attack and defense strategies can justify and evaluate cloud security. Due to the quantitative aspects of game analysis, security and reliability are viewed as a quantitative evaluation that provides a cloud security and reliability calculation.

To analyze the security in the cloud, the game theory uses its fundamental components, which include the players, the set of strategies available to each player, and the payoffs or outcomes associated with each combination of strategies. By analyzing the actions and behaviors of the players, game theory seeks to identify equilibrium points or states in which no player can improve their payoff by unilaterally changing their strategy. These components are discussed as follows.

1.3.1 Components of Game Theory

The game theory consists of players, strategies, actions, and payoff functions [36].
Players: The individuals or entities making their own decisions regarding the game are called players. Each individual or entity has unique objectives and preferences. Entities may be individuals, organizations, or institutions; players' primary objective is to select the action that maximizes their utility. The attacker and the defender are the two players competing with each other in cloud security.
Actions: Each player's decisions are actions. Each move of the player requires a decision. Each player may or may not be aware of the actions of all other players. The attacker's action is to exploit the cloud resources, and the defender will try to guard the resources.
Payoffs: The amount of benefit each player receives from an event is the player's

payoff. Depending on the decisions made, the player may receive a positive or negative payoff. Each player's payout is known after the completion of the game. With a successful attack on the cloud, the attacker will gain the payoff, and the defender will lose it. The defender will gain a payoff with the correct detection of an attack.

Strategies: Each player's set of actions to win the game will be defined concerning the opponent's past and anticipated actions. This collection of actions is referred to as Strategies. The attacker's strategy is to perform different security attacks, and the defender's strategy is to defend the attack with its various modules.

The different components of game theory are used to build different game models. Depending on the situation, these game models can be implemented to provide insights into the behavior and decision-making of the attacker and defender in a real-world scenario. The different games are discussed as follows.

1.3.2 Types of Games

The game theory recognizes several types of games, each with its own set of rules, strategies, and payoffs. The different types of games are discussed below [37].

Cooperative Games: The games in which all players must exhibit cooperative behavior. These games are between coalitions of players, not between two individuals. Since the attacker and defender will never cooperate. So, the cooperative game model can not be used in cloud security.

Non-cooperative Games: The games in which players act independently and compete with other players. All players' primary objective is to maximize their payouts. This model perfectly fits with cloud security as the attacker and defender are competing with each other. This model is used in this Thesis to design and develop models discussed in Chapters 4, 5, 6 and 7.

Static Games: All players involved in a game make a single decision at the beginning of the game. No player has any knowledge of the other player's behavior. As the attacker keeps changing his strategy, the static games do not fit in cloud security.

Dynamic Games: Each player in a dynamic game will have some knowledge of the behavior of the other players, and the game will be conducted in multiple stages. Players will make their decisions based on the opponent's behavior. The dynamic game can be used to model a scenario between the attacker and the defender as both keep changing their strategies based on previous outcomes.

Complete Information Games: All participants in the game will have complete knowledge of the opponent's behavior. Each player is aware of the strategies

employed by all other competitors. The defender does not know the attacker's behavior, so this model does not work for cloud security.

Incomplete Information Games: None of the game players know their opponents. The player cannot devise an ideal strategy to win the game. This model can depict the security incident as the attacker and the defender having little or no knowledge about each other.

Perfect Information Games: A game in which players know their opponent's previous moves before making a move. The defender system logs the network traffic to know the attacker's previous moves. But the attacker may or may not know the defense system's strategies.

Imperfect Information Games: Games in which at least one player is unaware of their opponent's previous moves. It will be tough for players to move if they are unaware of their opponent's behavior. Cloud Security falls under this game category as the attacker can perform new attacks that are not available in the database of the IDS.

Stackelberg Games: In these games, one player, the attacker, makes their decision first, and the defender decides based on the attacker's decision. This type of game is often used to model scenarios where one player has a significant advantage over the other player.

Differential Games: These games model dynamic scenarios where two or more players compete over time. Unlike traditional games, where a single decision or action determines the outcome, the outcome in differential games is determined by the player's actions over an extended period. As the attacker's actions are dynamic, this game can be used to analyze cloud security.

Stochastic Games: These games model scenarios where the outcomes are uncertain and depend on chance events. Random events influence the outcome in stochastic games. The game is played over a finite or infinite horizon. The attacker can perform any random attack infinite times; stochastic games can be used to model cloud security.

The security industry has adopted a non-cooperative, dynamic game model from the different game types in game theory. The attacker strategies in security attacks keep on changing and are not static. The dynamic model analysis is crucial to achieving optimal effect because dynamic models represent real-time cloud security issues. And for cloud-defense analysis, an incomplete information game model is employed [38]. The game theory senses the attacker's strategies and helps the IDS to make correct decisions by starting the accurate detection module. The working of the IDS with its various modules is discussed in the next Section.

1.4 Intrusion Detection System (IDS) to handle security attacks in cloud environment

IDS is much more popular and is based on autonomic computing. It tends to detect intrusions automatically and then alerts the admin. It can be installed on the switch, server, and virtual machine (VM) to detect attacks. It is a tool that monitors the traffic for malicious activities and policy violations and produces log files for the maintenance team. Detection methodologies for the IDS can be categorized as Signature-based Detection (SD), Anomaly-based Detection (AD), and Stateful Protocol Analysis (SPA) [39]. SD matches the signatures or patterns to the stored database containing the known attacks' knowledge base. It is also called knowledge-based detection. It is the simplest way of finding known attacks. The limitations include high time consumption to update the knowledge base and ineffective to detect unknown attacks. AD detects intrusions by detecting the anomaly behavior, i.e., when there is unknown, unexpected, or deviated behavior in the network. It is also termed behavior-based detection. The pros of AD include effective detection of new attacks and less dependent on the OS. The cons of AD are unavailability during rebuilding behavior profiles and weak profile accuracy. SPA traces the protocol states and depends on the provider-developed generic profiles to specific protocols. Another name is specification-based detection, also called SPA. The SPA includes the benefits of tracing the protocol states, and the main limitation is resource consumption. The common challenge of the IDS is sometimes it cannot detect the intrusions accurately [40]. The degree of accuracy can be checked by the False Positive Rate (FPR) and Detection Rate (DR). FPR alarm triggers if the IDS wrongly discovers the intrusion and DR is the intrusion successfully detected by the IDS.

The game theory helps the IDS to start the most accurate module in case of a security attack. To further improve the functioning of the IDS, the emerging techniques of Machine Learning and Deep Learning are used. The following Section discusses these two techniques.

1.4.1 Optimization of IDS with Machine Learning

Machine learning (ML) enables software applications to predict outcomes with greater precision without being explicitly programmed to do so [41]. The input for ML algorithms is historical data used to predict future output values. Many of today's leading companies, including Facebook, Google, and Uber, utilize ML

extensively. Classical ML is frequently categorized according to how an algorithm learns to make more precise predictions. There are four fundamental learning methodologies: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning [42], discussed in Chapter 3. ML in security defense is essential because it gives IDS a view of attackers' trends and patterns. Data scientists build various algorithms and models based on the type of information to improve the accuracy of the IDS. To validate the proposed model, it is tested on a real-time dataset, and the prediction accuracy is measured. Various techniques like bagging and boosting are there to increase prediction accuracy, but stacking is the most promising [43].

The stacking method can ensemble many models for classification or regression. Stacking is used to investigate a variety of potential solutions to a problem. It takes on a learning challenge by employing various models, each of which can master a specific subset of the problem's domain but not the whole thing [44]. Since there are several ways to construct a learner, a more nuanced prediction can be made in the intermediate stage. After that, a second model is brought in to learn from the first model's intermediate forecasts of the same objective. Figure 1.4 depicts the architecture of stacking.

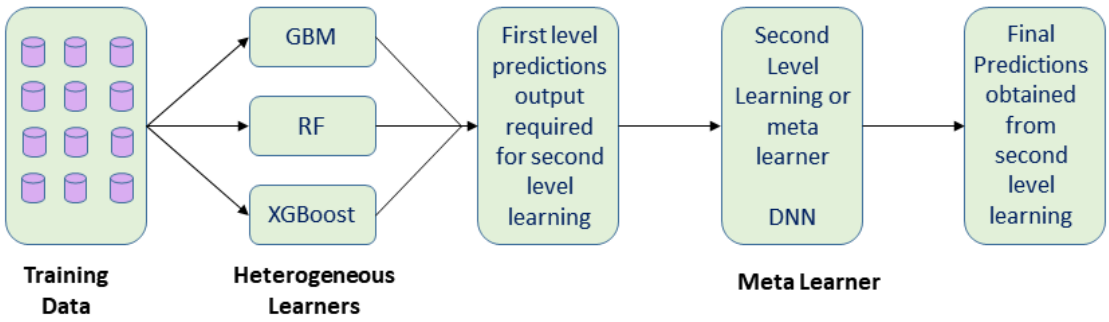


Figure 1.4: Working of Stacking in Machine Learning

In Chapter 7, stacking is used, and the overall performance of IDS is boosted, and in many cases, the final model outperforms all of the intermediate models combined. Assume M heterogeneous weak learners and one-second-level learners, *i.e.* meta-learners. On the training set, $k-$fold cross-validation is performed using M learners, and predictions are made as $p_1, p_2, ...,$ and p_m at the first level. The predictions from the first level are treated as features, i.e., N features are created. Then a new dataset $[]_{N*M}$ and $[]_y$ is created for second-level training using these N features and the y target variable, where $N * M$ is a matrix of features. The new dataset is trained with a meta-learner, and predictions are made at the second level of training. The three heterogeneous base learners are used to make first-level

predictions comprising Random Forest (RF), Gradient Boost Machine (GBM), and XGBoost. The description and working of these three models are given below:

Random Forest (RF): It is a predictor that combines multiple decision trees on distinct subsets of data and averages the results to improve the predictive accuracy of IDS on the dataset. Rather than relying on a single decision tree, the RF collects the results from each tree and predicts the output value using the majority voting rule [45]. It can solve both classification and regression problems. The structure of RF is represented by Figure 1.5. In classification, the result with the highest vote count is selected. The mean of each tree's predictions in regression is calculated and used as the final resul**Gradient Boosting Machine (GBM):**

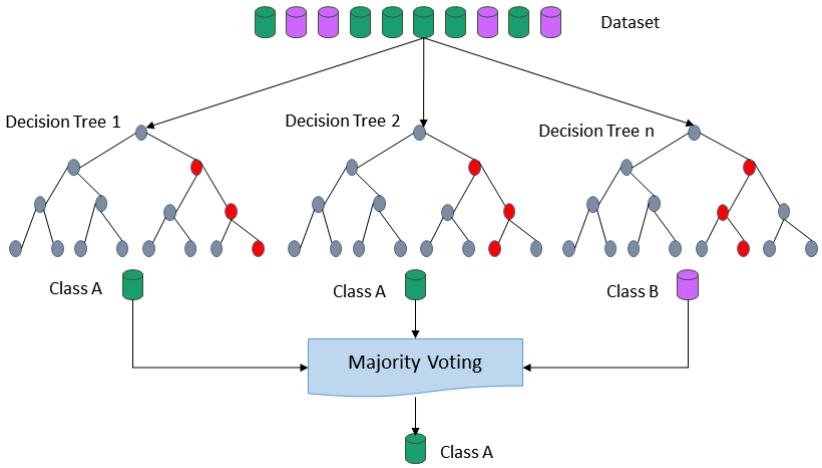


Figure 1.5: Structure of Random Forest Model

It operates by combining multiple weak learners to create a single strong learner. The weak learners correspond to a series of connected decision trees, with each tree attempting to reduce the errors of the previous tree [46]. The goal of GBM is the minimization of the loss function. Both the loss function and the base-learner models can be arbitrarily specified. Given a specific loss function $\psi(y, f)$ and a custom base-learner $h(x, \theta)$, it cannot be easy to obtain the solution to the parameter estimates in practice. To address this issue, it is suggested to select a new function $h(x, \theta_t)$ that is most parallel to the negative gradient $g_t(x_i)_{i=1}^{N} = 1$ along the observed data as given in Eq. 1.1.

$$g_t(x) = E_\gamma\big[\frac{\delta\chi(y, f(x))}{\delta f(x)}|x\big]_{f(x)=\hat{f}^{t-1}(x)} \tag{1.1}$$

Instead of searching for a general solution for the boost increment in the function space, the new function increment that correlates most strongly with $-g_t(x)$ can be selected. This enables substituting a potentially difficult optimization task

with the traditional least-squares minimization task. The structure of GBM is explained in Figure 1.6.

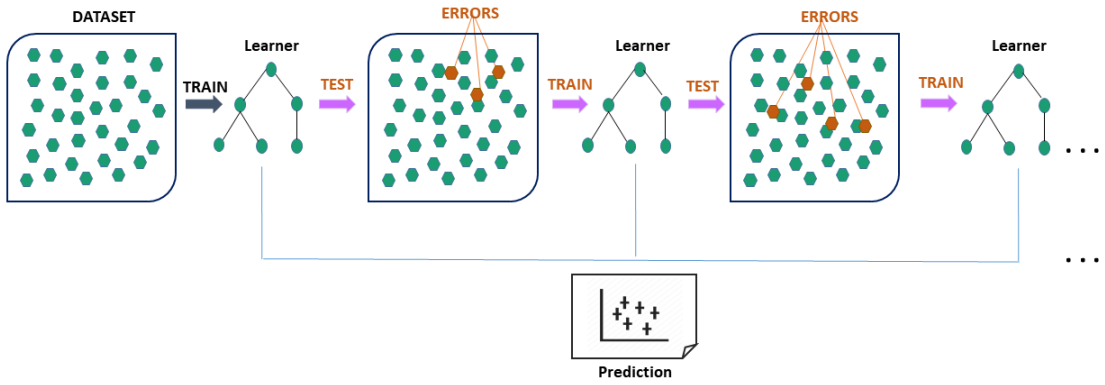


Figure 1.6: Structure of Gradient Boosting Machine Model

XGBoost: Extreme gradient boosting is an advanced implementation of the gradient-boosting algorithm. A new model that predicts the errors or residuals from the previous models is added and combined with the previous model. This new model is retrained to remove errors from previous models, followed by additional retraining until all errors are eliminated. The stochastic gradient boosting reduces error by subsampling each column and row. Regularized gradient boosting minimizes overfitting with *L1* (Lasso regression) and *L2* (ridge regression) [47]. This improves performance and ensures that the IDS produces accurate results. It is a highly efficient, dependable, and portable ML model that recursively reduces error until accurate and efficient results are obtained. XGBoost is faster than gradient boosting, stochastic gradient boosting, and regularised gradient boosting due to its support for parallel processing. XGBoost requires 0.04 s to process 100 nodes and 0.16 s to process 1,000 nodes. It takes 0.24 seconds to traverse 10,000 nodes, demonstrating a linear increase in time. Therefore, it has a time complexity of $O(logn)$.

Different machine learning models used in this work are discussed that can be used to enhance the working of the IDS. To further increase the prediction accuracy of the IDS, a deep learning approach is used, which is discussed in the next Section.

1.4.2 Optimization of IDS with Deep Learning

Besides ML, Deep learning (DL) is also a promising approach. DL is a sub-field of ML and AI that mimics how humans acquire specific types of knowledge. It is a crucial component of data science, encompassing statistics and predictive model-

ing. It is highly advantageous for data scientists tasked with collecting, analyzing, and interpreting large amounts of data; it makes these processes more efficient and streamlined [48]. Each algorithm in the hierarchy applies a nonlinear transformation to its input and uses what it has learned to construct a statistical model as its output. Iterations are repeated until the output achieves an acceptable level of precision. The term *"deep"* was inspired by the number of processing layers data must pass through. Most DL models rely on an advanced ML algorithm, an artificial neural network. Consequently, DL is also known as deep neural learning and deep neural networking [49]. There are various neural network types, such as recurrent neural networks, convolutional neural networks, artificial neural networks, and feed-forward neural networks, and each has advantages for particular applications [50]. DL is currently implemented in most image recognition tools, natural language processing, speech recognition applications, and security techniques like IDS. In Chapter 5, 6 and 7, Deep Neural Network is used, which is discussed as follows.

Deep Neural Network (DNN): DNNs are feed-forward ANNs with multiple hidden layers of neurons used for classification tasks of varying complexity. For DNN to perform well with the text, voice, sounds, and other functions, innovative thought is required. A DNN system employs multiple layers of nodes to extract high-level functions from incoming data [51]. It involves transforming the facts into a more creative and abstract component. DNN consists of more than one hidden layer (h). Each input, hidden, and output layer consists of neuron-like nodes. The i^{th} layer's output is the input for the j^{th} layer. Applying functional transformations and activation functions with weights ($w_{i,j}$) and bias (b) values yields the final output y. Figure 1.7 displays the structure of DNN. The various activation

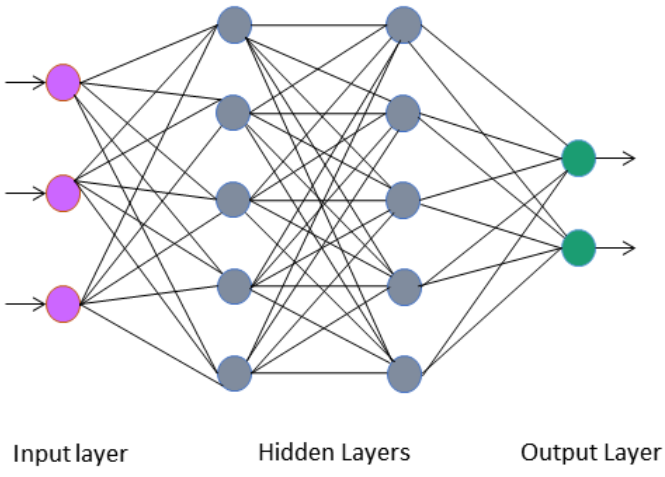


Figure 1.7: Structure of Deep Neural Network

functions, such as *tanh*, *Rectified Linear Unit (ReLU)*, and *sigmoid*, are used to perform computations. The current study uses a ReLU activation function with five hidden layers. This procedure is known as a forward pass. During the training of the IDS with DNN, the output values are compared to the predicted value to determine the total error, also known as the total cost, of the neural network. The error is then propagated backward over the whole link structure of the network while considering each connection's relative weights. It can determine the number of errors associated with that weight's contribution to the overall cost. And thus, it calculates the modification required to make to that weight to get the output of our DNN closer to the predicted output. Each model has many hyperparameters that can be tuned for optimizing the IDS [52]. In contrast to model parameters, which are learned during training, model hyperparameters are predetermined by the data scientist. Tuning various hyperparameter values is referred to as hyperparameter tuning. In Chapter 5, hyperparameter tuning is done, and different techniques used for the hyperparameter technique are discussed as follows:

Grid Search: This is the most fundamental method for tuning hyperparameters. A grid of hyperparameter values is defined. The tuning algorithm sequentially performs an exhaustive search of this space and trains a model for each possible combination of hyperparameter values [53]. The grid search algorithm trains multiple models (one for each combination) and then retains the optimal hyperparameter value combination. Because the number of models to train grows exponentially as the number of hyperparameters increases, grid search is not used in practice for IDS as it can be very inefficient regarding time and computation.

Random Search: Random search differs from grid search in that values are sampled from a statistical distribution for each hyperparameter [54]. A sampling distribution for each hyperparameter to conduct a random search is defined. Random search can control or limit the number of utilized hyperparameter combinations. In contrast to grid search, where every possible combination is evaluated, random search allows us to train a fixed number of models and then terminate the tuning algorithm. It is also possible to set the number of search iterations based on time or resources. Since random search tests hyperparameter sets at random, it risks missing the optimal set of hyperparameters and missing out on peak model performance.

Bayesian Optimisation: Bayesian optimization is a Sequential Model-Based Optimization (SMBO) algorithm that uses the previous iteration's results to determine the next set of candidate hyperparameter values. This method uses intelligence to select the next set of hyperparameters to improve the model's per-

formance instead of mindlessly searching the hyperparameter space (as in grid and random search) [55]. This process is repeated iteratively until it converges on an optimal solution. Bayesian optimization generates a probabilistic model by associating hyperparameters with the probability that the objective function will be met. The previous iterations' results help the IDS predict the attacks better.

DL techniques can learn complex patterns in network data, resulting in highly accurate and efficient IDS. They also have the added benefit of adapting to new and evolving types of attacks. Overall, using ML and DL techniques in IDS is a promising area of research, with the potential for significant improvements in detection rates and decreased false positives. As the sophistication and complexity of security attacks continue to increase, the importance of IDS and the application of advanced ML and DL techniques continue to grow.

This Chapter discusses the concept of cloud computing with its various security issues. The tools and techniques used to detect security attacks are elaborated. The role of game theory in cloud security, including its components and various game models, is presented. At last, ML and DL models are discussed, which further improve the detection accuracy of the IDS. The contribution and organization of the Thesis are discussed in forthcoming Sections.

1.5 Thesis Contribution

The major contributions of the Thesis are:

- Analysis and classification of related research work:
  - The design and issues related to the security of the existing approaches in the cloud environment are presented.
  - An overview of security techniques related to Game Theory, ML, and DL is provided.
- Four Game-Theoretic Models are developed, which assist the IDS in detecting the different types of attacks in different scenarios more efficiently. These models are:
  - **GTM-CSec:** This model is based on the two-player Non-Cooperative Game. It could be implemented where IDS and honeypots cooperate as defenders to tackle external security attacks. Different security attacks are detected by starting the defender's most accurate module (signature-based, anomaly-based, and honeypot).

- **BOGTA:** In the model, the graphical method for the Game Theory is used to select the most appropriate module of the IDS or honeypot. ML algorithms are also used to improve the detection rate and lower the false positive rate of the IDS.
- **GTA-IDS:** This model includes information theory and Game Theory and is best used to detect internal security attacks. Information entropy of the incoming traffic is measured, which leads the Game Model to make more accurate decisions.
- **NCGTM:** In this model, different IDS modules, namely signature-based, anomaly-based, and hybrid-based, are considered to defend against regular and sophisticated attacks. This model can best be used where a single IDS protects the network.

- The results of designed and developed game models show the improvement in the existing defender system's detection rate, accuracy, and payoffs. Also, the False Positive Rate (FPR) is reduced with the proposed models.

1.6 Thesis Organisation

The Thesis is organized into the following eight Chapters:

Chapter 1: Introduction

This Chapter discusses the basics of Cloud Computing, its applications, characteristics, and critical challenges for the Cloud. It focuses on various security issues and how to tackle security attacks. Different tools and techniques of security in the cloud environment are discussed. Game Theory and ML techniques are discussed in detail.

Chapter 2: Literature Survey

A comprehensive survey of research carried out in cloud computing security is discussed in the Chapter. Different game-theoretic models are highlighted that are used to handle security in the cloud environment. Different ML and DL methods to increase the overall efficiency of the IDS are also elaborated.

Chapter 3: Research Methodology

The Chapter discusses various software and hardware requirements for security in the Cloud environment. Also, various techniques used in research methodology, including Game Theory and ML, are explained.

Chapter 4: GTM-CSec: Game Theoretic Model for External Cloud Attacks

To make the defense mechanism more efficient, a model is proposed for intelligent systems monitoring network traffic regularly. The proposed model, namely GTM-CSec, uses game theory to monitor the network traffic intelligently and adjust the configuration of the IDS along with the honeypot for better results. The two players, Attacker and Defender, compete against each other to gain maximum Payoff. The game theory model is updated with probability values for the defender's and attacker's strategies to predict the best strategy. Ultimately, the best strategy is delineated with the help of the Nash Equilibrium.

Chapter 5: BOGTA Model: Bayesian Optimized Game-Theoretic Approach

In this Chapter, Game Theory and Bayesian Optimised Deep Neural Network (DNN) Model for IDS are used to eradicate the limitations of the GTM-CSec. The GTM-CSec model breaks the $2X3$ matrix into different parts and solves them in seven cycles. The BOGTA covers every occurrence in a single cycle with the graphical method of Game Theory that reduces the time complexity of the model. The delineated equations are solved by the saddle point method. Further, these models are trained and tested on three datasets.

Chapter 6: GTA-IDS: Game Theoretic Approach to Enhance IDS detection for internal cloud attacks

To make the defender more efficient and accurate, a game-theoretic model, GTA-IDS, is devised based on non-zero-sum non-cooperative game theory. The model first filters the incoming traffic with the information entropy. The filtered traffic is fed to the game model, and a mixed strategy Bayes Nash Equilibrium (BNE) is reached. The probability by which an attacker should attack and a Defender should defend is calculated. Further, the model is simulated on the NSL-KDD dataset.

Chapter 7: NCGTM: A Non-Cooperative Game-Theoretic Model to Assist Hybrid IDS in Cloud Environment

The NCGTM Model is based on non-cooperative game theory. The two players, attacker and defender, compete against each other to gain maximum Payoff. The payoffs of the two players are calculated. The best strategy for the malicious node and defender is derived and analyzed. Further, ML approaches are used to increase the efficiency of the IDS. The model is implemented on an ISOT-CID dataset to test and validate its real-time work.

Chapter 8: Conclusion and Future Scope

This Chapter concludes the Thesis outcome and provides insight into the future scope of the research work for security in the Cloud environment. The future

scope of the presented game-theoretic and machine-learning approaches involves choosing more parameters and targeting a particular attack rather than a general model.

Chapter 2
Literature Survey

This Chapter reviews the literature on security models in the cloud environment. The security models for handling internal and external attacks are discussed and analyzed. Further, the classification of the game theoretic models for security in cloud computing is done. It includes Stochastic, Differential, Non-Cooperative, Stackelberg, and other game models(developed by the researchers). The analysis of each game is done and represented in tabular form. The contribution of Machine Learning (ML) and Deep Learning (DL) approaches used to optimize Intrusion Detection System (IDS) is also discussed. From the literature survey, the future directions of the research from the existing work and the gaps are identified. Based on the present gaps, research objectives are delineated.

2.1 Security Models in Cloud Computing

This Section is divided into two parts: security models to tackle external attacks and security models to handle internal attacks. Both models are discussed in detail in the forthcoming Sections.

2.1.1 Security models to handle external attacks

Tsu-Yang Wu *et al.* [56] developed a lightweight authentication system to use in cloud computing scenarios enabled by the Internet of Things. A formal security study uses a real or a random model with an automatic verification program. The experimental results concluded that after examining its security and performance, the developed framework is reasonably secure and has satisfactory performance.

M. M. Kamruzzaman *et al.* [57] developed a 6G Wireless Communication-assisted Security Management System (WC-SM-AI) to improve security by using energy-efficient 6G real-time communication framework and the upgraded Deep Neural Network (DNN) security module. Network longevity and spectrum efficiency are improved by energy-efficient 6G real-time communication while lowering energy usage and latency. On the other side, DNN provided a more secure network connection by boosting data integrity, privacy, and access. The experi-

mental findings indicate that the security of 6G networks is more efficient than the existing solution.

Munwar Ali *et al.* [58] suggested a practical approach to secure data by introducing a new service model called Confidentiality-based Classification-as-a-Service (C2aaS), which processes data in advance of cloud storage by handling data dynamically by the data security level. The proposed service model outperforms traditional approaches regarding cloud system efficiency and the safety of sensitive data.

Qian He *et al.* [59] conducted a study of the fundamental ideas of cloud computing and the applications of cloud computing regarding the significance of safety concerns. By utilizing data mining and the decision tree method, the proposed algorithm can increase the level of security in the cloud computing platform. The suggested technique is more straightforward to execute in practice because of its low computational overhead and independence from the number of clients.

P. Abirami *et al.* [60]developed a crypto-deep neural network to improve a distributed secure outsourcing mechanism. The suggested architecture includes a cloud server, a web server, a data center, and an agent, which protects cloud-based services from impersonation attacks using crypto-deep neural networks (CDNNCS). Compared to a secure linear algebraic equation system, the proposed framework boosted cloud customers' confidence in the cloud. Delay, Jitter, Throughput, and Goodput metrics are used to describe the performance, and it is found that CDNNCS improved packet loss by 10% compared to the status quo and enhanced response time by 5%.

Further, a revolutionary Customized Network Security (CNS) for cloud service is presented by Jin He *et al.* [61] that not only protects cloud services from external and internal assaults but also provides cloud users with individualized protection for their networks. CNS is created by changing the Xen hypervisor, and its viability as a practical answer to widespread cloud computing promotion is demonstrated through a series of experiments. The experimental findings indicate the efficiency of the proposed work.

Ding Li *et al.* [62] proposed a heterogeneous integrated network resource management algorithm based on information security transmission to solve the problems of the Internet of Things (IoT), which has many errors and is not very secure. The resource management algorithms are improved, and a model is created for resource management algorithms based on information security transmission. The effectiveness of the resource management algorithm is determined through experimentation, demonstration, and analysis. It is evident from the results that

resource management errors are reduced, and security performance is enhanced using the proposed work.

A Security-Aware Mechanism and Ontology-based Data Access Control (SA-ODAC) is designed by Gangasandra M. Kiran *et al.* [63] to address the lack of robust security and access control in cloud computing. The operational methods comprising the Secure Awareness Technique (SAT) for encryption, splitting, and decryption and Ontology-based Data Access Control (ODAC) for controlling unauthorized access are used. The suggested architecture included a secret sharing approach for handling the SAT method's key. The algorithm is implemented in MATLAB, and its performance is checked in terms of delay, encryption time, encryption time, and ontology processing time, as well as compared to Role-based Access Control (RBAC), context-aware RBAC, context-aware task RBAC, security analysis of advanced encryption standard, and data encryption standard. The results show that the SA-proposed ODAC's data access control and security strategy is superior to most proposed work.

Rajendra Patil *et al.* [64] presented a Hypervisor-level Distributed Network Security (HLDNS) framework that is installed on each processing server in cloud computing. At each server, intrusions are checked by watching the network traffic from and to the virtual, internal, and external networks related to the Virtual Machines (VMs) running on that server. The new fitness functions are added to a Binary Bat Algorithm (BBA) to get the feasible features from network traffic in the cloud. The features found are then used with the Random Forest classifier to find intrusions in cloud network traffic, and intrusion alerts are made. The proposed security framework is tested on the cloud network testbed at the National Institutes of Technology (NIT), Goa, and with recent UNSW-NB15 and CICIDS-2017 intrusion datasets. The experiment is performed, and it is evident from the results the proposed work performed well with the best performance.

Tian Wang *et al.* [65] devised a buffer queue in the fog computing layer that would send the results back to the cloud without going through intermediate stages to maximize the utilization of resources. Then, the first round of allocations is achieved by extending the Kuhn-Munkres algorithm. Finally, the additional scheduling of the initially assigned resources is assessed. The results showed that the solution is superior to conventional scheduling techniques, cutting down scheduling rounds and computing expenses by 24.04% and 57.78% and 9.88 to 31.51%, respectively.

Christos Stergiou *et al.* [66] provides an overview of the IoT and Cloud Computing, focusing on safety concerns. In this study, the technologies mentioned ear-

lier (Cloud Computing and IoT) are integrated to understand better their shared characteristics and how they might improve one another. Further, how Cloud computing has advanced IoT infrastructure and how the IoT can benefit from Cloud Computing's enhanced functionality is demonstrated. The results discuss the potential risks of merging the IoT with Cloud Computing.

K. Loheswaran *et al.* [67] presented a renaissance system paradigm for the storing of confidential data in the cloud by including four entities comprising data owners, the cloud, a cloud service provider, and an independent Third Party Auditor (TPA). The proposed model is a semi-trusted proxy agent performed in place of the data owner to reinstall the data blocks collected throughout the data restoration process. The experimental results appeared reliable and can be relied upon, and it is also impartial for cloud servers and data owners.

Quality of Service (QoS) driven techniques for cloud environments are defined by Bruno Guazzelli Batista *et al.* [68] based on the findings of a performance evaluation of a service using various security mechanisms. The findings suggest that service performance can be maintained despite the added load from security systems by adjusting the virtualized computational resources in a cloud setting. It is also found that the response variables are directly affected by this shift in the available resources.

Feda Alshahwan *et al.* [69] investigated the safety features of a system for delivering advanced mobile web services by describing the various components related to security and offering a security framework to guarantee authentication and privacy between clients and mobile hosts. The experiment is performed, and the results evidenced that the proposed work is performed efficiently with good results compared to existing cutting-edge technologies.

A security model is created by Jinan Shen *et al.* [70], which is divided into domains, and distinct security rules are applied independently to three domains comprising the domain of data storage, the domain of data processing, and the domain of data transmission. In addition, the security policies of upper-level apps are modified according to the specific security requirements of those applications. The results of performed experiments indicate that the suggested security model is workable and lightweight because it can provide differentiated security protection with minimal additional infrastructure.

A detailed conceptual survey of the models mentioned above is shown in Table 2.1.

Table 2.1: Analysis of security models to handle external attacks in cloud security

Year	**Title**	**Technique Used**	**Work Done**	**Future Directions**
2022 [56]	Rotating behind Security: A Lightweight Authentication Protocol Based on IoT-Enabled Cloud Computing Environments	Security Analysis model	Privacy in IoT Services.	Can be employed in Fog and edge computing environments for better results.
2022 [57]	6G wireless communication assisted security management using cloud edge computing.	DL technique	Security management in 6G wireless communication networks.	The Blockchain and intelligent technologies can be improved.
2022 [58]	A Confidentiality-based data Classification-as-a-Service (C2aaS) for cloud security	Encryption/ Decryption technique	C2aaS service is provided to encrypt the confidential data.	Keys can be assigned to sub-files to control data division in the future.
2021 [59]	A Novel Method to Enhance Sustainable Systems Security in Cloud Computing Based on the Combination of Encryption and Data Mining	Decision Tree (DT) Learning and Homomorphic Technique	Security in the cloud computing platform.	Memory overeheads and computational cost can be reduced in future studies.
2020 [60]	Enhancing cloud security using crypto-deep neural network for privacy preservation in trusted environment	Crypto Deep Neural Network (CDNN)	CDNN cloud security is used to handle impersonation attacks.	Incorporate different DL models to forecast the unpredictable nature of node communication.

2020 [61]	Customized Network Security for Cloud Service	Spec analysis and log classification algorithm	Network security of services in cloud computing.	Proposed CNS can be applied on real datasets for performance validation.
2020 [62]	Security information transmission algorithms for IoT based on cloud computing	DES, MD, RSA	Information security in IoT	Resource management can be improved for better performance.
2020 [63]	Enhanced security-aware technique and ontology data access control in cloud computing.	SAT and ODAC	Security and access control in cloud computing.	SAT and ODAC will be applied in edge computing to provide security.
2019 [64]	Designing an efficient security framework for detecting intrusions in a virtual network of cloud computing.	BBA, Random Forest (RF)	Security in a virtual cloud environment network.	The proposed work can be used to detect network attacks.
2019 [65]	Solving Coupling Security Problem for Sustainable Sensor-Cloud Systems Based on Fog Computing	Kuhn-Munkres algorithm	Fog Computing based cloud security.	Inconvenient information can be achieved using sensor cloud which can be resolved.
2018 [66]	Secure integration of IoT and Cloud Computing	AES and RSA Algorithm	Security in IoT and Cloud Computing	The security challenges presented in this research can be minimized.

2016 [67]	Renaissance System Model Improving Security and Third Party Auditing in Cloud Computing	Encryption Technique	Securing data storage in cloud computing.	ML approaches can be employed for better results.
2016 [68]	A QoS-driven approach for cloud computing addressing performance attributes and security	SHA-1, AES, and RSA signature.	QoS-driven techniques for cloud environment	Prototype development with actual hardware will be planned for follow-up research.
2016 [69]	Security framework for RESTful mobile cloud computing Web services	Web service servlet, HTTP listener, request handler, parser, fuzzy logic, augmented offloading, orchestrator, response composer, Keytool Generate Certificate	Security in Restful mobile web services.	Key management can be discussed in further studies.
2015 [70]	A domain-divided configurable security model for cloud computing-based telecommunication services		Security model in cloud computing transmission domain.	To investigate the feasibility, reliability, and scalability, a simulated experimental platform will be built using current resources.

2.1.2 Security models to handle internal attacks

Piotr Nawrocki *et al.* [71] presented a model for allocating tasks that considers security by setting up an allocation algorithm and providing specific data privacy needs of end users. Further, a simulation environment called MocSecSim is developed for testing the proposed algorithms. It is evident from the simulations and experiments that the proposed model improves the security of calculations by a significant amount compared to existing methods.

Seongmo AN *et al.* [72] set up CloudSafe and checked Amazon Web Services (AWS) for security flaws. The in-depth evaluation of four distinct security countermeasure choices, including vulnerability patching, virtual patching, network hardening, and moving target defense, is performed. The project's deployment implementation feasibility for virtual patching, network hardening, and moving target defense is determined. The experiment is performed, and the findings show that the suggested tool CloudSafe is efficient and helpful in assisting security administrators in choosing the most appropriate countermeasures to safeguard their cloud.

A new multi-tier security architecture called SmartX Multi-Sec is proposed by Shin JS *et al.* [73] to ensure the safety of distributed edge clouds. Using an onion-like structure of physical, virtualized, and containerized cloud nodes, SmartX Multi-Sec hides the underlying networking architecture among multi-site edge clusters. The network traffic at each layer of the abstracted topology is monitored, visualized, and filtered using the collective DevSecOps automation functions. To validate the adaptability and scalability of flow-centric security for multi-site cloud-native edge clouds, a Proof-of-Concept (PoC) version of the SmartX Multi-Sec architecture is built, proving the proposed work effectively enhances the distributed edge cloud's safety.

Danish Ahmad *et al.* [74] created a model using AI techniques to protect cloud privacy. Data sanitization and restoration are the two main parts of the proposed privacy-protection system. Shark Smell Optimization (SSO) and Jaya Algorithm (JA) are hybridized into Jaya-based Shark Smell Optimization (J-SSO), which is then used to generate a key. A best key pair is generated using a multi-objective function based on the parameters like the degree of modification, hiding ratio, and information preservation ratio. The results showed that the proposed algorithm improved cloud security compared to the existing works.

Chenquan Gan *et al.* [75] tackled the problem of malware spreading between virtual machines by presenting a dynamical propagation model to investigate the drives of malware distribution and the role of antivirus software in VMs. A theo-

retical analysis of the model is conducted to understand the malware spreads in a cloud environment where it is compromised using differential dynamics. The efficiency of the proposed work is validated, and the numerical simulations performed well compared to existing works.

Haralambos Mouratidis *et al.* [76] introduces a novel security modeling language and a set of original analytic methodologies to capture and analyze security needs for cloud computing systems by integrating cloud computing, security, and goal-oriented requirements engineering, which help elicit, model, and analyze cloud infrastructures' security needs. Further, three analysis techniques are proposed that underpin an automated process that, when presented with a model of a cloud computing system built with the proposed language, updates that model with new security knowledge, such as threats and vulnerabilities, mitigation strategies, assets, and actor responsibilities. The experimental finding shows the effectiveness of the proposed work.

Muhammad Imran Tariq *et al.* [77] suggested an agent-based information security framework for hybrid cloud computing as an all-inclusive approach incorporating cloud-related approaches. Software and intelligent agents' ideas are introduced, and accurate data is collected and passed to the proposed framework to take action against the threat agents. Experiments and results show that ISO/IEC 27005:2011 is the best way for ISRA after comparing it to ABISF and other well-known ISRA approaches. It is also evidenced that the fuzzy-logic analysis ensured data safety in cloud computing settings.

Syed Rizwi *et al.* [78] provided a paradigm and mechanism for evaluating a Cloud Service Provider (CSP) security strength based on the customer's security preferences. A security evaluation paradigm that includes conceptual and quantitative models, including linear and non-linear equations, is provided. The conceptual model increased the theoretical understanding of cloud security challenges. The quantitative model provided scientific approaches for establishing a security index score for any CSP or comparing numerous CSPs for different security preferences. The final security index value for a given CSP is determined from the experiments, and the safety index calculations are also analyzed for several CSPs.

A cloud security evaluation approach, named Cloud-Trust, is introduced by Dan Gonzale *et al.* [79]. Cloud-Trust estimates high-level security metrics to measure the degree to which a Cloud Computing Service (CCS) or cloud service provider protects user data. Four multi-tenant IaaS cloud architectures with different cloud security policies are compared and contrasted using Cloud-Trust.

The results reveal that even with the most basic security measures in place, there remains a significant risk of a CCS breach (compromising sensitive data).

Laurence T. Yang *et al.* [80] suggested a unique parallel block Wiedemann technique on the cloud to reduce communication costs for solving large and sparse linear systems over GF(2). The presented algorithm comprises three steps, including strip partitioning, cyclic partitioning, and modified strip partitioning to parallelize the acceleration of the block Wiedemann algorithm. The experiment and results show that the parallel block Wiedemann algorithm significantly improves GNFS performance compared to other existing algorithms regarding execution time and speedup.

Minhaz Ahmed Khan *et al.* [81] gives an overview of security threats and how to fix them. The goal of this contribution is to analyze and classify the ways that the main security problems work and the possible solutions that are written about. The parameters are used to compare the risks that cloud platforms face. Further, the different frameworks used to find and stop intrusions and deal with security problems are compared. Moreover, how cloud security issues might change and how to fix them are also discussed.

Murat Yesil *et al.* [82] provided an in-depth analysis of the primary elements contributing to fulfilling the security requirements posed by each model and the factors and considerations that should be prioritized. In addition, the strategies and instruments for implementing security, confidentiality, and integrity of the information or data that forms the basis of modern technology are investigated and analyzed. The proposed work concluded that employing data hiding methods regarding access security in cloud computing architecture and the security of the stored data would be highly useful in securing information.

An innovative Network Security Architecture for Cloud Computing called NetSecCC is presented by Jin He *et al.* [83], which offers security to both the internal and the external traffic in cloud computing and achieves flexible scalability concerning a load of virtual middleboxes. Experiments and simulations are performed, and it is validated that NetSecCC is a compelling architecture with minimal performance overhead. It can be applied to extensive practical promotion in cloud computing.

The analysis of the security models to tackle internal attacks is done in Table 2.2.

Table 2.2: Analysis of security models to handle internal attacks in cloud security

Year	**Title**	**Technique**	**Work Done**	**Future Directions**
2022 [71]	Modeling adaptive security-aware task allocation in mobile cloud computing	Security Filteration Algorithm	Data Privacy with task allocation.	Model Optimization will be done.
2022 [72]	Toward Automated Security Analysis and Enforcement for Cloud Computing Using Graphical Models for Security	Hierarchical Attack Representation Model	Automated Security in cloud computing	Anomaly detection can be defined as detecting zero-day attacks.
2021 [73]	SmartX Multi-Sec: A Visibility-Centric Multi-Tiered Security Framework for Multi-Site Cloud-Native Edge Clusters	DevSecOps automation features with extended Berkeley Packet Filter	Monitoring and filtering of network traffic is done using DevSecOps automation function.	SmartX Multi-data Sec's analysis module will be enhanced using intelligent intrusion detection algorithms.
2020 [74]	A multi-objective privacy preservation model for cloud security using hybrid Jaya-based shark smell optimization	Hybridized J-SSO	Hybrid metaheuristic is used to preserve data.	Advanced optimization techniques can be applied in future.

2020 [75]	Dynamical Propagation Model of Malware for Cloud Computing Security	Dynamic Propagation (Local Stability, Global Stability, & Equilibrium)	Malware propagation among VMs in IaaS environment.	The proposed work can be extended to control strategies which lacked in the current study.
2019 [76]	A security requirements modeling language for cloud computing environments	Resource knowledge base, cloud and threat model	Security requirements for cloud computing	Computational time will be reduced in future work.
2018 [77]	Agent Based Information Security Framework for Hybrid Cloud Computing	Agent Technique, Fuzzy logic simulations	The decision system introduced by agents for cloud security.	The proposed framework can be validated using different scenarios.
2017 [78]	A security evaluation framework for cloud security auditing	Conceptual and qualitative model	Security index score is calculated with Linear and non-linear equations	Fuzzy Logic can be used to develop a set of evaluation rules for security purposes.

2017 [79]	Cloud-Trust—a Security Assessment Model for Infrastructure as a Service (IaaS) Clouds	Bayesian Network (BN) Model	Computed conditional probabilities using a BN	Extension to PaaS as well as SaaS services.
2016 [80]	Parallel GNFS algorithm integrated with parallel block Wiedemann algorithm for RSA security in cloud computing	General Number Field Sieve (GNFS) algorithm	Security of RSA algorithm	Wiedemann method can be used to reduce transmission time and enhance parallel performance.
2016 [81]	A survey of security issues for cloud computing.	-	Security issues and threats in cloud.	ML and DL approaches for IDS can be considered in future.
2014 [83]	NetSecCC: A scalable and fault-tolerant architecture for cloud computing security	Security Management, Fault Tolerant Technique	Network Security in cloud computing.	NetSecCC can be applied in fog and edge computing security.

Different models used for security in the cloud environment are studied and discussed in detail. It is realized that different researchers have proposed many models and techniques to defend the security attacks. Besides the nature of the attack, it is essential to understand the attacker's behavior. Game theory helps understand the attacker's nature and predicts his strategies. The following Section discusses different game theory models to understand the attacker's strategies.

2.2 Game Theory Security Models in Cloud Computing

This Section is divided into five different game theoretic models for cloud security. These are Stochastic games, Differential games, Non-Cooperative games, Stackelberg games, and other game models. Each model is discussed in detail below.

2.2.1 Stochastic Game Theoretic Security models

Lukasz Gaza *et al.* [84] presented the Epistemic Games with Conditional Believes paradigm for automating security decisions concerning Cloud Computing system security. The paper lays forth a process for factoring in one's assumptions about the rationality of one's opponent. The given model makes it possible to contemplate an assault on a Cloud system seriously. Experimental evaluation of the Cloud Sim simulator validates the proposed solution. Cloud services providers can use the presented approach to identify measures against cyber attacks on their data and infrastructure.

Tengchao Ma *et al.* [85] came up with the idea for a lightweight, adaptive intelligent defense strategy, which is implemented on the client and did not require any configuration support from Domain Name System (DNS). In the first step, the process of attack and defense as a static stochastic game with incomplete knowledge and restricted rationality is described, followed by the use of Deep Reinforcement Learning (DRL) which is proposed to ensure monotonic progress. The simulations are performed on the Alibaba Cloud, and the efficiency of the proposed solution is demonstrated against numerous attacks, which shows a success rate of around 97.5%.

Qimeng Li *et al.* [86] created a security game evaluation model based on dynamic games and examined the system from the perspectives of confidentiality, integrity, and availability of the game. The game process between cloud-based security defense and hostile attackers is examined. The optimal defense strategy

of the Smart Grid (SG) is studied, and the security risk value of the system is determined by calculating the equilibrium solution of the model. The simulation demonstrated that the developed game model and framework could improve the accuracy of risk assessment of nodes and optimize the defense strategy.

El Mehdi Kandoussi *et al.* [87] developed an integrated defensive system combining honeypots with virtual machine migration. In addition, the proposed model quantitatively determines the potential attack paths and is further divided into two distinct subsets: attack paths that are merely investigated and attack paths that are studied and exploited based on the black box intrusion stages. Further, in conjunction with the stochastic game theory, the attack graph simulates the interaction between the attacker and the defender. The experiment is performed, and the numerical findings illustrate the effectiveness of the suggested security game model.

Amandeep Singh Sohal *et al.* [88] suggested a cyber security system that employs the Markov model, IDS, and Virtual Honeypot Device (VHD) to detect malicious edge devices in a fog computing setting. The four classes of edge devices are efficiently classified using a two-stage hidden Markov model. VHD is created to keep a log repository of all known harmful devices, which helps the system protect itself against future attacks. The experiment was performed, and the results showed that the suggested cyber security architecture effectively detects malicious devices and decreases the number of false IDS alarms.

Junjie LV *et al.* [89] presented a model for assessing the security risks associated with cloud services using stochastic game nets. With the help of visual aids, cloud services' virtualization security risk scenario is presented in detail, and the virtualization security risk factors are evaluated with precision. The investigation findings demonstrated the robust capacity to model intricate and variable security problems in cloud services. The results assist cloud service providers and tenants in implementing necessary countermeasures.

Guishang Fan *et al.* [90] suggested a stochastic game model that combines stochastic Petri nets and game theory to characterize the attack-defense behaviors in cloud computing. The best defensive tactic is determined by computing a physical machine's Nash equilibrium (NE) of the attack-defense process. The related theories of Petri nets and the attainable states of the attack-defense game model are utilized. Further, an enforcement algorithm is presented to pick the defense strategy to counterattack behavior as rapidly as possible. The case study findings and the simulation reveal that the suggested method can swiftly adjust to new cloud application requirements by enhancing the safety of cloud computing.

Research is conducted by Priti Narwal *et al.* [91] in which analysis of the security risks that OpenStack private clouds face and the effects of those risks is done. The most common DOS (Denial-of-Service) attacks on the Dynamic Host Configuration Protocol (DHCP) server on the private cloud platform are examined. The vulnerabilities in an OpenStack networking component called Neutron are evaluated and carried out through a rogue DHCP server. At last, a game-theory-based cloud architecture is proposed that can assist with detecting and preventing denial-of-service assaults in OpenStack.

Liu *et al.* [92] investigated the existing security mechanism that the organizations used to protect against the attack. The stochastic evolutionary coalition game framework is proposed for sensor services. The parameters of resource availability and QoS are checked at every game stage, and the attacker's strategies are delineated from previous stages. The evolutionary coalition algorithms are used to defend the attacker's strategy. The Markov Chain is constructed by which the sensor nodes can learn the best strategy and maximize the payoffs of the defender. The proposed framework can also be used in other cloud platforms.

The analysis of the stochastic game-theoretic security models in cloud computing is discussed in Table 2.3.

2.2.2 Differential Game Theoretic Security Models

Hengwei *et al.* [93] used complex networks and differential game theory to explore the real-time defense decision problem. First, a scale-free propagation model has been created for real-time security analysis. The assault and defensive strategy, revenue calculation based on confrontation analysis, and network node security state modification based on equilibrium strategy have been then discussed. Finally, a simulated network has been created for tests. The proposed work performed well and showed an improvement of 17.3% when compared with existing works.

Li *et al.* [94] designed a differential game model of IDS in cloud computing. The rate of detecting the intrusions and the false alarms generated by the IDS is considered. To investigate the game model, two players are chosen, out of which one is a rational cloud resource defender and the other is a malicious user. The interactions and strategies between the players are analyzed until the Nash equilibrium is achieved. This work leads to deciding the optimal strategies by the IDS for the defense of the cloud resources by theoretical foundations of the game theory.

Table 2.3: Analysis of stochastic game-theoretic security models in cloud computing

Year	**Title**	**Work Done**	**Future Directions**
2022 [84]	Epistemic Games with Conditional Believes for Modelling Security Threats Defence in Cloud Computing Systems	Automatic Security Decisions in Cloud Computing	More advanced game models that blend Stackelberg Games, Epistemic Game Theory, and conditional beliefs can be used.
2020 [85]	Intelligent-Driven Adapting Defense Against the Client-Side DNS Cache Poisoning in the Cloud	Intelligent defense strategy to secure attacks on cloud environment.	Global optimization can be applied for better performance.
2020 [86]	A Risk Assessment Method of Smart Grid in Cloud Computing Environment Based on Game Theory	Security risk assessment of cloud-based Game Theory SG	Analyse and evaluate the cloud-based SG system's information security concerns based on the third-party audit.
2019 [87]	Toward an integrated dynamic defense system for strategically detecting attacks in cloud networks using stochastic game	Game Security model in the cloud computing environment.	Dynamic stochastic game model can be used to reduce the inefficient migrations and to adjust the configuration of the defender's strategy.

2018 [88]	A cyber security framework to identify malicious edge device in fog computing and cloud-of-things environments	Prevention of malicious edge devices in fog computing environment	The proposed work can be integrated with ethical hacking to deal wiith hacked devices.
2018 [89]	Virtualisation security risk assessment for enterprise cloud services based on stochastic game nets model	Security risk assessment for cloud services.	Investigate security vulnerabilities and expand the model application scenario to assess cloud service security risk.
2017 [90]	A game-theoretic method to model and analyze the attack-defense strategy of resource service in cloud application	Attack-Defence strategies in cloud environment	QoS parameters, as well as integration of offset tools, can be considered for better results.
2017 [91]	Game-Theory based Detection and Prevention of DoS Attacks on Networking Node in Open Stack Private Cloud	DoS attacks security in OpenStack Private Cloud	Cooperative game theory can be used in future for prevention from DoS attacks with better performance.
2015 [92]	A stochastic evolutionary coalition game model of secure and dependable virtual service in Sensor-Cloud	Maximizes the expected sum of discounted payoffs using min-max Q learning.	Proposed work can be used in other cloud computing services.

Cheng [95] designed a non-cooperative differential game model between IDS and attackers in Wireless Sensor Networks (WSN). The sensor is clustered in this model, and each cluster head node is assigned with the IDS. Nash Equilibrium finds the solution of the game between IDS and the attacker. The optimal strategies that increase the payoffs of the IDS are obtained. The Theoretic model is simulated, and the results are obtained in favor of IDS.

The analysis of the differential game theoretic security models in cloud computing is discussed in Table 2.4.

Table 2.4: Analysis of differential game theoretic security models in cloud computing

Year	**Title**	**Work Done**	**Future Directions**
2017 [93]	A differential game approach for real-time security defense decision in scale-free network.	Network attack–defense game model to solve the equilibrium strategy.	RL can be applied in future to provide network security.
2017 [94]	A differential game model of intrusion detection system in cloud computing	Detection of malicious attacks to improve defensive ability	Ml approaches can be used for IDS.
2016 [95]	A Differential Game Model Between Intrusion Detection System and Attackers for Wireless Sensor Networks	Improve IDS strategies to secure WSNs.	Combine the differential model with WSN for better performance.

2.2.3 Non-Cooperative Game Theoretic Security Models

Yi Han *et al.* [96] focused on one risk attack, the co-resident attack at the virtual machine level, by figuring out the attackers and legal users. Then, the users are grouped using clustering analysis and semi-supervised learning. Finally, the problem is modeled as a two-player security game in which the best moves for each player are analyzed. From experimental findings, it is found that the proposed mechanism raises the attacker's total cost by one to two orders of magnitude.

Kai Li *et al.* [97] proposed a dynamic game theory-based edge computing container security risk assessment approach. The dynamic game theory-based

container system security model targets the complex container security attack and defensive process. The NE solution of the model is constructed by merging the attack and defensive matrices, and the dynamic process of the security defense-malicious attacker mutual game is investigated. The model's feedback NE solution calculates the attackers' optimal strategy. Finally, the simulation tool solves the feedback NE solution of the two players in the proposed model, and the experimental setting validates the risk assessment approach.

A game theory is employed by Kaho Wan *et al.* [98] to examine the interplay between many actors when the cloud is under attack. If the cloud service provider made a logical strategy choice, the resultant NE demonstrates that collateral damage is uncommon; however, the NE may shift if the cloud service provider does not treat cloud users equally. The cloud provider's prejudices may affect its strategies in response to Distributed Denial of Service (DDoS) attacks, leading to unintended harm for users who are not the intended targets.

Dongao Zhang *et al.* [99] developed a cloud-based framework to safely outsource finding mixed-strategy Nash equilibria. Before being uploaded and stored in the cloud, the payout matrix in the proposed work is first protected with additive homomorphic encryption and then decrypted. The cloud server can calculate Nash Equilibria on the encrypted data by combining secure multi-party computing approaches. This is accomplished without compromising the privacy of the cloud's users. Experiments are carried out to validate the method's efficiency, evaluations of the precision are carried out, and finally, an analysis of the computational complexity is done, which is significantly less compared to existing works.

Yan Sun *et al.* [100] developed a model for an optimal defense strategy based on a differential game to solve the optimal defense strategy of edge nodes in both infinite and finite horizons. This is done by the characteristics of limited resources of edge devices, which are combined with the theory of dynamic gameplay. The optimal defense strategy of the edge nodes is simulated under various scenarios. The simulation demonstrates that the edge nodes may acquire the optimal defense effect with the minimal consumption of resources when working together to construct the defensive system.

Qianmu Li *et al.* [101] used game theory in edge computing networks to propose a data-driven imitation intrusion detection game-model-based technique called GLIDE, based on dynamic intrusion detection. In-depth analyses of how players' earnings and utility computing techniques change over various deployment options are provided. The experiment is performed, and by determining the ideal deployment plan for the multi-redundancy edge computing terminal intrusion detection

service in the edge computing network, it is found that the defender won the game.

Poria Pirozmand *et al.* [102] developed the performance of IDS using game theory. The suggested method thoroughly analyzed the attacker infiltration mode and the behavior of the IDS as a two-player and non-participatory dynamic game. The NE solution is employed to generate specialized subgames. The simulation results of the suggested method revealed that using IDS based on cloud fog in the Internet of Things can be highly effective in recognizing attacks with the fewest errors in this network.

To protect the hypervisor from potential dangers, Kashish Prabhakar *et al.* [103] developed a game strategy block similar to a tower defense game and established rules for security depending on those principles. In addition, an effort is made to develop a utility function called the Virtual Machine Vitality Measure (VMVM), intended to shed light on the current state of the virtual machines in the virtual environment.

Sane Branade *et al.* [104] proposed a new method for mitigating VM escape attacks in which the attacker's goal is to co-locate their VMs and the target VMs on the same physical server. Consequently, various fundamental VM allocation policies using a game-theoretic technique are simulated to provide quantitative analyses. In each VM allocation policy, the attacker determines when to start the VMs, how many VMs to start, and the VMs' security level. The experiment is performed on CloudSim under the VM allocation policies. Our findings indicate that the Round Robin allocation policy is the least safe against VM escape attacks.

Soodeh Hosseini *et al.* [105] presented a model for identifying vulnerable cloud computing data centers. This model is established based on examining the cloud computing system conducted in the game theory field. The game theory is used as a mathematical tool to develop the model. By game theory, the authors measured the degree to which data centers in the cloud computing network are susceptible to attack.

A framework is created by Hao Wu *et al.* [106] for the game-theoretic study of collaborative security detection by considering the conflict between the defender and the attacker. NE in a game model with a perfect agreement is studied, and its existence and uniqueness are determined. A computation approach based on iterative learning that yields the NE is provided. Complete and partial consensus with an infinite and finite number of iterations are considered, and the relationship between the Nash equilibriums of the game models is analyzed quantitatively. The experiment was performed, and the findings showed that the proposed work performed well compared to the current work.

Amin Nezarat *et al.* [107] presented a group of mobile agents that operate in the cloud environment as the sensors of invalid operations. Authors differentiated an attack from legitimate requests, as well as determined the severity of the attack and its point of origin. A game of non-cooperation is played with the person suspected of being the attacker, and then the NE value and utility are calculated. The simulation results reveal that the proposed strategy is accurate 86% of the time while detecting attacks.

Fahad Yaman Choudhary *et al.* [108] presented an innovative method for warding off Economic Denial of Sustainability (EDoS) attacks using game theory. A static game scenario is created to depict an interactive game between an attacker and a defender. The suggested EDoS Eye approach could locate a superior strategic threshold value via NE. This threshold value is capable of efficiently blocking EDoS traffic. Additionally, adding honeypots within the proposed model helps lower the false rate. The simulation outcomes further validated the efficiency of the proposed work.

Fan *et al.* [109] proposed a model to analyze resource service attack and defense strategy in cloud applications using a game theoretic approach. The game model works on stochastic Petri-nets, which define the behavior of the attacker and the defender. An enforcement algorithm is designed to find the possible attack path and enforce strategies according to the path. The Nash equilibrium is computed in a physical machine to choose the best strategy. The simulation results verified that the defense strategy chosen by the defender quickly tackled the attacker and thus improved the security of the cloud.

The analysis of the non-cooperative game theoretic security models in cloud computing is discussed in Table 2.5.

2.2.4 Stakelberg Game-Theoretic Security Models

Xueqin Liang *et al.* [110] created a formal economic model of Hybrid Encrypted Cloud Data Deduplication *(H-DEDU)* by quantifying the benefits to data holders, data owners, and CSP. Then, a Stackelberg game is built to model the interplay between the various parties involved in the system. The requirements for a subgame perfect NE are examined, and a gradient-based approach is presented to aid stakeholders in selecting near-optimal strategies. The experiment is performed, and it is evident that the proposed approach effectively obtains the NE of the Stackelberg game in extensive testing.

Table 2.5: Analysis of non-cooperative game theoretic security models in cloud computing

Year	**Title**	**Work Done**	**Future Directions**
2022 [96]	Game Theoretical Approach to Defend Against Co-Resident Attacks in Cloud Computing: Preventing Co-Residence Using Semi-Supervised Learning	Game Theory and security in Virtual Machines	The defense mechanism on normal users can be considered for practical use.
2021 [97]	Security Risk Assessment Method of Edge Computing Container Based on Dynamic Game	Security risk assessment of edge computing	ML and DL can be applied for security risk assessment in future.
2021 [98]	Game-Theoretic Modeling of DDoS Attacks in Cloud Computing	Security threat in cloud computing	Examine a recurring dynamic game.
2021 [99]	Privacy-Preserving Outsourced NE Computation in Cloud Computing	Cloud-based framework to secure outsourcing the task of computing mixed-strategy	The proposed approach can be employed in cloud computing applications to secure players' privacy.

2020 [100]	Optimal defense strategy model based on the differential game in edge computing	Defense model for complicated dynamic edge computing environment	Work on computation overhead can be done in future.
2020 [101]	GLIDE: A Game Theory and Data-Driven Mimicking Linkage Intrusion Detection for Edge Computing Networks	Game Theory in the field of the edge computing network.	Darwin's theory of biological evolution and Lamarck's genetic theory can be used to provide security.
2020 [102]	Intrusion Detection into Cloud-Fog-Based IoT Networks Using Game Theory	Cloud-Fog based Intrusion detection system	The proposed framework can be used in real-time applications.
2019 [103]	Securing Virtual Machines on Cloud through Game Theory Approach	VM security in game theory based Cloud environment	Security of end devices using cotainer security can be applied in future.
2019 [104]	A Game Theoretic Approach for Virtual Machine Allocation Security in Cloud Computing	VM allocation security in cloud computing	Consider cost, coverage, power consumption, and workload balancing under alternative VM allocation policies.
2019 [105]	Game theory approach for detecting vulnerable data centers in cloud computing network	Vulnerability of malware data centers and cloud network structure in game theory	Concentrate on the degradability of service in optical data centers.

2018 [106]	A Game Theory Based Collaborative Security Detection Method for Internet of Things Systems	Game theoretical framework for the collaborative security detection in IoT systems	Determine the strict criteria under which the iterative learning algorithm will converge.
2017 [107]	A game theoretic-based distributed detection method for VM-to-hypervisor attacks in cloud environment	Game theory and virtualization security	The proposed work is time consuming, so better model can be applied to reduce time complexity.
2017 [108]	EDoS eye: A game theoretic approach to mitigate economic denial of sustainability attack in cloud computing	Combat EDoS attacks using game theory in cloud computing.	To develop a dynamic game model where players can change strategy in a single game.
2017 [109]	A game-theoretic method to model and analyze the attack-defense strategy of resource service in cloud application	Help the defender to select optimal defense strategy and quickly, securely allocate resources.	Consider QoS for cloud computing security.

A recurring Bayesian Stackelberg game with the help of a risk assessment framework and a live-migration defense mechanism using an ML-based technique is presented by Omar Abdel Wahab *et al.* [111] that collects malicious data from VMs using honeypots and employs one-class Support Vector Machine *(SVM)* to learn the distributions of the attackers' types. The optimal distribution of the detection load over the VMs in the hypervisor is determined. The findings showed that the proposed work outperforms state-of-the-art detection and defensive strategies such as Collabra, probabilistic migration, Stackelberg, max-min, and fair allocation using data from Amazon's data center and AWS honeypots.

El Mehdi Kandoussi *et al.* [112] suggested a defense system combining virtual machine migration and honeypot. Security policies are used to discuss how well the proposed system would work. The proposed model quantifies the possible attack paths and divides them into two groups. These groups are based on the black box intrusion steps—the attack graph and the stochastic game theory model of how the attacker and the defender work together. The experiment is performed, and the numerical results show that the proposed security game model works well compared to existing works.

Agnieszka Jakóbik *et al.* [113] introduced the Stackelberg Game as a model for the robotic management of security decisions in cloud computing by allowing the description of attack-defense scenarios. Defenders and attackers face off against one another in this game. First, the attacker tries to attack, then the combined efforts of the attackers are represented by the second player. The second player reacts to the leader's actions and pursues their own goals when making choices. The black-box approach is introduced to determine an attacker's tactics. The use of several ANN processing pipelines derives the utility function. The experimental findings facilitate the cloud service provider's ability to identify the appropriate defensive measure.

Agnieszka Jakóbik *et al.* [114] defines a model based on Stackelberg games and enabling the automatic selection of provider-level security decisions in systems that make use of cloud computing. The experiment is performed that will result in the most significant gain for the defense while providing the most negligible benefit to the attacker. The model's effectiveness in automatically selecting protection strategies from the point of view of the cloud provider is experimentally validated, which shows that the proposed work is best compared to existing techniques.

The analysis of the Stackelberg game-theoretic security models in cloud computing is discussed in Table 2.6.

Table 2.6: Analysis of Stackelberg game-theoretic security models in cloud computing

Year	**Title**	**Work Done**	**Future Directions**
2021 [110]	Investigating the Adoption of Hybrid Encrypted Cloud Data Deduplication With Game Theory	Encryption of cloud data using Game Theory	Multiple CSPs competing in a worldwide market will be considered.
2021 [111]	Resource-Aware Detection and Defense System against Multi-Type Attacks in the Cloud: Repeated Bayesian Stackelberg Game	Detection and defense mechanism for VMs risk assessment.	Optimized data-driven solutions can be provided in future for cyber security.
2020 [112]	Toward an integrated dynamic defense system for strategic detecting attacks in cloud networks using stochastic game	Security measures as intrusion prevention and detection in cloud environment	A dynamic stochastic game can be employed to reduce inefficient migrations and adapt the defender's security settings based on the attacker's interaction history with the VM.
2020 [113]	Stackelberg game modeling of cloud security defending strategy in the case of information leaks and corruption	Game Theory and security decisions in cloud computing	More advanced algorithms combined with Nash algorithms.
2018 [114]	Stackelberg games for modeling defense scenarios against cloud security threats	Automatic Defence Strategies in Cloud Computing Security.	The obtained mathematical formulation can also be used for predicting attack techniques on cloud systems in which the attacker behaves rationally.

2.2.5 Other Game Theoretic Security Models

E Balamurugan *et al.* [115] designed a framework referred to as IDSGT-DNN to enhance the cloud IDS system's level of security. A mechanism for standard data processing and an attacker mechanism is incorporated with game theory through IWA to locate the best possible solutions. The performance of the proposed IDSGT-DNN model is assessed using the CICIDS-2017 dataset. The experiment is performed, and the proposed IDSGT-DNN network is compared to an existing approach. The simulation analysis results demonstrated that the suggested IDSGT-DNN achieves a higher accuracy, detection rate, and precision, as well as F-score, AUC, and FPR.

Quang HE *et al.* [116] makes the initial effort to address the edge DDoS mitigation (EDM) issue by treating it as a constraint optimization problem and is demonstrated as an NP-hard problem. The edge DDoS attacks are mitigated with an innovative game-theoretical technique called EDMGame. The EDM problem is formulated as a prospective EDM Game that can admit a NE. A decentralized algorithm is used to locate the NE, which solves the EDM problem. The experiment is performed, and the results demonstrate that the proposed approach can successfully and efficiently resolve the EDM problem.

T.P. Anithaashri *et al.* [117] has come up with a new framework to solve the problems of cloud security by using a mathematical method called "novel dominant game strategy" in software-defined networks. The framework sets up an improved version of an intrusion detection and prevention system and gives cloud services high security when deployed. The author has tested algorithms on 25 test beds with different numbers of nodes in a simulation of the cloud. The best result is that network services through the cloud worked 90% of the time. The work is compared to an algorithm for a lexicographic game, showing that cloud services are not very useful during peak hours. Together, the new framework and the most popular game strategy in software-defined networks solve the problems caused by fuzzy logic and lexicographic games, which makes software-defined networks better at providing network services.

Banavath Balaji Naik *et al.* [118] proposed a secure VM allocation against co-resident attacks utilizing support value-based game policy to reduce cloud security risks and maximize efficiency, coverage and link utilization. For each multi-objective value, the support value is calculated. Finally, game theory selects the optimal VM. Experimental results showed that the suggested method surpasses existing methods in coverage, efficiency, maximum link utilization, execution time, latency, throughput, power consumption, workload balance, and standard devia-

tion.

Pan Jun Sun *et al.* [119] came up with a quantitative weight model of privacy information, used evolutionary game theory to set up a game model of attack protection, made an algorithm for choosing the best protection strategy, and used the limited rational constraints to make an evolutionary solution method. An improved evolutionary game model of attack protection is built. The stability of the equilibrium point is further studied by the Jacobian matrix method, and the optimal selection strategy is found under different conditions. Lastly, experiments show that the model is correct and valid compared to other state-of-the-art works.

A cloud-based platform is suggested by Mina Emami *et al.* [120] to administer IoT service selection and composition in fog computing. To optimize CPU utilization, power consumption, and latency of IoT workflows in cloud-assisted fog computing settings, a multi-objective evolutionary game theory is offered that is enhanced by an evaporation-based water cycle algorithm (EG-ERWCA). The experiment is performed, and the simulation findings demonstrate a 2.66-fold increase in the overall quality of service compared to competing methods.

Fanyu Kong *et al.* [121] presented a security reputation model called SCNN-DGT based on a Convolutional Neural Network (CNN) called S-AlexNet and dynamic game theory by ensuring the confidentiality of patient's health records within the IoT. First, a CNN is trained with the S-AlexNet algorithm on textual input that makes up the user's health data. Then, a recommendation incentive approach is suggested based on dynamic game theory. The experimental study is conducted to test the correctness of the model, and it is found that the proposed work can tackle the problems of low reliability of the health data screening index and low precision of credit distinction in cloud environments.

Talal Hababi *et al.* [122] uses the Goal-Question-Metric (GQM) approach to develop a set set of parameters that quantitatively describes the Security-SLA in the Cloud. Further, a hedonic coalitional game model is used to form a Cloud federation, with a preference relation by considering the quality of security and the standing of individual CSPs. Moreover, an algorithm for federation creation that allows CSPs to join federations while incurring minimal security losses is proposed. From the experiments, it is found that the presented framework reduced the frequency and severity of Security-SLA breaches in the created federations, proving its effectiveness in practice.

Baris Bulent Karlar *et al.* [123] has proposed a brand new and highly effective encryption method that uses XTR *i.e.* effective and compact subgroup trace representation. These interactions are constructed in the direction of cryptographic

tools that address a natural optimization problem in game theory and financial economics. From the simulations, it is found that game theory and its optimization have provided a suitable framework for designing a crypto-cloud computing system that is seen as a powerful technique that satisfies the needs of many participants and users of the cloud.

Fadlullah *et al.* [124] optimized the QoS and security in next-generation heterogeneous networks by game theory. Imposing security in any network leads to lower QoS parameters. A GT-QoSec framework is designed to balance the security and the QoS parameters. The transition matrices are considered, and the NE is obtained in the expected number of steps.

Rao *et al.* [125] explained the role of game-theoretic models in defending cyber infrastructures from cyber-physical attacks. The proposed framework is based on two game models. The boolean attack-defense model is considered between attacker and defender. The probability of successful attacks and defense is calculated, and the Nash Equilibrium is computed in polynomial time. The analytical results are general and can be applied to high-performance computing infrastructures.

Duy La *et al.* [126] presented a game model in honeypot-enabled networks for IoT. The theoretic game approach is used in which attackers use various attacks to deceive the defender. The defender, in turn, uses honeypots to trap the attacker. The Bayesian game of incomplete information is modeled, and the equilibrium is calculated for the one-shot and repeated games. The defender used the mixed strategy to decide whether to deploy a honeypot or not to keep the energy level optimum and the attacker's success rate low.

Wang *et al.* [127] studied the existing security problems in ad hoc networks and designed a mean field game theoretic approach to enhance security. In the mean-field game, the game is played between many players, and any players can use their strategies to defend the network. The designed model is fully distributed, and each node should know the aggregate effect on the other nodes. The simulation of the mean-field game is done, and the results show low energy utilization and security value loss.

The analysis of the other game theoretic security models in cloud computing is discussed in Table 2.7.

Different researchers have proposed different security models based on game theory. The commonly used games are Stackelberg games, Differential games, Non-Cooperative games, and Stochastic games. From a security scenario, the game is modeled between the attacker and the defender.

Table 2.7: Analysis of other game theoretic security models in cloud environment

Year	**Title**	**Area**	**Future Directions**
2022 [115]	Network optimization using defender system in cloud computing security based intrusion detection system with game theory deep neural network (IDSGT-DNN)	Cloud security in IDS	Proposed work can be applied to real time application scenarios like intelligent transportation system.
2022 [116]	A Game-Theoretical Approach for Mitigating Edge DDoS Attack	Mitigating DDoS attacks in edge computing	Latency models can be integrated for domain-specific applications.
2021 [117]	Enhancing the Cloud Security using Novel Dominant Game Strategy	Security issues in cloud computing technologies.	Machine intelligence will automate the cloud security monitoring system.
2021 [118]	Secure virtual machine allocation against attacks using support value-based game policy	Security threats of cloud and allocation of best VM	Better game strategy such as bayesain Nash equilibrium can be used for better performance.

2020 [119]	Research on the Optimization Management of Cloud Privacy Strategy Based on Evolution Game	Evolutionary game model for privacy security in cloud computing	Address the attack-protect strategy set, trustworthy third-party supervision, and incentive coefficient measurement.
2020 [120]	A modified water cycle evolutionary game theory algorithm to utilize QoS for IoT services in cloud-assisted fog computing environments	Cloud-assisted fog computing to enhance QoS parameters.	The model optimization can be done for better performance.
2019 [121]	A Security Reputation Model for IoT Health Data Using S-AlexNet and Dynamic Game Theory in Cloud Computing Environment	Data Privacy in IoT	Optimizing the CNN, the performance can be improved in future studies.
2018 [122]	Towards Security-Based Formation of Cloud Federations: A Game Theoretical Approach	Security in cloud federation	More in-depth discussion of the proposed work can be done to build the Security-SLA baseline.

2016 [123]	A game-theoretical and cryptographical approach to crypto-cloud computing and its economic and financial aspects	Game Theory and optimization in crypto-cloud computing	The cooperative game theory and obligation rules can be included for efficient encryption algorithms.
2016 [124]	GT-QoSec: A Game-Theoretic Joint Optimization of QoS and Security for Differentiated Services in Next Generation Heterogeneous Networks	Next generation heterogeneous networks	GT-QoSec can be applied to cloud computing for providing security.
2016 [125]	Defense of Cyber Infrastructures Against Cyber-Physical Attacks Using Game-Theoretic Models	Attack-defense model for cyber security.	More details of system-specific details of Cloud computing infrastructure should be considered.
2016 [126]	Deceptive Attack and Defense Game in Honeypot-Enabled Networks for the Internet of Things	honeypot based deception mechanism.	The proposed work can be applied on real-time datasets to prove its effectiveness.
2016 [127]	Dynamic GameModel of Botnet DDoS Attack and Defense	DDoS attack protection on the servers.	DL algorithms can be used for IDS.

These two players have opposite goals. So, the best game model is the non-cooperative game model that can be used to analyze the behavior, actions, and strategies of the attacker and the defender. The best defender system used in the modern era is the IDS, based on Machine Learning and Deep Learning techniques. The survey of both these techniques to optimize the IDS is discussed in the next Section.

2.3 IDS in Cloud Computing

The Section is divided into IDS with Machine Learning (ML) techniques and IDS with Deep Learning (DL) techniques, used to optimize IDS. The feature extraction and optimization techniques applied to real-time datasets are discussed in detail. Further, the performance is evaluated using three performance parameters: accuracy, detection rate (DR), and False Positive Rate (FPR). These are discussed as follows.

2.3.1 Machine Learning Techniques to Optimize IDS for Cloud security

Machine Learning (ML) also plays a vital role in IDS. Various researchers have worked on IDS using ML; *e.g.*, Md. AlaminTalukder *et al.* [128] presented a new hybrid approach incorporating ML and DL to improve detection rates while maintaining reliability. The developed technique is compared to a wide range of ML and DL methods to identify a more effective algorithm to implement in the pipeline. Finally, the best network intrusion model is selected using a battery of benchmarked performance analysis criteria. The proposed technique is highly effective on two different datasets (KDDCUP'99 and CIC-MalMem-2022), with an accuracy of 99.99% and 100%, respectively, with no overfitting or Type-1 and Type-2 problems.

Sivamohan Krishnaveni *et al.* [129] developed a system that uses feature selection and classification with ensemble techniques effectively and efficiently detect intrusions. Combining the three best feature selection methods (gain-ratio, chi-squared, and information gain) into a single model, this proposal provides a qualifying result and four leading classifiers (SVM), Logistic Regression (LR), Naive Bayes (NB), and DT) via weighted majority voting. Honeypots, Kyoto, and NSL-KDD are used in all trials. With an accuracy of 98.29%, False Alarm Rate (FAR) of 0.012%, Detection Rate (DR) of 97%, and Area Under Curve (AUC) of 0.9921

%, the results show that the suggested intrusion detection based on the Honeypot dataset is better and more efficient than other approaches.

Sumathi Sokkalingam *et al.* [130] proposed an IDS paradigm combining ML and traditional methods. Using the publicly available benchmark NSL-KDD dataset, the performance of the proposed IDS model is improved by applying a 10-fold cross-validation technique to accomplish feature selection, lowering data dimensionality. The parameters of an SVM are optimized with a combination of the Harris Hawks optimization (HHO) and the particle swarm optimization (PSO) techniques. The proposed SVM with hybrid optimization HHO-PSO ML IDS model exhibits superior DDoS detection to the selected performance metrics.

Zouhair Chiba *et al.* [131] proposed an intelligent method to automatically design an efficient and effective anomaly network based on deep neural networks by utilizing a cutting-edge hybrid optimization framework called ISAGASAA. The Improved Self-Adaptive Genetic Algorithm (ISAGA) is a novel self-adaptive heuristic search algorithm that combines the Simulated Annealing Algorithm with Self-Adaptive Genetic Algorithm (SAA). The experiment is performed, and the testing results show that the proposed work can detect intrusions with high detection accuracy and low false alarm rate and is superior to existing work.

The state-action-reward-state-action (SARSA) Markov decision process for honeypot design is developed by Abbasgholi Pashaei *et al.* [132]. Two agents are used, one to classify and the other to describe the environment. The environmental agent tries to give the classifying agent as little money as possible. Hence, the proposed method increased the level of interaction to find honeypots faster by keeping track of aggressive behavior. The proposed method is compared to two types of malicious ICS attacks, such as MITM and DDoS, and the results show that the proposed model is more accurate and has a higher F-measure of 98% which is far better than traditional non-linear IDS models.

An ML-based hybrid IDS is proposed by Ammar Aldallal *et al.* [133] in which SVM and Genetic Algorithm (GA) are used. The SVM is tuned using GA, and the experiment is performed on the CICIDS dataset. Further, the results are compared with the NSL-KDD dataset, and it is evident from the results that the proposed work performed well with an accuracy of 99%.

Gopal Singh Khushwaha *et al.* [134] presented a Self-adaptive evolutionary extreme learning machine (SaE-ELM) to detect the DDoS attack. The presented work consists of two adaptive operations: adapting the crossover operator and classifying the number of hidden layer neurons. The experiment is performed on four datasets: NSL-KDD, ISCX IDS 2012, UNSWNB15, and CICIDS 2017.

The results showed that the proposed work gives an accuracy of 86.90%, 98.90%, 89.17%, and 99.9%, respectively, for the ab datasets mentioned above.

Zina Chkirbene *et al.* [135] suggested a revolutionary weighted classes categorization system to protect the network from malicious nodes using supervised ML algorithms. To improve the accuracy of seldom detectable attacks, authors integrate a supervised ML algorithm with prior network node information and a specially constructed best-effort iterative approach. The experiment is performed on the UNSWNB15 dataset, and it is shown from the results that the proposed work maximizes the number of reliably detectable classes and improves overall detection accuracy.

Ishu Gupta *et al.* [136] put forth a cutting-edge methodology called ML and probabilistic analysis-based model (MLPAM) that enables many participants to share their data for various objectives safely. The four ML models comprise SVM, RF, K-Nearest Neighbor (KNN), and NB, which specify the communication protocol and access rules for the several untrusted parties that process the data for the owners. The suggested model offers a robust system for detection and prevention, hence reducing the risk related to the leakage. The experimental findings show the proposed model ensures high accuracy of 97%.

An ML-based IDS for heterogeneous client networks in mobile clouds with data fusion is presented by Saurabh Dey *et al.* [137], which can protect mobile clouds from DDoS and Man in the Middle (MITM) assaults. Moreover, adopting a cloud-of-cloud approach can aid in securing VM on cloud premises. The suggested approach comprises two steps: multi-layer traffic filtering and decision-based VM. The suggested technique can be conceptualized as a cognitive system that performs traffic screening based on gathered location and Operating System (OS) data and selects the appropriate VM. The experimental findings show the proposed work's effectiveness compared to existing methodologies.

Xueluan Gong *et al.* [138] conducted a comprehensive study of previous methods, assault strategies, and countermeasures for model extraction in the cloud. The state-of-the-art assault techniques are divided into two groups, according to whether the attack's goal is to steal the model's parameters, hyperparameters, architecture, or functionality. Further, defense strategies are divided into output disturbance and query observation. A comprehensive overview of the various techniques is provided, and a thorough comparison of attack and defense strategies is also made concerning other methods.

The analysis of the ML techniques used to optimize IDS in cloud computing is discussed in Table 2.8.

Table 2.8: Analysis of IDS using ML techniques for cloud security

Year	**Title**	**Dataset Used**	**ML Model**	Limitations
2023 [128]	A dependable hybrid ML model for network intrusion detection	KDDCUP'99 and CIC-MalMem-2022	XGBoost and SMOTE	The work is limited to old datasets. The latest datasets can be used for better results.
2022 [129]	Network intrusion detection based on ensemble classification and feature selection method for cloud computing	Honeypots, Kyoto, and NSL-KDD datasets	SVM, LR, NB, and DT	Advanced feature optimization techniques can be used to improve performance.
2022 [130]	An intelligent intrusion detection system for distributed denial of service attacks: A support vector machine with hybrid optimization algorithm based approach	NSL-KDD dataset	SVM with hybrid Harris Hawks optimization	HHO algorithm has low exploring ability because Hawks must wait many minutes to hours for prey.
2022 [131]	Automatic Building of a Powerful IDS for The Cloud Based on Deep Neural Network by Using a Novel Combination of Simulated Annealing Algorithm and Improved Self-Adaptive Genetic Algorithm	Kyoto version 2015 and CICIDS-001 datasets	Self Adaptive Genetic Algorithm	The proposed method cannot separate backdoor and analysis assaults. The False Positive Rate (FPR) of 3.6% is high and should be lowered.

2022 [132]	Early Intrusion Detection System using honeypot for industrial control networks	Cyber-kit Datasets	SARSA	Different reinforcement learning (RL) methods can be used to reduce the learning time.
2021 [133]	Effective Intrusion Detection System to Secure Data in Cloud Using ML	CICIDS	SVM tuned with GA	The computation cost to implement the proposed work is quite high, which needs to be considered in future work.
2021 [134]	Optimized extreme learning machine for detecting DDoS attacks in cloud computing	NSL-KDD, ISCX IDS 2012, UNSWNB15, and CICIDS 2017	SaE-ELM	Advanced DL algorithms can be applied for the detection of DDoS attacks.
2020 [135]	Machine Learning Based Cloud Computing Anomalies Detection	UNSWNB15	Weighted Model	The performance achieved with the presented work can be improved using other ML methods.
2020 [136]	MLPAM: A Machine Learning and Probabilistic Analysis-Based Model for Preserving Security and Privacy in Cloud Environment	Glass, Iris, Wine, and Balance Scale	SVM, RF, KNN, and NB	The unique keys for the data items can be used to address the issue of shared data key.

2019 [137]	A machine learning-based intrusion detection scheme for data fusion in mobile clouds involving heterogeneous client networks	Own generated dataset	K-means clustering	During filtration, basic type of traffic from FTP protocol is considered. Better performance can be achieved using traffic from HTTP and SMTP protocols.
2020 [138]	Model Extraction Attacks and Defenses on Cloud-Based Machine Learning Models	11 online available datasets	SVM, DNN, LR	Cloud based DNNs can be applied for better results.

2.3.2 Deep Learning Techniques to Optimize IDS for Cloud security

C. Kavitha *et al.* [139] developed a Deep Learning Model (DLM) and filter-based ensemble feature selection (FEFS) for use in detecting intrusions in the cloud. At first, we filtered our intrusion statistics from KDDCup-99 and NSL-KDD databases. The information is used to verify the accuracy of the proposed approach. To facilitate the training procedure in the DLM, the necessary features are chosen based on the FEFS. The selected features are then sent to the classifier. A recurrent neural network (RNN) and Tasmanian devil optimization (TDO) are combined to create the DLM. The TDO aids in the selection of the best weighting parameter for the RNN. The proposed method is implemented in MATLAB and evaluated using a variety of performance metrics (sensitivity, F measure, precision, sensitivity, recall, and accuracy).

Bishwajeet Kumar Pandey *et al.* [140] presented the Exponential Shuffled Shepherded Optimization (ExpSSOA) Algorithm, a DL method for intrusion detection. The suggested ExpSSOA melds together EWMA (exponential weighted moving average) and SSOA (shuffled shepherded optimization algorithm) (SSOA). MQTT-IOT-IDS2020 and Apache Web Server Samples are used to test the proposed ExpSSOA-based Deep Maxout network for intrusion detection. Experimental results on the Apache webserver dataset show that the recommended ExpSSOA-Deep max-out network provides a better outcome, with an accuracy of 0.883%.

To identify potentially malicious actions in a cloud environment, Loheswaran Karuppusamy *et al.* [141] proposed a Chronological Salp Swarm Algorithm-based Deep Belief Network. The fitness function reveals the best strategy for intrusion detection by rewarding a minor error value. In this case, the suggested algorithm fine-tunes the weights to produce an efficient and optimal answer to the intrusion problem. As a result of improved exploitation and search space exploration, the suggested Chronological Salp Swarm Algorithm-based Deep Belief Network achieved superior performance. Using the KDD cup dataset and the BoT-IoT dataset, the proposed method is compared to existing state-of-the-art techniques. It is found that the proposed work performed well with an accuracy of 96% on the KDD cup dataset and 97% on the BoT-IoT dataset, respectively.

Andrea Sharron *et al.* [142] used a Bi-DLDA (Bi-directional LSTM followed by a dense layer, a dropout layer, and a layer with attention mechanism) in conjunction with a sparse autoencoder and stacked contractive autoencoder (S-SCAE)

to detect intrusions in a cloud environment. In addition, a cloud IDS collected the NSL-KDD dataset's data traffic, using the S-SCAE + Bi-DLDA algorithm to evaluate whether or not a received packet is malicious. The experiment is performed and the detection performance of the proposed system is evaluated using several metrics, including precision, recall rate, and accuracy. The results of the simulations show that the suggested model has an accuracy of over 98%, a recall rate of over 99%, and a precision of 99%.

Abdulaziz Fatani *et al.* [143] created innovative techniques for feature extraction and selection that take advantage of the benefits offered by swarm intelligence (SI) algorithms to improve the capabilities of the IDS. A mechanism for extracting features is devised based on the traditional CNN. Four widely recognized public datasets, namely CIC2017, NSL-KDD, BoT-IoT, and KDD99, are used to evaluate the effectiveness of the proposed work. The experiment is performed, and the findings demonstrate that the created strategy performs exceptionally well when measured against various evaluation metrics.

To identify DDoS attacks, Devrim Akgun *et al.* [144] suggested an IDS including preprocessing steps and a DL model. Different DNN, CNN, and Long Short Term Memory (LSTM)-based models are tested for this purpose, and their detection and real-time performance are assessed. The CIC-DDoS2019 dataset is utilized to test the proposed model. According to the test findings, the CNN-based inception-like model produced the best results among the offered models, with an accuracy rates of 99.9% for binary and 99.30% for multiclass classification. The findings from the suggested IDS system and preprocessing techniques showed superior performance compared to recent studies.

V. Gowdhaman *et al.* [145] suggested a DNN-based IDS. The best characteristics from the dataset are selected using a cross-correlation technique, which is then employed as the building blocks of a DNN to look for intrusions. The experimental findings showed that the proposed DNN efficiently recognizes attacks and outperforms traditional ML models, including SVM, DT, and RF, with 96% accuracy.

M. Mayuranathan *et al.* [146] uses a hybrid DL method to create an ideal security solution for detecting intrusions in cloud computing (EOS-IDS). The improved heap optimization (IHO) method is utilized during preprocessing to guarantee high-quality data by omitting irrelevant information. Finally, a chaotic red deer optimization (CRDO) method is provided for reducing the high dimensionality of data by rigorous feature selection. Next, a deep Kronecker neural network (DKNN) is demonstrated for detecting and classifying cloud attacks and

intrusions. The efficacy of the proposed EOS-IDS method is demonstrated by an evaluation and comparison to other current IDS strategies on two benchmark datasets comprising DARPA IDS and CSE-CIC-IDS2018. It is evident from the results that the proposed work performed well with an accuracy of 97.118%.

Omar A. Alzubi *et al.* [147] proposed a model for effective seeker optimization in the context of a machine learning-enabled IDS (ESOML-IDS) applicable to both the Fog and edge computing environment. A novel ESO-based feature selection (FS) strategy is developed for selecting an appropriate subset of features to detect the occurrence of intrusions. Further, the authors combined complete learning particle swarm optimization (CLPSO) that included a Denoising Autoencoder (DAE). The experimental outputs proved that the ESOML-IDS model produced better results than the existing state-of-the-art techniques.

The analysis of the DL techniques used to optimize IDS in cloud computing is discussed in Table 2.9.

2.4 Gaps Identification for game-theoretic model for security in cloud environment

The work done by different researchers related to security in cloud computing is discussed in this Chapter. It is concluded that security is the main challenge which is a hurdle in the success of cloud computing. Various models are discussed to provide security, and some researchers discussed the benefits of IDS in providing security in cloud computing. The gaps identified from the literature survey are as follows:

- Li *et al.* [94] proposed and implemented a differential game to improve the detection rate of the IDS in the cloud environment. The focus is increasing the detection rate and decreasing false positives and negatives. The researcher mentioned parameters like response time and overhead that can be considered in future work.

- A security model for virtual services in Sensor-Cloud is proposed by Liu *et al.* [92]. The IDS is used, and the attacker's strategies and resource consumption are checked at each stage. Security issues like jamming communication between sensor and cloud, using different sensors as zombies to flood the cloud servers, and the researchers did not consider malicious sensors that are the bottleneck in sensor-cloud secure communications.

Table 2.9: Analysis of IDS using DL techniques for cloud security

Year	**Title**	**Dataset Used**	**ML Model**	Limitations
2023 [139]	Filter-Based Ensemble Feature Selection and Deep Learning Model for Intrusion Detection in Cloud Computing	KDDCup-99 and NSL-KDD	RNN and TDO	Model validation on other datasets including CICIDS, and UNSW can be done in future.
2023 [140]	ExpSSOA-Deep maxout: Exponential Shuffled shepherd optimization based Deep maxout network for intrusion detection using big data in cloud computing framework	MQTT-IOT-IDS2020 dataset, and the Apache Web Server dataset	Deep Maxout Network model	Basic DL model is used to experiment.
2022 [141]	Chronological salp swarm algorithm based DBN for intrusion detection in the cloud using fuzzy entropy	KDD cup dataset and BoT-IoT dataset	Chronological SSA-based DBN	The proposed solution has the limitation of not being able to identify previously unseen attacks.
2022 [142]	An Intelligent Intrusion Detection System Using Hybrid Deep Learning Approaches in Cloud Environment	NSL-KDD dataset	LSTM	The work is limited to NSL-KDD dataset only. More datasets can be used for better predictions.

2022 [143]	Advanced Feature Extraction and Selection Approach Using Deep Learning and Aquila Optimizer for IoT Intrusion Detection System	CIC2017, NSL-KDD, BoT-IoT, and KDD99	CNN with Aquila optimizer (AQU)	The AQU has slow convergence, which affects the performance of the proposed work.
2022 [144]	A new DDoS attacks intrusion detection model based on deep learning for cybersecurity	CIC-DDoS2019	CNN, DNN, and LSTM	AI techniques along with game theory can be applied for IDS.
2022 [145]	An intrusion detection system for wireless sensor networks using deep neural network	NSL-KDD	DNN	Detection accuracy decreases when the number of network attacks increases.
2022 [146]	An efficient optimal security system for intrusion detection in a cloud computing environment using hybrid deep learning technique	DARPA IDS and CSE-CICIDS2018	CRDO with DKNN	Parameter optimization can be done for efficient intrusion detections.
2022 [147]	Optimized Machine Learning-Based Intrusion Detection System for Fog and Edge Computing Environment	UNSWNB15	ESOML-IDS, DAE	Other datasets comprising NSLKDD, honeypot, and CICIDS can also be considered for intrusion detection in fog and edge computing.

- Cheng [95] considered a differential game between the attacker and the IDS in Wireless Sensor Networks. The optimal strategy for IDS is achieved by Nash Equilibrium in their work. Still, there is no consideration for false positives, false negatives, detection rate, and missing rate of the IDS. Significant work can be done on these parameters.

- Fan *et al.* [109] analyzed the attack-defense mechanism of resource services in cloud applications. The defense strategies build to tackle the attack are only effective in certain attack behaviors and can fail in multi-stage attacks. This work has a scope to secure the virtual machines and hypervisor.

- Fadlullah *et al.* [124] optimized the QoS and security in next-generation heterogeneous networks. With decoupling the control and data plane, the attack surface for Software Defined Networks (SDN) is augmented compared to traditional networks. SDN in the cloud environment still faces many threats like unauthorized access, data leakage, malicious applications, and DoS, which must be solved to make the cloud environment more secure.

- The defense of cyber infrastructure in the cloud environment is discussed by Rao *et al.* [125]. The solution is provided with the boolean and component attack-defense game. The defense model considers only a starting point of the game-theoretic analysis in the uniform attacks and thus cannot work for multi-stage attacks. A dynamic game should be proposed to handle multi-stage attacks.

- The security defense using honeypot-enabled networks in the Internet of Things is spotted by La *et al.* [126]. The honeypots are deployed in the network to deceive the attacker, and both the one-shot and repeated games are considered. A specific threshold value is set, after which the honeypots are deployed. If an attacker uses mixed strategies by changing the frequency of the attacks, this model fails to deploy security. Hence, it has been analyzed that further work can be done in this area.

- Wang *et al.* [127] applied a dynamic game model over botnet DDoS attacks on the servers. The firewall configuration is done dynamically with the help of a dynamic game between the attacker and the defender system. It fails to detect the unknown DDoS attacks not presented in KDB (Knowledge Database). The anomaly module of the IDS should be considered and optimized to detect unknown attacks.

Different researchers have proposed various ML and DL approaches to optimize IDS. Various metaheuristic approaches have been discussed to optimize the parameters of ML and DL models. It is found that among all the ML and DL algorithms, RF, GBM, and DNN performed best in the optimization of IDS. Further, it is found that none of the authors worked on the Bayesian optimization technique. Therefore, we have proposed a Bayesian optimization to optimize IDS in the present study.

2.5 Objectives

Based on the literature review and research gaps, the following objectives are delineated:

1. To study the existing game theoretic models and different types of security requirements in cloud environment.
2. To devise a game-theoretic model to enhance security in cloud environment.
3. To analyze the devised model in order to derive an optimal defense strategy.
4. To test and validate the derived strategy in simulated environment.

Different security models based on game theory are proposed to fulfill the abovementioned objectives. ML and DL techniques are also used to enhance the working of IDS further. The research methodology adopted to formulate these models is discussed in the next Chapter.

Chapter 3

Research Methodology

The Chapter discusses the methodology to address the research objectives of the Thesis. The methodology is classified into three categories, i.e., requirement specifications, game-theoretic approach, and ML model, as shown in Figure 3.1. The requirement specifications include the exhaustive study of security devices like IDS concerning hardware, software configuration, and their calibration for separating malicious and non-malicious data. The game theory approach includes proposing a game theory model, delineating strategies for both players and reaching the NE stage. The ML model includes training and testing the game model on a real-time dataset. The dataset used to test and validate the proposed model is discussed in this Chapter.

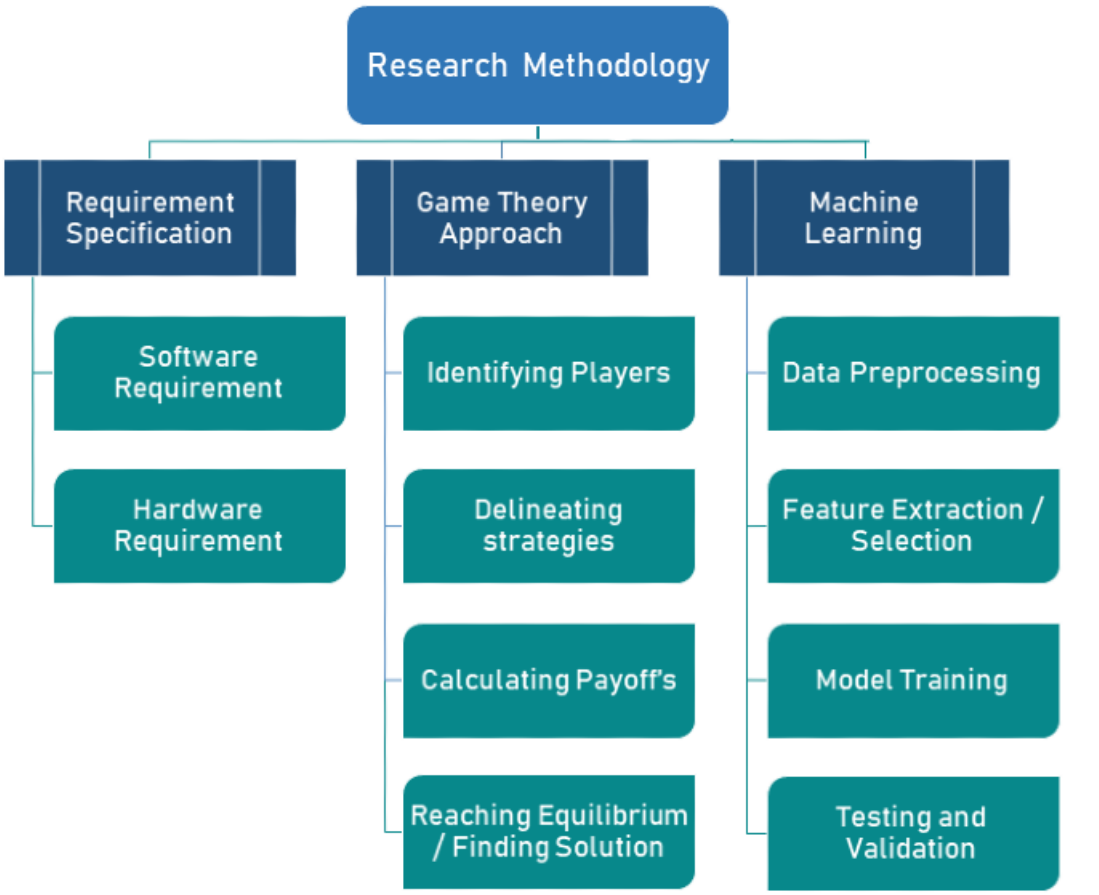


Figure 3.1: Flow of Research Methodology to attain the proposed objectives

3.1 Requirement Specifications

To address the objectives stated for the Thesis, the foremost requirement is to understand the required specifications for hardware and software. For the same, the specifications are described.

3.1.1 Software Requirements

The following tools are used to address the research objectives.

***Anaconda (Spyder)*:** *Spyder* is a free and open-source scientific environment written in Python and designed for scientists, engineers, and data analysts [148]. It combines the advanced editing, analysis, debugging, and profiling capabilities of a comprehensive development tool with the data exploration, in-depth inspection, and beautiful visualization capabilities of a scientific package in an unmatched way. In this research, game theory models have been implemented in Python. The libraries are installed and used to experiment, including *numpy, matplotlib, time, pandas, keras* and *schedule*. Further, the *KNNimpute* method is also implemented in Python, which is required to impute the missing values. The ML models are implemented in *R* language for which RStudio is required, which is discussed below.

RStudio: *R* is a programming language and environment for statistical computing and data visualization. *R* offers a vast array of statistical and graphical techniques and is highly extendable [149]. One of *R's* strengths is the ease with which publication-quality plots, complete with mathematical symbols and formulas, can be generated. The source code for *R* is free software under the terms of the GNU General Public License from the Free Software Foundation. The *RStudio* Integrated Development Environment (IDE) is a set of integrated tools designed to enhance *R* and Python productivity. It consists of a console, a syntax-highlighting editor that supports direct code execution, and an assortment of robust tools for plotting, viewing history, debugging, and managing your workspace. In the present research, ML models are implemented in RStudio. All the required packages are installed using install.packages() command. These installed packages are used for model training and testing, including *h2o, Dplyr, caret, Hmeasure, ParBayesianOptimization,* and *tidyverse*. The normalization of the dataset is also done in *R* using the *scale function*. Finally, the performance of the presented work is evaluated using the *EvaluationParameter* function, which is a part of the *HMeasure* library. The following Section discusses the hardware requirements to implement the above-discussed software requirements.

3.1.2 Hardware Requirements

System's Configuration: The minimum system configuration required for the Thesis is described.

- Processor: Intel® Core™ i5 processor.
- Memory: 4 GB of RAM, 8 GB recommended.
- Hard Disk: 20 GB of hard disk space required, 10 GB additional hard disk space required for installation.
- Display: 1024 X 768 or higher-resolution display with 24 bits colors.

3.2 Strategic Modelling with Game Theory

The process of modeling the strategic interaction between two or more players in a situation with predetermined rules and outcomes is game theory. A solution to a game describes the optimal decisions of the players, who may have similar, opposing, or mixed interests, and the possible outcomes of these decisions. Game participants are assumed to be rational and seek to maximize their payoffs [150]. Mathematically a strategic game (G) can be given as:

$$G = (P, (S_j), (U_j)) \tag{3.1}$$

Here, S_j is the strategy of Player P and U_j is the payoff of that P^{th} player [151]. Figure 3.2 shows the essential elements of game theory.

Different stages in game theory to attain the proposed objectives are discussed as follows.

Identifying Players: A player is a participant in a game and a decision-maker whose objective is to choose actions that result in his most desired outcomes or outcomes determined by chance. A game requires at least two players to qualify it. Additionally, the players must be able to interact with each other. The players are rational, with complete and transitive preference orderings. In this work, the players identified are the attacker and the defender. Both players have opposite goals and will try to exploit each other. The attacker's goal is to perform malicious activities and exploit cloud resources. The goal of the defender and this research is to analyze the attacker's behavior and predict his strategies.

Delineating Strategies: Strategies are the actions that players take in a game

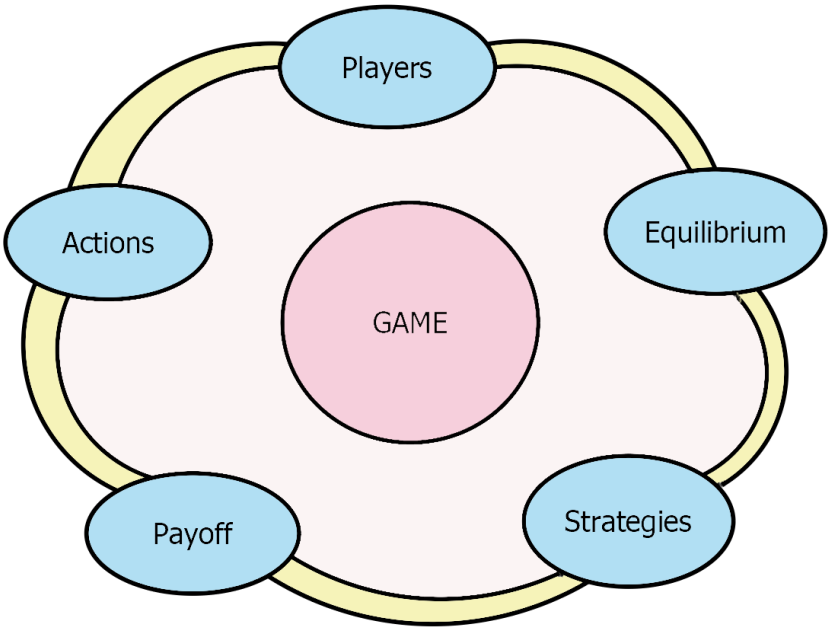


Figure 3.2: Basic Elements of Game Theory

based on the potential circumstances that may arise. Frequently, a player's strategy is based on their self-interest and what other players are anticipated to do. Players commonly know the game's rules, available strategies, and possible outcomes. Depending on the scenario, the attacker's strategies are to perform a simple or regular attack like brute force or a more advanced or sophisticated attack. The defender's strategy is to detect the attack by starting the accurate defense module.

Calculating Payoffs: The payoff of a game is the incremental gain/benefit or loss/cost that accrues to a player as a result of executing its strategy in light of the opponent's strategy. To showcase the payoffs of both players, a payoff matrix is used. It is a table in which the strategies of one player are listed in rows, and those of the other player are listed in columns, and the cells display payoffs to each player, with the row player's payoff listed first. By calculating the payoffs, the attacker's strategy can be analyzed. Also, it aids in determining the existence of a dominant strategy and equilibrium.

Reaching Equilibrium/ Finding Solution: To end the game is to achieve equilibrium, the point at which all players have decided and an outcome is determined. NE refers to the optimal outcome of a game in which there is no incentive to deviate from the initial strategy. The dominance property, min-max method, saddle point theorem, and graphical methods can be used to find the NE.

After analyzing the attacker's behavior, the defender can start the most accurate module to defend the attack. ML and DL enhance the accuracy of different IDS modules, which is discussed in the next Section.

3.3 Enhancing IDS Accuracy with Machine Learning and Deep Learning

ML allows a machine to learn from data, identify patterns, and make decisions based on that learning without being explicitly programmed. There are several types of ML, including supervised, unsupervised, semi-supervised, and reinforcement learning.

Supervised Models: In supervised learning, the machine is trained on labeled data, meaning that the desired output is already known. The target variables are used to determine which variables can increase the model's efficiency. It is possible for supervised learning to occur when the training data contains both input and output values. Each data set containing inputs and the expected output is called a supervisory signal. When the inputs are fed into the model, the training is based on the deviation of the processed result from the documented result [152].

Unsupervised Models: Unsupervised feature selection refers to a feature selection method that does not require the output label class and is utilized for unlabeled data. Unsupervised learning involves identifying data patterns. Then, additional information is utilized to fit patterns or clusters. This is also an iterative process that increases precision based on the correlation to expected patterns or clusters. This method produces no reference output dataset [153].

Semi-supervised Learning: A combination of labeled and unlabeled data is used to train the algorithm. This combination will typically consist of a minimal amount of labeled data and a vast amount of unlabeled data. The basic procedure first involves similar clustering data using an unsupervised learning algorithm, then labeling the remaining unlabeled data with the existing labeled data. Typical use cases of this type of algorithm share a common characteristic: acquiring unlabeled data is relatively inexpensive, whereas labeling said data is extremely costly, [154].

Reinforcement Learning: This class of models includes algorithms that use estimated errors as rewards or penalties. If the error is significant, the penalty is severe, and the reward is modest. The penalty is low, and the reward is high if the error is minor. The most relevant characteristics of reinforcement learning are trial-error search and delayed reward. For a model to learn which action is optimal, reinforcement feedback is required, known as the reinforcement signal. Q-learning is an example of a model that belongs to reinforcement learning.[155].

In the present research, supervised learning models comprising Random Forest (RF), Gradient Boosting Machine (GBM), XGboost, and Deep Neural Network

(DNN) are used to experiment as the dataset present is labeled, and the desired output is already known. These models are implemented in R studio. ML involves four steps, including data preprocessing, feature selection, model training, testing, and validation to perform specific tasks discussed in forthcoming Sections.

3.3.1 Data Preprocessing

Data preprocessing is preparing unstructured data for an ML model. Typically, data from the real world contains noise and missing values and may be unusable, preventing its direct use in ML models. Data preprocessing is required to clean the data and make it suitable for an ML model, improving its accuracy and efficiency [156]. The process of preprocessing data includes the following:

- Getting the Dataset
- Importing Libraries
- Importing the Dataset
- Handling Missing data
- Encoding Categorical data
- Normalization and Transformation of data

The current study uses the *scale* function of *R* language to normalize the dataset. The 10% missing values are removed using *na.omit* function of RStudio followed by the imputation of remaining missing values using *KNNimpute* function of the Python language. The dataset is preprocessed and passed on, featuring selection methods to select the relevant features required for training and testing. The feature selection methods are discussed below.

3.3.2 Feature Selection

Feature selection refers to selecting a subset of features (also known as predictors or variables) from the data to use in the machine learning model. This is important because not all features in the data are equally important or relevant for solving a particular problem. By removing irrelevant or redundant features, feature selection can improve the model's performance, reduce overfitting, and make the model easier to interpret. Different methods used for feature selection models are represented in Figure 3.3 and elaborated in detail as follows.

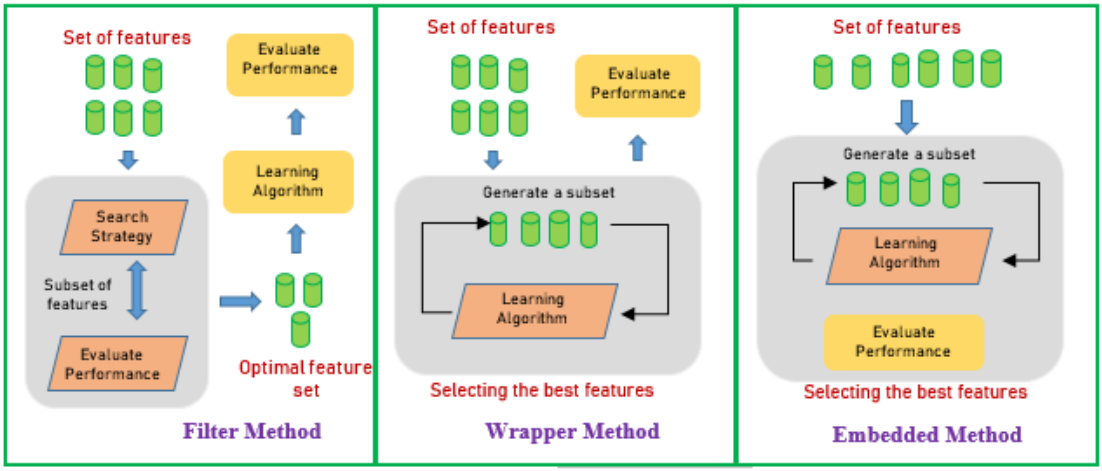


Figure 3.3: Feature Selection Methods in Machine Learning [157]

Filter Method: In this method, features are eliminated based on their correlation with the output. The correlation is used to determine if the features are positively or negatively correlated with the output labels, and based on the results, the features are eliminated [158]. For example, Information Gain, the Chi-Square Test, and Fisher's Score. The goal is to improve model performance, reduce overfitting, and enhance interpretability.

Wrapper Method: Wrapper method is a type of feature selection technique in ML that involves evaluating the performance of different subsets of features by training an ML model and evaluating its performance. The feature selection process is repeated until an optimal set of features is found. The feature selection process is guided by the ML model's performance. The features that improve the model's performance are kept, while those that do not are discarded. The process is repeated multiple times until an optimal set of features is found. Forward Selection, Reverse Elimination, Recursive Feature Elimination etc. are different wrapper methods. [159] .

Embedded Model: This method combines the Filter and Wrapper to produce the optimal subset. This method handles the iterative ML process while keeping computation cost to a minimum [160]. For example, Lasso and Ridge Regression.

In this Thesis, Chapter 7, *FSelector* is used to select the features which work by calculating the information gain value. The rank is assigned to each feature based on the computed information gain, and the top-ranked features are selected for model training. Further, In Chapter 5, Recursive Feature Elimination (RFE) is used to select the features which work by selecting the features recursively. A small subset is selected, then passed to model training for performance evaluation. The process is repeated for all the subsets, and the best subset with the best accuracy is selected for model training. The model training is discussed below.

3.3.3 Model Training

Model training is the process of using a set of labeled data to estimate the parameters of an ML model. Model training aims to find the values of the model parameters that minimize the difference between the model's predictions and the actual target values in the training data [161]. The model training process starts with defining a loss function that measures the difference between the model's predictions and the actual target values. The loss function is used to guide the optimization process, and the goal is to find the values of the model parameters that minimize the loss. An existing optimization algorithm is used to find the values of the model parameters that minimize the loss function. Standard optimization algorithms include random, grid, and Bayesian optimization. The optimization algorithm updates the values of the model parameters in each iteration based on the loss function. Once the optimization process has converged and the values of the model parameters have been estimated, the model is said to be trained. The trained model can then make predictions on new, unseen data. In Chapter 5, Bayesian optimization tunes the parameters. Chapter 6 uses Gradient Boosting Machine (GBM), Random Forest (RF), and Deep Neural Networks (DNN) models to train the dataset. In Chapter 7, stacking of RF, GBM, and XGBoost as first-level training and DNN at second-level training is performed to train the dataset.

3.3.4 Testing and Validation

Testing and validation are two important ML processes to assess a trained model's performance. They help determine how well the model generalizes to new, unseen data and whether it is overfitting or underfitting the data [162]. Testing refers to evaluating a trained model's performance on a separate, independent dataset not used during training. The purpose of testing is to estimate the model's performance on new, unseen data and to provide an unbiased estimate of its generalization performance. The test set should be large enough to provide a reliable estimate of the model's performance but not so large that it significantly reduces the size of the training set.

Conversely, validation refers to evaluating a model's performance during training to tune its hyperparameters. Hyperparameters are parameters set before the model is trained and control the behavior of the learning algorithm. Common hyperparameters include the learning rate, the number of hidden units in a neural network, or the depth of a decision tree. Validation is performed by dividing

the training set into two parts: a validation set and a training set. The model is trained on the training set, and its performance is evaluated on the validation set. This process is repeated several times with different hyperparameters, and the hyperparameters that result in the best performance on the validation set are selected. It is important to remember that validation should not be used to evaluate the performance of the final model, as it is performed on a portion of the training set and may result in overfitting the validation set. Testing should be used to evaluate the performance of the final model. The performance parameters used to test the performance in the present research are accuracy, Detection Rate (DR), and False Positive Rate (FPR).

Different datasets are used to test and validate the proposed models: BOGTA, GTA-IDS and NCGTM. These datasets are NSL-KDD, ISOT-CID, UNSW-NB15, CICIDS and Bot-IoT. The description of each dataset is given in the next Section.

3.4 Dataset Used

Five different real-time datasets are considered and discussed below:
NSL-KDD: The NSL-KDD data set is an updated version of the KDD'99 data set. This is a useful benchmark data set for researchers to compare various intrusion detection methods. The configuration comprises one training set and two testing sets with 125,973 and 22,544 records. Each NSL-KDD record has 41 features (e.g., protocol type, Logged in, Duration, etc.). These features are represented as numeric, nominal, and binary, defined as continuous or discrete, and labeled as normal or attack. The NSL-KDD dataset is divided into four classes. The types of attacks considered to test the model are *DoS, probe, R2L,* and *U2L* [163].
ISOT-CID: The ISOT Cloud Intrusion Dataset (ISOT-CID) is the first public dataset of its sort gathered from a production cloud environment. The dataset contains over 2.5 terabytes of data spanning normal operations and a diverse array of attack vectors. It was gathered in two phases and spanned several months for the VM instances and numerous days and time slots for the Hypervisors. The benign/normal data comes from online apps and administrative tasks such as monitoring the health of virtual machines, rebooting, updating, generating files, SSHing into the machines, and signing in to a remote server. Over 160 authentic visitors created web traffic, comprising over 60 human users and genuine traffic generated by 100 robots completing duties such as account registration, reading/posting and commenting on blogs, and exploring other websites. ISOT-CID

is intended to represent a real cloud dataset; it is raw and is not processed, modified, or changed in any way. This is critical for industry and research in designing and assessing realistic cloud computing intrusion models. The dataset is divided into two classes, i.e., benign and malicious. There are 10,42,558 samples, of which 4,13,484 are malicious and 6,29,074 are benign. The malicious samples are divided into regular and sophisticated samples, with 2,85,453 and 1,28,022 samples in each type [164][165].

UNSW-NB15: UNSW-NB15 dataset was created by the IXIA PerfectStorm tool in the Cyber Range Lab of UNSW Canberra for generating a hybrid of real modern normal activities and synthetic recent attack behaviors. The *tcpdump* tool was utilized to capture 100 GB of the raw traffic files. UNSW-NB15 consists of nine types of attacks: Fuzzers, Analysis, Backdoors, DoS, Exploits, generics, Reconnaissance, Shellcode, and Worms. Therefore two classes were there, one for attacks and the other for normal samples. UNSW-NB15 has 175341 and 82332 training and testing instances with 49 features [166].

CICIDS: The CICIDS2017 dataset includes common attacks that are both safe and up-to-date, making it an accurate representation of data from the actual world. In addition, it incorporates the findings of the network traffic analysis performed by CICFlowMeter, complete with labeled flows organized according to the time stamp, source, and destination IP addresses, source and destination ports, protocols, and attack types. CICIDS is divided into 93500 and 28481 training and testing samples, respectively, and contains 49 features. [167]

Bot-IoT: Bot-IoT [23] dataset includes DDoS, DoS, OS and Service Scan, Keylogging, and Data ex-filtration attacks, with the DDoS and DoS attacks further organized, based on the protocol used. To ease the handling of the dataset, 5% of the original dataset is extracted via select MySQL queries. The extracted 5% comprises four files of approximately 1.07 GB and about 3 million records [168].

3.5 Summary

This Chapter discusses different tools and techniques, hardware and software specifications. The game theory components used for modeling different strategies are also presented. The ML and DL steps are elaborated to optimize the IDS, including data preprocessing, feature selection, model training, testing, and validation. A detailed description of different datasets used to test and validate the proposed models is also given. Four models are proposed based on this research methodology, discussed in the forthcoming Chapters.

Chapter 4

GTM-CSec: Game Theoretic Model for External Cloud Attacks

In this Chapter, a non-cooperative game-theoretic model GTM-CSec is developed to tackle external security attacks on the Cloud. A game is modeled between the attacker and the defender competing against each other to gain maximum payoffs. The attacker's primary goal is to exploit the cloud services with different security attacks, and the defender's goal is to detect these security attacks. The signature-based and anomaly-based modules of IDS, and honeypot are used as a defender system. Different strategies for both players are delineated, and a mixed strategy Nash Equilibrium (NE) is attained to conclude the game. The payoffs of the attacker and defender are analyzed and compared. The results show that with a proper detection module, the payoffs of the defender system come out to be more than the attacker.

4.1 Overview of GTM-CSec

Cloud computing and its multifarious services are a boon for digital businesses' setup. On the other side, security is the primary concern for the Cloud. The Cloud's top security threats are data breaches, hijacking accounts, Distributed Denial of Services (DDoS), malware injection, inside threats, insecure Application Programming Interfaces (APIs), and data loss. Also, the increasing number of IoT devices is a significant threat to the cloud [66]. With various factors like power consumption, product life, and overall efficiency, the vendors compromise with the security aspects of these devices. Some devices like CCTV cameras are situated in public places that can be easily accessed and are vulnerable. Several devices are available with default or weak authentication methods that can be easily guessed by a Brute Force attack [169]. As an illustration, the Mirai DDoS in 2016 occurred in which intelligent devices like CCTVs, and DVRs were used as bots [170]. This attack affected enterprises dreadfully and is one of the most significant DDoS attacks, with a traffic rate of 600 Gbps to 1.5 Tbps. The bot army is increasing daily because of IoT device usage [171]. This imposes a massive

threat on cloud servers or other resources. The scenario is shown in Figure 4.1.

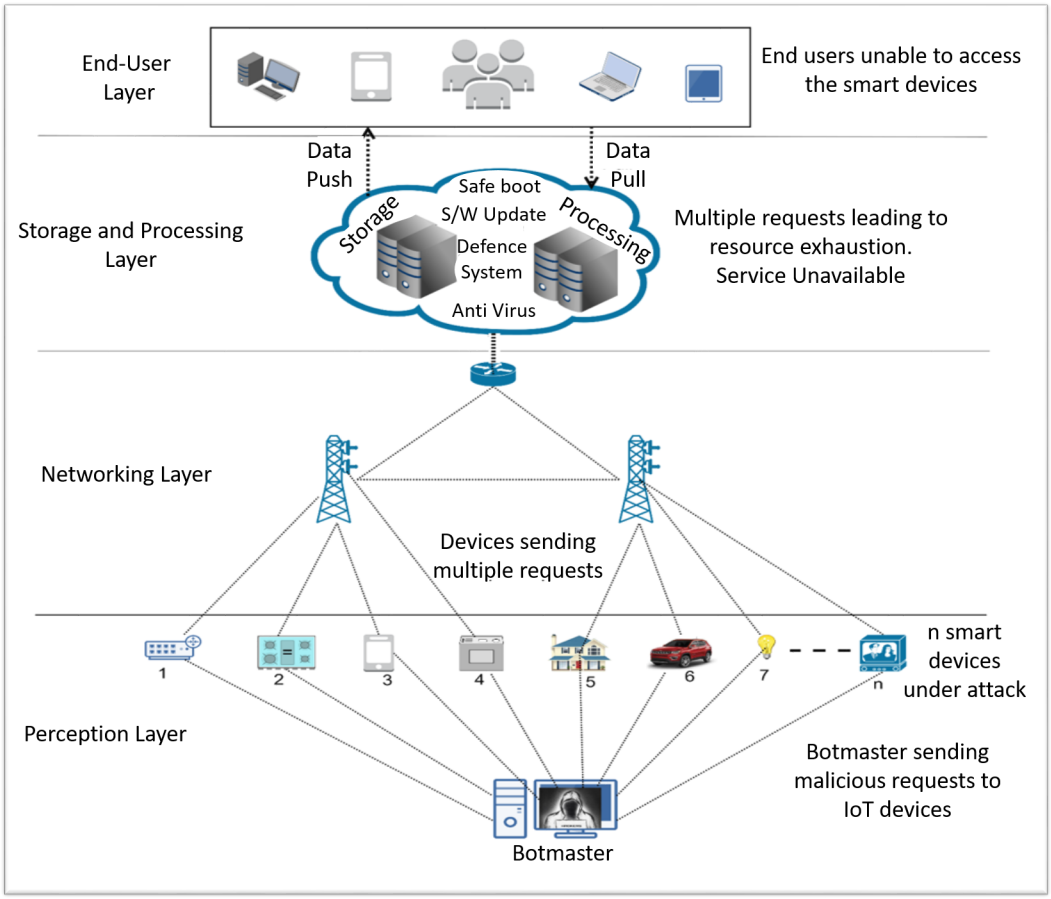


Figure 4.1: Cloud under Attack of IoT Botnets

The perception layer consists of IoT devices where the attacker uses a botmaster machine to find the vulnerable devices with the help of a Brute Force attack [172]. The vulnerable devices are then hacked and used as zombies. The zombies send malicious requests to the Cloud to make the server unavailable or corrupt the data. The communication between the IoT devices and the Cloud occurs through the networking layer. The attacker tries to compromise the cloud operations at the storage and processing layer. If the attacker successfully invades the cloud security, the services provided by the Cloud may not be assessed by the consumers at the end-user layer.

4.1.1 Motivation

The motivation is to address the growing security concerns in cloud computing and IoT, which have become increasingly vulnerable to cyber attacks due to their complex and distributed nature. The main aim is to propose a model that can

intelligently select the most suitable detection module and provide a more effective defense system against attackers. For this purpose, *GTM-CSec* model is proposed and developed to intelligently monitor the network traffic and adjust the configuration of the Intrusion Detection System (IDS) along with the honeypot for better results. The game theory model is updated with probability values for the defender's and attacker's strategies to predict the best strategy. Ultimately, the best strategy is delineated with the help of the NE.

4.2 GTM-CSec Model to tackle external cloud attacks

The *GTM-CSec* model uses a non-cooperative game model (discussed in Chapter 2 to present the relationship between the Attacker (A) and the Defender (D). Mathematically, the proposed non-cooperative game model (G) can be given as:

$$G = \{(D, A)(S_D, S_A)(U_D, U_A)\} \tag{4.1}$$

where S_D and S_A are the strategies space, U_D and U_A are the payoff functions for the defender and the attacker respectively. The framework of the *GTM-CSec* is shown in Figure 4.2. The framework is divided into four layers; Attack Layer, Security Layer, Back-end Layer, and End-user Layer. In the Attack Layer, the attacker or the botmaster sends malicious requests to the IoT devices and controls these devices from Command and Control (C&C) Server [173]. The attacker attacks according to its best strategy to attain maximum payoff. The Security Layer comes into action to defend against the attacker's attack. This layer consists of the IDS, including signature-based and anomaly-based detection modules, a honeypot-based detection module, and the *GTM-CSec* Model with the decision-maker. The IDS has two modules, Signature Detection (SD) and Anomaly Detection (AD). SD best defends the known or regular attacks as these attacks are already in the database. The SD checks with the database and gives results more efficiently and quickly. On the other hand, the AD deals best with unknown or sophisticated attacks. If there is a deviation in the behavior of the incoming traffic or packets, the AD techniques give a true positive signal indicating there is an attack. The honeypots reside on a network and contain dummy data to trap attackers. After the detection system, legitimate traffic is sent to the cloud for safe networking. The defender plays its best strategy to defend the attacker's attack. The Back-end Layer contains all the data and resources on the cloud servers that must be

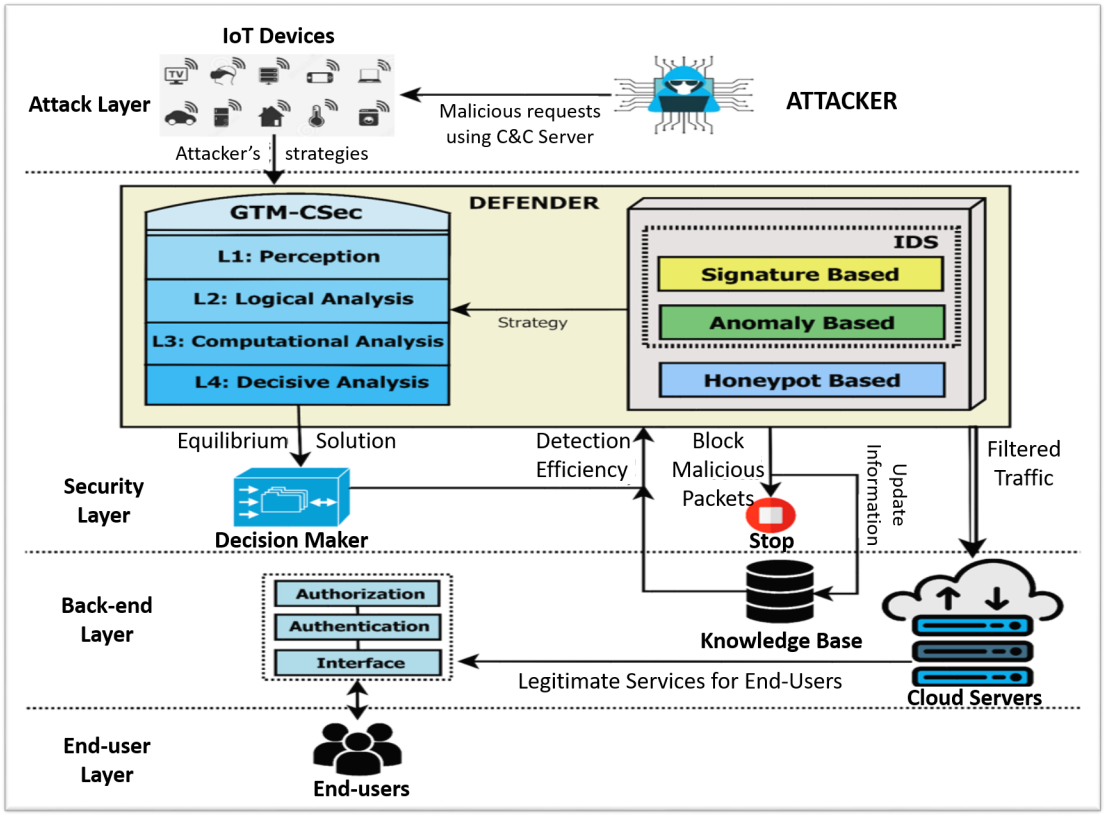


Figure 4.2: GTM-CSec framework for Security in Cloud Environment

protected from any cost. At last, there is an End-user layer where all cloud computing services are leveraged. The *GTM-CSec* Model is further extended into four layers; *L1: Perception Layer, L2: Logical Analysis, L3: Computational Analysis,* and *L4: Decisive Analysis.*

4.2.1 GTM-CSec Layers

L1: Perception Layer: In the first layer, the strategies of both the attacker and the defender are fed to the *GTM-CSec* module. The parameters are initialized and passed to the next phase according to players' strategies. Table 4.1 represents the terminology used to define the payoff functions. The main parameters considered are the energy consumption for the IDS (E_{ids}) and Honeypots (H_{hp}). The gain or benefit (B_{ds}) the defender system gets by successfully detecting the attack and the value of the assets (V) under attack for a particular time (t). The assets can be any cloud resources, including a server, hypervisor, virtual machine, or API. The attacker attacks with some resources, which leads to the exhaustion of the resources (R_{at}) during the attack leading to the negative impact on its payoffs. The attacker waits for some time which costs him (W_{at}). The gain (G_{at}) which

the attacker gets after successfully invading the cloud adds up to the payoff of the attacker. The whole model is designed considering the time parameter *(t)* for which the attacker attacks and the defender defends. The attack remains for a particular time that is used with each parameter. The defender will respond to the attack at the same time t. The detection rate of the signature module, anomaly module, and honeypot is represented by μ, α, and γ, respectively. The False Positive Rate of the IDS is given by β.

Table 4.1: Parameters used to develop GTM-CSec Model

Description	**Symbol**
Energy consumed by IDS	E_{ids}
Energy consumed by Honeypot	H_{hp}
Gain for successfully detecting	B_{ds}
Value of the assets under attack	V
Resource consumption by attacker	R_{at}
Gain for successfully attacking	G_{at}
Waiting time cost for attacker	W_{at}
Detection rate of IDS for signature	μ
Detection rate of IDS for anomaly	α
Detection rate of honeypot	γ
False Positive Rate of Defender	β

L2: Logical Analysis: It calculates the payoffs of both the players from the initialized parameters, which helps the defender system to understand the strategies of the attacker. $S_A = \{S_{A1}, S_{A2}, S_{A3}\}$ are the attacker's strategies and are given in Table 4.2.

Table 4.2: Attacker's Strategies for Attacking the Cloud

Representation	**Strategy**
S_{A1}	Attacking the cloud servers with regular attacks
S_{A2}	Wait for particular time period *(t)*
S_{A3}	Attacking the cloud with new or sophisticated attacks.

The defender's strategy set can be represented as $S_D = \{S_{D1}, S_{D2}, S_{D3}\}$ and are elaborated in Table 4.3.

Table 4.3: Defender's Strategies for Defending the Cloud

Representation	**Strategy**
S_{D1}	Monitoring with signature-based IDS
S_{D2}	Monitoring with anomaly-based IDS
S_{D3}	Monitoring with honeypot.

The strategies and relationship between the defender and attacker in the form of a tree are represented by Figure 4.3. For each strategy of the attacker, the

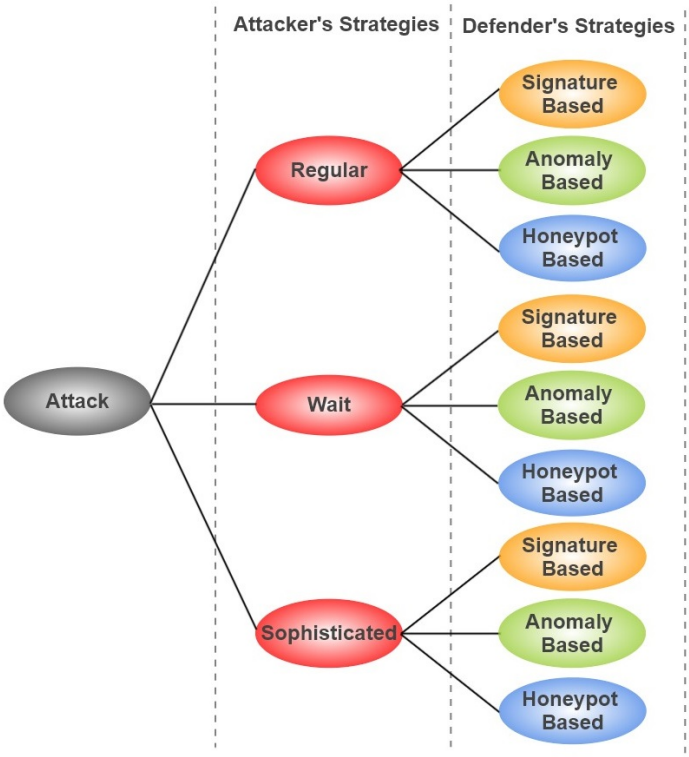


Figure 4.3: Tree representation of defender and attacker with their strategies

defender has three strategies. The *GTM-CSec* model helps the defender system to decide the best response to the attacker's strategy.

L3: Computational Analysis: The calculated payoffs are converted into matrices in Layer 3 to compute the equilibrium. The strategy space of the defender and the attacker is represented as rows and columns in M.

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} = \begin{bmatrix} (S_{D1}, S_{A1}) & (S_{D1}, S_{A2}) & (S_{D1}, S_{A3}) \\ (S_{D2}, S_{A1}) & (S_{D2}, S_{A2}) & (S_{D2}, S_{A3}) \\ (S_{D3}, S_{A1}) & (S_{D3}, S_{A2}) & (S_{D3}, S_{A3}) \end{bmatrix} \tag{4.2}$$

The m_{11} in M, describes the strategy of using signature-based detection of IDS (S_{D1}) and regular attack of the attacker (S_{A1}). The attacker attacks with regular techniques already known to the defender. The defender monitors with a signature-based method. Hence it will successfully monitor the attacks. The payoff of the IDS depends upon the energy consumed and the benefit gained for the successful monitoring and is given by Eq. 4.3:

$$U_{11}(D) = B_{ds}(t) - E_{ids}(t) \tag{4.3}$$

The attacker loses the resources he used because of the unsuccessful attack. The payoff of the attacker is given by Eq. 4.4:

$$U_{11}(A) = -R_{at} \tag{4.4}$$

Similarly, there are payoffs for the other strategies. The complete payoff functions of defender and attacker are described by Matrices M_D and M_A as shown in Eq. 4.5 and Eq. 4.6 respectively.

$$M_D = \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix} = \begin{bmatrix} B_{ds}(t) - E_{ids}(t) & -E_{ids}(t) & -E_{ids}(t) - V(t) \\ -E_{ids}(t) & -E_{ids}(t) & B_{ds}(t) - E_{ids}(t) \\ B_{ds}(t) - H_{hp}(t) & -H_{hp}(t) & -H_{hp}(t) - V(t)) \end{bmatrix} \tag{4.5}$$

$$M_A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} -R_{at}(t) & -W_{at}(t) & G_{at}(t) - R_{at}(t) \\ G_{at}(t) - R_{at}(t) & -W_{at}(t) & -R_{at}(t) \\ -R_{at}(t) & -W_{at}(t) & G_{at}(t) - R_{at}(t)) \end{bmatrix} \tag{4.6}$$

d_{11} and a_{11} are the corresponding payoff functions of strategy Matrix element m_{11}.
L4: Decisive Analysis: This layer calculates the equilibrium from the payoff matrices. If pure strategy NE exists, the control is directly passed to the decision-maker to start the accurate module. Otherwise, a mixed strategy NE is calculated if both players do not agree on a single outcome. The defender system filters the traffic, sends legitimate traffic to the cloud servers, and blocks illegitimate traffic. It also updates its knowledge base to increase detection accuracy further. The cloud provides services to its authenticated and authorized users by the interface layer. To calculate the NE, the scribing method is used [174]. The maximum benefit function is calculated with the help of Matrices M_D and M_A. If the maximum

benefit function of M_A matches the corresponding element of M_D, that is the pure strategy NE solution of this game model. The maximum profits for attacker by Eq. 4.6 are the a_{13}, a_{21} and a_{33}. Similarly, d_{11}, d_{23} and d_{31} are the maximum profits for the defender from Eq 4.5. The following two inferences are made:

Inference 1: There exists no pure strategy NE in this game model.

Proof: The subscript of the maximum profits of both Matrices M_D and M_A do not match. The maximum profit for the attacker lies in attacking the cloud with a sophisticated attack. But, sophisticated attacks consume more resources, and the attacker can't attack every time in a new way. So, the defender will choose to detect by anomaly method and thus bust the attacker. Then, the attacker will try to attack with the regular method because the defender has chosen the anomaly method. Again, the defender can choose the honeypot or the signature method and prevent the attacker from attacking. The system can not be stabilized as there exists no pure strategy NE.

Inference 2: The attacker will not get any profit while waiting for the attack, so he always chooses to attack.

Proof: From the maximum benefit subscript; a_{13}, a_{21} and a_{33}, it is clear that strategy S_{A2} has never been taken into account. To gain maximum benefit, the attacker will always choose between S_{A1} and S_{A3}, which means he will always choose to attack. From the above statement, the strategy S_{A2} can be neglected. After eliminating the strategy of attacker (S_{A2}), the strategy left for the attacker is to either choose a Regular Attack (RA) or the Sophisticated Attack (SA). A regular attack can easily be performed with fewer resources and experience. The sophisticated attack is more difficult to perform, but its success rate is more than the regular attack. The defender still has the same strategies. Thus, the updated game model is shown in Table 4.4.

Table 4.4: Payoff matrix of defender and attacker in cloud environment

S_A,S_D	**SD**	**AD**	**Honeypot**
RA	$B_{ds}(t)$ – $E_{ids}(t), -R_{at}(t)$	$-E_{ids}(t), G_{at}(t)$ – $R_{at}(t)$	$B_{ds}(t)$ – $H_{hp}(t), -R_{at}(t)$
SA	$-E_{ids}(t)$ – $V(t), G_{at}(t) - R_{at}(t)$	$B_{ds}(t)$ – $E_{ids}(t), -R_{at}(t)$	$-H_{hp}(t)$ – $V(t), G_{at}(t) - R_{at}(t)$

Suppose the attacker attacks with a regular attack, and the defender detects with signature-based IDS. Then, the payoff function of the attacker for this scenario can be given by Eq. 4.7.

$$U_{11}(A) = (1-\mu)G_{at}(t) - R_{at}(t) \tag{4.7}$$

Similarly, the payoff function can be represented for the attacker in Eq. 4.8.

$$U_{11}(D) = \mu B_{ds}(t) - (1-\mu)V(t) - E_{ids}(t) \tag{4.8}$$

The payoff matrix of the defender (M'_D) and the attacker (M'_A) for all possible scenarios are given by Eqs. 4.9 and 4.10 respectively.

$$M'_D = \begin{bmatrix} \mu B_{ds}(t) - V(t)(1-\mu) - E_{ids}(t) & -E_{ids}(t) - V(t) & \gamma B_{ds}(t) - V(t)(1-\gamma) - H_{hp}(t) \\ -E_{ids}(t) - V(t) & \alpha B_{ds}(t) - V(t)(1-\alpha) - E_{ids}(t) & -H_{hp}(t) - V(t) \end{bmatrix} \tag{4.9}$$

$$M'_A = \begin{bmatrix} (1-\mu)G_{at}(t) - R_{at}(t) & G_{at}(t) - R_{at}(t) & (1-\gamma)G_{at}(t) - R_{at}(t) \\ G_{at}(t) - R_{at}(t) & (1-\alpha)G_{at}(t) - R_{at}(t) & G_{at}(t) - R_{at}(t) \end{bmatrix} \tag{4.10}$$

4.3 GTM-CSec Model Analysis

According to *Inference 1*, no pure strategy exists; hence mixed strategies for the players are analyzed. The probabilities for each strategy used by the defender and attacker are assigned. To maximize the payoff and to calculate the probability, the partial derivative of the various payoffs of the defender is done concerning (*wrt*) probability of the defender. Similarly, the partial derivative of the payoff of the attacker is taken *wrt* probability of attacker. The saddle point of Game Theory is used and is explained with the toy example in three steps as follows:

Step 1: Find out the minimax and maximin values.

$$\begin{array}{lcc} & \text{Player B} & \text{Row min} \\ Player A & \begin{bmatrix} 9 & 7 \\ 5 & 11 \end{bmatrix} & \begin{matrix} 7 \\ 5 \end{matrix} \\ \text{Column Max:} & \begin{matrix} 9 & 11 \end{matrix} & \end{array} \tag{4.11}$$

From Eq. 4.11, 7 is maximin and 9 is minimax. Since this game's minimax and maximin values are not equal, this game has no saddle point.

Step 2: The 22 matrix finds the oddments for both row and column. To find oddments, take the difference between the highest outcome and the smallest outcome of the first row and put it on the right side of the second row (Eq. 4.12), *i.e.* the difference between 9 and 7 is 2, and it is placed at the right of the second row. Similarly, take the difference between the highest and the lowest outcome in the second row and put it at the right of the first row, *i.e.* the difference between 11

and 5 is 6, and is placed at the right of the first row. Similarly, find oddments for columns as well. The result is shown in Eq. 4.12

$$\begin{array}{cccc} & \text{Player B} & & \text{Oddments} \\ Player A & \begin{bmatrix} 9 & 7 \\ 5 & 11 \end{bmatrix} & & \begin{matrix} 6 \\ 2 \end{matrix} \\ \text{Oddments:} & \begin{matrix} 4 & 4 \end{matrix} & & \end{array} \tag{4.12}$$

Step 3: The probabilities for each row are calculated. Let x and (1 – x) be the probabilities of the selection of strategies of player A, and y and (1 – y) be the probabilities of the selection of strategies of player B as shown in Eq. 4.13.

$$\begin{array}{cccc} & \text{Player B} & & \text{Probabilities} \\ Player A & \begin{bmatrix} 9(A_{11}) & 7(A_{12}) \\ 5(A_{21}) & 11(A_{22}) \end{bmatrix} & & \begin{matrix} \text{x} \\ \text{(1-x)} \end{matrix} \\ \text{Probabilities:} & \begin{matrix} y & (1-y) \end{matrix} & & \end{array} \tag{4.13}$$

The calculation of probabilities x and $(1-x)$ are shown in Eq. 4.14 and 4.15.

$$x = \frac{A_{22} - A_{21}}{(A_{11} + A_{22}) - (A_{12} + A_{21})} = \frac{3}{4} \tag{4.14}$$

$$(1-x) = \frac{1}{4} \tag{4.15}$$

Probability y and $(1-y)$ calculation is given by Eq. 4.16 and 4.17.

$$y = \frac{A_{22} - A_{12}}{(A_{11} + A_{22}) - (A_{12} + A_{21})} = \frac{1}{2} \tag{4.16}$$

$$(1-y) = \frac{1}{2} \tag{4.17}$$

Based on the above-discussed toy example, the probabilities of the attacker's and the defender's strategies can be calculated with the help of Eqs. 4.9 and 4.10. These are discussed in the next Section.

4.3.1 Probabilities of attacker and defenders strategies

The attacker has two strategies: regular attack and sophisticated attack. The defender has three strategies: signature-based detection, anomaly-based detection, and honeypots. Seven possible cases are delineated from the given strategies and shown in Table 4.5. *Case 1* to *Case 7* are discussed as follows:

Table 4.5: Description of different cases

Cases	**Defender's Strategies**		
Case 1	Signature	Anomaly	Honeypot
Case 2	Signature	Anomaly and Honeypot	
Case 3	Anomaly	Signature and Honeypot	
Case 4	Honeypot	Signature and Anomaly	
Case 5	Signature	Anomaly	
Case 6	Signature	Honeypot	
Case 7	Anomaly	Honeypot	

Case 1: In this case, the defender uses signature-based, anomaly-based, and honeypot-based strategies separately. The probability by which the attacker attacks the cloud with regular attack and sophisticated attacks is given by f and *(1-f)*, respectively. The defender performs the signature-based, anomaly-based, and honeypot-based monitoring with the probability of $\frac{2p}{5}$, $\frac{2p}{5}$ and $\frac{p}{5}$. The probability of both the modules of IDS is taken the same and more than the honeypot. Sophisticated attacks are relatively more challenging to perform than regular attacks. Hence, the attacker would choose the regular method more often than the sophisticated attacks. The updated total benefits of both defender (U_{Dsah}) and attacker (U_{Asah}) with the probabilities are represented by the Eq. 4.18 and 4.19 respectively.

$$\begin{aligned} U_{Dsah} =&(\frac{2p}{5})(f)U_{11}(D) + (\frac{2p}{5})(f)U_{12}(D) + (\frac{p}{5})(f)U_{13}(D) \\ &+ (\frac{2p}{5})(1-f)U_{21}(D) + (\frac{2p}{5})(1-f)U_{22}(D) + (\frac{p}{5})(1-f)U_{23}(D) \end{aligned} \tag{4.18}$$

$$\begin{aligned} U_{Asah} =& (\frac{2p}{5})(f)U_{11}(A) + (\frac{2p}{5})(f)U_{12}(A) + (\frac{p}{5})(f)U_{13}(A) \\ &+ (\frac{2p}{5})(1-f)U_{21}(A) + (\frac{2p}{5})(1-f)U_{22}(A) + (\frac{p}{5})(1-f)U_{23}(A) \end{aligned} \tag{4.19}$$

The partial derivatives of defender payoffs (U_{Dsah}) and attacker's payoff (U_{Asah}) w.r.t probability p and f respectively is given by Eq. 4.20 and Eq. 4.21:

$$\begin{aligned} \frac{\partial U_{Dsah}}{\partial p} =& \frac{pf}{5}[B_{ds}(t) + V(t)][2\mu - 2\alpha + \gamma] + \frac{2p\alpha}{5}[B_{ds}(t) + V(t)] - \\ & 4pE_{ids}(t) - pV(t) - \frac{p}{5}H_{hp}(t) = 0 \end{aligned} \tag{4.20}$$

$$\frac{\partial U_{Asah}}{\partial f} = \frac{pfG_{at}(t)}{5}[2\mu - 2\alpha + \gamma] + pG_{at}(t) - pR_{at}(t) - \frac{2p}{5}\alpha G_{at}(t) = 0 \quad (4.21)$$

By solving Eq. 4.20 and 4.21, $p, f, (1-f)$ are determined as:

$$p = \infty \quad (4.22)$$

$$f = \frac{2\alpha}{5(2\alpha - 2\mu - \gamma)} + \frac{1}{[2\mu - 2\alpha + \gamma][B_{ds}(t) + V(t)]}[4E_{ids}(t) + H_{hp}(t) + 5V(t)] \quad (4.23)$$

$$(1-f) = \frac{2\mu + \gamma}{(2\mu - 2\alpha + \gamma)} + 1 - \frac{1}{[2\mu - 2\alpha + \gamma][B_{ds}(t) + V(t)]}[4E_{ids}(t) + H_{hp}(t) + 5V(t)] \quad (4.24)$$

Case 2: The signature-based IDS works individually, while anomaly-based and honeypot work in coordination. The probability of using signature-based is given by s, and the combined probability of anomaly and honeypot is given by *(1-s)*. The attacker has the same strategies, namely regular attacks and sophisticated attacks with g and *(1-g)* probability. The defender strategy is given by Eq. 4.25:

$$U_{DS} = (s)(g)U_{11}(D) + (s)(1-g)U_{21}(D) + (1-s)(g)[U_{12}(D) + U_{13}(D)] + (1-s)(1-g)[U_{22}(D) + U_{23}(D)] \quad (4.25)$$

The payoff of the attacker is given by Eq. 4.26:

$$U_{AS} = (s)(g)U_{11}(A) + (s)(1-g)U_{21}(A) + (1-s)(g)[U_{12}(A) + U_{13}(A)] + (1-s)(1-g)[U_{22}(A) + U_{23}(A)] \quad (4.26)$$

To maximize the payoff, the partial derivative is done of U_{DS} and U_{AS} *wrt* s and g respectively.

$$\frac{\partial U_{DS}}{\partial s} = sg\mu B_{ds}(t) + sg\mu V(t) + sV(t) + g\gamma B_{ds}(t) + g\gamma V(t) - sg\gamma B_{ds}(t) - sg\gamma V(t) - s\alpha B_{ds}(t) - s\alpha V(t) + sH_{hp}(t) - g\alpha B_{ds}(t) - g\alpha V(t) + sg\alpha B_{ds}(t) + sg\alpha V(t) \quad (4.27)$$

$$\frac{\partial U_{AS}}{\partial g} = 4G_{at}(t) - s\mu G_{at}(t) + s\alpha G_{at}(t) - 2sG_{at}(t) - g\gamma G_{at}(t) + g\alpha G_{at}(t) - sg\mu G_{at}(t) + sg\gamma G_{at}(t) - sg\alpha G_{at}(t) - \alpha G_{at}(t) - 2R_{at}(t) \quad (4.28)$$

Solving Eq. 4.27 and 4.28, we get *s*, *(1-s)*, *g* and *(1-g)* as follows:

$$s = \frac{\mu}{\mu + \alpha - \gamma} \tag{4.29}$$

$$(1 - s) = \frac{\gamma - \alpha}{\gamma - \mu - \alpha} \tag{4.30}$$

$$g = \frac{\mu - \gamma - 2\alpha}{\mu - \gamma - \alpha} - \frac{V(t)}{(\mu - \gamma - \alpha)(B_{ds}(t) + V(t))} - \frac{H_{hp}(t)}{(\mu - \gamma - \alpha)(B_{ds}(t) + V(t))} \tag{4.31}$$

$$(1 - g) = \frac{\alpha}{\mu - \gamma - \alpha} - [1 - \frac{V(t)}{(\mu - \gamma - \alpha)(B_{ds}(t) + V(t))} - \frac{H_{hp}(t)}{(\mu - \gamma - \alpha)(B_{ds}(t) + V(t))}] \tag{4.32}$$

For simplicity, $\frac{V(t)}{(\mu-\gamma-\alpha)(B_{ds}(t)+V(t))} - \frac{H_{hp}(t)}{(\mu-\gamma-\alpha)(B_{ds}(t)+V(t))}$ is represented by ϕ. So, Eq. 4.31, becomes:

$$g = \frac{\mu - \gamma - 2\alpha}{\mu - \gamma - \alpha} - \phi \tag{4.33}$$

and Eq. 4.32, becomes:

$$(1 - g) = \frac{\alpha}{\mu - \gamma - \alpha} - [1 - \phi] \tag{4.34}$$

Case 3: The anomaly-based IDS works individually, and the signature-based IDS and honeypot work together. *r* is the probability of anomaly-based detection and *(1-r)* is the probability of honeypot and signature-based detection. Here, *q* and *(1-q)* are the probabilities of RA and SA, respectively. The payoffs of the *Defender* and the *Attacker* are given by Eq. 4.35 and 4.36 respectively:

$$U_{DA} = (r)(q)U_{12}(D) + (r)(1-q)U_{22}(D) + (1-r)(q)[U_{11}(D) + U_{13}(D)] + (1-r)(1-q)[U_{21}(D) + U_{23}(D)] \tag{4.35}$$

$$U_{AA} = (r)(q)U_{12}(A) + (r)(1-q)U_{22}(A) + (1-r)(q)[U_{11}(A) + U_{13}(A)] + (1-r)(1-q)[U_{21}(A) + U_{23}(A)] \tag{4.36}$$

The partial derivatives of the calculated payoffs are calculated by taking partial derivatives of U_{DA} *wrt* r (Eq. 4.37) and U_{AA} *wrt* q (Eq. 4.38).

$$\frac{\partial U_{DA}}{\partial r} = r\alpha B_{ds}(t) - rV(t) + r\alpha V(t) - rq\alpha B_{ds}(t) - rq\alpha V(t) + r\mu B_{ds}(t) - 2qV(t) - q\gamma B_{ds}(t) + q\gamma V(t) - qH_{hp}(t) - rq\mu B_{ds}(t) - rq\mu V(t) - rg\gamma B_{ds}(t) - rq\gamma V(t) + rH_{hp}(t) + qE_{ids}(t) + 2qV(t) + qH_{hp}(t) \tag{4.37}$$

$$\frac{\partial U_{AA}}{\partial q} = rG_{at}(t) - r\alpha G_{at}(t) - rR_{at}(t) + rq\alpha G_{at}(t) - \mu q G_{at}(t) - q\alpha \\ G_{at}(t) + rq\mu G_{at}(t) + rq\gamma G_{at}(t) + 2G_{at}(t) - 2R_{at}(t) - 2rG_{at}(t) + 2rR_{at}(t) \tag{4.38}$$

Solving Eq. 4.37 and 4.38, we get *r*, *(1-r)*, *q* and *(1-q)* as follows:

$$r = \frac{\alpha}{\gamma + \mu + \alpha} \tag{4.39}$$

$$(1 - r) = \frac{\mu + \gamma}{\gamma + \mu + \alpha} \tag{4.40}$$

$$q = \frac{\mu + \gamma}{\gamma + \mu + \alpha} + \frac{V(t)}{(\mu + \gamma + \alpha)(B_{ds}(t) + V(t))} + \\ \frac{H_{hp}(t)}{(\mu + \gamma + \alpha)(B_{ds}(t) + V(t))} \tag{4.41}$$

$$(1 - q) = \frac{\alpha}{\gamma + \mu + \alpha} + [1 - \frac{V(t)}{(\mu + \gamma + \alpha)(B_{ds}(t) + V(t))} + \\ \frac{H_{hp}(t)}{(\mu + \gamma + \alpha)(B_{ds}(t) + V(t))}] \tag{4.42}$$

The term $\frac{V(t)}{(\mu+\gamma+\alpha)(B_{ds}(t)+V(t))} + \frac{H_{hp}(t)}{(\mu+\gamma+\alpha)(B_{ds}(t)+V(t))}$ is represented by ϵ. The Eq. 4.41 becomes:

$$q = \frac{\mu + \gamma}{\gamma + \mu + \alpha} + \epsilon \tag{4.43}$$

The Eq. 4.42 can be represented as:

$$(1 - q) = \frac{\alpha}{\gamma + \mu + \alpha} + [1 - \epsilon] \tag{4.44}$$

Case 4: The signature and anomaly-based work together with probability *(1-x)* and the honeypot works individually with probability *x*. The attacker attacks with two choices, regular attack with probability *y* and sophisticated attack with probability *(1-y)*. The defender's payoff is given by Eq. 4.45:

$$U_{DH} = (x)(y)U_{13}(D) + (x)(1 - y)U_{23}(D) + (1 - x)(y)[U_{11}(D) + \\ U_{12}(D)] + (1 - x)(1 - y)[U_{21}(D) + U_{22}(D)] \tag{4.45}$$

The payoff of the attacker is given by Eq. 4.46:

$$U_{AH} = (x)(y)U_{13}(A) + (x)(1 - y)U_{23}(A) + (1 - x)(y)[U_{11}(A) + \\ U_{12}(A)] + (1 - x)(1 - y)[U_{21}(A) + U_{22}(A)] \tag{4.46}$$

The partial derivatives of the defender are given by Eq. 4.47.

$$\begin{aligned}\frac{\partial U_{DH}}{\partial x} = xy\gamma B_{ds}(t) + xy\gamma V(t) - xH_{hp}(t) - xV(t) - xy\mu B_{ds}(t) - xy\mu \\ V(t) - x\alpha B_{ds}(t) + 2xV(t) - x\alpha V(t) + 2xE_{ids}(t) + xy\alpha B_{ds}(t) + xy\alpha V(t)\end{aligned} \tag{4.47}$$

The partial derivatives of the attacker are given by Eq. 4.48.

$$\begin{aligned}\frac{\partial U_{AH}}{\partial y} = -xy\gamma G_{at}(t) + xG_{at}(t) - xR_{at}(t) - \\ y\mu G_{at}(t) + xy\mu G_{at}(t) + 2G_{at}(t) - 2R_{at}(t) - \alpha G_{at}(t) - 2xG_{at}(t) + \\ 2xR_{at}(t) + x\alpha G_{at}(t) + y\alpha G_{at}(t) - xy\alpha G_{at}(t)\end{aligned} \tag{4.48}$$

By solving Eq. 4.47 and 4.48, $x, (1-x), y, (1-y)$ are calculated as:

$$x = \frac{\gamma}{\gamma + \alpha - \mu} \tag{4.49}$$

$$(1-x) = \frac{\mu - \alpha}{\mu - \gamma - \alpha} \tag{4.50}$$

$$\begin{aligned}y = \frac{\gamma - \mu - 2\alpha}{\gamma - \mu - \alpha} + \frac{V(t)}{(\gamma - \mu - \alpha)(B_{ds}(t) + V(t))} + \\ \frac{H_{hp}(t)}{(\gamma - \mu - \alpha)(B_{ds}(t) + V(t))} - \frac{2E_{ids}(t)}{(\gamma - \mu - \alpha)(B_{ds}(t) + V(t))}\end{aligned} \tag{4.51}$$

$$\begin{aligned}(1-y) = \frac{\alpha}{\gamma - \mu - \alpha} + [1 - \frac{V(t)}{(\gamma - \mu - \alpha)(B_{ds}(t) + V(t))} \\ + \frac{H_{hp}(t)}{(\gamma - \mu - \alpha)(B_{ds}(t) + V(t))} - \frac{2E_{ids}(t)}{(\gamma - \mu - \alpha)(B_{ds}(t) + V(t))}]\end{aligned} \tag{4.52}$$

For simplicity, $\frac{V(t)}{(\gamma-\mu-\alpha)(B_{ds}(t)+V(t))} + \frac{H_{hp}(t)}{(\gamma-\mu-\alpha)(B_{ds}(t)+V(t))} - \frac{2E_{ids}(t)}{(\gamma-\mu-\alpha)(B_{ds}(t)+V(t))}$ is represented by σ. The Eq. 4.51 becomes:

$$y = \frac{\gamma - \mu - 2\alpha}{\gamma - \mu - \alpha} + \sigma \tag{4.53}$$

$$(1-y) = \frac{\alpha}{\gamma - \mu - \alpha} + [1 - \sigma] \tag{4.54}$$

Case 5: Either the defender system chooses to detect the attack with the signature-based method or the anomaly-based method. The probability with signature-based detection is given by a, and the probability for anomaly-based is given by *(1-a)*. The attacker attacks with a regular attack by the probability of b and with a sophisticated attack with the probability of *(1-b)*. The defender

payoffs are given by Eq. 4.55.

$$\begin{aligned} U_{Dsa} = (a)(b)U_{11}(D) + (a)(1-b)U_{21}(D) + \\ (1-a)(b)U_{12}(D) + (1-a)(1-b)U_{22}(D) \end{aligned} \tag{4.55}$$

The payoff of the attacker is given by Eq. 4.56:

$$\begin{aligned} U_{Asa} = (a)(b)U_{11}(A) + (a)(1-b)U_{21}(A) + \\ (1-a)(b)U_{12}(A) + (1-a)(1-b)U_{22}(A) \end{aligned} \tag{4.56}$$

The partial derivatives of the defender are given by Eq. 4.57.

$$\begin{aligned} \frac{\partial U_{Dsa}}{\partial a} = ab\mu B_{ds}(t) + ab\alpha B_{ds}(t) + ab\mu V(t) + \\ ab\alpha V(t) - a\alpha B_{ds}(t) - a\alpha V(t) - b\alpha B_{ds}(t) - b\alpha V(t) \end{aligned} \tag{4.57}$$

The partial derivatives of the attacker are given by Eq. 4.58.

$$\begin{aligned} \frac{\partial U_{Asa}}{\partial b} = -ab\mu G_{at}(t) + 1 - \alpha G_{at}(t) - R_{at}(t) + a\alpha \\ G_{at}(t) + b\alpha G_{at}(t) - ab\alpha G_{at}(t) \end{aligned} \tag{4.58}$$

By solving Eq. 4.57 $a, (1-a)$ are calculated as:

$$a = \frac{\alpha}{\mu + \alpha} \tag{4.59}$$

$$(1-a) = \frac{\mu}{\mu + \alpha} \tag{4.60}$$

By solving Eq. 4.58 $b, (1-b)$ are calculated as:

$$b = \frac{\mu}{\mu + \alpha} \tag{4.61}$$

$$(1-b) = \frac{\alpha}{\mu + \alpha} \tag{4.62}$$

Case 6: The defender system chooses to detect the attack with the signature-based or the honeypot-based method. The probability with signature-based detection is given by *j*, and the probability for honeypot based is given by *(1-j)*. The attacker attacks with the regular attack with the probability of *k* and with a sophisticated attack with the probability of *(1-k)*. The defender payoffs are given

by Eq. 4.63.

$$U_{Dsh} = (j)(k)U_{11}(D) + (j)(1-k)U_{21}(D) + (1-a)(b)U_{13}(D) + (1-a)(1-b)U_{23}(D) \tag{4.63}$$

The payoff of the attacker is given by Eq. 4.64:

$$U_{Ash} = (a)(b)U_{11}(A) + (a)(1-b)U_{21}(A) + (1-a)(b)U_{13}(A) + (1-a)(1-b)U_{23}(A) \tag{4.64}$$

The partial derivatives of the defender are given by Eq. 4.65.

$$\frac{\partial U_{Dsh}}{\partial j} = jk\mu B_{ds}(t) + jk\mu V(t) + k\gamma B_{ds}(t) + k\gamma V(t) - jk\gamma B_{ds}(t) - jk\gamma V(t) - H_{hp}(t) - V(t) + jH_{hp}(t) - jE_{ids}(t) \tag{4.65}$$

The partial derivatives of the attacker are given by Eq. 4.66.

$$\frac{\partial U_{Ash}}{\partial k} = -jk\mu G_{at}(t) - k\gamma G_{at}(t) - jkG_{at}(t) + jk\alpha G_{at}(t) + jkR_{at}(t) + G_{at}(t) - R_{at}(t) + jkG_{at}(t) - jk_R at(t) \tag{4.66}$$

By solving Eq. 4.65 $j, (1-j)$ are calculated as:

$$j = \frac{\gamma}{\gamma - \mu} \tag{4.67}$$

$$(1-j) = \frac{\mu}{\mu - \gamma} \tag{4.68}$$

By solving Eq. 4.66 $k, (1-k)$ are calculated as:

$$k = \frac{E_{ids}(t) - H_{hp}(t)}{\mu - \gamma} \tag{4.69}$$

$$(1-k) = \frac{\mu - \gamma - E_{ids}(t) + H_{hp}(t)}{\mu - \gamma} \tag{4.70}$$

Case 7: The defender system chooses to detect the attack with the anomaly-based or the honeypot-based method. The probability of anomaly-based detection is given by u, and the probability for honeypot based is given by *(1-u)*. The attacker attacks with a regular attack by the probability of v and with a sophisticated attack with the probability of *(1-v)*. The defender payoffs are given by Eq. 4.71.

$$U_{Dah} = (u)(v)U_{12}(D) + (u)(1-v)U_{22}(D) + (1-u)(v)U_{13}(D) + (1-u)(1-v)U_{23}(D) \tag{4.71}$$

The payoff of the attacker is given by Eq. 4.72:

$$\begin{aligned} U_{Aah} = (a)(b)U_{12}(A) + (a)(1-b)U_{22}(A) + \\ (1-a)(b)U_{13}(A) + (1-a)(1-b)U_{23}(A) \end{aligned} \tag{4.72}$$

The partial derivatives of the defender are given by Eq. 4.73.

$$\begin{aligned} \frac{\partial U_{Dah}}{\partial u} = v\gamma B_{ds}(t) + v\gamma V(t) - uv\gamma B_{ds}(t) - uv\alpha V(t) + u\alpha B_{ds}(t) + \\ u\alpha V(t) - uv\alpha B_{ds}(t) - uv\alpha V(t) - H_{hp}(t) - V(t) \end{aligned} \tag{4.73}$$

The partial derivatives of attacker assuming $H_{hp}(t)$ is equal to $E_{ds}(t)$ is given by Eq. 4.74.

$$\frac{\partial U_{Aah}}{\partial v} = u\alpha G_{at}(t) + uv\alpha G_{at}(t) - u\gamma G_{at}(t) + uv\gamma G_{at}(t) + G_{at}(t) - R_{at}(t) \tag{4.74}$$

By solving Eq. 4.73 $u, (1-u)$ are calculated as:

$$u = \frac{\gamma}{\gamma + \alpha} \tag{4.75}$$

$$(1-u) = \frac{\alpha}{\alpha + \gamma} \tag{4.76}$$

By solving Eq. 4.74 $v, (1-v)$ are calculated as:

$$v = \frac{\alpha}{\alpha + \gamma} \tag{4.77}$$

$$(1-v) = \frac{\gamma}{\gamma + \alpha} \tag{4.78}$$

There can be another case in which the defender is at rest, and one of the signature-based, anomaly-based, or honeypots is working. But, as the attacker always chooses to attack, resting the defender could be a considerable risk. Hence, the NE is computed to find the stable solution from the above-stated cases. The *GTM-CSec* game model helps the detection system to secure the cloud efficiently. The steps of execution are shown in Algorithm 4.1.

The payoffs of the *defender* and *attacker* are represented in general as U_D and U_A. It corresponds to the payoffs of the respective cases. The values of parameters (*Step 1* of Algorithm 4.1) are taken as input to the game model (see Table 4.1). From the input values, the game model calculates the payoffs of the defender $U_{ij}(D)$ and the attacker $U_{ij}(A)$ and constructs their payoff matrices M'_D and M'_A respectively. If the pure strategy NE exists, the game model passes the solution

to the decision-maker, and the particular detecting module tackles the attack. Else, update the payoffs of the defender and the attacker with probability k and l respectively *(See Step 8)*. Calculate the payoffs and derive the mixed strategy NE. In *Step 10*, pass the results to the decision-maker and execute the delineated strategy. To test the effectiveness of the GTM-CSec Algorithm, its performance is tested against the attacker. The results obtained are discussed in the following Section.

Algorithm 4.1 GTM-CSec Algorithm to tackle external security attacks in cloud

1: Input: $B_{ds}(t), V(t), E_{ids}(t), H_{hp}(t), R_{at}(t), G_{at}(t)$ and $W_{at}(t)$.
2: Calculate: $U_{ij}(D)$ and $U_{ij}(A)$.
3: Construct: M'_D and M'_A.
4: **if** Pure Strategy NE **then** Goto Step 11.
5: **end if**
6: **Else** Input: μ, α, γ
7: **for** *Case 1 to 7* **do**
8: Update $U_{ij}(D)$ and $U_{ij}(A)$ with k and l.
9: Find mixed strategy NE.
10: **end for**
11: Start the accurate detection module.

4.4 Results and Discussions

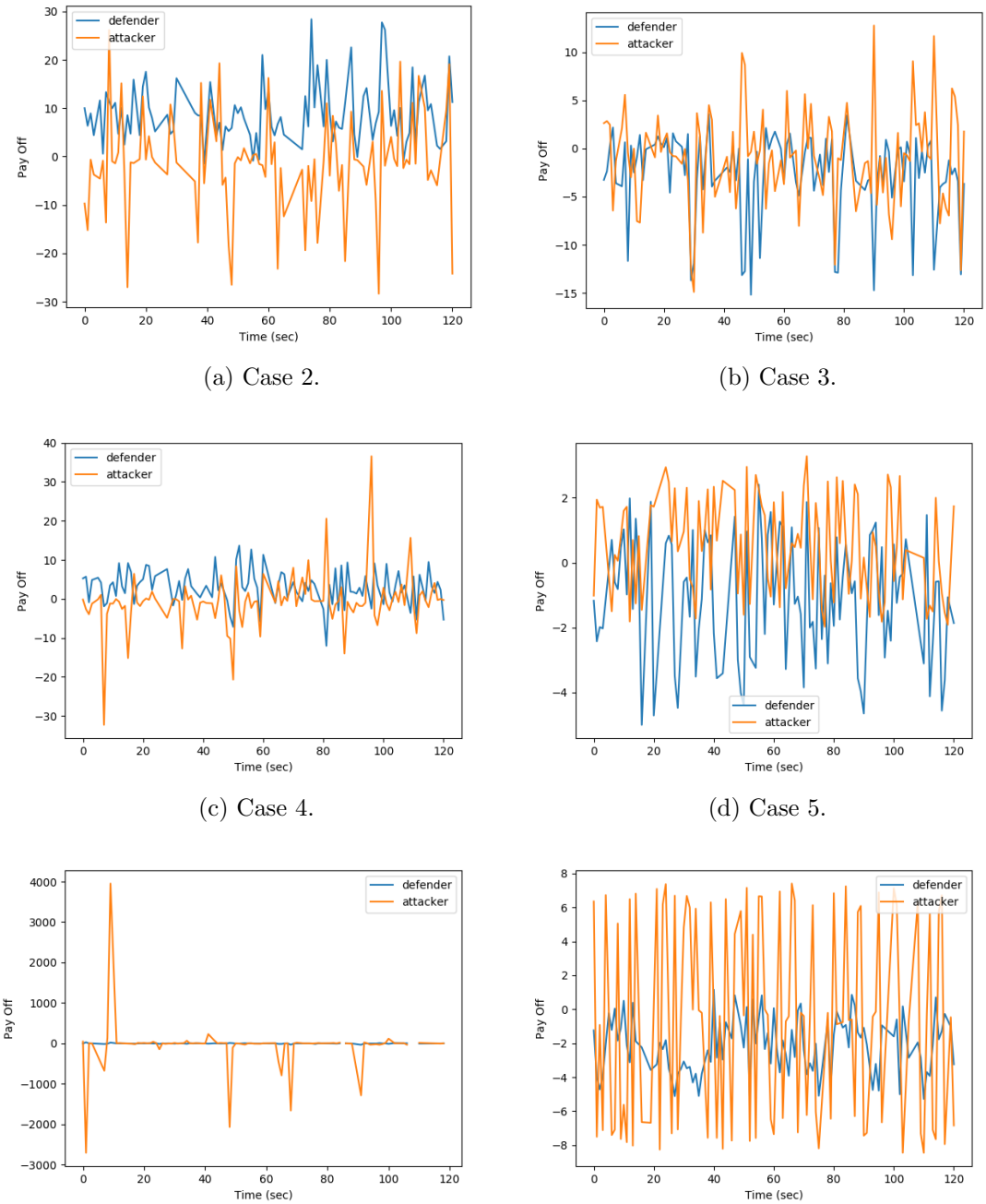
The *Cases 1-7* discussed in Section 4.3.1, after applying game theory, the following Nash Equilibrium Solution (NES) for the defender S_D and the attacker S_A are computed. These are shown in Eqs. 4.79 and 4.80.

$$\begin{aligned} S_D = [p,(1-p),s,(1-s),r,(1-r),x,(1-x),a,(1-a),j,(1-j),u,\\ (1-u)] = [\frac{\mu}{\mu+\alpha-\gamma},\frac{\gamma-\alpha}{\gamma-\mu-\alpha},\frac{\alpha}{\gamma+\mu+\alpha},\frac{\mu+\gamma}{\gamma+\mu+\alpha},\frac{\gamma}{\gamma+\alpha-\mu},\\ \frac{\mu-\alpha}{\mu-\gamma-\alpha},\frac{\alpha}{\mu+\alpha},\frac{\mu}{\mu+\alpha},\frac{\gamma}{\gamma-\mu},\frac{\mu}{\mu-\gamma},\frac{\gamma}{\gamma+\alpha},\frac{\alpha}{\alpha+\gamma}] \end{aligned} \tag{4.79}$$

$$\begin{aligned} S_A = [f,(1-f),g,(1-g),q,(1-q),y,(1-y),b,(1-b),k,(1-k),v,(1-v)]\\ = [\frac{\mu-\gamma-2\alpha}{\mu-\gamma-\alpha}-\phi,\frac{\alpha}{\mu-\gamma-\alpha}-(1-\phi),\frac{\mu+\gamma}{\gamma+\mu+\alpha}+\epsilon,\frac{\alpha}{\gamma+\mu+\alpha}+(1-\epsilon),\\ \frac{\gamma-\mu-2\alpha}{\gamma-\mu-\alpha}+\sigma,\frac{\alpha}{\gamma-\mu-\alpha}+(1-\sigma),\frac{\mu}{\mu+\alpha},\frac{\alpha}{\mu+\alpha},\frac{E_{ids}(t)-H_{hp}(t)}{\mu-\gamma},\\ \frac{\mu-\gamma-E_{ids}(t)+H_{hp}(t)}{\mu-\gamma},\frac{\alpha}{\alpha+\gamma},\frac{\gamma}{\gamma+\alpha}] \end{aligned} \tag{4.80}$$

If the values of both probabilities for each case come out to be equal, then the attacker and the defender have the maximum benefits. The simulation environment is built in Python script to verify and validate the *GTM-CSec* model. The algorithm is set up along with payoff functions and probabilities. The detection rate of defender system (μ, α, γ) is 0 to 0.99 (i.e., 0% - 100%). The asset's value should be equal to or less than the gain. Otherwise, there is no point in defending the asset because the defender will gain less than the asset's value. The simulation is run for several hours, and one instance of 120 seconds is tested for *Case 2,3,4,5,6,7* as shown in Figure 4.4 as the *Case 1* got rejected in NES. Both the players compete with each other along with their strategies. The values generated are stored and analyzed. In *Case 1*, the probability of the detection comes out to be infinite, which means if the defender waits long enough, it can detect the attack. Since the attack must be detected soon, this case is unfavorable for the defender. So, this case is rejected. The probability values of *Case 2* are proportional to the signature-based module detection rate, and the combination of anomaly-based and honeypot depends directly on their respective detection rates. It is represented by Figure 4.4a. Similarly, in *Case 3* and *Case 4* represented in Figure 4.4b and 4.4c, the probabilities of the functioning of detection modules are directly proportional to their detection rates. The more the detection rate of the module for the particular instance more will be chance of its working to defend against the attack. In *Case 5*, if the value of μ is fixed, and the value of α increases, the $a = b = \frac{\alpha}{\mu+\alpha}$ increases as shown in Figure 4.4d. It indicates the detection accuracy of the anomaly detection method increases. Then, the rational attacker will try to reduce the probability by adopting a sophisticated method, i.e., $b = \frac{\mu}{\mu+\alpha}$. From the defenders' perspective, the probability of signature-based detection will increase after the attacker's strategy. If the value of μ is fixed, the greater the α, the greater will be $a = b = \frac{\alpha}{\mu+\alpha}$. So it increases the probability of a signature-based method, so the attacker will adopt a regular method to attack. In *Case 6*, the payoff of the defender remains steady. The defender is not gaining, and the sudden peak in the attacker's payoff shows the successful attack. This case is not favorable for the defender. In *Case 7* which is shown in Figure 4.4e, if the value of α is fixed, and the value of γ increases, the $u = v = \frac{\gamma}{\gamma+\alpha}$ increases. It indicates the detection accuracy of the anomaly detection method increases. Then, the rational attacker will try to reduce the probability by adopting a sophisticated method, i.e., $b = \frac{\mu}{\mu+\alpha}$. From the defenders' perspective, the probability of signature-based detection will increase after the attacker's strategy. If the value of μ is fixed, the greater the α, the greater will be $u = v = \frac{\alpha}{\gamma+\alpha}$. So it

increases the probability of a honeypot-based method, so the attacker will adopt a regular method to attack. The players' payoffs for *Case 7* are shown in Figure 4.4f. The *GTM-CSec* chooses the best strategies out of the delineated strategies. The trends in Figure 4.4 show that Case 2 is very effective against the attacker for this particular instance.



(a) Case 2.

(b) Case 3.

(c) Case 4.

(d) Case 5.

(e) Case 6.

(f) Case 7.

Figure 4.4: Comparison of the attacker and defender's payoffs.

The payoffs of the defender in Case 2 are more than the attacker, which means

the defender is winning the game. So, for this particular instance, the defender system will automatically turn on the signature module and the combination of anomaly and honeypot. Similarly, the *GTM-CSec* model will calculate the payoffs and start the best-suited module for the next instance.

4.5 Summary

This Chapter proposes a non-cooperative game model (*GTM-CSec*) in which attack-defense scenarios are illustrated. The attacker has two strategies *i.e.* regular attack and sophisticated attack. Similarly, the defender has three strategies *i.e.* signature-based, anomaly-based, and honeypot-based detection methods. The following conclusions are made:

- The game model finds the pure or mixed strategy NE according to the scenario.
- The simulation results show the *GTM-CSec* model is very effective in defending against the attacker.
- Instead of using all three detection modules, the *GTM-CSec* algorithm chooses the best module for attack detection. This leads to an increase in the ability of the defender to tackle the attack most efficiently and intelligently.
- The *GTM-CSec* model is an independent module. Due to that, it can be easily implementable in real systems by coupling the game module with existing IDS and honeypots.

Chapter 5

BOGTA Model: Bayesian Optimized Game-Theoretic Approach

In this Chapter, Game Theory and Bayesian Optimized Deep Learning approaches are used to eliminate the shortcomings of the GTM-CSec model. The graphical method of game theory optimizes the decision-making process of the proposed model BOGTA by eliminating the least significant Strategy of the IDS. The hyperparameter tuning with Bayesian optimization further enhances the accuracy and detection rate of the Intrusion Detection System (IDS). Also, it lowers the FPR of the IDS. It is validated and tested on three datasets, namely UNSW-NB15, CICIDS, and Bot-IoT. The results show that BOGTA is more accurate and takes less computation time than GTM-CSec. The comparative analysis of IDS and BOGTA is done. After analysis, it is observed that the overall performance of IDS is better than the existing models.

5.1 Overview of BOGTA

The vast application areas of cloud computing pose an enormous security risk to cloud servers and other resources. Different hardware devices like IDS, Firewalls, and software like anti-virus are used to tackle these attacks. The GTM-CSec model (discussed in Chapter 4) defends against external attacks on the cloud environment, but its few limitations should be addressed to make it more efficient. In this work, the proposed model *BOGTA* covers every possibility of working each defender system module (signature, anomaly, or honeypot) in a single cycle. The graphical method of game theory reduces the time complexity of the model. A Bayesian Optimised Deep Neural Network technique is used to optimize the model to a greater extent, and hyperparameter tuning is done. Three different datasets for different detection modules are considered to train and test the IDS. The detection rate, accuracy, payoffs, and false positive rate are calculated and compared with the existing models. To test the signature-based module, the UNSW-NB15 dataset is used [175]. The anomaly-based detection is tested on a CICIDS dataset [176]. The Bot-IoT dataset tests the Honeypot module [177]. The BOGTA model

is discussed in the next Section. The *BOGTA* framework is shown in Figure 5.1. The User Layer consists of the end-users, devices, and attackers. The Network

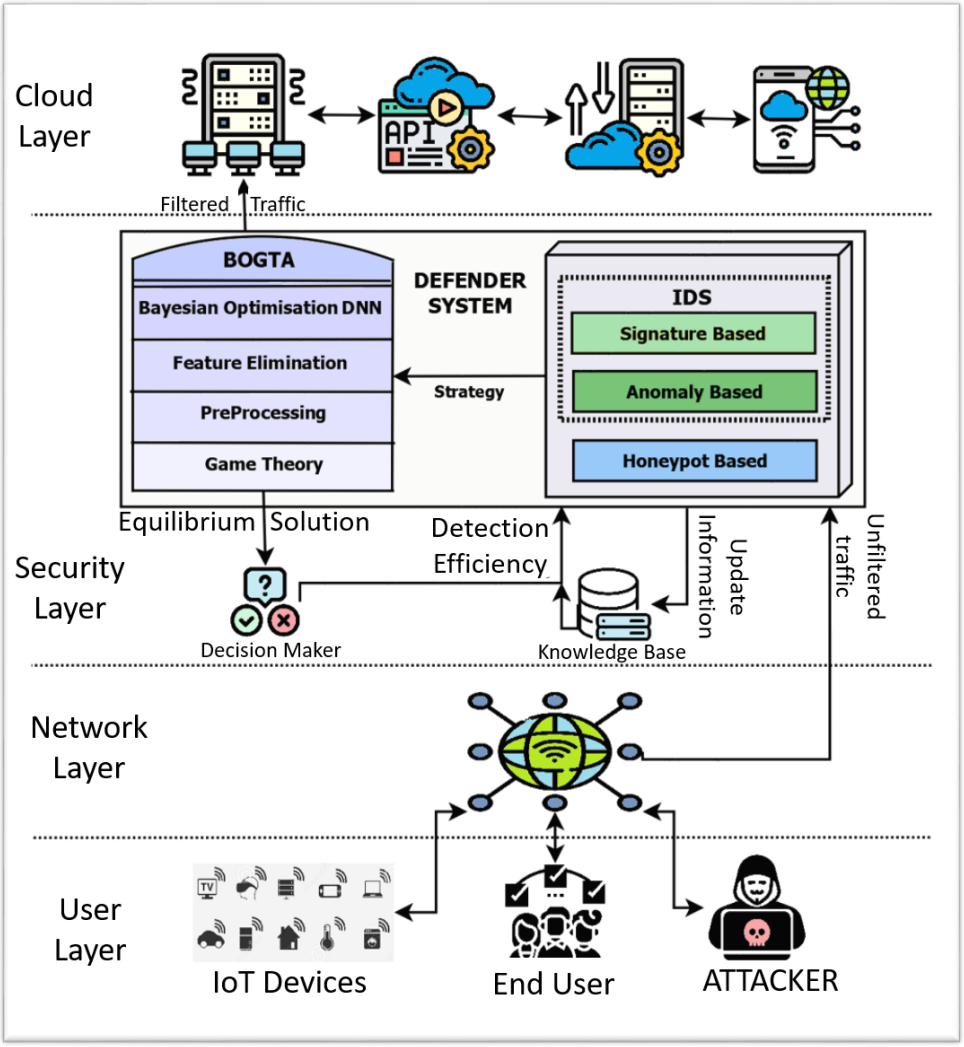


Figure 5.1: BOGTA deployment framework

Layer connects the User Layer with the Security Layer, where the defender system monitors the unfiltered traffic. The IDS has two modules signature-based and anomaly-based. Honeypots are also deployed in the network to catch malicious packets. *BOGTA* is divided into four phases *Game Theory*, *Preprocessing*, *Feature Elimination*, and *Bayesian Optimization DNN*. These four phases are discussed in detail throughout the Chapter. The filtered traffic from the defender system is sent to the Cloud Layer for safe computing.

5.1.1 Motivation

The ever-expanding horizons of cloud computing's applications underscore its profound impact on various domains. The diverse landscape of cloud-based services

and solutions demands a robust defense against potential security breaches. To fortify these digital bastions, organizations deploy a spectrum of tools ranging from specialized hardware devices like IDS and Firewalls to sophisticated software solutions like anti-virus programs. This research endeavors to confront the multifaceted security challenges embedded in cloud computing. It aims to design an innovative framework, christened as BOGTA (Bayesian optimized Game Theoretic approach), empowered by Deep Neural Networks (DNNs) and Game Theory. BOGTA cultivates a harmonious coexistence between these cloud security components, optimizing their respective roles to create a robust line of defense. Leveraging the predictive prowess of DNNs, the proposed model strategically allocates attacks to specific detection modules within the defender system, thereby minimizing False Alarm Rate and elevating Detection Rate (DR).

5.1.2 Game Theory Approach in BOGTA Model

The mathematical representation of the suggested non-cooperative game model G is as follows [178]:

$$G = (D, A)(Sd, Sa)(Ud, Ua) \tag{5.1}$$

where Sd and Sa are the strategies space, Ud and Ua is the payoff functions for the defender and the attacker, respectively. The strategy set for the attacker can be represented as $Sa = \{Sa_1, Sa_2\}$ and is given in Table 5.1.

Table 5.1: Attacker's Strategies to Attack the Cloud

Representation	**Strategy**
Sa_1	Regular attacks
Sa_2	Sophisticated attacks.

The defender's strategy set can be represented as $Sd = \{Sd_1, Sd_2, Sd_3\}$ and are elaborated in Table 5.2.

Table 5.2: Defender's Strategies to Defend the Cloud

Representation	**Strategy**
Sd_1	Detection with Signature based IDS
Sd_2	Detection with Anomaly based IDS
Sd_3	Detection with Honeypot.

The terminology used to define the payoff functions is given by Table 5.3.

Table 5.3: Parameters to develop BOGTA

Description	**Symbol**
Energy consumed by IDS	ϵ_i
Energy consumed by Honeypot	χ_h
Gain for successfully detecting	κ_d
Value of the assets under attack	υ
Resource consumption by attacker	ρ_a
Gain for successfully attacking	τ_a
Waiting time cost for attacker	ω_a
Detection rate of IDS for signature	ξ
Detection rate of IDS for anomaly	α
Detection rate of honeypot	λ

The main parameters considered are the energy consumption for the IDS *(ϵ_i)* and Honeypots *(χ_h)*, the gain or benefit *(κ_d)* the defender system gets by successfully detecting the attack and the value of the assets *(υ)* under attack for particular period *(t)*. The assets can be any cloud resources, including a server, hypervisor, virtual machine, or API. The detection rate of the signature and anomaly modules is given by ξ and α, respectively. The detection rate of a honeypot is given by λ. The attacker attacks with some resources, which leads to the exhaustion of the resources *(ρ_a)* during the attack leading to the negative impact on its payoffs. The attacker waits for some time which costs him *(ω_a)*. The gain *(τ_a)* which the attacker gets after successfully invading the cloud adds up to the payoff of the attacker. The whole model is designed considering the time parameter *(t)* for which the attacker attacks and the defender defends. The attack remains for a particular time that is used with each parameter. The defender will respond to the attack at the same time *t*. The attacker attacks with regular techniques already known to the defender, and the defender monitors with the signature-based method. Hence it will successfully monitor the attacks. The payoff of the IDS depends upon the energy consumed and the benefit gained for the successful monitoring and is given by Eq. 5.2:

$$Ud_{11} = \kappa_d(t) - \epsilon_i(t) \tag{5.2}$$

The attacker loses the resources he used because of the unsuccessful attack. The payoff of the attacker is given by Eq. 5.3:

$$Ua_{11} = -\rho_a \tag{5.3}$$

Similarly, there are payoffs for the other strategies. The complete payoff functions of defender and attacker are described by Matrices M_D and M_A as shown in Eq. 5.4 and Eq. 5.5 respectively.

$$M_D = \begin{bmatrix} \kappa_d(t) - \epsilon_i(t) & -\epsilon_i(t) - \upsilon(t) \\ -\epsilon_i(t) & \kappa_d(t) - \epsilon_i(t) \\ \kappa_d(t) - \chi_h(t) & -\chi_h(t) - \upsilon(t)) \end{bmatrix} \tag{5.4}$$

$$M_A = \begin{bmatrix} -\rho_a(t) & \tau_a(t) - \rho_a(t) \\ \tau_a(t) - \rho_a(t) & -\rho_a(t) \\ -\rho_a(t) & \tau_a(t) - \rho_a(t)) \end{bmatrix} \tag{5.5}$$

To calculate the Nash Equilibrium (NE), the scribing method is used [179]. The maximum benefit function is calculated with the help of Matrices M_D and M_A. If the maximum benefit function of M_A matches the corresponding element of M_D, that is the pure strategy NE solution of this game model. The maximum profits for attacker by Eq. 5.5 are the a_{12}, a_{21} and a_{32}. Similarly, d_{11}, d_{22} and d_{31} are the maximum profits for the defender from Eq 5.4.

An attacker's most profitable strategy is launching a Sophisticated Attack (SA) on the cloud. However, complex assaults cost more resources, and the attacker cannot launch a fresh attack every time. Therefore, the defense will choose to detect an abnormality, preventing the attacker. Since the defense has selected the unorthodox approach, the attacker will next attempt to attack using the regular way. Again, the defense might select between the honeypot or signature approach to thwart an attack. The system cannot be stabilized since no pure NE strategy exists. To gain maximum benefit, the attacker will always choose between Sa_1 and Sa_2, which means he will always choose to attack. The Regular Attack (RA) can be readily executed with fewer assets and experience. The sophisticated attack is more complex to execute than the standard attack but has a higher success rate. The defender's tactics remain unchanged. Thus, the updated game model is shown in Table 5.4.

Table 5.4: Payoffs of defender and attacker

Sa,Sd	**Signature based**	**Anomaly based**	**Honeypot**
RA	$\kappa_d(t) - \epsilon_i(t), -\rho_a(t)$	$-\epsilon_i(t), \tau_a(t) - \rho_a(t)$	$\kappa_d(t) - \chi_h(t), -\rho_a(t)$
SA	$-\epsilon_i(t) - \upsilon(t), \tau_a(t) - \rho_a(t)$	$\kappa_d(t) - \epsilon_i(t), -\rho_a(t)$	$-\chi_h(t) - \upsilon(t), \tau_a(t) - \rho_a(t)$

If the attacker attacks with a regular attack and the defender detect with signature-based IDS. Then, the payoff function of the attacker for this scenario can be given by Eq. 5.6.

$$Ua_{11} = (1 - \xi)\tau_a(t) - \rho_a(t) \tag{5.6}$$

Similarly, for attackers, the payoff function can be represented in Eq. 5.7.

$$Ud_{11} = \xi\kappa_d(t) - (1 - \xi)\upsilon(t) - \epsilon_i(t) \tag{5.7}$$

The defender's payoffs are given in Matrix (M'_D) in Eq 5.8.

$$M_{D'} = \begin{bmatrix} \xi\kappa_d(t) - \upsilon(t)(1-\xi) - \epsilon_i(t) & -\epsilon_i(t) - \upsilon(t) & \lambda\kappa_d(t) - \upsilon(t)(1-\lambda) - \chi_h(t) \\ -\epsilon_i(t) - \upsilon(t) & \alpha\kappa_d(t) - \upsilon(t)(1-\alpha) - \epsilon_i(t) & -\chi_h(t) - \upsilon(t) \end{bmatrix} \tag{5.8}$$

The attacker payoffs are given in Matrix (M'_A) in Eq 5.9.

$$M_{A'} = \begin{bmatrix} (1-\xi)\tau_a(t) - \rho_a(t) & \tau_a(t) - \rho_a(t) & (1-\lambda)\tau_a(t) - \rho_a(t) \\ \tau_a(t) - \rho_a(t) & (1-\alpha)\tau_a(t) - \rho_a(t) & \tau_a(t) - \rho_a(t) \end{bmatrix} \tag{5.9}$$

The *BOGTA* algorithm is discussed in Algorithm 5.1.

5.1.2.1 Graphical method to solve game theoretic model

The final payoff matrices are $3X2$, which can be solved with the graphical method of the game theory. The goal of the graphical method is to reduce the $3X2$ matrix into $2X2$ so that it can be solved by the saddle point method. The steps of the graphical method by considering a toy example are discussed as follows:

Step 1: The size of the payoff matrix is reduced by applying the dominance property.

Step 2: Let x be the probability of selection of alternative 1 by player A and (1–x) be the probability of selection of alternative 2 by player A as shown by Eq 5.10.

Algorithm 5.1 BOGTA Algorithm to optimize GTM-CSec

1: Input: $\kappa_d(t), \upsilon(t), \epsilon_i(t), \chi_h(t), \rho_a(t), \tau_a(t)$ and $\omega_a(t)$.
2: Output: Increased Detection Rate, Decreased FPR
3: Begin
4: Calculate: Ud_{ij} and Ua_{ij} where $1 \geq i, j \leq 2$.
5: Construct: M'_D and M'_A.
6: **if** Pure Strategy NE **then**
7: Goto Step 17.
8: **end if**
9: **Else**
10: Input: ξ, α, λ
11: x be the probability of the attacker and $(1-x)$ be the probability of the defender
12: Calculate the updated Ud_{ij} and Ua_{ij}
13: Calculate min-max point
14: Eliminate the extra column of M'_D and M'_A.
15: Calculate the values of x and $(1-x)$
16: Find mixed strategy NE.
17: Start the accurate detection module.
18: End

$$
\begin{array}{cc}
 & \begin{array}{ccc} \multicolumn{3}{c}{\textit{Player B}} \\ 1 & 2 & 3 \end{array} \\
\textit{PlayerA} \begin{array}{c} 1 \\ 2 \end{array} & \begin{bmatrix} -4 & 2 & -6 \\ 3 & -9 & 4 \end{bmatrix} \begin{array}{c} x \\ (1-x) \end{array}
\end{array} \tag{5.10}
$$

The expected gain function of player A, relative to each of player B's options, is determined as shown in Table 5.5.

Table 5.5: Expected Payoff Function and Gain of Both Players

B's Alternative	**A's expected Payoff function**	**A's expected gain**	
		$x=0$	$x=1$
1	$-4x+3(1-x)=-7x+3$	3	-4
2	$2x-9(1-x)=11x-9$	-9	2
3	$-6x+4(1-x)=-10x+4$	4	-6

To accomplish this, the column values of B's alternatives are multiplied by the probabilities of their selection by player A. For instance, the first alternative

of player B is column 1, so multiply -4 with x and 3 with $(1-x)$ and add the resulting expression to obtain the expected payoff function for player A. Similarly, the second option for player B is column 2, so multiply 2 by x and -9 by $(1-x)$ and add them. Similarly, player B's third option corresponds to column number 4, so multiply -6 by x and 4 by $(1-x)$ and add the results.

Step 3: The gain function on a graph by assuming a suitable scale is plotted. If B selects the first alternative *i.e.* first strategy when $x = 0$, A's expected gain is 3, and when $x = 1$, A's expected gain is -4. If B selects the second alternative *i.e.* second strategy when $x = 0$, A's expected gain is -9, and when $x = 1$, A's expected gain is 2. If B selects the third alternative when $x = 0$, A's expected gain is 4; when $x = 1$, A's expected gain is -6.

Step 4: The highest intersection point in the lower boundary of the graph, which acts as the Maximum point as A is the Maximin player as shown in Figure 5.2. The lower boundary is ABC, and the highest point among A, B, and C is B. This intersection point B is called the Maximin point.

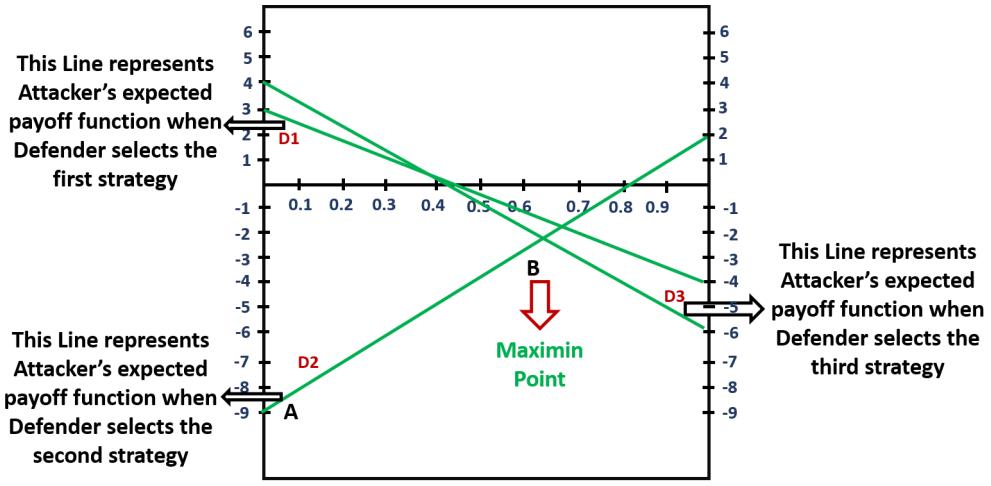


Figure 5.2: Maximin point of $3X2$ Matrix by graphical method of game theory

Step 5: The two lines passing through the maximin point are considered to form a $2X2$ payoff matrix as shown in Eq. 5.11.

$$\begin{array}{cc} & \begin{array}{cc} Player\ B & \\ 2 & 3 \end{array} \\ Player\,A \begin{array}{c} 1 \\ 2 \end{array} & \begin{bmatrix} 2 & -6 \\ -9 & 4 \end{bmatrix} \end{array} \tag{5.11}$$

After calculating the maximin point, the least effective strategy of the defender is eliminated. The remaining $2X2$ matrix can be easily solved by the game theory

saddle point method. The calculated values are used to find the attacker and defender payoffs. According to these payoffs, the most suitable module of the defender system is used to detect the attack. The next step is to optimize the working of IDS to increase the prediction and accuracy of detecting security attacks. It is achieved by Preprocessing, Feature Elimination, and Bayesian Optimization, which are discussed in the forthcoming Sections.

5.1.3 Preprocessing of Dataset to implement BOGTA

The dataset typically has noise and missing values and may be in an unsuitable format, preventing its direct use in ML models. Data preprocessing is required to clean the data and make it suitable for an ML model, improving its accuracy and efficiency. For the three datasets, namely UNSW-NB15, CICIDS, and Bot-IoT, the first 10% missing values are removed using *na.omit* function of RStudio. *na.omit* searches for the rows having 10% missing values and deletes the complete row. Further, the remaining missing values are imputed using *KNNimpute* function. KNNimpute fills the missing values by imputing the value from the nearest neighbor. The data is pre-processed, and the relevant features are selected using the feature selection methods discussed below.

5.1.4 Feature Elimination to Train BOGTA

Recursive feature elimination (RFE) eliminates irrelevant features by returning optimal relevant features to the target variable. This technique starts with building a model on the complete set of features and calculates a rank for each feature. The features with the minor rank are removed, the model is rebuilt, and the rank is computed for the remaining features. This is repeated until the desired sets of features are achieved. Therefore, it will return the essential features by recursively eliminating the minor important features [47]. By doing so, all model dependencies and col-linearity exits are eliminated. Once the features are selected, the model training uses a Bayesian Optimized Deep Neural Network (DNN), discussed in the next Section.

5.1.5 Bayesian Optimisation of BOGTA

DNN consists of more than one hidden layer (h). Each input, hidden, and output layer consists of neuron-like nodes. The i^{th} layer's output is the input for the j^{th} layer. Applying functional transformations and activation functions with weights

($w_{i,j}$) and bias (b) values yields the final output y. For example, the following Eq. 5.12 yields the output y.

$$y_i^h = f(\sum_{k=1}^{K}(w_{i,jj}x_j + b_i)) \tag{5.12}$$

where h represents the hidden layers, x_j represents the input at layer j^{th}, $w_{i,j}$ represents the weights, and b represents the bias value. In the equation, the activation function is nonlinear. Various activation functions, such as *tanh*, Rectified Linear Unit *(ReLU)*, and *sigmoid*, can be used to perform various computations. The current study uses a $ReLU$ activation function with five hidden layers. The hyper-tuning of parameters is done using the Bayesian Optimization technique to improve the performance of the trained model. Three techniques are available to optimize the hyperparameters: Grid Search, Random Search, and Bayesian model-based optimization. The comparison of these three techniques is given in Table 5.6. The lowest score is depicted by +, and +++ is the highest. Bayesian

Table 5.6: Comparison of Hyperparameter techniques

Hyperparameter technique	**Ease of Approach**	**Less Computational Cost**	**Better Results**
Grid Search	+++	+++	+
Random Search	+++	+++	++
Bayesian Optimisation	++	+++	+++

Optimization performs better than the other two techniques because it is a sequential model-based optimization algorithm that uses the previous iteration's results to determine the next set of candidate hyperparameter values. This method selects the next set of hyperparameters to improve the model's performance instead of mindlessly searching the hyperparameter space (as in grid search and random search) [180]. This process is repeated iteratively until it converges on an optimal solution. Bayesian optimization generates a probabilistic model by associating hyperparameters with the probability that the objective function will be met. The workflow of Bayesian optimized DNN is shown in Figure 5.3. The Bayesian approach, as compared to random or grid search, keep track of past evaluation results, which are used to form a probabilistic model mapping hyperparameters to a probability of a score on the objective function given by Eq. 5.13.

$$P(score \mid hyperparameters) \tag{5.13}$$

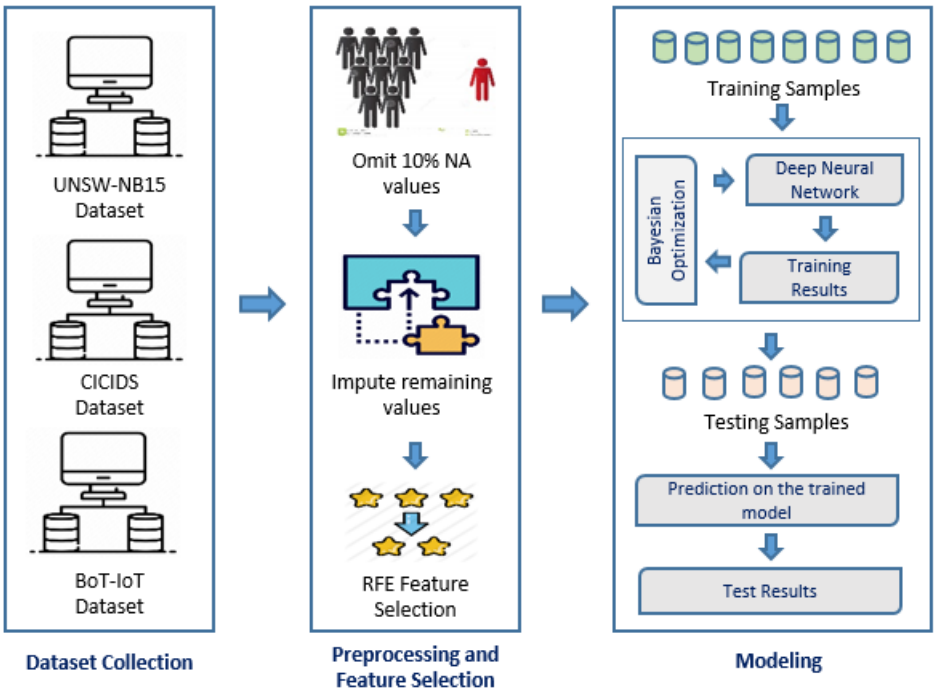


Figure 5.3: Workflow of Bayesian Optimised Deep Neural Network

The Algorithm of Bayesian Optimization is given in Algorithm 5.2.

Algorithm 5.2 Bayesian Optimization

1: Input: Training Dataset $D = \{p_n, q_n\}_{n=1}^{i-1}$, Acquisition function o
2: Output: $max(o)$
3: Begin
4: For i= 1 to n do
5: Optimize the Acquisition function fn
6: Calculate p_i using $p_i = argmax_x o(p|D_{1:i-1})$
7: Calculate objective function i.e. $q_i = f(p_i)$
8: Update posterior of function fn by adding $D_{1:i-1}$
9: End For
10: End

The Bayesian approach builds a probability model of the objective function and uses it to select the most promising hyperparameters to evaluate the true objective function. To model the hyperparameters that yield the best score on the validation set metric, Eq. 5.14 is used [181].

$$x^* = arg \underset{x \in \chi_h}{(min)} \ f(x) \tag{5.14}$$

Here $f(x)$ represents an objective score to minimize $RMSE$ or error rate evaluated on the validation set; x^* is the set of hyperparameters that yields the lowest value

of the score, and x can take on any value in the domain X. The flowchart of the *BOGTA* is discussed in Figure 5.4. The traffic is passed through the Game Theoretic Model and used to generate a dataset to train and test the IDS. The Game Theoretic Model checks NE and starts the most optimal detection module of the defender System. The filtered traffic is sent to the cloud servers.

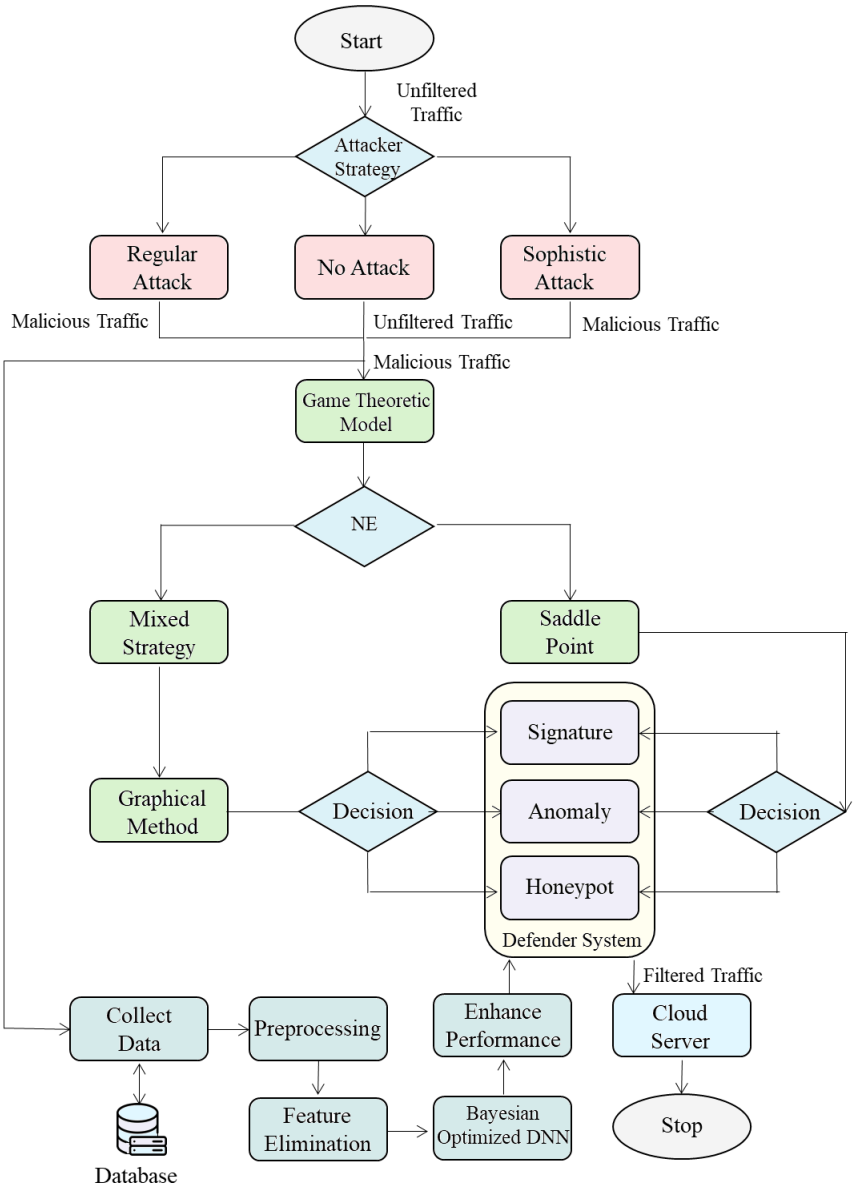


Figure 5.4: Flowchart of the BOGTA

5.2 Analysis of BOGTA

The analysis of the *BOGTA* model can be done by discussing three strategies of the defender system. These are discussed as follows:

Strategy 1: Either the defender system chooses to detect the attack with the signature-based method with probability x or the anomaly-based method with probability *(1-x)*. The attacker attacks with a regular attack by the probability of y and with a sophisticated attack with the probability of *(1-y)*. Eq. 5.15 gives the defender's partial derivatives.

$$\begin{aligned}\frac{\partial U_{Dsa}}{\partial x} = xy\xi\kappa_d(t) + xy\alpha\kappa_d(t) + xy\xi\upsilon(t) + xy \\ \alpha\upsilon(t) - x\alpha\kappa_d(t) - x\alpha\upsilon(t) - y\alpha\kappa_d(t) - y\alpha\upsilon(t)\end{aligned} \tag{5.15}$$

The partial derivatives of the attacker are given by Eq. 5.16.

$$\frac{\partial U_{Asa}}{\partial y} = -xy\xi\tau_a(t) + 1 - \alpha\tau_a(t) - \rho_a(t) + x\alpha\tau_a(t) + y\alpha\tau_a(t) - xy\alpha\tau_a(t) \tag{5.16}$$

By solving Eq. 5.15 $x, (1-x)$ are calculated as:

$$x = \frac{\alpha}{\xi + \alpha}, (1-x) = \frac{\xi}{\xi + \alpha} \tag{5.17}$$

By solving Eq. 5.16 $y, (1-y)$ are calculated as:

$$y = \frac{\xi}{\xi + \alpha}, (1-y) = \frac{\alpha}{\xi + \alpha} \tag{5.18}$$

Strategy 2: The defender system chooses to detect the attack with the signature-based probability of x or the honeypot-based probability by *(1-x)*. The attacker attacks with the regular attack with the probability of y and with a sophisticated attack with the probability of *(1-y)*. The defenders derivative is given by Eq. 5.19.

$$\begin{aligned}\frac{\partial U_{Dsh}}{\partial x} = xy\xi\kappa_d(t) + xy\xi\upsilon(t) + y\lambda\kappa_d(t) + y\lambda\upsilon(t) - \\ xy\lambda\kappa_d(t) - xy\lambda\upsilon(t) - \chi_h(t) - \upsilon(t) + x\chi_h(t) - x\epsilon_i(t)\end{aligned} \tag{5.19}$$

The partial derivatives of the attacker are given by Eq. 5.20.

$$\begin{aligned}\frac{\partial U_{Ash}}{\partial y} = -xy\xi\tau_a(t) - y\lambda\tau_a(t) - xy\tau_a(t) + xy\alpha \\ \tau_a(t) + xy\rho_a(t) + \tau_a(t) - \rho_a(t) + xy\tau_a(t) - xy_R at(t)\end{aligned} \tag{5.20}$$

By solving Eq. 5.19 $x, (1-x)$ are calculated as:

$$x = \frac{\lambda}{\lambda - \xi}, (1-x) = \frac{\xi}{\xi - \lambda} \tag{5.21}$$

By solving Eq. 5.20 $y, (1-y)$ are calculated as:

$$y = \frac{\epsilon_i(t) - \chi_h(t)}{\xi - \lambda}, (1-y) = \frac{\xi - \lambda - \epsilon_i(t) + \chi_h(t)}{\xi - \lambda} \tag{5.22}$$

Strategy 3: The defender system chooses to detect the attack with the anomaly-based with x probability or the honeypot-based method by the probability of *(1-x)*. The attacker attacks with a regular attack by the probability of y and with a sophisticated attack with the probability of *(1-y)*. Eq. 5.23 gives the partial derivatives of the defender.

$$\begin{aligned} \frac{\partial U_{Dah}}{\partial x} &= y\lambda\kappa_d(t) + y\lambda\upsilon(t) - xy\lambda\kappa_d(t) - xy\alpha\upsilon(t) + x\alpha \\ &\kappa_d(t) + x\alpha\upsilon(t) - xy\alpha\kappa_d(t) - xy\alpha\upsilon(t) - \chi_h(t) - \upsilon(t) \end{aligned} \tag{5.23}$$

The partial derivatives of attacker assuming $\chi_h(t)$ is equal to $E_{ds}(t)$ is given by Eq. 5.24.

$$\frac{\partial U_{Aah}}{\partial y} = x\alpha\tau_a(t) + xy\alpha\tau_a(t) - x\lambda\tau_a(t) + xy\lambda\tau_a(t) + \tau_a(t) - \rho_a(t) \tag{5.24}$$

By solving Eq. 5.23 $x, (1-x)$ are calculated as:

$$x = \frac{\lambda}{\lambda + \alpha}, (1-x) = \frac{\alpha}{\alpha + \lambda} \tag{5.25}$$

By solving Eq. 5.24 $y, (1-y)$ are calculated as:

$$y = \frac{\alpha}{\alpha + \lambda}, (1-y) = \frac{\lambda}{\lambda + \alpha} \tag{5.26}$$

The calculated values are used to find the attacker and defender payoffs. The model is analyzed after developing the BOGTA model with game theory, ML, and DL techniques. To validate the working of BOGTA, its effectiveness is proven mathematically and is tested on real-time datasets comprising UNSW-NB15, CI-CIDS and BOT-IoT. The results are discussed in the next Section.

5.3 Results and Discussions

This Section is divided into two parts. In the first part, mathematical validation of BOGTA is done by analyzing and comparing the payoffs of the attacker and the defender. In the second part, the BOGTA model is tested on three real-time datasets to check the accuracy, detection rate, and FPR of the IDS.

5.3.1 Mathematical validation of BOGTA

The Game Theory model is implemented in Python to verify and validate the proposed model. Both players' payoffs are calculated. The graphical method to solve the $2X3$ matrix selects the best Strategy out of the three. The *BOGTA* model is compared with the GTM-CSec model as shown in Table 5.7.

Table 5.7: Comparison of GTM-CSec and BOGTA

GTM-CSec	**BOGTA**
Solved game model in 7 cycles	Solved game model in a single cycle
It takes more time to make a decision	Takes less time to make a decision
No Optimisation with modern technique	Optimisation done with Optimised Bayesian DNN Model
Not implemented on a real-time dataset	Implemented on three real-time datasets
Only payoffs of the defender are calculated	DR, Payoffs, Accuracy, and FPR are calculated

The payoffs of the attacker and defender for the GTM-CSec Model are depicted in Figure 5.5 and for *BOGTA* in Figure 5.6. The graphs show that the payoff of the defender remains higher in *BOGTA* compared to the GTM-CSec. Both models are tested on several iterations to evaluate the execution time they take to start the accurate module of the defender system. Figure 5.7 shows that the *BOGTA* takes significantly less time than the GTM-CSec Model.

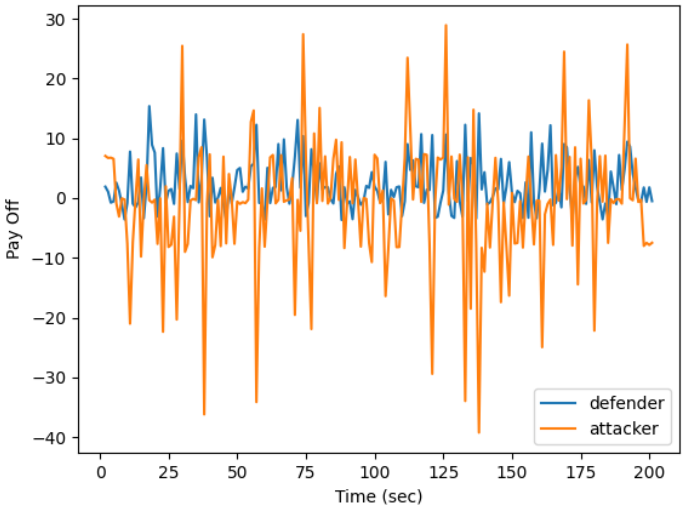


Figure 5.5: Payoffs of attacker and defender of GTM-CSec

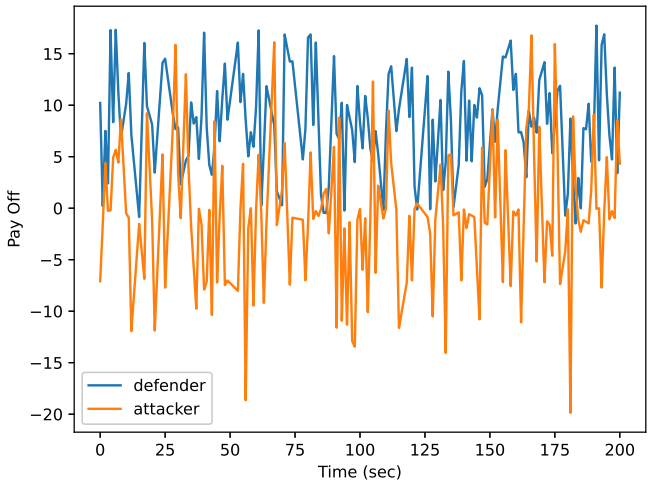


Figure 5.6: Payoffs of attacker and defender of BOGTA

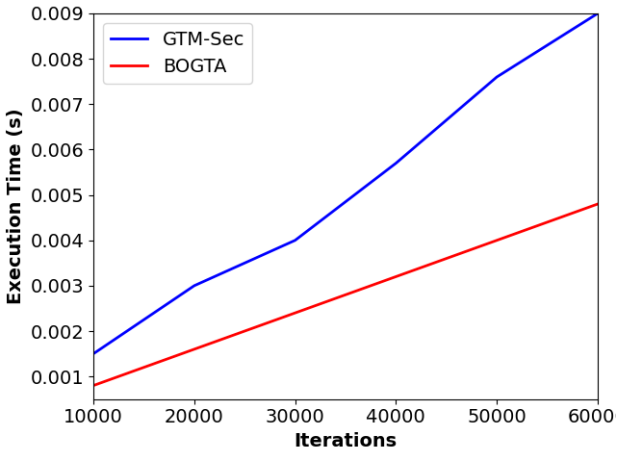


Figure 5.7: Comparison of BOGTA and GTM-CSec about execution time

5.3.2 Validation of BOGTA with real-time datasets

The game theory model is validated using Bayesian Optimised DNN. Three real-time datasets comprising UNSW-NB15, CICIDS, and BoT-IoT are collected, which were then preprocessed using *na.omit* and *KNNImpute* function. The description and the results gathered from the datasets are discussed as follows:

UNSW-NB15: It consists of nine types of attacks, namely, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms [182]. Therefore two classes were there, one for attacks and the other for normal samples. UNSW-NB15 has 175, 341 and 82, 332 training and testing instances with 49 features. Figures 5.8 show the accuracy and detection rate comparison of BOGTA with Kumar *et al.* [183] on the UNSW-NB15 dataset. The results show BOGTA

has performed better with a 10.29% increase in accuracy and a 9.66% increase in the detection rate of the signature-based module of IDS.

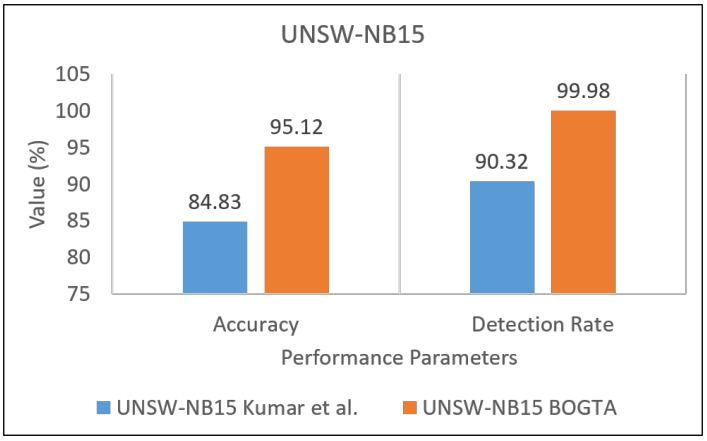


Figure 5.8: Comparison of Accuracy and Detection Rate of *BOGTA* on UNSW-NB15 with Kumar *et al.*

CICIDS: The dataset contains benign and the most up-to-date common attacks, which resembles the actual real-world data (PCAPs) [176]. CICIDS has divided into 93,500 and 28,481 training and testing samples, respectively, with 49 features. The results are shown in Figure 5.9. BOGTA outperforms the existing model, R. *et al.* [184] with a 2.1% increase in accuracy and 3.75% increase in the detection rate of the anomaly-based module of the IDS.

Bot-IoT: It includes DDoS, DoS, OS and Service Scan, Keylogging, and Data ex-filtration attacks, with the DDoS and DoS attacks further organized, based on the protocol used [185]. To ease the handling of the dataset, we extracted 5% of the original dataset via select MySQL queries. The extracted 5% comprises

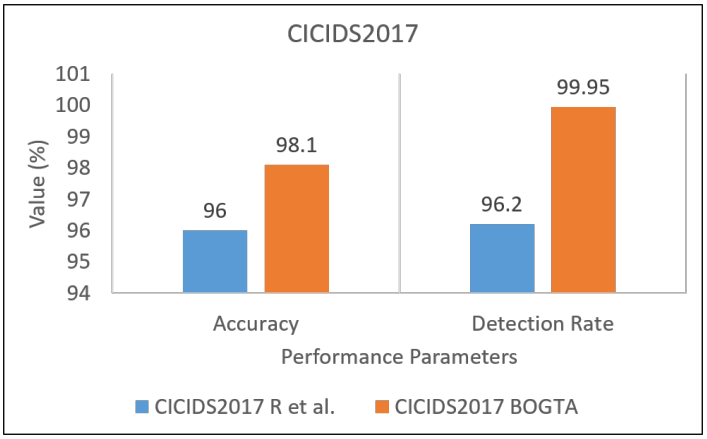


Figure 5.9: Comparison of Accuracy and Detection Rate of BOGTA on CICIDS with R. *et al.*

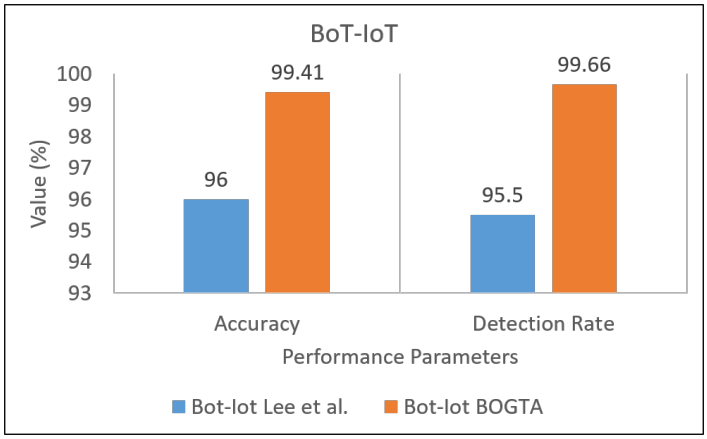


Figure 5.10: Comparison of Accuracy and Detection Rate of BOGTA on Bot-IoT with Lee *et al.*

four files of approximately 1.07 GB and about 3 million records. The collected dataset is in the raw form, which needs to be preprocessed. Then RFE is used to extract relevant features and passed to DNN for training. The dataset was divided into 70% training and 30% testing dataset. The parameters of DNN are hyper-tuned using Bayesian Optimization, and performance is evaluated on the testing dataset. The results are generated and compared with existing techniques. Figure 5.10 compares the *BOGTA* model with the existing model, Lee *et al.* [186]. The accuracy and detection rate of the honeypot module is 3.41% and 4.16% more than the existing model.

The FPR of all three models on three datasets is presented in Figure 5.11. The BOGTA has the lowest payoffs of all the existing methods [183], [184] and [186]. The plots depict that *BOGTA* has performed well with an accuracy of 95.12%,

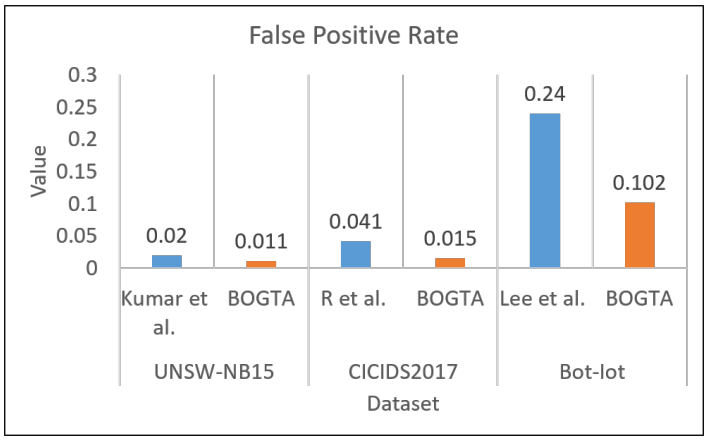


Figure 5.11: Comparison of False Positive Rate of BOGTA with existing models (Kumar *et al.*, R. *et al.* and Lee *et al.*)

98.1%, and 99.41% in the UNSW-NB15, CICIDS, and Bot-IoT datasets, respec-

tively. The Detection Rate comes out to be 99.98%, 99.95%, 99.66%, and FPR are calculated as 0.011%, 0.015%, 0.102% in signature, anomaly, and honeypot module, respectively.

5.4 Summary

This Chapter is inspired by game theory and the Bayesian Optimised Deep Learning model and consists of three detection modules: signature-based, anomaly-based, and honeypot-based. *BOGTA* assists the defender system in handling external security attacks. The contribution of this Chapter is:

- A novel lightweight module based on game theory is developed to study the non-cooperative behavior between the attacker and the defender in Cloud Environment.
- The strategies of the attacker are analyzed with different strategies of the defender, and higher payoffs of the defender as compared to the attacker show the effectiveness of *BOGTA*.
- The overall performance of the IDS improves with lower time complexity, higher detection rate, and lower false positive rate with *BOGTA*.
- The model is trained, implemented, and tested on UNSW-NB15, CICIDS 2017, and Bot-IoT datasets. The results show that *BOGTA* has taken significantly less time to decide than GTM-CSec. The performance increased drastically with an improvement of 9.66%, 3.75%, and 4.16% in detection rates of the three modules.
- The accuracy is increased by 10.29%, 2.1%, and 3.41%. The FPR of BOGTA is reduced to 0.01%, 0.026%, and 0.138%. The higher detection rate values and lower FPR values depict the adequate performance of *BOGTA*.

Chapter 6

GTA-IDS: Game Theoretic Model to Enhance IDS Detection for Internal Cloud Attacks

In this Chapter, information theory and game theory approaches are used to tackle internal security attacks on the cloud that can be done with social engineering. The information entropy tracks the abnormal flow in the traffic, and it is sent to the game-theoretic model for further analysis. A game is modeled between the malicious node and the defender system. Different strategies for both players are delineated, and a mixed strategy Nash Equilibrium (NE) is attained to conclude the game. The Intrusion Detection System (IDS) is used as a defender system and is trained using ML techniques, namely Random Forest (RF), Gradient Boosting Machine (GBM), and Deep Neural Network (DNN). The GTA-IDS is implemented on a benchmark NSL-KDD dataset to check the accuracy, Detection Rate (DR), and False Positive Rate (FPR) of the defender system.

6.1 Overview of GTA-IDS

Cloud computing is a rapidly growing technology used in the current era. Besides external attacks, the cloud should also be protected from internal attacks. Internal attacks can be performed with social engineering, and the reasons include poor access management, angry employee leavers, human error, and employee bribery. The malware can be injected to make a hypervisor, virtual machine, or cloud server into a malicious node. Also, malware attacks increased by 53% in the last year in India. An analysis shows 3.9 trillion malware attacks in 216 countries worldwide. Another research by SonicWall Capture Labs shows a 30% rise in IoT malware (32.4 million) in September 2020 [187]. So, this issue needs to be addressed because modern malware is intelligent and challenging to detect. To protect the assets of the cloud from malicious nodes, there is a need to build a solid defending system. Defender systems like IDS need to be more optimized to detect these intelligent malicious packets for the proposed *GTA-IDS* model, which works on information theory and game theory. Information theory is a

mathematical sub-field concerned with data transmission over a noisy channel. A tenet of information theory quantifies the amount of information in a message. Calculating information entropy is the foundation for techniques including feature selection, decision tree construction, and fitting classification models.

6.1.1 Motivation

The motivation of this research is to address the security implications of IoT devices and propose a solution to improve the efficiency and accuracy of the Defender system using a game theoretic approach. The main aim is to enhance the detection of Intrusion Detection System (IDS) in cloud environments by modeling the interaction between the Attacker and the Defender as a game and finding the optimal strategies for both parties. A non-zero non-cooperative game is designed between the malicious node and the defender system. Both malicious node and defender use their best strategies to increase their payoffs. By delineating the Bayesian Nash Equilibrium (BNE), the probability of attacking and defending the Asset is calculated. Further, Machine Learning (ML) is used to lower the false positive rate (FPR) and increase the detection rate (DR) of the IDS.

6.2 GTA-IDS Model to tackle internal security attacks in cloud

The defender system must always work efficiently and accurately with minimum energy requirements [179]. GTA-IDS is designed based on a non-zero non-cooperative game to make the defender system more efficient. The framework of the proposed model is depicted in Figure 6.1. The packet is entered into the defender system comprising Entropy Module, GTA-IDS, and the IDS. These three models are described in detail in the following Section.

6.2.1 Entropy Module

The entropy module consists of three layers: *Capture Layer*, *Calculation layer*, and *Process layer*. The packets are captured at the *Capture Layer*. The information entropy of the packet is calculated at the *Calculation Layer* by using relative entropy or Kullback Leibler distance formula [188]. It is given by the Eq. 6.1:

$$D(P||Q) = \sum_i p_i log(\frac{p_i}{q_i}) \tag{6.1}$$

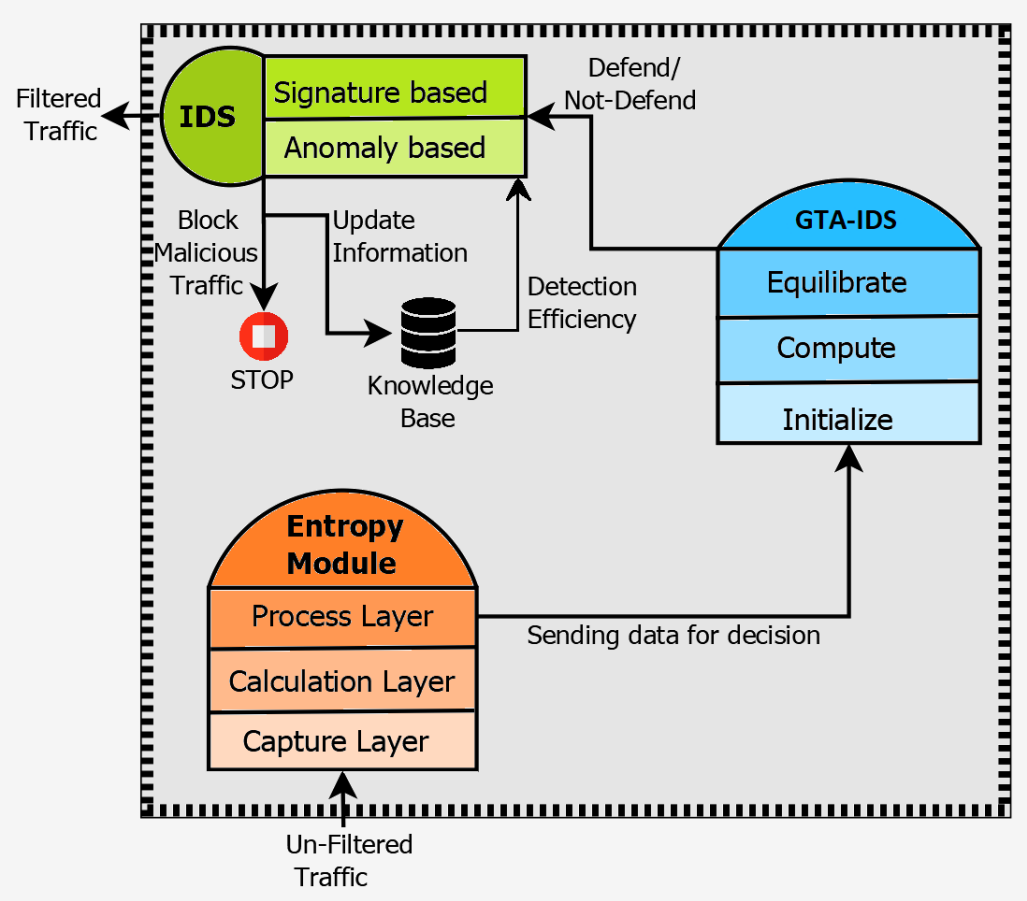


Figure 6.1: Framework of GTA-IDS Model

where Q is the distribution calculated on the legitimate set of traffic, P is the distribution that can be of the malicious form. If the value of p_i is greater than q_i, the value of the logarithm will be greater than 1, resulting in a higher total value. So, a threshold value is placed, which is given by Threshold High (TH_h) and Threshold Low (TH_l), if the Change in Entropy (CiE) increases by TH_h, the packet will be treated as malicious and will directly be blocked. If the value lies between TH_h and TH_l, the packet propagation to the network will be decided on anomaly and signature tests. Traffic will be forwarded to the cloud network if the value is below TH_l. The decision is taken at the *Process Layer* by calculating the entropy of the information changed. According to the decision made by the Entropy Module, the packet is passed to the GTA-IDS model, and the process is discussed in Figure 6.2 with the help of a flowchart.

6.2.2 GTA-IDS Module

This module is divided into three stages: *Initialize*, *Compute*, and *Equilibrate*. These three stages are discussed as follows:

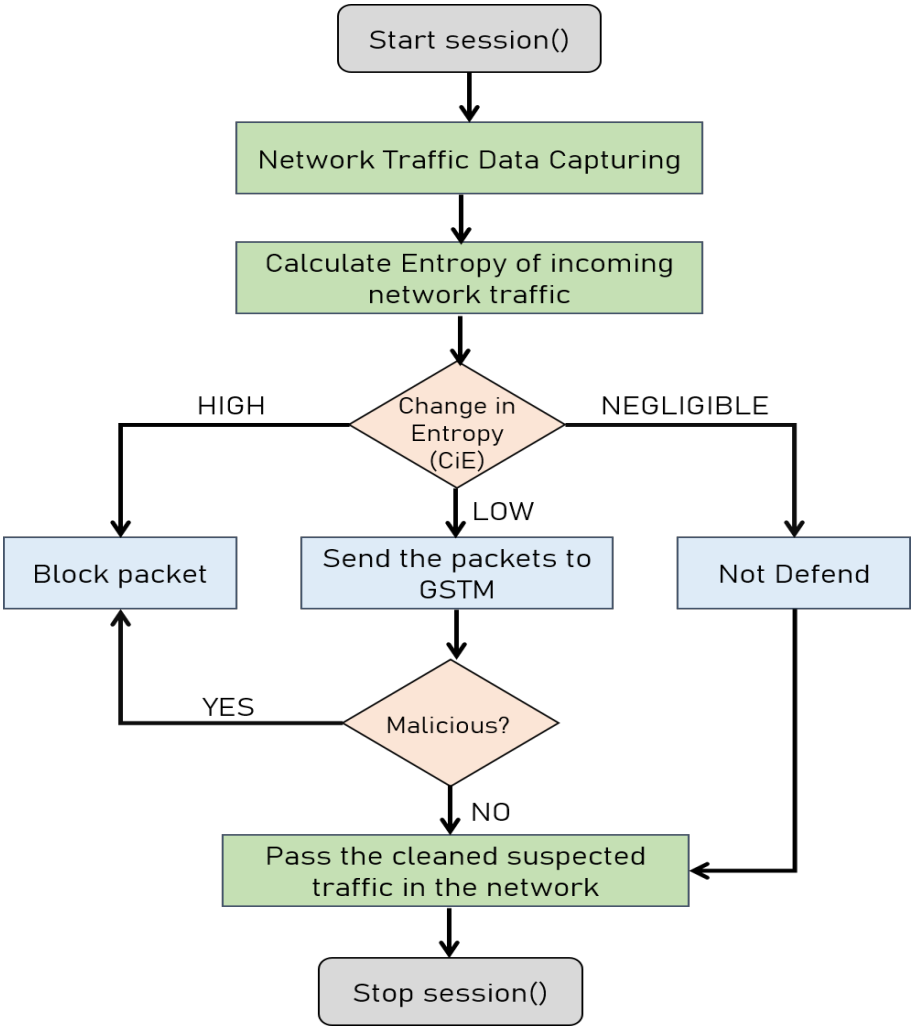


Figure 6.2: Working steps of GTA-IDS Model

Initialize: In this stage, the interaction between two players, namely malicious node (P_a) and defender (P_d), is designed. The malicious node has two pure strategies *[Attack, Not-Attack]*. The defender has two strategies, namely *[Defend, Not-Defend]*. However, the strategy of the normal node is *[Not-Attack]*. It can be represented as Figure 6.3. The defender has no prior knowledge about the maliciousness of the nodes. So, the game of imperfect information can be mathematically given as $G = (P, S, U)$ [189] where:

- $P = P_a, P_d$ are the two participating and competitive players, namely the malicious node and the defender, respectively.

- $S = S_a X S_d$ is the game strategy space where strategies of the malicious node are represented as S_a and strategies of the defender are given by S_d.

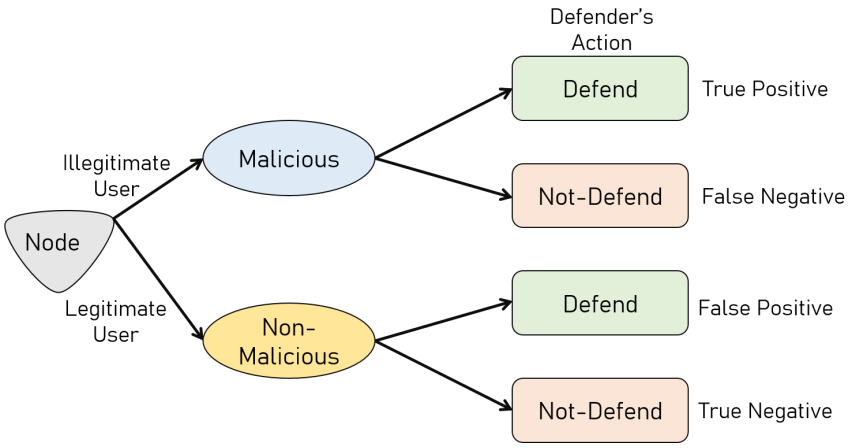


Figure 6.3: Strategies of malicious node and defender in GTA-IDS Model

- $U = U_a X U_d$ are the payoffs gained by the malicious node and the defender by their strategy space S. U_a is the payoff of the malicious node with S_a and U_d is the payoff gained by the defender with strategy S_d. The other parameter used in the game model is represented in Table 6.1.

Table 6.1: Parameters used to develop GTA-IDS Model

Description	**Symbol**
Resources consumed by defender	E_{ds}
Detection rate of defender	λ
False Positive Rate of defender	σ
Value of the Asset to be protected	ψ
Malicious node	d
Number of malicious nodes	z
Timestamp	T_a
User ID of Trusted nodes	T_{id}
Threshold value of entropy(High)	TH_h
Threshold value of entropy(Low)	TH_l
Change in Entropy	CiE

The energy or the resources the defender consumes while defending the traffic is E_{ds}. If the defender is at rest, this factor will be zero. The detection rate *i.e.*, the true positive rate, or the accuracy of the defender is defined by λ. The false

positive rate means the defender identified the legitimate node as malicious and blocked it. It is given by σ. The value of λ and σ lies between 0 and 1 *i.e.* λ,σ $\epsilon[0,1]$. The asset the defender needs to protect can be a hypervisor, a server, software, or any other helpful cloud entity. The Asset's value should always be greater than the resources the defender consumes and more significant than the resources exhausted on the malicious node's side. Suppose the asset value is lower than the resources consumed by the defender. In that case, it will not prevent the attack as, ultimately, it is gaining a low value (players are rational in game theory). Otherwise, the malicious node and the defender will continuously lose the game whenever he attacks, similar to the case with the defender. The attacking or malicious nodes are represented by *d*. The number of malicious nodes used is given by *z*. The time the node keeps sending malicious packets is given by T_a. Some nodes connected with the cloud servers are put in the trust category. These nodes are assumed to meet the security requirements to connect to the cloud. The id of these trusted devices is given by T_{id}.

Compute: Based on the parameters, the payoffs of both players are calculated and are given in Table 6.2. It shows the relationship between the malicious node and the defender. All the possible combinations of events *(Attack, Defend), (Attack, Not-Defend), (Not-Attack, Defend)* and *(Not-Attack, Not-Defend)* are discussed. The normal node will never attack the cloud servers, so its payoff will be

Table 6.2: Payoff matrix of malicious node and defender.

	Defend	**Not-Defend**
Attack	$(T_aE_{ds}-2\lambda)\psi-zd, (2\lambda-E_{ds})\psi-T_aE_{ds}$	$(\psi-zd)T_a, -T_a\psi$
Not-Attack	$0, (-\sigma\psi-E_{ds})T_a$	0,0

zero. Although, the defender has two strategies to defend or not to defend. By defending the normal node, the defender is just wasting resources. So, the best strategy for the defender system is not to defend the normal node.

Equilibrate : After building the game model, there is a need to analyze this model to compute the NE, which is discussed in Section 6.3. Both players will have the best strategies and payoffs at NE.

6.3 Analysis of GTA-IDS Model

The purpose of payoffs shown in Table 6.2 is to calculate the probability of the defender successfully detecting the attack. We assume there is some initial belief

of maliciousness of node and is given by probability p_0. So, $(1 - p_0)$ is the initial belief of a non-malicious node. An extensive game model between the malicious node and the defender is shown in Figure 6.4. The different possibilities of a node and the defender are presented. These possibilities are discussed in Table 6.3.

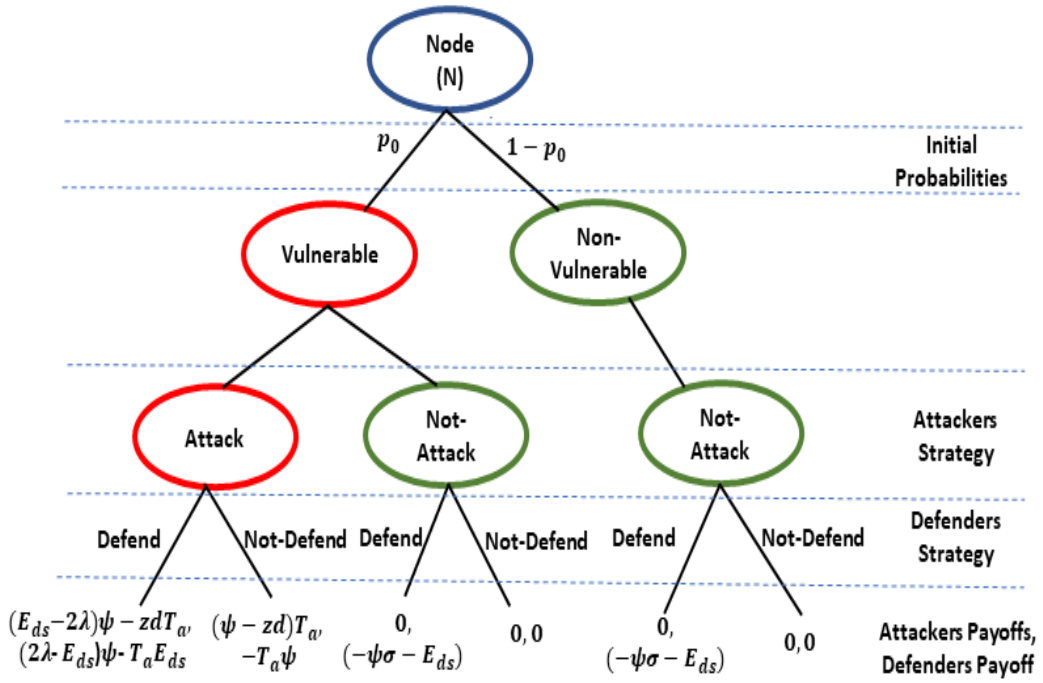


Figure 6.4: An extensive game theory model between defender and malicious node

Table 6.3: Description of different cases in GTA-IDS Model

Case	Malicious Node's Strategy	Defender's Strategy
Case 1	Attack	Defend/Not-Defend
Case 2	Attack/Not-Attack	Defend
Case 3	Attack	Defend
Case 4	Attack	Defend, Not-Defend

6.3.1 Computation of Nash Equilibrium (NE)

To calculate the NE, the four cases are discussed in detail as follows:

Case 1: The node is malicious and chose its pure strategy as *Attack*. The pure strategies of the defender are *Defend* and *Not-Defend*. The payoffs of defender in this case is given Eq 6.2 and Eq 6.3:

$$U(Defend) = p_0[(2\lambda - E_{ds})\psi - T_a E_{ds}] - (1 - p_0)[(\sigma\psi + E_{ds})T_a] \tag{6.2}$$

$$U(\textit{Not-Defend}) = -p_0 T_a \psi \tag{6.3}$$

If the defender chooses to *Defend*, it will result in true positive means this strategy will gain the defender. The defender successfully prevents the asset from getting damaged. If the defender chooses the strategy *Not-Defend*, it will lose the asset, and the malicious node will be completely in gain.

Case 2: The pure strategy of defender is *Defend*, the payoffs of the malicious node for strategy *Attack* and *Not-Attack* is given in Eq 6.4 and Eq 6.5 respectively:

$$U(\textit{Attack}) = p_0[(T_a E_{ds} - 2\lambda)]\psi - zdT_a \tag{6.4}$$

$$U(\textit{Not-Attack}) = 0 \tag{6.5}$$

U(*Attack*) represents the profit of the attack by exploiting the asset. If the malicious node chooses the strategy *Not-Attack*, neither will it lose nor gain anything. So, the payoff, in this case, will be 0.

Case 3: When the malicious node has chosen its strategy as *Attack*, the defender will choose its pure strategy *Defend*.

If U(*Defend*) > U(*Not-Defend*): If the defender plays its *Defend* strategy, the node will stop attacking because of getting caught every time. Hence, the defender and malicious node keep changing strategy in this case, so it does not follow NE.

If U(*Defend*) < U(*Not-Defend*): the defender chooses *Not-Defend*, and then the malicious node strategy will always be *Attack* which follows the NE.

Case 4: If the malicious node chooses *Not-Attack* strategy, the beneficial move for the defender is to *Not-Defend*. But if the defender tends to choose the *Not-Defend* strategy, the malicious node changes its strategy to *Attack*. Thus, this case will also not follow NE.

So, in some cases, there does not exist NE. So, we need to find the mixed strategy NE. Let the attacking probability of the malicious node is *p*. Thus, *(1-p)* is the probability of not-attacking. Similarly, *q* is the probability of defending the Asset, and *(1-q)* is the probability of not defending. From Figure 6.4, the net payoffs can be deduced by Eq. 6.6 and Eq. 6.7

$$\begin{aligned} U(\textit{Defend}) = {} & pp_0[(2\lambda - (E_{ds})\psi - T_a E_{ds}] - (1-p)p_0[(\sigma\psi - E_{ds})T_a] - \\ & (1-p)[(\sigma\psi - E_{ds})T_a] \end{aligned} \tag{6.6}$$

$$U(\textit{Not-Defend}) = -pp_0 T_a \psi \tag{6.7}$$

For the malicious node, the payoffs for strategies *Attack* and *Not-Attack* are given

by Eq. 6.8 and Eq. 6.9:

$$U(Attack) = p_0[q((T_aE_{ds} - 2\lambda)\psi - zdT_a) + (1-q)((\psi - zd)T_a)] \tag{6.8}$$

$$U(Not\text{-}Attack) = 0 \tag{6.9}$$

For the sake of Equilibrium, we equate U(*Defend*) and U(*Not-Defend*) and probability for attack(p) and is given by Eq. 6.10:

$$p = \frac{\sigma\psi T_a + E_{ds}T_a}{[2\lambda - E_{ds} + T_a(\sigma+1)]p_0\psi} \tag{6.10}$$

The probability of defending is given by Eq. 6.11:

$$q = \frac{zd - \psi T_a}{\psi(T_aE_{ds} - 2\lambda - T_a)} \tag{6.11}$$

So, to gain maximum payoffs, a malicious node will attack with probability p, and the defender will defend with probability q.

The analysis of the game model clearly shows that there does not exist any pure strategy NE. However, there exists a mixed strategy Bayesian Nash Equilibrium (BNE) in which strategies depend upon the probability of the attack and defense. The GTA-IDS Algorithm is given by 6.1:

6.4 Training of IDS

To further increase the accuracy, detection rate, and false positive rate of the IDS, it is trained with ML models, namely Random Forest (RF), Gradient Boost Machine (GBM), and Deep Neural Network (DNN). The working of these models is discussed as follows:

Random Forest (RF): It is a predictor that combines multiple decision trees on distinct subsets of data and then averages the results to improve the predictive accuracy of the dataset. In RF, random vector α_k is generated using distribution without requiring previous random vectors $\alpha_1, ...\alpha_{m-1}$. The tree is constructed with α_k and the training set to generate the classifier $p(x, \alpha_m)$ from the input vector x. α represents independent random numbers chosen at random within the interval $[1, m]$. The production of many trees aids in voting for trendy classes. This technique generates a random forest. Therefore, a random forest classifier is produced by grouping various classifiers of tree structure $p(x, \alpha_m), m = 1$. The margin function in the ensemble of classifiers $p_1(x), p_2(x), ...p_m(x)$, with a ran-

dom training set selected according to a random vector's distribution (x, y). In Equation 6.12, the indicator function is $I()$, the margin value; M is the classification confidence, and an is the empirical probability of any classifier outcome $p_m(x)$. The probability subscripts x, y are utilized to define Eq. 6.13 for the generalization error E^* [190].

$$M(x, y) = \hat{a}_m I(pm(x) = y) - \underset{i \neq y}{max} \hat{a}_m (pm(x) = i) \tag{6.12}$$

$$E^* = P_{x,y}(M(x, y) < 0) \tag{6.13}$$

Algorithm 6.1 GTA-IDS Algorithm to tackle internal cloud security attacks

Input: Packet entering the defending system
Output: Non-Malicious packets entering the cloud
startsession()
if $T_{id} == True_{id}$ **then** ▷ $True_{id}$ is the authenticated id in Database
return Pass
else if $CiE > TH_h$ **then**
return Fail
else if $(CiE < TH_h)$ and $(CiE > TH_l)$ **then**
    initialize GTA-IDS module
    Calculate: $U_{ij}(D)$ and $U_{ij}(A)$.
    Construct: M'_D and M'_A.
    if Pure Strategy NE **then** Goto Step 19.
    end if
Else Input: λ
    for *Case 1 to 4* **do**
        Update $U_{ij}(D)$ and $U_{ij}(A)$ with k and l.
        Find mixed strategy NE.
    end for
    Start the accurate detection module.
end ifreturn Pass
stopsession()

Gradient Boost algorithm (GBM): It operates by combining multiple weak learners to create a single strong learner. The weak learners correspond to a series of connected decision trees, with each tree attempting to reduce the errors of the previous tree. Instead of searching for a general solution for the boost increment in the function space, one can select the new function increment that correlates most

strongly with $-g_t(x)$. This enables the substitution of a difficult optimization task with the traditional least-squares minimization task with Eq. 6.14

$$(\rho_t, \theta_t) = \underset{\rho,\theta}{argmin} \sum_{i=1}^{N} [-g_t(x_i) + \rho h(x_i, \theta)]^2 \tag{6.14}$$

In conclusion, the complete form of gradient boosting algorithm proposal can be formulated to train the IDS [191]. The exact form of the algorithm derived will depend heavily on the design decisions made for $\psi(y, f)$ and $h(x, \theta)$.

Deep Neural Networks (DNNs): DNNs are feed-forward neural networks with multiple hidden layers of neurons used for classification tasks of varying complexity. DNN consists of more than one hidden layer (h). Each input, hidden, and output layer consists of neuron-like nodes. The i^{th} layer's output is the input for the j^{th} layer. Applying functional transformations and activation functions with weights ($w_{i,j}$) and bias (b) values yields the final output y. For example, the following Eq. 6.15 yields the output y.

$$y_i^h = f(\sum_{k=1}^{K} (w_{i,jj} x_j + b_i)) \tag{6.15}$$

where h represents the hidden layers, x_j represents the input at layer j^{th}, $w_{i,j}$ represents the weights, and b represents the bias value. The models above train the IDS to increase its accuracy, Detection Rate (DR), and False Positive Rate (FPR). The results obtained by applying these techniques are discussed in the next Section.

6.5 Results and Discussions

A Python code is developed to determine the best strategy for the malicious node and defender during the *[Attack, Defend]* scenario. The parameters' values are randomly varied to understand the players' best strategy. The payoffs of the defender and the malicious node are shown in Figure 6.5. It can be observed that the payoffs of the defender are almost stable. Still, the payoff of the malicious node fluctuates at a high rate but in a negative direction. This means the malicious node is trying its best with different parameters and methods each time but is not winning. So, it is clear that the malicious node's payoffs are lower than the defender's. The defender's strategy is to use minimum energy and minimize the attacking time, while the malicious node's strategy is to gain maximum payoff by increasing the attack time and using fewer devices. Also, the best payoff for the malicious node is to attack the cloud with probability p, and the defender should

defend the cloud with probability q to gain maximum payoff as given in Eq. 6.1 and Eq. 6.11.

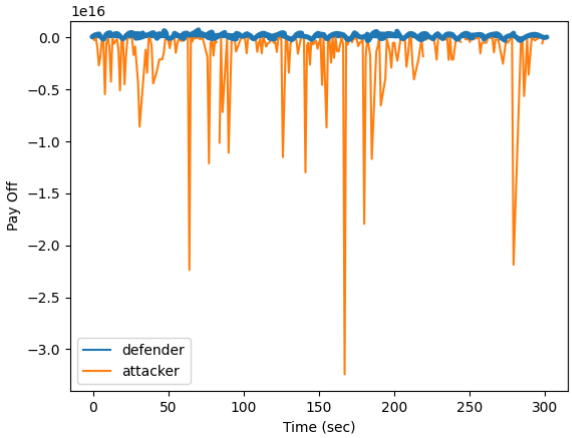


Figure 6.5: Comparison of malicious node's and defender's payoff

GTA-IDS is tested on a real dataset NSL-KDD, and the results are compared with other existing techniques like NIDS [192], AS-IDS [193] and Hybrid [194]. The NSL-KDD dataset is an enhanced version of the KDDCUP'99 dataset. This dataset comprises a training and testing dataset with 125973 and 22544 records. Each NSL-KDD record has 41 features (e.g., protocol type, Logged in, and Duration, *etc.*). These features are represented as numeric, nominal, and binary, defined as continuous or discrete, and labeled as normal or attack. The NSL-KDD dataset is divided into four classes. The types of attacks considered to test the model are DoS, probe, R2L, and U2L. The Z-score achieves the normalization of the dataset. After the normalization, the three models, Gradient Boosting Machine(GBM), Random Forest, and Deep Learning, are trained using ML. After training with these models, the Detection Rate (DR) and the False Positive Rate (FPR) of the IDS are calculated. The DR and FPR values are tested on the GTA-IDS model. The payoffs of the defender are calculated from the DR and FPR values that are further delineated from the NSL-KDD Dataset.

Table 6.4 shows the comparative analysis of GTA-IDS with the existing techniques. GTA-IDS performed well with a detection rate and payoff value of 99.5 and 0.84, respectively. The FPR is also the lowest of all techniques, with a value of 0.05.

The graphical representation of the payoffs of the IDS respective to its detection rate is shown in Figure 6.6.

The trend clearly shows the payoff value increases and decreases with the increase and decrease of the DR. In other words, the payoff gained by the defender is directly proportional to the DR of the defender. It also shows that the DR and

Table 6.4: Comparative Analysis of GTA-IDS with existing models

Technique	Model	detection rate	FPR (in %)	payoff
NIDS [192]	KNN	0.71	2.27	0.66
	CNN	0.7	2.25	0.7
	C4.5	0.76	3.88	0.58
	IBK	0.71	2.28	0.22
AS-IDS [193]	AS-IDS	0.96	3.4	0.72
Hybrid [194]	Hybrid	0.94	0.77	0.69
GTA-IDS	GBM	0.995	0.07	0.84
	RF	0.97	0.05	0.75
	DNN	0.997	0.06	0.81

payoff of GTA-IDS are highest compared to the other methods.

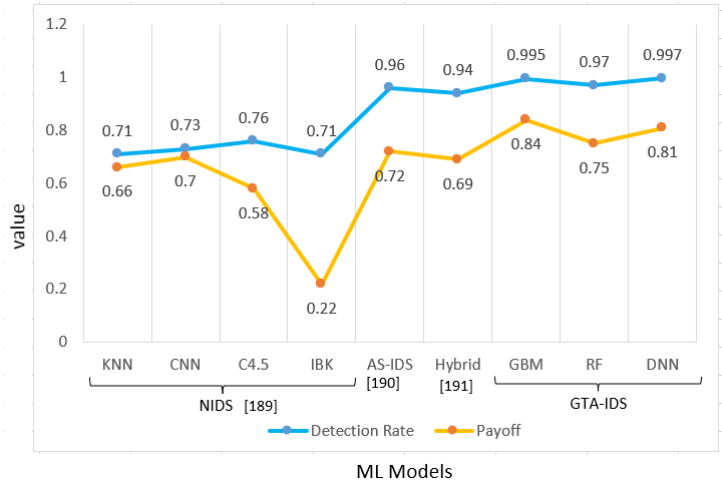


Figure 6.6: Comparison of defender's payoff and detection rate with existing models

The relation between the FPR and the payoff of the defender is presented in Figure 6.7. With the employment of RF, GBM, and DNN in the GTA-IDS module, the values of the FPR swiftly decrease. Also, the FPR of GTA-IDS is the lowest when compared to other models comprising NIDS [192], AS-IDS [193], and Hybrid [194] The low FPR leads to an increase in the payoff of the defender. The overall comparison of GTA-IDS with other models is given in Figure 6.8. The bars of the graphs clearly show the GTA-IDS performs better than other techniques in DR, FPR, and payoff. The clubbing of the GTA-IDS module with the IDS increases the DR and the payoff of IDS by 0.997 and 0.84, respectively. It also decreases the FPR of the IDS to 0.07.

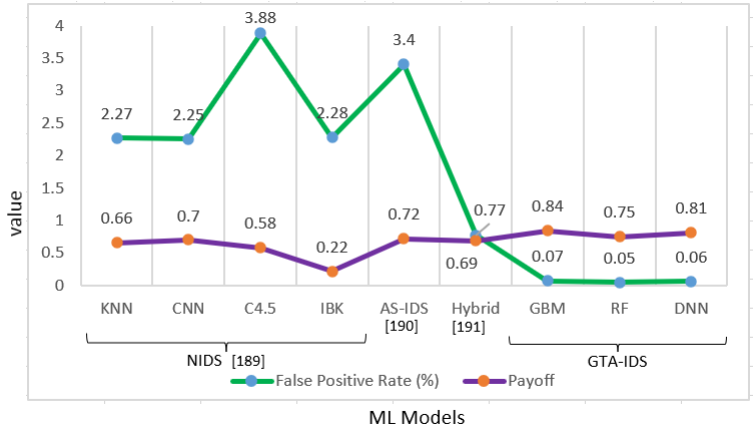


Figure 6.7: Comparison of defender's payoff and False Positive Rate with existing models

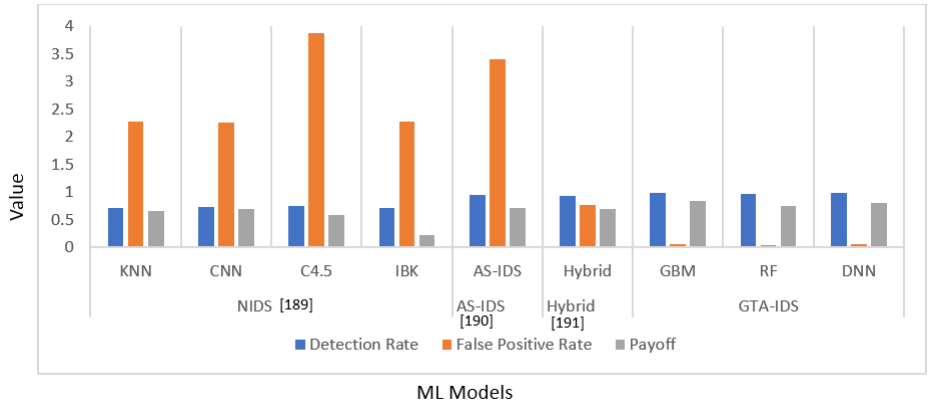


Figure 6.8: Comparison of GTA-IDS Model with existing models

6.6 Summary

This Chapter proposes a game-theoretic model (GTA-IDS) to make the defender more efficient and accurate based on non-zero-sum non-cooperative game theory and information theory to fight the internal malicious nodes. The game model is analyzed after specifying the players' payoffs. The probability of a malicious node attacking and a defender defending is calculated. The following observations are made:

- A lightweight module (GTA-IDS) is added to work parallel with the existing defender system (IDS) and assists it in working more efficiently by calculating the prior probabilities and entropy of the traffic flowing.

- The energy of the defender system can be saved by activating it only during times of need. When there is a high risk, GTA-IDS will suggest the defender

start the defending, and when there is no risk, the defender can rest.

- The best strategy for the malicious node is to use fewer nodes and increase the time for the attack. The defender's best strategy is to minimize the energy cost and the attack timing.
- The model is applied to the NSL-KDD dataset, and results are compared with NIDS, AS-IDS, and Hybrid IDS. With the addition of the GTA-IDS module, the detection rate of the IDS comes to be 99.5%, and FPR comes to be 0.07% which is better than the existing techniques.

Chapter 7

NCGTM: A Non-Cooperative Game Theoretic Model to Assist Hybrid IDS in Cloud Environment

In this Chapter, a Non-Cooperative Game-Theoretic Model (NCGTM) is proposed and developed to enhance the decision-making process of the hybrid Intrusion Detection System (IDS). The attacker's strategies are analyzed against the strategies of the IDS to make better predictions. NCGTM model assists the IDS in determining whether to use the signature, anomaly, or hybrid modules. Different Machine Learning (ML) models comprising Random Forest (RF), Gradient Boosting Machine (GBM), XGBoost, and Deep Neural Network (DNN) are stacked together to train the IDS, which is required to increase the detection rate (DR) and lower the False Positive Rate (FPR). The NCGTM is tested on a real-time cloud dataset and compared with the existing models. NCGTM performs better than the existing approaches by giving better DR and lower FPR.

7.1 Overview of NCGTM

Cloud computing provides excellent help in processing, handling, and storing big data. The attackers try to hack the servers or steal the users' sensitive information. Another motivation for the attackers can be to deface the firm by stopping the cloud servers from providing services by performing attacks like DDoS [195]. The whole scenario is depicted in Figure 7.1. The attacker tries inserting the malicious code into the devices called nodes. The nodes are divided into three categories; *Trusted, Vulnerable and Malicious Nodes*. The nodes with maximum security implications and placed in search places are Trusted nodes. These nodes are impossible to attack. The vulnerable nodes can be attached to sophisticated attacks. The nodes having the most minor security implications are malicious nodes. These are very easy for the attacker to perform security attacks. The malware can be transmitted to the Cloud through the network by malicious and vulnerable nodes. In Figure 7.1, the *red packets* show the infection, and the *black packets* show the unfiltered traffic. If this unfiltered traffic directly enters the cloud

environment, the provider can lose millions of dollars and users' trust. Thus an efficient defender system is required to detect and stop malicious packets. Only then the cloud applications can be safely used by the users. Many tools and techniques are used to provide security. Different researchers have proposed many algorithms, but attackers still find some loopholes and attack the services offered by Cloud. The new approach, much more popular nowadays, is IDS based on autonomic computing. Detection methodologies for the IDS can be categorized as Signature-based Detection (SD), Anomaly-based Detection (AD), and Hybrid Intrusion Detection System (HIDS) [196]. Hybrid IDS is the combination of the SD and AD IDS. To enhance the working of IDS, the NCGTM model is proposed based on non-cooperative game theory. The working of NCGTM is discussed in the next Section.

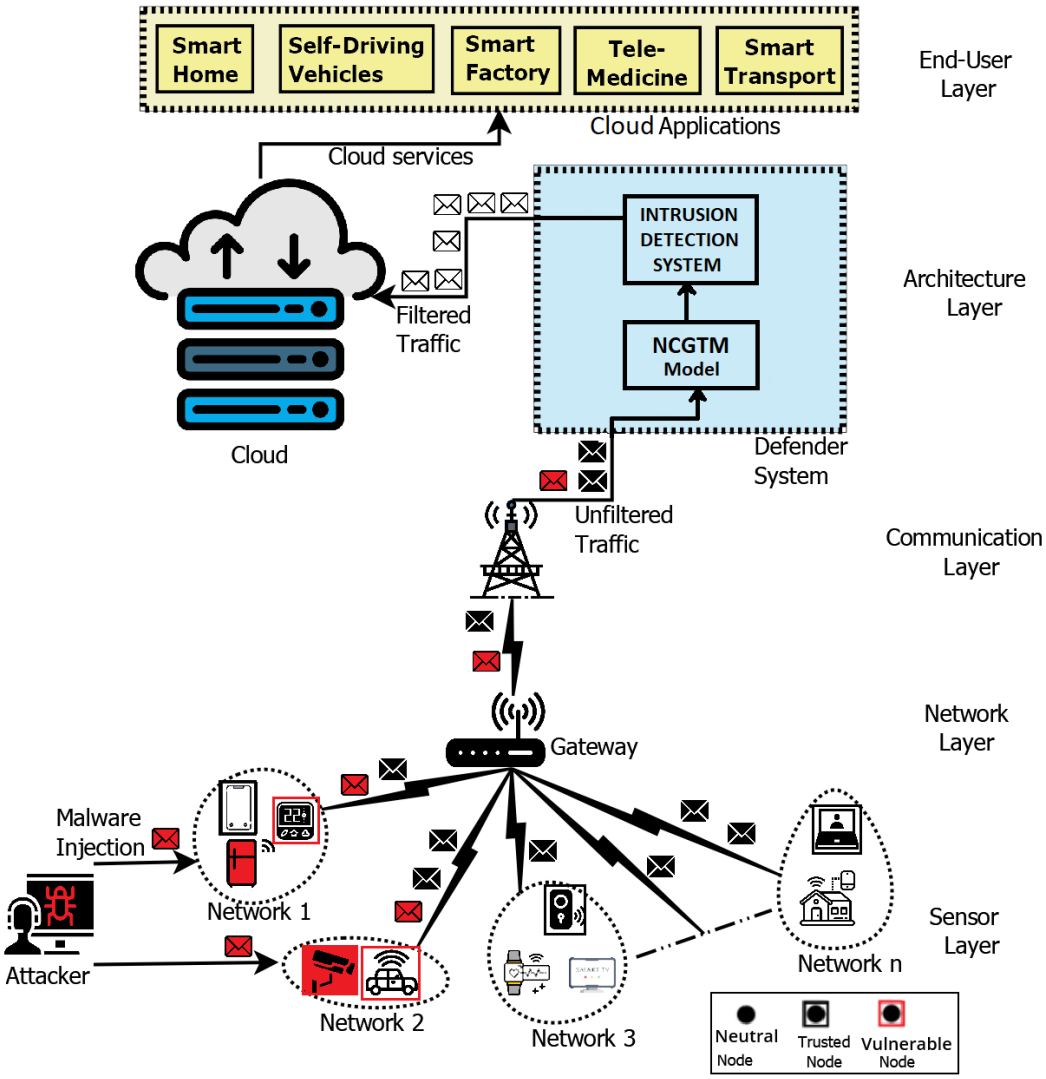


Figure 7.1: IoT bots attacking the Cloud with malware injection

7.1.1 Motivation

In the realm of cloud security, the emergence of the Internet of Things (IoT) has ushered in unparalleled connectivity and convenience, but it has also magnified the challenges of safeguarding our digital landscapes. The ubiquity of IoT devices, ranging from secure installations to public spaces, introduces a complex network teeming with both trusted nodes and potential vulnerabilities. The motivation behind this proposed research is to redefine the boundaries of IDS efficacy, unveiling a novel approach that leverages the power of Game Theory. The introduction of the Non-Cooperative Game-Theoretic Model (NCGTM) is a paradigm shift that addresses the formidable task of filtering the massive traffic cascading from billions of IoT devices. The NCGTM derives its strength from the intrinsic nature of Game Theory and Machine Learning (ML), offering a distinct probability-based approach to calculating outcomes.

7.2 NCGTM Model to assist Hybrid IDS

The IDS must always work efficiently and accurately. IDS uses two modules; signature-based and anomaly-based. These two modules can work independently or be tuned together, thus making the IDS hybrid [197]. To increase the smartness of the IDS, a game-theoretic model, NCGTM, is proposed based on a non-zero non-cooperative game. NCGTM is designed to model the interaction between two players, i.e., attacker and defender. The NCGTM model is shown in Figure 7.2.

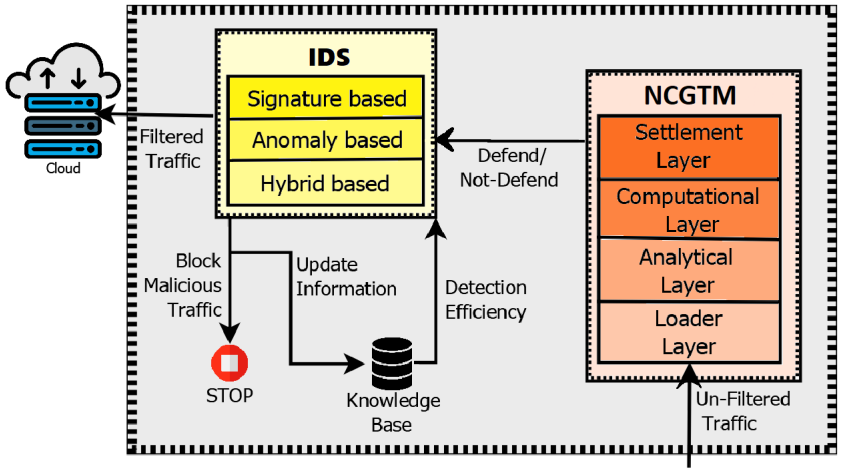


Figure 7.2: NCGTM Model and IDS

The working of the NCGTM model is discussed in the following Section.

7.2.1 Working of NCGTM

The NCGTM module is divided into four layers: Loader *Layer, Analytical Layer, Computational Layer,* and *Settlement Layer* which are discussed below:
Loader Layer: The values of the parameters are initialized and updated at every instance. The description of the parameters used to develop NCGTM is given in Table 7.1. All the cloud assets would be valued according to their possessions. The asset with more valuable information or task will be given more value. If the attacker succeeds in compromising that asset, it will gain a big payoff, and simultaneously the defender will suffer a massive loss if it cannot defend it.

Table 7.1: Parameters used to develop NCGTM

Description	**Symbol**
Resources consumed by defender	θ
Gain for successful defence	δ
Detection rate of signature module	\$
Detection rate of anomaly module	α
Detection rate of hybrid module	β
Value of the asset under attack	ψ
Resources consumed by attacker	π
Gain for successful attack	ω
Degree of the maliciousness	ϵ
Vulnerable nodes	V
Malicious nodes	N

The parameters used in this model are explained as follows. The energy or the resources the defender consumes while monitoring the traffic is θ. If the defender is not working at any instance, this factor will become zero. The detection rate of the signature, anomaly, and hybrid modules, is defined by \$, α, and β receptively. The asset which the defender needs to protect can be a hypervisor, a server, software, or any other helpful entity of the Cloud. The asset's value represented by ψ should always be greater than the resources consumed by the defender and more significant than the resources exhausted on the attacker side. Otherwise, both players will continuously lose the Game. The resources that the

attacker consumes to perform the regular or sophisticated attack are represented with π. If the attacker is not attacking, the value of the π can be 0. The gain for the attacker after the particular attack is given by ω. The ϵ represents the degree of the maliciousness of the packet. After the initialization of parameters, the strategy of both players is delineated in *Analytical Layer*.

Analytical Layer: The strategies of the attacker and defender are delineated in this layer. The strategy set for the attacker can be represented as $A_S = \{A_{S1}, A_{S2}, A_{S3}\}$ and are given in Table 7.2

Table 7.2: Attacker's Strategies to perform attacks

Representation	**Strategy**
A_{S1}	Sophisticated Attack
A_{S2}	Regular Attack.
A_{S3}	Wait for particular time period *(t)*

In Figure 7.3, the attacker attacks the trusted, neutral, and vulnerable nodes using three strategies in Table 7.2. The trusted nodes are very secure, so the best strategy for the attacker is not to attack these nodes. The neutral nodes can be compromised but only with sophisticated attacks. The vulnerable nodes can easily be attacked with regular attacks. Figure 7.3 represents the best strategy for every node. On the other hand, the defender also has three strategies and plays its

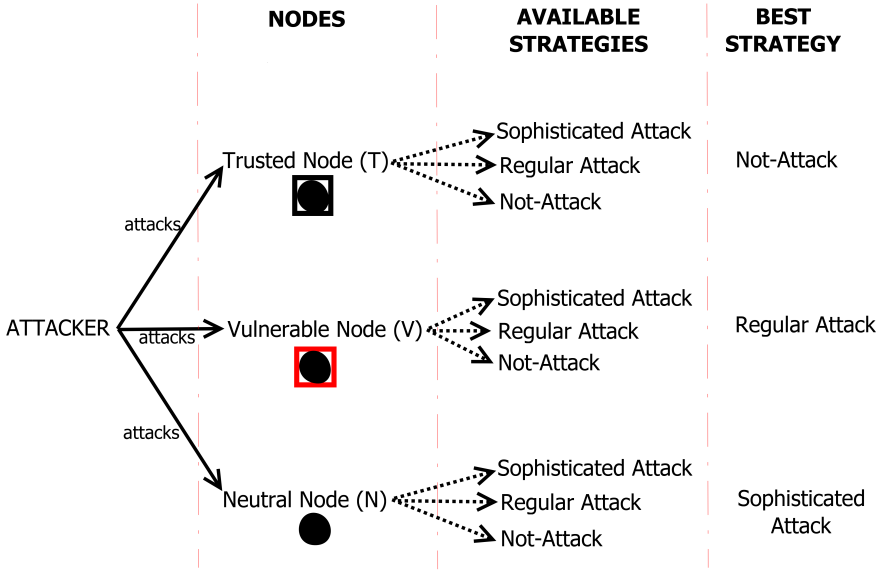


Figure 7.3: Strategies of the attacker in NCGTM Model

best strategy to defend against the attacker's attack. The defender's strategy set can be represented as $D_S = \{D_{S1}, D_{S2}, D_{S3}\}$ and are elaborated in Table 7.3.

Table 7.3: Defender's Strategies to detect attacks

Representation	Strategy
D_{S1}	Monitoring with signature-based IDS
D_{S2}	Monitoring with anomaly-based IDS
D_{S3}	Monitoring with hybrid IDS.

Figure 7.4 represents all the strategies and the corresponding best strategy of the defender. The defender will not monitor the trusted nodes to conserve energy.

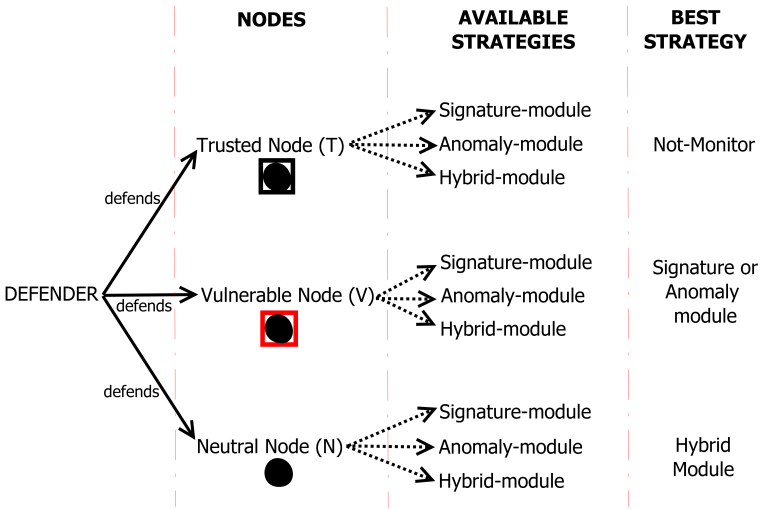


Figure 7.4: Strategies of the defender in NCGTM Model

For vulnerable nodes, the defender can use any of the strategies. However, the hybrid module would be neglected for regular attacks to conserve energy. Only the other two modules will be used. For sophisticated attacks, only the hybrid module will be used. According to the strategies of both players, the payoff matrices are formed in *Computational layer*.

Computational Layer: The attacker and defender's payoff matrices are generated according to the pre-defined strategies in the *Analytical Layer*. The parameter values generated from the *Loader layer* are passed to the general matrices in the second layer. The equations in these matrices are solved with the values. Both the general matrices having equations are converted to the actual number matrices and are passed to the next layer. Eq. 7.1 gives the general payoff matrix with a

defense and attack strategy.

$$X = \begin{bmatrix} (S_{D1}, S_{A1}) & (S_{D1}, S_{A2}) & (S_{D1}, S_{A3}) \\ (S_{D2}, S_{A1}) & (S_{D2}, S_{A2}) & (S_{D2}, S_{A3}) \\ (S_{D3}, S_{A1}) & (S_{D3}, S_{A2}) & (S_{D3}, S_{A3}) \end{bmatrix} \tag{7.1}$$

$$\mathbf{X_A} = \begin{pmatrix} -\pi(t).V & -\pi(t).V + w(t) + \psi & -\pi(t).V \\ -\pi(t).N + w(t) + \psi & -\pi(t).N & -\pi(t).N \\ 0 & 0 & 0 \end{pmatrix} \tag{7.2}$$

$$\mathbf{X_D} = \begin{pmatrix} -\theta(t).\epsilon + \delta(t) + \psi & -\theta(t).\epsilon - \psi & -2.\theta(t).\epsilon + \delta(t) + \psi \\ -\theta(t).\epsilon - \psi & -\theta(t).\epsilon + \delta(t) + \psi & -2.\theta(t).\epsilon + \delta(t) + \psi \\ -\theta(t) & -\theta(t) & -2\theta(t) \end{pmatrix} \tag{7.3}$$

From Matrix X_A and X_D given in Eq. 7.2 and Eq. 7.3, the dominance property is applied to eliminate the unnecessary column. The last row of the Matrix X_A contains the 0 payoffs. If the attacker waits for the attack, there is no gain for him. So, an attacker will continuously attack the nodes to gain maximum profit. Table 7.4 shows the updated new payoff matrix.

Table 7.4: Payoff matrix of defender and attacker

A_S/D_S	**SD**	**AD**	**Hybrid based**
Regular Attack	$-\theta(t).\epsilon + \delta(t) + \psi, -\pi(t).V$	$-\theta(t).\epsilon - \psi, -\pi(t).V + w(t) + \psi$	$-2.\theta(t).\epsilon + \delta(t) + \psi, -\pi(t).V$
Sophisticated Attack	$-\theta(t).\epsilon - \psi, -\pi(t).N + w(t) + \psi$	$-\theta(t).\epsilon + \delta(t) + \psi, -\pi(t).N$	$-2.\theta(t).\epsilon + \delta(t) + \psi, -\pi(t).N$

The *Computational Layer* computes the payoffs of the defender and the attacker with the detection rate of the particular module. Suppose the attacker attacks with a regular attack, and the defender detects with signature-based IDS. Then, the payoff function of the attacker for this scenario can be given by Eq. 7.4.

$$U_{11}(A) = (1 - \$)(\omega(t) + \psi) - \pi(t).V \tag{7.4}$$

Similarly, the payoff function can be represented in Eq. 7.5 for attackers.

$$U_{11}(D) = (\$)(\delta(t) + 2\psi) - \theta(t).\epsilon - \psi \tag{7.5}$$

As explained in Eq. 7.4 and Eq. 7.5, all the attacker and defender payoffs are calculated to complete the payoff matrix. The payoff matrix of the attacker and the defender for all possible scenarios are given by Matrices X'_A and X'_D, respectively.

$$\mathbf{X'_A} = \begin{bmatrix} (1-\$)(\omega(t)+\psi)- & \pi(t).V & \omega(t)+\psi-\pi(t).V & -\pi(t).V \\ \omega(t)+\psi)-\pi(t).N & (1-\alpha)(\omega(t)+\psi)-\pi(t).N & (1-\beta)(\omega(t)+\psi)-\pi(t).N & \end{bmatrix} \tag{7.6}$$

$$\mathbf{X'_D} = \begin{bmatrix} (\$)(\delta(t)+2\psi)- & \theta(t).\epsilon-\psi & -\theta(t).\epsilon-\psi & (\delta(t)-2\theta(t).\epsilon+\psi) \\ -\theta(t).\epsilon-\psi & (\alpha)(\delta(t)+2\psi)-\theta(t).\epsilon-\psi & (\beta)(\delta(t)+2\psi)-2\theta(t).\epsilon-\psi) & \end{bmatrix} \tag{7.7}$$

The detailed working of the whole framework is discussed with the help of an activity diagram in Figure 7.5.

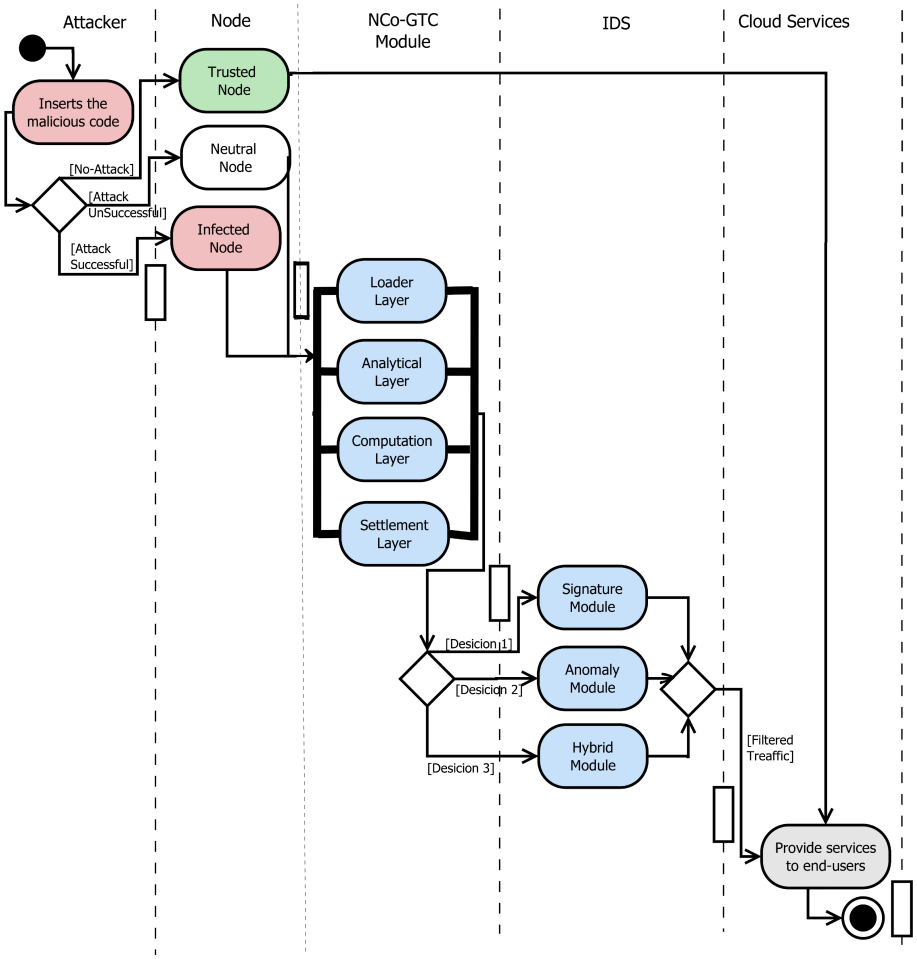


Figure 7.5: Activity diagram of the NCGTM framework

7.3 Analysis of the NCGTM Model

Settlement Layer: At NE, both players will have the best strategies and payoffs. However, there exists no pure strategy Nash Equilibrium (NE). The explanation is that if the attacker attacks the trusted node with a regular or sophisticated attack, it will waste resources as the trusted node can not be compromised. Also, there will be no monitoring for the trusted node by the defender as this node can not send malicious packets. If the attacker tries to attack the neutral node with a regular attack, this strategy will not work because of the security implications on these nodes. Again, the attacker can try to attack the node with a sophisticated attack and win the situation. As a rational player, the defender will defend the node with the hybrid module, as the attack from the regular node would be more sophisticated. The third strategy of the attacker is to attack the vulnerable node with a regular or sophisticated attack. Logically, the attacker will try the regular attack more often, as it will consume less time and resources. Further, regular attacks are easier to perform. Developing source code for a sophisticated attack is more complex and consumes more resources. An attacker will always choose the regular attack over the sophisticated attack for the vulnerable nodes. Conversely, the defender can detect regular attacks more quickly than sophisticated ones. So, choosing the attack's regular strategy more often will not work as the defender will always detect the attack. So, to solve this problem, a Game-theoretic model needs to be analyzed to find the mixed strategy NE for this scenario. The mixed strategy NE is calculated using the graphical method from game theory modeling. Since the Game has no saddle point, the size of the $3X2$ Matrix is needed to reduce to $2X2$ by dominance property. The graphical method is discussed in detail in Chapter 5. The updated matrix consists of three possibilities in the form of three different Cases shown in Table 7.5.

Table 7.5: Different cases and their description

Cases	**defender's Strategies**
Case 1	Signature or Anomaly
Case 2	Anomaly or Hybrid
Case 3	Signature or Hybrid

These different cases are analyzed, and the values of the probabilities of strategies of the defender and attacker are calculated. These cases are described below.

Case 1: Either the defender system chooses to detect the attack with the signature-based method or the anomaly-based method. The probability with signature-based detection is given by x, and the probability for anomaly-based is given by *(1-x)*. The attacker attacks with a regular attack by the probability of q and with a sophisticated attack with the probability of *(1-q)*. The defender payoffs are given by Eq. 7.8.

$$U_{DS} = (x)(q)U_{11}(D) + (x)(1-q)U_{21}(D) + (1-x)(q)[U_{12}(D)] + (1-x)(1-q)[U_{22}(D)] \tag{7.8}$$

The payoff of the attacker is given by Eq. 7.9:

$$U_{AS} = (x)(q)U_{11}(A) + (x)(1-q)U_{21}(A) + (1-x)(q)[U_{12}(A)] + (1-x)(1-q)[U_{22}(A)] \tag{7.9}$$

To maximize the payoff, the partial derivative is done of U_{DS} and U_{AS} *wrt* x and q respectively.

$$\frac{\partial U_{DS}}{\partial x} = xq\$\delta(t) + 2xq\$\psi - xq\theta(t)\epsilon - xq\psi + xq\theta(t)\epsilon + xq\psi + \alpha\delta(t) + 2\alpha\psi - \theta\epsilon - \psi - x\alpha\delta(t) - 2x\alpha\psi - q\alpha\delta(t) - 2\alpha q\psi + xq\alpha\delta(t) + 2xq\alpha\psi \tag{7.10}$$

$$\frac{\partial U_{AS}}{\partial q} = -xq\$\omega(t) - xq\$\psi - q\pi(t)V - \pi(t)N + \omega(t) + \psi - \alpha\omega(t) - \alpha\psi + x\alpha\omega(t) + x\alpha\psi + q\pi(t)N + q\alpha\omega(t) + q\alpha\psi - xq\alpha\omega(t) - xq\alpha\psi \tag{7.11}$$

Solving Eq. 7.10 and 7.11, we get x, *(1-x)*, q and *(1-q)* as follows:

$$x = \frac{\pi(t)V + \pi(t)N + \omega(t) + \psi}{\$\omega(t) + \$\psi + \omega + \psi} \tag{7.12}$$

$$(1-x) = \frac{\$\omega(t) + \$\psi + \pi(t)V + \pi(t)N}{\$\omega(t) + \$\psi + \omega + \psi} \tag{7.13}$$

$$q = \frac{\alpha\delta(t) + 2\alpha\psi}{\$\delta(t) + 2\$\psi + \alpha\delta(t) + 2\alpha\psi} \tag{7.14}$$

$$(1-q) = \frac{\$\delta(t) + 2\$\psi}{\$\delta(t) + 2\$\psi + \alpha\delta(t) + 2\alpha\psi} \tag{7.15}$$

Case 2: The defender system chooses to detect the attack with the anomaly-based method or the hybrid method. The probability of anomaly-based detection

is given by x, and the probability for a hybrid based is given by *(1-x)*. The attacker attacks with a regular attack by the probability of q and with a sophisticated attack with the probability of *(1-q)*. The defender payoffs are given by Eq. 7.16.

$$U_{DS} = (x)(q)U_{12}(D) + (x)(1-q)U_{22}(D) + (1-x)(q)[U_{13}(D)] + (1-x)(1-q)[U_{23}(D)] \tag{7.16}$$

The payoff of the attacker is given by Eq. 7.17:

$$U_{AS} = (x)(q)U_{12}(A) + (x)(1-q)U_{22}(A) + (1-x)(q)[U_{13}(A)] + (1-x)(1-q)[U_{23}(A)] \tag{7.17}$$

To maximize the payoff, the partial derivative is done of U_{DS} and U_{AS} *wrt* x and q respectively.

$$\frac{\partial U_{DS}}{\partial x} = x\alpha\delta(t) + 2x\alpha\psi - xq\alpha\delta(t) - 2xq\alpha\psi + q\theta(t)\epsilon + x\theta(t)\epsilon + \beta\delta(t) + 2x\psi - 2\theta(t)\epsilon - \psi - x\beta\delta(t) - 2x\beta\psi - q\beta\delta(t) - 2\beta q\psi + xq\beta\delta(t) - 2xq\theta(t)\epsilon - 2xq\psi \tag{7.18}$$

$$\frac{\partial U_{AS}}{\partial q} = xq\omega(t) + xq\psi - x\alpha\psi - x\alpha\omega(t) - x\pi(t)N - xqV + \omega(t) + \psi - \beta\omega(t) - \beta\psi - xN + x\beta\omega(t) + x\beta\psi - x\pi(t)N - q\omega(t) - q\psi + q\beta\omega(t) + q\beta\psi + q\pi(t)N - xq\beta\psi - xq\beta\omega(t) \tag{7.19}$$

Solving Eq. 7.18 and 7.19, we get x, *(1-x)*, q and *(1-q)* as follows:

$$x = \frac{\pi(t)(V-N) + (1-\beta)(\omega(t)+\psi)}{(1-\beta)(\omega(t)+\psi)} \tag{7.20}$$

$$(1-x) = \frac{\pi(t)(V+N)}{(1-\beta)(\omega(t)+\psi)} \tag{7.21}$$

$$q = \frac{\alpha\delta(t) + 2\alpha\psi + \theta(t)\epsilon + 2\psi - \beta\delta - 2\beta\psi}{\alpha\delta(t) + 2\alpha\psi - \beta q - 2\beta\psi + 2\theta(t)\epsilon + 2\psi} \tag{7.22}$$

$$(1-q) = \frac{\theta(t)\epsilon - \beta\delta(t)}{\$\delta(t) + 2\$\psi + \alpha\delta(t) + 2\alpha\psi} \tag{7.23}$$

Case 3: The defender system chooses to detect the attack with the signature-based method or the hybrid method. The probability with signature-based detection is given by x, and the probability for a hybrid based is given by *(1-x)*.

The attacker attacks with the regular attack with the probability of q and with a sophisticated attack with the probability of *(1-q)*. The defender payoffs are given by Eq. 7.24.

$$U_{DS} = (x)(q)U_{11}(D) + (x)(1-q)U_{21}(D) + (1-x)(q)[U_{13}(D)] + (1-x)(1-q)[U_{23}(D)] \tag{7.24}$$

The payoff of the attacker is given by Eq. 7.25:

$$U_{AS} = (x)(q)U_{11}(A) + (x)(1-q)U_{21}(A) + (1-x)(q)[U_{13}(A)] + (1-x)(1-q)[U_{23}(A)] \tag{7.25}$$

To maximize the payoff, the partial derivative is done of U_{DS} and U_{AS} *wrt* x and q respectively.

$$\frac{\partial U_{DS}}{\partial x} = \beta\delta(t) + 2\beta\psi - 2\theta(t)\epsilon - \psi - 2xq\beta\psi + x\theta(t)\epsilon - \beta q\delta(t) - 2q\beta\psi + q\theta(t)\epsilon + xq\beta\delta(t) + 2xq\beta\psi - xq\theta(t)\epsilon - xq\psi \tag{7.26}$$

$$\frac{\partial U_{AS}}{\partial q} = xq\omega(t) - 2xq\beta\omega - 2xq\beta\psi - qpi(t)V + \omega(t) + \psi - \beta\omega(t) - \beta\psi - \pi(t)N + x\beta\omega(t) + x\beta\psi - q\omega(t) - q\psi + q\beta\omega(t) + q\beta\psi + q\pi(t)N \tag{7.27}$$

Solving Eq. 7.26 and 7.27, we get x, *(1-x)*, q and *(1-q)* as follows:

$$x = \frac{\pi(t)(V-N) + (1-\beta)(\omega(t)+\psi)}{\omega(t) - 2\beta\omega(t) - 2\beta\psi} \tag{7.28}$$

$$(1-x) = \frac{\pi(t)(V-N) + \psi - \beta(\omega(t)-\psi)}{\omega(t) - 2\beta\omega(t) - 2\beta\psi} \tag{7.29}$$

$$q = \frac{2\beta\psi - \theta(t)\epsilon - \beta\delta(t)}{\beta\delta(t) + 2\beta\psi - \theta(t)\epsilon - \psi} \tag{7.30}$$

$$(1-q) = \frac{\psi}{\psi - \beta\delta(t) - 2\beta\psi + \theta(t)\epsilon} \tag{7.31}$$

The Algorithm of the proposed NCGTM model is presented in Algorithm 7.1. The ISOT Cloud Intrusion Dataset (ISOT-CID) analyzes NCGTM on a real dataset. ISOT-CID represents a real cloud dataset [198][165]. The dataset achieved is in raw form, which needs to be pre-processed for better performance. For pre-processing, normalization using a z-score is done. Further, after pre-processing,

feature selection using FSelector [199] is done to select the most relevant features. FSelector works by computing an information gain for each feature. Once the information gain is computed, a rank is generated for each feature. The features are sorted in ascending order, and the top-ranked features are selected for further processing.

Algorithm 7.1 NCGTM Algorithm

1: Input: $\theta, \delta, \psi, \pi, \omega, VandN$.
2: Calculate: $U_{ij}(D)$ and $U_{ij}(A)$ where $1 \geq i, j \leq 2$.
3: Update the values of parameters in X'_D and X'_A matrices.
4: **if** Pure Strategy NE **then** Calculate NE Start IDS.
5: **end if**
6: **Else** Input: $\$, \alpha, \beta$
7: **if** Dominance property exists **then** Reduce the size of the matrix.
8: **end if**
9: **Else** Initialize y and $(1 - y) = 0$
10: Calculate: $U_{ij}(D)$ and $U_{ij}(A)$
11: Plot the $U_{ij}(D)$ and $U_{ij}(A)$ values
12: Calculate Minimax point
13: Form a $2X2$ matrix from the minimax point.
14: Calculate: y and $(1 - y)$
15: Update: $U_{ij}(D)$ and $U_{ij}(A)$ with k and l.
16: Calculate NE.
17: Start IDS.

After, the pre-processed dataset is divided into 80 percent training data and 20 percent testing data. Stacking is performed on the data to train the dataset. Stacking aims to investigate numerous potential solutions to a problem and tackle a learning problem with various models, each of which can master a specific subset of the problem field. Thus, it is possible to construct several distinct learners, each of which can be used to construct an intermediate prediction. A second model can be added using intermediate predictions to learn the final goal. This research uses three heterogeneous models, Gradient Boost Machine (GBM), Random Forest (RF), and XGBoost, to train the data at the first level. The training of the heterogeneous models is done in parallel. Once the first-level predictions are made, a new model called the meta-model is added to the intermediate predictions for second-level predictions. The deep neural network (DNN) is used as a meta-model. After training with these models, the DR and the FPR of the IDS are calculated,

which are discussed in the next Section.

7.4 Results and Discussions

The results are divided into two parts. In the first part, the payoff of the attacker and defender are calculated and analyzed. The second part tests the IDS on the real-time dataset to find the DR and FPR.

7.4.1 Payoffs of attacker and defender

The analysis of the game model shows a mixed strategy Bayesian Nash Equilibrium (BNE) exists in which strategies depend on probability. A Python code is developed to determine the best strategy for the attacker and defender during the *[Attack, Defend]* scenario. The parameters' values are randomly varied to understand the players' best strategy. Figure 7.6 shows the payoffs of the defender and the attacker on the Y-axis and the time elapsed (in seconds) on the X-axis. This screenshot of the simulation is taken on various time instances (40 sec (Figure 7.6a), 100 sec (Figure 7.6b), 200 sec (Figure 7.6c), and 300 sec (Figure 7.6d)) when the attacker attacks the System. The negative payoff shows the depletion of the resources and the energy used by the defender or the attacker. From Figure 7.6, it is clear that the defender's payoffs are high than the attacker, which means the defender is effectively handling the security attacks. To further validate the model in a real-time scenario, it is tested on ISOT-CID cloud data and is discussed in the next Section.

7.4.2 Detection Rate and False Positive Rate of IDS

Further, the NCGTM is validated on the real-time cloud dataset. The ISOT Cloud Intrusion Dataset (ISOT-CID) has been used to test the NCGTM model. There is a total of $10,42,558$ samples, of which $4,13,484$ are malicious samples, and $6,29,074$ are benign samples. The malicious samples are divided into regular and sophisticated samples with a total of $2,85,453$ and $1,28,022$ samples in each type, respectively. The Table 7.6 shows the complete dataset description. The dataset achieved is in raw form and hence needs to be preprocessed. For preprocessing, z-score normalization scales the data from -1 to 1. There is a total of 75 features in the dataset. Out of the 75 features, only a few relevant features are essential. Therefore, to extract those relevant features, feature extraction is performed using the *FSelector* package [199], which works by calculating the information gain and

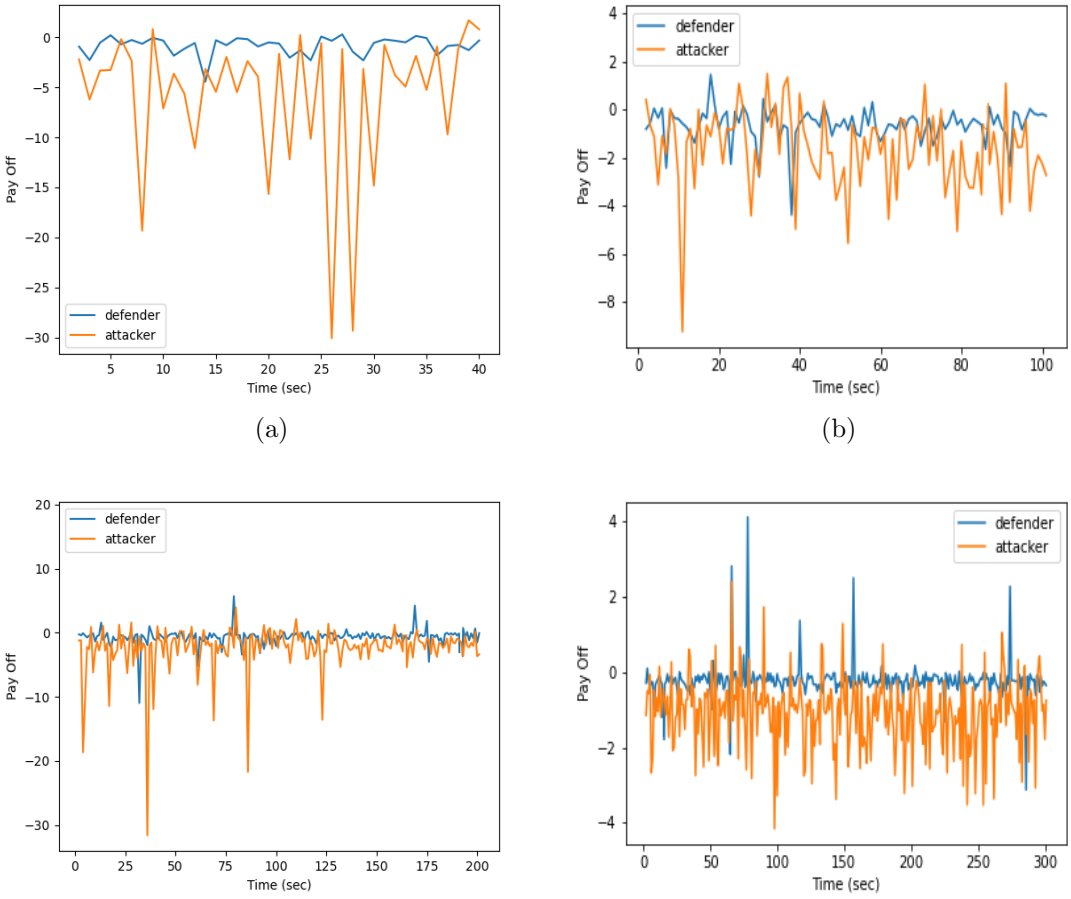


Figure 7.6: Defender's and attacker's payoffs at different time windows

generates a rank for each feature. The top 15 features are selected and passed to stacking for model training and testing. After training, the DR and the FPR of the IDS have been calculated. The DR and FPR values are tested on the NCGTM model. The payoffs of the defender are calculated from the DR and FPR values that are further delineated from the ISOT-CID dataset. The trend in Figure 7.7 clearly shows that the payoff value increases and decreases with the increase and decrease of the DR. In other words, the payoff gained by the defender is directly proportional to the DR of the defender. It shows that the DR and payoff of the NCGTM are highest compared to the K Nearest Neighbor (KNN), Convolutional Neural Network (CNN), C4.5, AS-IDS, IBK, and Hybrid methods. The attacker's payoff is the lowest in the NCGTM model compared to other models. Also, with the addition of the NCGTM, the DR and the payoff of the defender increases gradually. The comparison of the FPR and the payoff of the NCGTM with existing approaches is presented in Figure 7.8. With the NCGTM

Table 7.6: Description of ISOT-CID Dataset

Dataset	Sample Count
Total Samples	10,42,558
Benign Samples	6,29,074
Malicious Samples	4,13,484
Regular Attack Samples	2,85,463
Sophisticated Samples	1,28,022

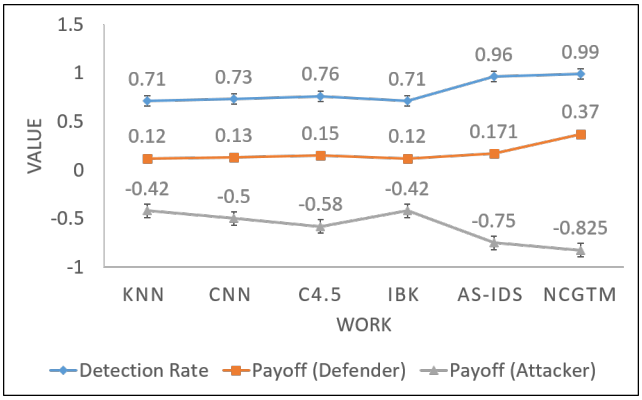


Figure 7.7: Comparison of Detection Rate of NCGTM with existing models

module, the values of the FPR swiftly decrease. This leads to an increase in the payoff of the defender. In NCGTM, the FPR rate is lowest at 0.078.

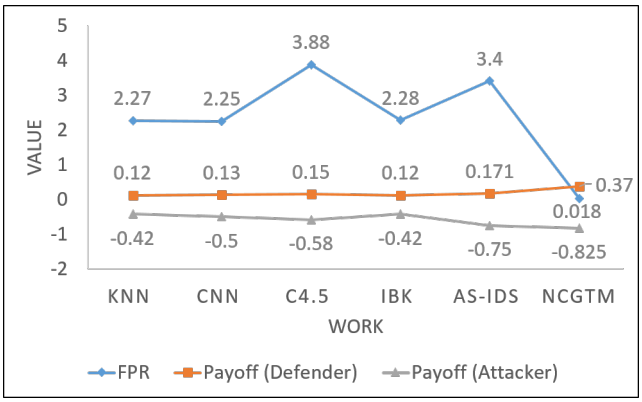


Figure 7.8: Comparison of False Positive Rate of NCGTM with existing models

7.5 Summary

The NCGTM Model is based on non-zero-sum non-cooperative game theory. The players, malicious node and defender, compete against each other to gain max-

imum payoff. The game model is analyzed after specifying the players' payoffs. The analyses show that there exists no pure strategy NE. However, a mixed strategy BNE is reached. The probability by which an attacker should attack and a defender should defend is calculated, and the following observations are made:

- A novel lightweight module based on Game Theory is developed to study the non-cooperative behavior between the attacker and the defender in Cloud Environment.
- The NCGTM is implemented and validated on a real-time cloud data set using an ML stacked ensemble framework comprising RF, GBM, XGBoost, and DNN models.
- With NCGTM, the overall performance of the IDS improves with a higher detection rate of 99% and lower FPR of 0.01.

Chapter 8
Conclusions and Future Work

This Chapter brings out the overall conclusions of the research work carried out in this Thesis. The suggestions regarding future research directions and possible work extensions are also discussed.

8.1 Conclusion

In this Thesis, an attempt is made to solve the security problems faced by Cloud Computing. The following points can be concluded from this research work, and the following novel contributions can be attributed.

1. The study of the existing approaches in the field of security in cloud computing is discussed. The study is divided into three Sections. The first Section analyzes security models for internal and external attacks. The second Section discusses different game models to handle cloud attacks. The third Section discusses Machine Learning (ML) and Deep Learning (DL) techniques to optimize IDS in the cloud environment. (Chapter 2)

2. The Chapter details the methodology to address the research objectives of the Thesis. The Thesis aims to design and develop an intelligent security system for the cloud environment. For this purpose, the methodology is classified into three categories, i.e., requirement specifications, game-theoretic approach, and ML model. The requirement specifications include the exhaustive study of security devices like IDS concerning hardware, software configuration, and their calibration for separating malicious and non-malicious data. The game theory approach includes proposing a model, delineating strategies for both players, and reaching the NE stage. The ML model includes training and testing the game model on a real-time dataset. The dataset used to test and validate the proposed model is discussed in this Chapter.

3. Four Game-Theoretic Models are developed, which assist the Intrusion Detection System (IDS) to detect different security attacks more efficiently. These models are:

- ***GTM-CSec:*** A non-cooperative game-theoretic model (*GTM-CSec*) is developed to tackle external security attacks on the cloud. A game is modeled between the attacker and the defender competing against each other to gain maximum payoffs. The signature-based and anomaly-based modules of IDS and honeypot are used as a defender system. The payoffs of the attacker and defender are analyzed and compared. The results show that with a proper detection module, the payoffs of the defender system come out to be more than the attacker. (Chapter 4)
- ***BOGTA:*** Game Theory and Bayesian Optimized Deep Learning approaches are used to eliminate the shortcomings of the GTM-CSec model. The graphical method of game theory optimizes the decision-making process of the proposed model BOGTA by eliminating the least significant strategy of the IDS. The hyper-parameter tuning with Bayesian optimization further enhances the accuracy and detection rate of the Intrusion Detection System (IDS). Also, it lowers the FPR of the IDS. The results show that BOGTA is more accurate and takes less computation time than GTM-CSec. The overall performance of IDS with BOGTA is better than the existing models. (Chapter 5)
- ***GTA-IDS:*** This model uses information theory and game theory approaches to tackle internal security attacks on the cloud. The information entropy tracks the abnormal flow in the traffic, and it is sent to the game-theoretic model for further analysis. A game is modeled between the malicious node and the defender system. The IDS is used as a defender system and is trained using ML techniques, namely Random Forest (RF), Gradient Boosting Machine (GBM), and Deep Neural Network (DNN). The GTA-IDS is implemented on a benchmark NSL-KDD dataset to check the accuracy, detection rate (DR), and false positive rate (FPR) of the defender system (Chapter 6)
- ***NCGTM:*** A Non-Cooperative Game-Theoretic Model (*NCGTM*) is proposed and developed to enhance the decision-making process of the hybrid Intrusion Detection System (IDS). The attacker's strategies are analyzed against the strategies of the IDS to make better predictions. NCGTM model assists the IDS in determining whether to use the signature, anomaly, or hybrid modules. Different ML models comprising RF, GBM, XGBoost, and DNN are stacked together to train the IDS, which is required to increase the DR and lower the FPR. NCGTM

performs better than the existing approaches by giving better DR and lower FPR. (Chapter 7)

4. The proposed Game models are validated on real-time datasets using ML and Deep Learning (DL) Algorithms. The results show an improvement in the defender's detection rate, accuracy, and payoffs. Also, the FPR is reduced with the proposed models.

The comparison of the *GTM-CSec, GTA-IDS, NCGTM,* and *BOGTA* is presented in Table 8.1.

Table 8.1: Comparison of GTM-CSec, GTA-IDS, NCGTM, and BOGTA

Work	**Work Done**		**Performance**			
	Game Theory	**ML and DL**	**A**	**B**	**C**	**D**
GTM-CSec	NE. Performed in seven cycles.	-	0.80	-	-	-
BOGTA	Improvement of GTM-CSec. Performed in one cycle	Bayesian Optimization with DNN	0.84	0.991	0.011	99.41%
GTA-IDS	BNE.	GBM, RF, and DNN	0.81	0.98	0.06	96.9%
NCGTM	Graphical method.	Stacking with GBM, XGBoost, RF and DNN	0.82	0.99	0.01	98.12%

A-payoff, B-DR, C-FPR, D-Accuracy

8.1.1 Deployment Cost

This Section discusses the actual cost in terms of deployment and run-time overhead due to the application of the proposed techniques in a real testbed deployment. The deployment is done on Amazon EC2, which is a free software for

the first year. This includes 750 hours of Linux and Windows *t2.micro* instances (*t3.micro* for the regions in which *t2.micro* is unavailable), each month for one year.
The run time overhead is calculated in terms of performance monitoring and load testing which is discussed as follows:

1. **Performance Monitoring:** To measure the run time overhead of the proposed techniques, we have computed the CPU utilization on the local host and CPU utilization on the Amazon AWS cloud. The total CPU utilization on the local host is 45%, and on Amazon AWS cloud is 17% which is utilizing almost 28% less as compared to the local host. Moreover, the total number of packets send is $20K - 25K$ on the Amazon AWS cloud.

2. **Load Testing:** The load testing is performed on both the local host and Amazon EC2 cloud, and it has been found that the deployed application of Amazon AWS handles the load efficiently. As there are four modules in the proposed research, they all are tested on different no. of seconds starting from $100s$ to $4000s$ and it has been found that the local host sometimes runs out of time due to the increasing number of seconds. On the other side, Amazon AWS performed efficiently without any server error in the increasing number of seconds.

8.2 Future Work

Research is an iterative and continuous procedure. The work presented in the Thesis focuses on solving security issues in Cloud Computing and IDS. The future scope of this work is as follows:

1. Modifications can be done in parallel by implementing the proposed GTM-CSec, GTA-IDS, and NCGTM algorithms that may improve its performance. A parallel implementation of the algorithms may be designed for a distributed and shared memory architecture.

2. The number of parameters can be increased in game models to cover every detail, which can increase the complexity but can give promising and better results.

3. Instead of developing a general model for all security attacks, a more specific model for a particular attack can be made by choosing only parameters related to that attack.

4. New machine learning and deep learning approaches can be explored for accurate and fast predictions.

References

[1] Ali Sunyaev and Ali Sunyaev. Cloud computing. *Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies*, pages 195–236, 2020. https://doi.org/10.1006/s00454-016-00156-0.

[2] Paolo Bellavista, Antonio Corradi, Luca Foschini, Sabato Luciano, and Michele Solimando. A simulation framework for virtualized resources in cloud data center networks. *IEEE Journal on Selected Areas in Communications*, 37(8):1808–1819, 2019. https://doi.org/10.1109/JSAC.2019.2927066.

[3] Peter Mell and Tim Grance. The nist definition of cloud computing. 2011.

[4] Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*, pages 5–13. Ieee, 2008. https://doi.org/10.1109/HPCC.2008.172.

[5] Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08*, pages 1–10. Ieee, 2008. https://doi.org/10.1109/GCE.2008.4738445.

[6] Jayachander Surbiryala and Chunming Rong. Cloud computing: History and overview. In *2019 IEEE Cloud Summit*, pages 1–7. IEEE, 2019. https://doi.org/10.1109/CloudSummit47114.2019.00007.

[7] Subhankar Dhar. From outsourcing to cloud computing: evolution of it services. *Management research review*, 35(8):664–675, 2012. https://doi.org/10.1108/01409171211247677.

[8] Fang Liu, Jin Tong, Jian Mao, Robert Bohn, John Messina, Lee Badger, and Dawn Leaf. Nist cloud computing reference architecture. *NIST special publication*, 500(2011):292, 2011.

[9] Tinankoria Diaby and Babak Bashari Rad. Cloud computing: a review of the concepts and deployment models. *International Journal of Information Technology and Computer Science*, 9(6):50–58, 2017. https://doi.org/10.5815/ijitcs.2017.06.07.

[10] Mohsen Attaran and Jeremy Woods. Cloud computing technology: improving small business performance using the internet. *Journal of Small*

Business & Entrepreneurship, 31(6):495–519, 2019. https://doi.org/10.1080/08276331.2018.1466850.

[11] Shreshth Tuli, Sukhpal Singh Gill, Minxian Xu, Peter Garraghan, Rami Bahsoon, Schahram Dustdar, Rizos Sakellariou, Omer Rana, Rajkumar Buyya, Giuliano Casale, et al. Hunter: Ai based holistic resource management for sustainable cloud computing. *Journal of Systems and Software*, 184:111124, 2022. https://doi.org/10.1016/j.jss.2021.111124.

[12] Fatiha Houacine. *Service-Oriented Architecture for the Mobile Cloud Computing*. PhD thesis, Paris, CNAM, 2016.

[13] Aaqib Rashid and Amit Chaturvedi. Cloud computing characteristics and services: a brief review. *International Journal of Computer Sciences and Engineering*, 7(2):421–426, 2019. https://doi.org/10.26438/ijcse/v7i2.421426.

[14] Salim Bitam, Abdelhamid Mellouk, and Sherali Zeadally. Vanet-cloud: a generic cloud computing model for vehicular ad hoc networks. *IEEE Wireless Communications*, 22(1):96–102, 2015. https://doi.org/10.1109/MWC.2015.7054724.

[15] Mohammed Mohammed Sadeeq, Nasiba M Abdulkareem, Subhi RM Zeebaree, Dindar Mikaeel Ahmed, Ahmed Saifullah Sami, and Rizgar R Zebari. Iot and cloud computing issues, challenges and opportunities: A review. *Qubahan Academic Journal*, 1(2):1–7, 2021. https://doi.org/10.48161/qaj.v1n2a36.

[16] Wei Cai, Ryan Shea, Chun-Ying Huang, Kuan-Ta Chen, Jiangchuan Liu, Victor CM Leung, and Cheng-Hsin Hsu. A survey on cloud gaming: Future of computer games. *IEEE Access*, 4:7605–7620, 2016. https://doi.org/10.1109/ACCESS.2016.2590500.

[17] Kirandeep Kaur, Arjan Singh, and Anju Sharma. A systematic review on resource provisioning in fog computing. *Transactions on Emerging Telecommunications Technologies*, 34(4):e4731, 2023. https://doi.org/10.1002/ett.4731.

[18] Arokia Paul Rajan et al. Evolution of cloud storage as cloud computing infrastructure service. *Journal of Computer Engineering*, 1(1):38–45, 2013. https://doi.org/10.48550/arXiv.1308.1303.

[19] Muhammad Anshari, Yabit bin Alas, and Lim Sie Guan. Pervasive knowledge, social networks, and cloud computing: e-learning 2.0. *Eurasia Journal of Mathematics, Science and Technology Education*, 11(5):909–921, 2017. https://doi.org/10.12973/eurasia.2015.1360a.

[20] Wenyuan Yang and Yuesheng Zhu. A verifiable semantic searching scheme by optimal matching over encrypted data in public cloud. *IEEE Transactions on Information Forensics and Security*, 16:100–115, 2020. https://doi.org/10.1109/TIFS.2020.3001728.

[21] Srijita Basu, Arjun Bardhan, Koyal Gupta, Payel Saha, Mahasweta Pal, Manjima Bose, Kaushik Basu, Saunak Chaudhury, and Pritika Sarkar. Cloud computing security challenges & solutions-a survey. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 347–356. IEEE, 2018. https://doi.org/10.1109/CCWC.2018.8301700.

[22] Bader Alouffi, Muhammad Hasnain, Abdullah Alharbi, Wael Alosaimi, Hashem Alyami, and Muhammad Ayaz. A systematic literature review on cloud computing security: threats and mitigation strategies. *IEEE Access*, 9:57792–57807, 2021. https://doi.org/10.1109/ACCESS.2021.3073203.

[23] Nalini Subramanian and Andrews Jeyaraj. Recent security challenges in cloud computing. *Computers & Electrical Engineering*, 71:28–42, 2018. https://doi.org/10.1016/j.compeleceng.2018.06.006.

[24] Akram H Shaikh and BB Meshram. Security issues in cloud computing. In *Intelligent Computing and Networking*, volume 146, pages 63–77. Springer, 2021. https://doi.org/10.1007/978-981-15-7421-4_6.

[25] Freeha Khan, Jung Hwan Kim, Lars Mathiassen, and Robin Moore. Data breach management: An integrated risk model. *Information & Management*, 58(1):103392, 2021. https://doi.org/10.1016/j.im.2020.103392.

[26] Gartner. Gartner top security and risk trends. https://www.gartner.com/smarterwithgartner/gartner-top-security-and-risk-trends-for-2021, 2021. Last accessed: 16 September 2021.

[27] Kennedy A Torkura, Muhammad IH Sukmana, Feng Cheng, and Christoph Meinel. Continuous auditing and threat detection in multi-cloud infrastructure. *Computers & Security*, 102:102124, 2021. https://doi.org/10.1016/j.cose.2020.102124.

[28] Linbin Wen. Cloud computing intrusion detection technology based on bp-nn. *Wireless Personal Communications*, 126:1–18, 2021. https://doi.org/10.1007/s11277-021-08569-y.

[29] Gururaj Ramachandra, Mohsin Iftikhar, and Farrukh Aslam Khan. A comprehensive survey on security in cloud computing. *Procedia Computer Science*, 110:465–472, 2017. https://doi.org/10.1016/j.procs.2017.06.

124.

[30] Leila Benarous, Salim Batim, and Abdelhamid Mellouk. *Security in Vehicular Networks: Focus on Location and Identity Privacy*. John Wiley & Sons, 2022.

[31] Hamed Tabrizchi and Marjan Kuchaki Rafsanjani. A survey on security challenges in cloud computing: issues, threats, and solutions. *The journal of supercomputing*, 76(12):9493–9532, 2020. https://doi.org/10.1007/s11227-020-03213-1.

[32] B Thirumaleshwari Devi, S Shitharth, and MA Jabbar. An appraisal over intrusion detection systems in cloud computing security attacks. In *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, pages 722–727. IEEE, 2020. https://doi.org/10.1109/ICIMIA48430.2020.9074924.

[33] Michael Maschler, Shmuel Zamir, and Eilon Solan. *Game theory*. Cambridge University Press, 2020.

[34] Tamer Basar and Georges Zaccour. *Handbook of dynamic game theory*. Springer, 2018. https://doi.org/10.1007/978-3-319-44374-4.

[35] Yuzhao Wu, Yongqiang Lyu, and Yuanchun Shi. Cloud storage security assessment through equilibrium analysis. *Tsinghua Science and Technology*, 24(6):738–749, 2019. https://doi.org/10.26599/TST.2018.9010127.

[36] Erol Akçay. Deconstructing evolutionary game theory: coevolution of social behaviors with their evolutionary setting. *The American Naturalist*, 195(2):315–330, 2020. https://doi.org/10.1086/706811.

[37] Mbazingwa Elirehema Mkiramweni, Chungang Yang, Jiandong Li, and Wei Zhang. A survey of game theory in unmanned aerial vehicles communications. *IEEE Communications Surveys & Tutorials*, 21(4):3386–3416, 2019. https://doi.org/10.1109/COMST.2019.2919613.

[38] Quanyan Zhu and Stefan Rass. Game theory meets network security: A tutorial. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 2163–2165, 2018. https://doi.org/10.1145/3243734.3264421.

[39] Zhiqiang Liu, Bo Xu, Bo Cheng, Xiaomei Hu, and Mehdi Darbandi. Intrusion detection systems in the cloud computing: a comprehensive and deep literature review. *Concurrency and Computation: Practice and Experience*, 34(4):e6646, 2022. https://doi.org/10.1002/cpe.6646.

[40] Shahab Shamshirband, Mahdis Fathi, Anthony T Chronopoulos, Antonio Montieri, Fabio Palumbo, and Antonio Pescapè. Computational intelligence

intrusion detection techniques in mobile cloud computing environments: Review, taxonomy, and open research issues. *Journal of Information Security and Applications*, 55:102582, 2020. https://doi.org/10.1016/j.jisa.2020.102582.

[41] Jason Bell. What is machine learning? *Machine Learning and the City: Applications in Architecture and Urban Design*, pages 207–216, 2022. https://doi.org/10.1002/9781119815075.ch18.

[42] Deepak Kshirsagar and Sandeep Kumar. An ensemble feature reduction method for web-attack detection. *Journal of Discrete Mathematical Sciences and Cryptography*, 23(1):283–291, 2020. https://doi.org/10.1080/09720529.2020.1721861.

[43] Bohdan Pavlyshenko. Using stacking approaches for machine learning models. In *2018 IEEE second international conference on data stream mining & processing (DSMP)*, pages 255–258. IEEE, 2018. https://doi.org/10.1109/DSMP.2018.8478522.

[44] Christian Janiesch, Patrick Zschech, and Kai Heinrich. Machine learning and deep learning. *Electronic Markets*, 31(3):685–695, 2021. https://doi.org/10.1007/s12525-021-00475-2.

[45] Jaime Lynn Speiser, Michael E Miller, Janet Tooze, and Edward Ip. A comparison of random forest variable selection methods for classification prediction modeling. *Expert systems with applications*, 134:93–101, 2019. https://doi.org/10.1016/j.eswa.2019.05.028.

[46] Rebwar M Nabi, Saeed Soran Ab M, and Habibollah Harron. A novel approach for stock price prediction using gradient boosting machine with feature engineering (gbm-wfe). *Kurdistan Journal of Applied Research*, 5(1):28–48, 2020. http://dx.doi.org/10.24017/science.2020.1.3.

[47] Ashima Singh, Arwinder Dhillon, Neeraj Kumar, M Shamim Hossain, Ghulam Muhammad, and Manoj Kumar. ediapredict: An ensemble-based framework for diabetes prediction. *ACM Transactions on Multimidia Computing Communications and Applications*, 17(2s):1–26, 2021. https://doi.org/10.1145/3415155.

[48] Rohit Kumar Gupta, Amit Ranjan, Md Ashraf Moid, and Rajiv Misra. Deep-learning based mobile-traffic forecasting for resource utilization in 5g network slicing. In *Internet of Things and Connected Technologies: Conference Proceedings on 5th International Conference on Internet of Things and Connected Technologies (ICIoTCT), 2020*, pages 410–424. Springer, 2021. https://doi.org/10.1007/978-3-030-76736-5_38.

[49] Samaneh Mahdavifar and Ali A Ghorbani. Application of deep learning to cybersecurity: A survey. *Neurocomputing*, 347:149–176, 2019. https://doi.org/10.1016/j.neucom.2019.02.056.

[50] Chetan L Srinidhi, Ozan Ciga, and Anne L Martel. Deep neural network models for computational histopathology: A survey. *Medical Image Analysis*, 67:101813, 2021. https://doi.org/10.1016/j.media.2020.101813.

[51] Mahmoud Abbasi, Amin Shahraki, and Amir Taherkordi. Deep learning for network traffic monitoring and analysis (ntma): A survey. *Computer Communications*, 170:19–41, 2021. https://doi.org/10.1016/j.comcom.2021.01.021.

[52] Puneet Kumar, Shalini Batra, and Balasubramanian Raman. Deep neural network hyper-parameter tuning through twofold genetic approach. *Soft Computing*, 25(13):8747–8771, 2021. https://doi.org/10.1007/s00500-021-05770-w.

[53] Daniel Mesafint Belete and Manjaiah D Huchaiah. Grid search in hyperparameter optimization of machine learning models for prediction of hiv/aids test results. *International Journal of Computers and Applications*, 44(2):875,886, 2021. https://doi.org/10.1080/1206212X.2021.1974663.

[54] Annibale Panichella. A systematic comparison of search-based approaches for lda hyperparameter tuning. *Information and Software Technology*, 130:106411, 2021. https://doi.org/10.1016/j.infsof.2020.106411.

[55] Ravpreet Kaur and Sarbjeet Singh. A comprehensive review of object detection with deep learning. *Digital Signal Processing*, 132:103812, 2022. https://doi.org/10.1016/j.dsp.2022.103812.

[56] Tsu-Yang Wu, Qian Meng, Saru Kumari, and Peng Zhang. Rotating behind security: A lightweight authentication protocol based on iot-enabled cloud computing environments. *Sensors*, 22(10):3858, 2022. https://doi.org/10.3390/s22103858.

[57] MM Kamruzzaman. 6g wireless communication assisted security management using cloud edge computing. *Expert Systems*, 40:e13061, 2022. https://doi.org/10.1111/exsy.13061.

[58] Munwar Ali, Low Tang Jung, Ali Hassan Sodhro, Asif Ali Laghari, Samir Birahim Belhaouari, and Zeeshan Gillani. A confidentiality-based data classification-as-a-service (c2aas) for cloud security. *Alexandria Engineering Journal*, 64:749–760, 2023. https://doi.org/10.1016/j.aej.2022.10.056.

[59] Qian He and Hong He. A novel method to enhance sustainable systems

security in cloud computing based on the combination of encryption and data mining. *Sustainability*, 13(1):101, 2020. https://doi.org/10.3390/su13010101.

[60] P Abirami and S Vijay Bhanu. Enhancing cloud security using crypto-deep neural network for privacy preservation in trusted environment. *Soft Computing*, 24:18927–18936, 2020. https://doi.org/10.1007/s00500-020-05122-0.

[61] Jin He, Kaoru Ota, Mianxiong Dong, Laurence T Yang, Mingyu Fan, Guangwei Wang, and Stephen S Yau. Customized network security for cloud service. *IEEE Transactions on Services Computing*, 13(5):801–814, 2017. https://doi.org/10.1109/TSC.2017.2725828.

[62] Li Ding, Zhongsheng Wang, Xiaodong Wang, and Dong Wu. Security information transmission algorithms for iot based on cloud computing. *Computer Communications*, 155:32–39, 2020. https://doi.org/10.1016/j.comcom.2020.03.010.

[63] Gangasandra Mahadevaiah Kiran and Narasimhaiah Nalini. Enhanced security-aware technique and ontology data access control in cloud computing. *International Journal of Communication Systems*, 33(15):e4554, 2020. https://doi.org/10.1002/dac.4554.

[64] Rajendra Patil, Harsha Dudeja, and Chirag Modi. Designing an efficient security framework for detecting intrusions in virtual network of cloud computing. *Computers & Security*, 85:402–422, 2019. https://doi.org/10.1016/j.cose.2019.05.016.

[65] Tian Wang, Yuzhu Liang, Yujie Tian, Md Zakirul Alam Bhuiyan, Anfeng Liu, and A Taufiq Asyhari. Solving coupling security problem for sustainable sensor-cloud systems based on fog computing. *IEEE Transactions on Sustainable Computing*, 6(1):43–53, 2019. https://doi.org/10.1109/TSUSC.2019.2904651.

[66] Christos Stergiou, Kostas E Psannis, Byung-Gyu Kim, and Brij Gupta. Secure integration of iot and cloud computing. *Future Generation Computer Systems*, 78:964–975, 2018. https://doi.org/10.1016/j.future.2016.11.031.

[67] K Loheswaran and J Premalatha. Renaissance system model improving security and third party auditing in cloud computing. *Wireless Personal Communications*, 90(2):1051–1066, 2016. https://doi.org/10.1007/s11277-016-3296-7.

[68] Bruno Guazzelli Batista, Carlos Henrique Gomes Ferreira, Danilo

Costa Marim Segura, Dionisio Machado Leite Filho, and Maycon Leone Maciel Peixoto. A qos-driven approach for cloud computing addressing attributes of performance and security. *Future Generation Computer Systems*, 68:260–274, 2017. https://doi.org/10.1016/j.future.2016.09.018.

[69] Feda AlShahwan, Maha Faisal, and Godwin Ansa. Security framework for restful mobile cloud computing web services. *Journal of Ambient Intelligence and Humanized Computing*, 7(5):649–659, 2016. https://doi.org/10.1007/s12652-015-0308-5.

[70] Jinan Shen, Deqing Zou, Hai Jin, Bin Yuan, and Weiqi Dai. A domain-divided configurable security model for cloud computing-based telecommunication services. *The Journal of Supercomputing*, 75(1):109–122, 2019. https://doi.org/10.1007/s11227-015-1587-5.

[71] Piotr Nawrocki, Jakub Pajor, Bartlomiej Sniezynski, and Joanna Kolodziej. Modeling adaptive security-aware task allocation in mobile cloud computing. *Simulation Modelling Practice and Theory*, 116:102491, 2022. https://doi.org/10.1016/j.simpat.2022.102491.

[72] Seongmo An, Asher Leung, Jin B Hong, Taehoon Eom, and Jong Sou Park. Toward automated security analysis and enforcement for cloud computing using graphical models for security. *IEEE Access*, 10:75117–75134, 2022. https://doi.org/10.1109/ACCESS.2022.3190545.

[73] Jun-Sik Shin and Jongwon Kim. Smartx multi-sec: A visibility-centric multi-tiered security framework for multi-site cloud-native edge clusters. *IEEE Access*, 9:134208–134222, 2021. https://doi.org/10.1109/ACCESS.2021.3115523.

[74] Danish Ahamad, Shabi Alam Hameed, and Mobin Akhtar. A multi-objective privacy preservation model for cloud security using hybrid jaya-based shark smell optimization. *Journal of King Saud University-Computer and Information Sciences*, 34(6):2343–2358, 2022. https://doi.org/10.1016/j.jksuci.2020.10.015.

[75] Chenquan Gan, Qingdong Feng, Xulong Zhang, Zufan Zhang, and Qingyi Zhu. Dynamical Propagation Model of Malware for Cloud Computing Security. *IEEE Access*, 8:20325–20333, 2020. https://doi.org/10.1109/ACCESS.2020.2968916.

[76] Haralambos Mouratidis, Shaun Shei, and Aidan Delaney. A security requirements modelling language for cloud computing environments. *Software and Systems Modeling*, 19(2):271–295, 2020. https://doi.org/10.1007/s10270-019-00747-8.

[77] Muhammad Imran Tariq. Agent based information security framework for hybrid cloud computing. *KSII Transactions on Internet and Information Systems (TIIS)*, 13(1):406–434, 2019. http://doi.org/10.3837/tiis.2019.01.023.

[78] Syed Rizvi, Jungwoo Ryoo, John Kissell, William Aiken, and Yuhong Liu. A security evaluation framework for cloud security auditing. *The Journal of Supercomputing*, 74(11):5774–5796, 2018. https://doi.org/10.1007/s11227-017-2055-1.

[79] Dan Gonzales, Jeremy M Kaplan, Evan Saltzman, Zev Winkelman, and Dulani Woods. Cloud-trust—a security assessment model for infrastructure as a service (iaas) clouds. *IEEE Transactions on Cloud Computing*, 5(3):523–536, 2015. https://doi.org/10.1109/TCC.2015.2415794.

[80] Laurence T Yang, Gaoyuan Huang, Jun Feng, and Li Xu. Parallel gnfs algorithm integrated with parallel block wiedemann algorithm for rsa security in cloud computing. *Information Sciences*, 387:254–265, 2017. https://doi.org/10.1016/j.ins.2016.10.017.

[81] Minhaj Ahmad Khan. A survey of security issues for cloud computing. *Journal of network and computer applications*, 71:11–29, 2016. https://doi.org/10.1016/j.jnca.2016.05.010.

[82] Murat Yesilyurt and Yildiray Yalman. New approach for ensuring cloud computing security: using data hiding methods. *Sādhanā*, 41(11):1289–1298, 2016. https://doi.org/10.1007/s12046-016-0558-8.

[83] Jin He, Mianxiong Dong, Kaoru Ota, Minyu Fan, and Guangwei Wang. Netseccc: A scalable and fault-tolerant architecture for cloud computing security. *Peer-to-Peer Networking and Applications*, 9(1):67–81, 2016. https://doi.org/10.1007/s12083-014-0314-y.

[84] Łukasz Gaża and Agnieszka Jakóbik. Epistemic games with conditional believes for modelling security threats defence in cloud computing systems. 2022.

[85] Tengchao Ma, Changqiao Xu, Zan Zhou, Xiaohui Kuang, Lujie Zhong, and Luigi Alfredo Grieco. Intelligent-driven adapting defense against the client-side dns cache poisoning in the cloud. In *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pages 1–6. IEEE, 2020. https://doi.org/10.1109/GLOBECOM42002.2020.9322430.

[86] Qimeng Li, Pengpeng Lv, Mengqiang Wang, Zhizhi Zhang, Shaoyin Wang, Pengbo Fang, and Lifang Gao. A risk assessment method of smart grid in cloud computing environment based on game theory. In *2020 IEEE*

5th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA), pages 67–72. IEEE, 2020. https://doi.org/10.1109/ICCCBDA49378.2020.9095625.

[87] Libingyi Huang, Guoqing Jia, Weidong Fang, Wei Chen, and Wuxiong Zhang. Towards security joint trust and game theory for maximizing utility: Challenges and countermeasures. *Sensors*, 20(1):221, 2019. https://doi.org/10.3390/s20010221.

[88] Amandeep Singh Sohal, Rajinder Sandhu, Sandeep K Sood, and Victor Chang. A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments. *Computers & Security*, 74:340–354, 2018. https://doi.org/10.1016/j.cose.2017.08.016.

[89] Junjie Lv and Juling Rong. Virtualisation security risk assessment for enterprise cloud services based on stochastic game nets model. *IET Information Security*, 12(1):7–14, 2018. https://doi.org/10.1049/iet-ifs.2017.0038.

[90] Guisheng Fan, Liqiong Chen, and Huiqun Yu. A game theoretic method to model and analyze attack-defense strategy of resource service in cloud application. *Concurrency and Computation: Practice and Experience*, 29(12):e4131, 2017. https://doi.org/10.1002/cpe.4131.

[91] Priti Narwal, Shailendra N Singh, and Deepak Kumar. Game-theory based detection and prevention of dos attacks on networking node in open stack private cloud. In *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions)(ICTUS)*, pages 481–486. IEEE, 2017. https://doi.org/10.1109/ICTUS.2017.8286057.

[92] Jianhua Liu, Shigen Shen, Guangxue Yue, Risheng Han, and Hongjie Li. A stochastic evolutionary coalition game model of secure and dependable virtual service in sensor-cloud. *Applied Soft Computing*, 30:123–135, 2015. https://doi.org/10.1016/j.asoc.2015.01.038.

[93] Hengwei Zhang, Yan Mi, Xiaohu Liu, Yuchen Zhang, Jindong Wang, and Jinglei Tan. A differential game approach for real-time security defense decision in scale-free networks. *Computer Networks*, 224:109635, 2023. https://doi.org/10.1016/j.comnet.2023.109635.

[94] Zhi Li, Haitao Xu, and Yanzhu Liu. A differential game model of intrusion detection system in cloud computing. *International Journal of Distributed Sensor Networks*, 13(1):1550147716687995, 2017. https://doi.org/10.1016/j.asoc.2015.01.038.

[95] Zhi-mi Cheng. A differential game model between intrusion detection

system and attackers for wireless sensor networks. *Wireless Personal Communications*, 90(3):1211–1219, 2016. https://doi.org/10.1007/s11277-016-3386-6.

[96] Yi Han, Tansu Alpcan, Jeffrey Chan, Christopher Leckie, and Benjamin IP Rubinstein. A game theoretical approach to defend against co-resident attacks in cloud computing: Preventing co-residence using semi-supervised learning. *IEEE Transactions on information Forensics and Security*, 11(3):556–570, 2015. https://doi.org/10.1109/TIFS.2015.2505680.

[97] Kai Li, Dawei Yang, Liang Bai, and Tianjun Wang. Security risk assessment method of edge computing container based on dynamic game. In *2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*, pages 195–199. IEEE, 2021. https://doi.org/10.1109/ICCCBDA51879.2021.9442570.

[98] Kaho Wan and Joel Coffman. Game-theoretic modeling of ddos attacks in cloud computing. In *Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing*, pages 1–10, 2021. https://doi.org/10.1145/3468737.3494093.

[99] Dongao Zhang, Ziyan Cheng, Peijia Zheng, Lin Chen, and Weiqi Luo. Privacy-preserving outsourced nash equilibrium computation in cloud computing. In *Advances in Artificial Intelligence and Security: 7th International Conference, ICAIS 2021, Dublin, Ireland, July 19-23, 2021, Proceedings, Part II 7*, pages 580–592. Springer, 2021. https://doi.org/10.1007/978-3-030-78618-2_48.

[100] Yan Sun, Yaobing Li, Xuehong Chen, and Jun Li. Optimal defense strategy model based on differential game in edge computing. *Journal of Intelligent & Fuzzy Systems*, 39(2):1449–1459, 2020. https://doi.org/10.3233/JIFS-179919.

[101] Qianmu Li, Jun Hou, Shunmei Meng, and Huaqiu Long. Glide: a game theory and data-driven mimicking linkage intrusion detection for edge computing networks. *Complexity*, 2020:1–18, 2020. https://doi.org/10.1155/2020/7136160.

[102] Poria Pirozmand, Mohsen Angoraj Ghafary, Safieh Siadat, and Jiankang Ren. Intrusion detection into cloud-fog-based iot networks using game theory. *Wireless Communications and Mobile Computing*, 2020, 2020. https://doi.org/10.1155/2020/8819545.

[103] Kashish Prabhakar, Kaushik Dutta, Rachana Jain, Mayank Sharma, and Sunil Kumar Khatri. Securing virtual machines on cloud through game the-

ory approach. In *2019 Amity International Conference on Artificial Intelligence (AICAI)*, pages 859–863. IEEE, 2019. https://doi.org/10.1109/AICAI.2019.8701229.

[104] Sane Bernard Ousmane, Babou Cheikh Saliou Mbacke, and Niang Ibrahima. A game theoretic approach for virtual machine allocation security in cloud computing. In *Proceedings of the 2Nd International Conference on Networking, Information Systems & Security*, pages 1–6, 2019. https://doi.org/10.1145/3320326.3320379.

[105] Soodeh Hosseini and Ramin Vakili. Game theory approach for detecting vulnerable data centers in cloud computing network. *International Journal of Communication Systems*, 32(8):e3938, 2019. https://doi.org/10.1002/dac.3938.

[106] Hao Wu and Wei Wang. A game theory based collaborative security detection method for internet of things systems. *IEEE Transactions on Information Forensics and Security*, 13(6):1432–1445, 2018. https://doi.org/10.1109/TIFS.2018.2790382.

[107] Amin Nezarat. A game theoretic method for vm-to-hypervisor attacks detection in cloud environment. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 1127–1132. IEEE, 2017. https://doi.org/10.1109/CCGRID.2017.138.

[108] Fahad Zaman Chowdhury, Mohd Yamani Idna Idris, Laiha Mat Kiah, and MA Manazir Ahsan. Edos eye: A game theoretic approach to mitigate economic denial of sustainability attack in cloud computing. In *2017 IEEE 8th Control and System Graduate Research Colloquium (ICSGRC)*, pages 164–169. IEEE, 2017. https://doi.org/10.1109/ICSGRC.2017.8070588.

[109] Guisheng Fan, Liqiong Chen, and Huiqun Yu. A game theoretic method to model and analyze attack-defense strategy of resource service in cloud application. *Concurrency and Computation: Practice and Experience*, 29(12), 2017. https://doi.org/10.1002/cpe.4131.

[110] Xueqin Liang, Zheng Yan, Robert H Deng, and Qinghua Zheng. Investigating the adoption of hybrid encrypted cloud data deduplication with game theory. *IEEE Transactions on Parallel and Distributed Systems*, 32(3):587–600, 2020. https://doi.org/10.1109/TPDS.2020.3028685.

[111] Omar Abdel Wahab, Jamal Bentahar, Hadi Otrok, and Azzam Mourad. Resource-aware detection and defense system against multi-type attacks in the cloud: Repeated bayesian stackelberg game. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2019. https://doi.org/10.

1109/TPDS.2020.3028685.

[112] El Mehdi Kandoussi, Mohamed Hanini, Iman El Mir, and Abdelkrim Haqiq. Toward an integrated dynamic defense system for strategic detecting attacks in cloud networks using stochastic game. *Telecommunication Systems*, 73(3):397–417, 2020. https://doi.org/10.1007/s11235-019-00616-1.

[113] Agnieszka Jakóbik. Stackelberg game modeling of cloud security defending strategy in the case of information leaks and corruption. *Simulation Modelling Practice and Theory*, 103:102071, 2020. https://doi.org/10.1016/j.simpat.2020.102071.

[114] Agnieszka Jakóbik, Francesco Palmieri, and Joanna Kołodziej. Stackelberg games for modeling defense scenarios against cloud security threats. *Journal of network and computer applications*, 110:99–107, 2018. https://doi.org/10.1016/j.jnca.2018.02.015.

[115] E Balamurugan, Abolfazl Mehbodniya, Elham Kariri, Kusum Yadav, Anil Kumar, and Mohd Anul Haq. Network optimization using defender system in cloud computing security based intrusion detection system withgame theory deep neural network (idsgt-dnn). *Pattern Recognition Letters*, 156:142–151, 2022. https://doi.org/10.1016/j.patrec.2022.02.013.

[116] Qiang He, Cheng Wang, Guangming Cui, Bo Li, Rui Zhou, Qingguo Zhou, Yang Xiang, Hai Jin, and Yun Yang. A game-theoretical approach for mitigating edge ddos attack. *IEEE Transactions on Dependable and Secure Computing*, 2021. https://doi.org/10.1109/TDSC.2021.3055559.

[117] TP Anithaashri, A Benjamin Joseph, and G Ravichandran. Enhancing the cloud security using novel dominant game strategy. In *2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 1507–1514. IEEE, 2021. https://doi.org/10.1109/ICIRCA51532.2021.9544645.

[118] Banavath Balaji Naik, Dhananjay Singh, and Arun Barun Samaddar. Secure virtual machine allocation against attacks using support value based game policy. *International Journal of Communication Systems*, 34(2):e4299, 2021. https://doi.org/10.1002/dac.4299.

[119] Pan Jun Sun. Research on the optimization management of cloud privacy strategy based on evolution game. *Security and Communication Networks*, 2020, 2020. https://doi.org/10.1155/2020/6515328.

[120] Mina Emami Khansari and Saeed Sharifian. A modified water cycle evolutionary game theory algorithm to utilize qos for iot services in cloud-assisted fog computing environments. *The Journal of Supercomputing*, 76(7):5578–

5608, 2020. https://doi.org/10.1007/s11227-019-03095-y.

[121] Fanyu Kong, Yufeng Zhou, Bin Xia, Li Pan, and Limin Zhu. A security reputation model for iot health data using s-alexnet and dynamic game theory in cloud computing environment. *IEEE Access*, 7:161822–161830, 2019. https://doi.org/10.1109/ACCESS.2019.2950731.

[122] Talal Halabi and Martine Bellaiche. Towards security-based formation of cloud federations: A game theoretical approach. *IEEE transactions on cloud computing*, 8(3):928–942, 2018. https://doi.org/10.1109/TCC.2018.2820715.

[123] Barış Bülent Kırlar, Serap Ergün, Sırma Zeynep Alparslan Gök, and Gerhard-Wilhelm Weber. A game-theoretical and cryptographical approach to crypto-cloud computing and its economical and financial aspects. *Annals of Operations Research*, 260(1):217–231, 2018. https://doi.org/10.1007/s10479-016-2139-y.

[124] Zubair Md Fadlullah, Chao Wei, Zhiguo Shi, and Nei Kato. Gt-qosec: A game-theoretic joint optimization of qos and security for differentiated services in next generation heterogeneous networks. *IEEE Transactions on Wireless Communications*, 16(2):1037–1050, 2017. https://doi.org/10.1109/TWC.2016.2636186.

[125] Nageswara SV Rao, Stephen W Poole, Chris YT Ma, Fei He, Jun Zhuang, and David KY Yau. Defense of cyber infrastructures against cyber-physical attacks using game-theoretic models. *Risk Analysis*, 36(4):694–710, 2016. https://doi.org/10.1111/risa.12362.

[126] Quang Duy La, Tony QS Quek, Jemin Lee, Shi Jin, and Hongbo Zhu. Deceptive attack and defense game in honeypot-enabled networks for the internet of things. *IEEE Internet of Things Journal*, 3(6):1025–1035, 2016. https://doi.org/10.1109/JIOT.2016.2547994.

[127] Kun Wang, Miao Du, Dejun Yang, Chunsheng Zhu, Jian Shen, and Yan Zhang. Game-theory-based active defense for intrusion detection in cyber-physical embedded systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 16(1):18, 2016. https://doi.org/10.1145/2886100.

[128] Md Alamin Talukder, Khondokar Fida Hasan, Md Manowarul Islam, Md Ashraf Uddin, Arnisha Akhter, Mohammand Abu Yousuf, Fares Alharbi, and Mohammad Ali Moni. A dependable hybrid machine learning model for network intrusion detection. *Journal of Information Security and Applications*, 72:103405, 2023. https://doi.org/10.1016/j.jisa.2022.103405.

[129] Sivamohan Krishnaveni, Sivanandam Sivamohan, Subramanian Sridhar, and

Subramani Prabhakaran. Network intrusion detection based on ensemble classification and feature selection method for cloud computing. *Concurrency and Computation: Practice and Experience*, 34(11):e6838, 2022. https://doi.org/10.1002/cpe.6838.

[130] Sumathi Sokkalingam and Rajesh Ramakrishnan. An intelligent intrusion detection system for distributed denial of service attacks: A support vector machine with hybrid optimization algorithm based approach. *Concurrency and Computation: Practice and Experience*, 34(27):e7334, 2022. https://doi.org/10.1002/cpe.7334.

[131] Zouhair Chiba, Moulay Seddiq El Kasmi Alaoui, Noreddine Abghour, and Khalid Moussaid. Automatic building of a powerful ids for the cloud based on deep neural network by using a novel combination of simulated annealing algorithm and improved self-adaptive genetic algorithm. *International Journal of Communication Networks and Information Security*, 14(1):93–117, 2022. http://dx.doi.org/10.17762/ijcnis.v14i1.5264.

[132] Abbasgholi Pashaei, Mohammad Esmaeil Akbari, Mina Zolfy Lighvan, and Asghar Charmin. Early intrusion detection system using honeypot for industrial control networks. *Results in Engineering*, page 100576, 2022. https://doi.org/10.1016/j.comcom.2021.01.021.

[133] Ammar Aldallal and Faisal Alisa. Effective intrusion detection system to secure data in cloud using machine learning. *Symmetry*, 13(12):2306, 2021. https://doi.org/10.3390/sym13122306.

[134] Gopal Singh Kushwah and Virender Ranga. Optimized extreme learning machine for detecting ddos attacks in cloud computing. *Computers & Security*, 105:102260, 2021. https://doi.org/10.1016/j.cose.2021.102260.

[135] Zina Chkirbene, Aiman Erbad, Ridha Hamila, Ala Gouissem, Amr Mohamed, and Mounir Hamdi. Machine learning based cloud computing anomalies detection. *IEEE Network*, 34(6):178–183, 2020. https://doi.org/10.1109/MNET.011.2000097.

[136] Ishu Gupta, Rishabh Gupta, Ashutosh Kumar Singh, and Rajkumar Buyya. Mlpam: A machine learning and probabilistic analysis based model for preserving security and privacy in cloud environment. *IEEE Systems Journal*, 15(3):4248–4259, 2020. https://doi.org/10.1109/JSYST.2020.3035666.

[137] Saurabh Dey, Qiang Ye, and Srinivas Sampalli. A machine learning based intrusion detection scheme for data fusion in mobile clouds involving heterogeneous client networks. *Information Fusion*, 49:205–215, 2019. https://doi.org/10.1016/j.inffus.2019.01.002.

[138] Xueluan Gong, Qian Wang, Yanjiao Chen, Wang Yang, and Xinchang Jiang. Model extraction attacks and defenses on cloud-based machine learning models. *IEEE Communications Magazine*, 58(12):83–89, 2020. https://doi.org/10.1109/MCOM.001.2000196.

[139] C Kavitha, Thippa Reddy Gadekallu, Balasubramanian Prabhu Kavin, Wen-Cheng Lai, et al. Filter-based ensemble feature selection and deep learning model for intrusion detection in cloud computing. *Electronics*, 12(3):556, 2023. https://doi.org/10.3390/electronics12030556.

[140] Bishwajeet Kumar Pandey, MRM Veeramanickam, Shabeer Ahmad, Ciro Rodriguez, and Doris Esenarro. Expssoa-deep maxout: Exponential shuffled shepherd optimization based deep maxout network for intrusion detection using big data in cloud computing framework. *Computers & Security*, 124:102975, 2023. https://doi.org/10.1016/j.cose.2022.102975.

[141] Loheswaran Karuppusamy, Jayavadivel Ravi, Murali Dabbu, and Srinivasan Lakshmanan. Chronological salp swarm algorithm based deep belief network for intrusion detection in cloud using fuzzy entropy. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, 35(1):e2948, 2022. https://doi.org/10.1002/jnm.2948.

[142] Andrea Sharon, Prarthna Mohanraj, Tanya Elizabeth Abraham, Bose Sundan, and Anitha Thangasamy. An intelligent intrusion detection system using hybrid deep learning approaches in cloud environment. In *Computer, Communication, and Signal Processing: 6th IFIP TC 5 International Conference, ICCCSP 2022, Chennai, India, February 24–25, 2022, Revised Selected Papers*, pages 281–298. Springer, 2022. https://doi.org/10.1007/978-3-031-11633-9_20.

[143] Abdulaziz Fatani, Abdelghani Dahou, Mohammed AA Al-Qaness, Songfeng Lu, and Mohamed Abd Elaziz. Advanced feature extraction and selection approach using deep learning and aquila optimizer for iot intrusion detection system. *Sensors*, 22(1):140, 2022. https://doi.org/10.3390/s22010140.

[144] Devrim Akgun, Selman Hizal, and Unal Cavusoglu. A new ddos attacks intrusion detection model based on deep learning for cybersecurity. *Computers & Security*, 118:102748, 2022. https://doi.org/10.1016/j.cose.2022.102748.

[145] V Gowdhaman and R Dhanapal. An intrusion detection system for wireless sensor networks using deep neural network. *Soft Computing*, 26(23):13059–13067, 2022. https://doi.org/10.1007/s00500-021-06473-y.

[146] M Mayuranathan, SK Saravanan, B Muthusenthil, and A Samydurai. An

efficient optimal security system for intrusion detection in cloud computing environment using hybrid deep learning technique. *Advances in Engineering Software*, 173:103236, 2022. https://doi.org/10.1016/j.advengsoft.2022.103236.

[147] Omar A Alzubi, Jafar A Alzubi, Moutaz Alazab, Adnan Alrabea, Albara Awajan, and Issa Qiqieh. Optimized machine learning-based intrusion detection system for fog and edge computing environment. *Electronics*, 11(19):3007, 2022. https://doi.org/10.3390/electronics11193007.

[148] Akhil Kadiyala and Ashok Kumar. Applications of python to evaluate environmental data science problems. *Environmental Progress & Sustainable Energy*, 36(6):1580–1586, 2017. https://doi.org/10.1002/ep.12786.

[149] Marley W Watkins. *A step-by-step guide to exploratory factor analysis with R and RStudio*. Routledge, 2020. http://dx.doi.org/10.4324/9781003120001.

[150] Mu Zhu, Ahmed H Anwar, Zelin Wan, Jin-Hee Cho, Charles A Kamhoua, and Munindar P Singh. A survey of defensive deception: Approaches using game theory and machine learning. *IEEE Communications Surveys & Tutorials*, 23(4):2460–2493, 2021. https://doi.org/10.1109/COMST.2021.3102874.

[151] Xiannuan Liang and Yang Xiao. Game theory for Network Security. *IEEE Communications Surveys & Tutorials*, 15(1):472–486, 2013. https://doi.org/10.1109/SURV.2012.062612.00056.

[152] Tammy Jiang, Jaimie L Gradus, and Anthony J Rosellini. Supervised machine learning: a brief primer. *Behavior Therapy*, 51(5):675–687, 2020. https://doi.org/10.1016/j.beth.2020.05.002.

[153] Muhammad Usama, Junaid Qadir, Aunn Raza, Hunain Arif, Kok-Lim Alvin Yau, Yehia Elkhatib, Amir Hussain, and Ala Al-Fuqaha. Unsupervised machine learning for networking: Techniques, applications and research challenges. *IEEE access*, 7:65579–65615, 2019. https://doi.org/10.1109/ACCESS.2019.2916648.

[154] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine learning*, 109(2):373–440, 2020. https://doi.org/10.1007/s10994-019-05855-6.

[155] Csaba Grossi. *Algorithms for reinforcement learning*. Springer Nature, 2022. https://doi.org/10.1007/978-3-031-01551-9.

[156] Saqib Hakak, Mamoun Alazab, Suleman Khan, Thippa Reddy Gadekallu, Praveen Kumar Reddy Maddikunta, and Wazir Zada Khan. An ensemble

machine learning approach through effective feature extraction to classify fake news. *Future Generation Computer Systems*, 117:47–58, 2021. https://doi.org/10.1016/j.future.2020.11.022.

[157] Arwinder Dhillon, Ashima Singh, and Vinod Kumar Bhalla. A systematic review on biomarker identification for cancer diagnosis and prognosis in multi-omics: From computational needs to machine learning and deep learning. *Archives of Computational Methods in Engineering*, 20:917–949, 2022. https://doi.org/10.1007/s11831-022-09821-9.

[158] Konstantin Hopf and Sascha Reifenrath. Filter methods for feature selection in supervised machine learning applications–review and benchmark. *arXiv preprint arXiv:2111.12140*, 2021.

[159] Yosef Masoudi-Sobhanzadeh, Habib Motieghader, and Ali Masoudi-Nejad. Featureselect: a software for feature selection based on machine learning approaches. *BMC bioinformatics*, 20(1):1–17, 2019. https://doi.org/10.1186/s12859-019-2754-0.

[160] Shoujing Zheng and Zishun Liu. The machine learning embedded method of parameters determination in the constitutive models and potential applications for hydrogels. *International Journal of Applied Mechanics*, 13(01):2150001, 2021. https://doi.org/10.1142/S1758825121500010.

[161] Bruce J Wittmann, Kadina E Johnston, Zachary Wu, and Frances H Arnold. Advances in machine learning for directed evolution. *Current opinion in structural biology*, 69:11–18, 2021. https://doi.org/10.1016/j.sbi.2021.01.008.

[162] Azidine Guezzaz, Younes Asimi, Mourade Azrour, and Ahmed Asimi. Mathematical validation of proposed machine learning classifier for heterogeneous traffic and anomaly detection. *Big Data Mining and Analytics*, 4(1):18–24, 2021. https://doi.org/10.26599/BDMA.2020.9020019.

[163] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. Ieee, 2009. https://doi.org/10.1109/CISDA.2009.5356528.

[164] Abdulaziz Aldribi, Issa Traore, and Belaid Moa. Data sources and datasets for cloud intrusion detection modeling and evaluation. *Cloud computing for optimization: foundations, applications, and challenges*, pages 333–366, 2018. https://doi.org/10.1007/978-3-319-73676-1_13.

[165] Abdulaziz Aldribi, Issa Traoré, Belaid Moa, and Onyekachi Nwamuo. Hypervisor-based cloud intrusion detection through online multivariate sta-

tistical change tracking. *Computers & Security*, 88:101646, 2020. https://doi.org/10.1016/j.cose.2019.101646.

[166] Mohanad Sarhan, Siamak Layeghy, Nour Moustafa, and Marius Portmann. Netflow datasets for machine learning-based network intrusion detection systems. In *Big Data Technologies and Applications: 10th EAI International Conference, BDTA 2020, and 13th EAI International Conference on Wireless Internet, WiCON 2020, Virtual Event, December 11, 2020, Proceedings 10*, pages 117–135. Springer, 2021. https://doi.org/10.1007/978-3-030-72802-1_9.

[167] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116, 2018. https://doi.org/10.5220/0006639801080116.

[168] Nickolaos Koroniotis, Nour Moustafa, Francesco Schiliro, Praveen Gauravaram, and Helge Janicke. A holistic review of cybersecurity and reliability perspectives in smart airports. *IEEE Access*, 8:209802–209834, 2020. https://doi.org/10.1109/ACCESS.2020.3036728.

[169] Sarbjeet Singh and Dilip Kumar. A public key authentication and privacy preserving model for securing healthcare system. *IETE Journal of Research*, 132:1–13, 2021. https://doi.org/10.1080/03772063.2021.1977189.

[170] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. Ddos in the iot: Mirai and other botnets. *Computer*, 50(7):80–84, 2017. https://doi.org/10.1109/MC.2017.201.

[171] Minhaj Ahmad Khan and Khaled Salah. Iot security: Review, blockchain solutions, and open challenges. *Future Generation Computer Systems*, 82:395–411, 2018. https://doi.org/10.1016/j.future.2017.11.022.

[172] Nonita Sharma, Renu Dhir, et al. Fog computing: An overview of iot applications with security issues and challenges. In *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*, pages 1–8. IEEE, 2021. https://doi.org/10.1109/ICRITO51393.2021.9596158.

[173] Zhuotao Liu, Hao Jin, Yih-Chun Hu, and Michael Bailey. Practical proactive ddos-attack mitigation via endpoint-driven in-network traffic control. *IEEE/ACM Transactions on Networking*, 26(4):1948–1961, 2018. https://doi.org/10.1109/TNET.2018.2854795.

[174] Lansheng Han, Man Zhou, Wenjing Jia, Zakaria Dalil, and Xingbo Xu. Intrusion detection model of wireless sensor networks based on game the-

ory and an autoregressive model. *Information Sciences*, 476:491–504, 2019. https://doi.org/10.1016/j.ins.2018.06.017.

[175] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 military communications and information systems conference (MilCIS)*, pages 1–6. IEEE, 2015. https://doi.org/10.1109/MilCIS.2015.7348942.

[176] K Vamsi Krishna, K Swathi, P Rama Koteswara Rao, and B Basaveswara Rao. A detailed analysis of the cidds-001 and cicids-2017 datasets. In *Pervasive Computing and Social Networking*, pages 619–638. Springer, 2022. https://doi.org/10.1007/978-981-16-5640-8_47.

[177] Nickolaos Koroniotis, Nour Moustafa, and Elena Sitnikova. A new network forensic framework based on deep learning for internet of things networks: A particle deep framework. *Future Generation Computer Systems*, 110:91–106, 2020. https://doi.org/10.1016/j.future.2020.03.042.

[178] Guillermo Owen. *Game theory*. Emerald Group Publishing, 2013.

[179] Komal Singh Gill, Sharad Saxena, and Anju Sharma. Gtm-csec: game theoretic model for cloud security based on ids and honeypot. *Computers & Security*, 92:101732, 2020. https://doi.org/10.1016/j.cose.2020.101732.

[180] Liang Du, Ruobin Gao, Ponnuthurai Nagaratnam Suganthan, and David ZW Wang. Bayesian optimization based dynamic ensemble for time series forecasting. *Information Sciences*, 591:155–175, 2022. https://doi.org/10.1016/j.ins.2022.01.010.

[181] Ian Dewancker, Michael McCourt, and Scott Clark. Bayesian optimization for machine learning: A practical guidebook. *arXiv preprint arXiv:1612.04858*, 2016.

[182] Nour Moustafa and Jill Slay. The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set. *Information Security Journal: A Global Perspective*, 25(1-3):18–31, 2016. https://doi.org/10.1109/MilCIS.2015.7348942.

[183] Vikash Kumar, Ditipriya Sinha, Ayan Kumar Das, Subhash Chandra Pandey, and Radha Tamal Goswami. An integrated rule based intrusion detection system: analysis on unsw-nb15 data set and the real time online dataset. *Cluster Computing*, 23(2):1397–1418, 2020. https://doi.org/10.1007/s10586-019-03008-x.

[184] Ravi Vinayakumar, Mamoun Alazab, KP Soman, Prabaharan Poornachandran, Ameer Al-Nemrat, and Sitalakshmi Venkatraman. Deep learning ap-

proach for intelligent intrusion detection system. *IEEE Access*, 7:41525–41550, 2019. https://doi.org/10.1109/ACCESS.2019.2895334.

[185] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Benjamin Turnbull. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems*, 100:779–796, 2019. https://doi.org/10.1016/j.future.2019.05.041.

[186] Seungjin Lee, Azween Abdullah, Nz Jhanjhi, and Sh Kok. Classification of botnet attacks in iot smart factory using honeypot combined with machine learning. *PeerJ Computer Science*, 7:e350, 2021. http://dx.doi.org/10.7717/peerj-cs.350.

[187] SonicWall. https://www.sonicwall.com/news/new-sonicwall-research-finds-aggressive-growth-in-ransomware-rise-in-iot-attacks/. Accessed on March 2019.

[188] Qi Zhang, Meizhu Li, and Yong Deng. Measure the Structure Similarity of Nodes in Complex Networks Based on Relative Entropy. *Physica A: Statistical Mechanics and its Applications*, 491:749–763, 2018. https://doi.org/10.1016/j.physa.2017.09.042.

[189] Basant Subba, Santosh Biswas, and Sushanta Karmakar. A Game Theory Based Multi Layered Intrusion Detection Framework for Wireless Sensor Networks. *International Journal of Wireless Information Networks*, 25:399–421, 2018. https://doi.org/10.1007/s10776-018-0403-6.

[190] Matthias Schonlau and Rosie Yuyan Zou. The random forest algorithm for statistical learning. *The Stata Journal*, 20(1):3–29, 2020. https://doi.org/10.1177/1536867X20909688.

[191] Elena-Adriana Minastireanu and Gabriela Mesnita. Light gbm machine learning algorithm to online click fraud detection. *J. Inform. Assur. Cybersecur*, 2019:263928, 2019. https://doi.org/10.5171/2019.263928.

[192] Fatima Zohra Belgrana, Nacéra Benamrane, Mohamed Amine Hamaida, Abdellah Mohamed Chaabani, and Abdelmalik Taleb-Ahmed. Network Intrusion Detection System using Neural Network and Condensed Nearest Neighbors with Selection of NSL-KDD influencing Features. In *2020 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)*, pages 23–29. IEEE, 2021. https://doi.org/10.1109/IoTaIS50849.2021.9359689.

[193] Yazan Otoum and Amiya Nayak. AS-IDS: Anomaly and Signature Based IDS for the Internet of Things. *Journal of Network and Systems Manage-*

ment, 29(3):1–26, 2021. https://doi.org/10.1007/s10922-021-09589-6.

[194] Ozgur Depren, Murat Topallar, Emin Anarim, and M Kemal Ciliz. An Intelligent Intrusion Detection System (IDS) for Anomaly and Misuse Detection in Computer Networks. *Expert systems with Applications*, 29(4):713–722, 2005. https://doi.org/10.1007/s10922-021-09589-6.

[195] Neha Agrawal and Shashikala Tapaswi. Defense mechanisms against ddos attacks in a cloud computing environment: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, 21(4):3769–3795, 2019. https://doi.org/10.1109/IoTaIS50849.2021.9359689.

[196] Surbhi Solanki, Chetan Gupta, Kalpana Rai, and Minal Saxena. An efficient hids system using machine learning algorithm and evidence theory. In *Proceedings of 2nd International Conference on Artificial Intelligence: Advances and Applications*, pages 21–28. Springer, 2022. https://doi.org/10.1007/978-981-16-6332-1_3.

[197] Ibraheem Aljamal, Ali Tekeoğlu, Korkut Bekiroglu, and Saumendra Sengupta. Hybrid intrusion detection system using machine learning techniques in cloud computing environments. In *2019 IEEE 17th international conference on software engineering research, management and applications (SERA)*, pages 84–89. IEEE, 2019. https://doi.org/10.1109/SERA.2019.8886794.

[198] Abdulaziz Aldribi, Issa Traore, Paulo Gustavo Quinan, and Onyekachi Nwamuo. Documentation for the isot cloud intrusion detection benchmark dataset (isot-cid).

[199] Tiejun Cheng, Yanli Wang, and Stephen H Bryant. Fselector: a ruby gem for feature selection. *Bioinformatics*, 28(21):2851–2852, 2012. https://doi.org/10.1093/bioinformatics/bts528.

Appendix

A.1 Github Links

The code of the proposed models can be accessible using the following link:
`https://github.com/komalgillz/ALL_models`

The datasets used in the proposed research comprising NSLKDD, ISOT, UNSWNB-15, CICIDS, and BoT-IoT are present at:
`https://github.com/komalgillz/Dataset-`

List of Publications

SCI Journals

1. Komal Singh Gill, Sharad Saxena, Anju Sharma, *"GTM-CSec: Game theoretic model for cloud security based on IDS and honeypot"*, Computers & Security, 92, 2020, 101732, ISSN 0167-4048. (IF 5.6)
   https://doi.org/10.1016/j.cose.2020.101732

2. Komal Singh Gill, Sharad Saxena, Anju Sharma, *"GTA-IDS: Game Theoretic Approach to enhance IDS detection in Cloud Environment"*, Computing and Informatics, 41(3), 665–688. (IF 0.974)
   https://doi.org/10.31577/cai_2022_3_665

3. Komal Singh Gill, Sharad Saxena, Anju Sharma, *"NCGTM: A Non-Cooperative Game-Theoretic Model to Assist IDS in Cloud Environment"*, IEEE Transactions on Industrial Informatics. (IF 12.3)
   https://doi.org/10.1109/TII.2023.3300452

4. Komal Singh Gill, Sharad Saxena, Anju Sharma, *"BOGTA using Bayesian Optimised Approach for Intelligent IDS"*, Concurrency and Computation: Practice and Experience. [Under Review]

International Conference

1. Gill, Komal Singh, Sharad Saxena, and Anju Sharma. "Taxonomy of security attacks on cloud environment: A case study on telemedicine." 2019 Amity International Conference on Artificial Intelligence (AICAI). IEEE.
   https://doi.org/10.1109/AICAI.2019.8701363