Mini Project Report

on

Analysis of Cyber Attacks in SaaS-based Cloud Computing Services

Submitted by

Team Members

20BCS059 – Himanshu Shekhar
20BCS125 – Somisetty Sai Praneeth
20BCS128 – Sreedeva Krupananda B Reddy

Under the guidance of

Dr. Malay Kumar
Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Information Technology Dharwad
Karnataka, India-580009

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DHARWAD

04/05/2023

CERTIFICATE

It is to certify that the work contained in the project report titled ***"Analysis of Cyber Attacks in SaaS-based Cloud Computing Services"*** by Himanshu Shekhar (20BCS059), Somisetty Sai Praneeth (20BCS125), and Sreedeva Krupananda B Reddy (20BCS128) has been submitted for the fulfillment of the requirement for the degree of ***"Bachelor of Technology in Computer Science & Engineering"***. Their work has been found satisfactory and hereby approved for submission.

Signature of the Supervisor

Dr. Malay Kumar

Assistant Professor

Department of Computer Science and Engineering

IIIT-Dharwad

DECLARATION

We declare that this written project submission represents our ideas and has not been taken from any other source. We have mentioned and cited the resources of any other publisher wherever needed in the report. We also declare that this work is done by following the principles of academic honesty and integrity and we have done this work with utmost sincerity towards our profession and the institute. We moreover understand the consequences of falsifying any information or misinterpretation of data and figures, hence we have tried to come up and present the best possible response and results.

Himanshu Shekhar – 20BCS059

Somisetty Sai Praneeth – 20BCS125

Sreedeva Krupananda B Reddy – 20BCS129

APPROVAL SHEET

The project entitled ***"Analysis of Cyber Attacks in SaaS-based Cloud Computing Services"*** by Himanshu Shekhar (20BCS059), Somisetty Sai Praneeth (20BCS125), and Sreedeva Krupananda B Reddy (20BCS128) is approved for the degree of Bachelor of Technology in Computer Science & Engineering.

Signature of the Supervisor

Dr. Malay Kumar

Assistant Professor

Department of Computer Science and Engineering

IIIT-Dharwad

Head of Department

Dr. Pavan Kumar C

Assistant Professor

Department of Computer Science and Engineering

IIIT-Dharwad

ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr. Malay Kumar**, Assistant Professor (Dept of Computer Science and Engineering) **Dr Sadhvi Manerikar**, Mini Project Coordinator for their constant support and guidance throughout the project. It is their constant effort that helped us in completing the project well and on time.

We would also like to thank the **Department of Computer Science IIIT-Dharwad** for the motivation and encouragement, and sincere thanks to **Dr. Pramod Yelmewad,** Mini Project Coordinator for providing us with the guidelines and resources to complete this report.

Last but not least a special thanks to **Library, IIIT-Dharwad** for helping us identify any plagiarism.

Contents

List of Figures

List of Tables

1 Introduction

1.1 SaaS (Software as a Service)

The term "cloud computing" refers to the process of storing data and using online computing resources. Nothing is stored on your computer by it. This phrase describes the on-demand accessibility of computer services including servers, databases, networking, and data storage. Cloud computing's main objective is to make data centers more widely accessible to users. Users can also access data on a faraway server.

In cloud computing, there are three types of service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

SaaS is a hosted software model which allows end-users to utilize the software and apps/web-based applications on the cloud via the internet without the need to install them on his/her machine. By utilizing SaaS, consumers may pay a per-user subscription cost without having to spend a significant sum of money on the installation and upkeep of required software and hardware. As long as they have a device with internet connectivity, users can access the service from anywhere in the globe at any time. Without having to worry about updates, users are always using the most recent version of the program.

The service provider is in charge of managing the cloud's infrastructure. Together with ensuring the security of applications and customers' access to them, the service provider also controls software and service agreements. Companies are driven to concentrate on the development of their software by recurring income. SaaS's inherent multi-tenancy and virtualization technologies give service providers centralized control and maximum resource usage. Examples of SaaS include email and office software.

1.2 SaaS Security Challenges

Despite these advantages, the security of the SaaS presents a great problem for companies, on the level of Confidentiality, integrity, authentication, location of data, etc. The major challenge that a company or user encounters when converting their software to a SaaS architecture is the lack of confidence in the security of their data owing to the worry of data leakage. Customers of cloud services are not in charge of managing the cloud infrastructure. The user should be aware of the security precautions the cloud provider has put in place.

The data security of the SaaS (Software as a Service) environment is a complex issue that covers several areas such as confidentiality, availability, integrity, authorization, backup and recovery, and transfer. Several solutions have been proposed, such as using SSL encryption, MAC, searchable encryption, homomorphic tokens, and flexible security policies. However, these solutions are limited by factors such as false certificate authorities, data error location, limited applicability to SQL-like databases, and data availability.

Similarly, application security is also a critical issue that has been extensively studied, but fine-grained solutions to tackle these issues are limited. User authentication is a crucial part of application security, and various approaches have been proposed, such as user identity verification, biometric cues-based authentication, behavior-based authentication, touch screen pattern, keystroke analysis, and implicit authentication. However, these approaches are limited by accuracy, training time, and the impact of different factors such as footwear and soft input methods.

2 Related Work

2.1 Attacks and their possible countermeasures in SaaS

Cyber Attacks on SaaS-based cloud computing services are a growing concern in today's digital landscape. In this model, as the software and associated data are stored on servers maintained by the service provider, and accessed remotely by users, it makes an attractive target for cyber attackers, who can potentially access a large number of sensitive information and systems in a single attack.

There are now 28 assaults on SaaS, which is this problem. It has to be made clear that not all assaults are possible with this service paradigm. 25 papers cover the 28 attacks in all. According to the data gathered from the many articles that were evaluated, the list of assaults and potential defenses is shown below.

1) **ARP Spoofing**: Attackers utilize spoofing techniques such as Address Resolution Protocol (ARP) spoofing to intercept data. By fooling a device into delivering communications to the attacker rather than the intended receiver, an attacker can carry out an ARP spoofing attack.

Because ARP serves as a communication mechanism linking a dynamic internet protocol (IP) address to a physical machine address, it causes ARP spoofing on a local area network (LAN) (MAC). Attackers can employ ARP spoofing for denial-of-service attacks, man-in-the-middle assaults, and other cyberattacks.

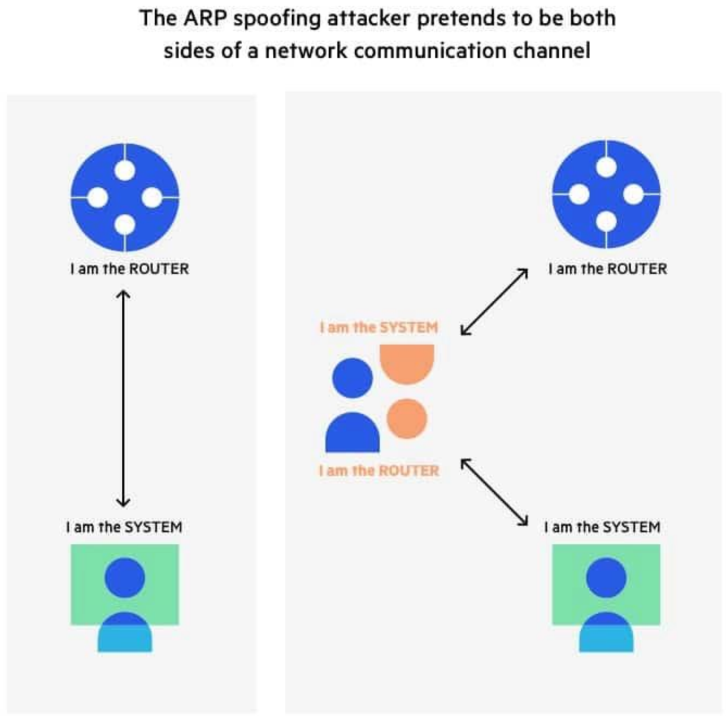


Fig 2.1.1 ARP Spoofing
source: www.imperva.com

ARP spoofing can be prevented by using:

a. Virtual Private Network (VPN) as it makes all communication encrypted, and worthless for an ARP spoofing attacker
b. Static ARP—The ARP protocol lets you define a static ARP entry for an IP address, and prevents devices from listening to ARP responses for that address.
c. Packet filtering solutions can be used as they identify poisoned ARP packets by seeing that they contain conflicting source information, and stop them before they reach devices on your network.

2) **Backdoor and debug options:** A backdoor may be included in the software code by the developer and debug options are features included in software applications to help developers identify and fix bugs during the development process. However, if these debug options are not removed before the software is released, they can be exploited by attackers to gain unauthorized access to the system.

An attacker can use a backdoor or debug option to gain access to sensitive data, modify or delete data, and perform other malicious activities. To prevent such attacks, SaaS providers should follow the best practices for secure software development, including:

a. Conduct security audits regularly and vulnerability assessments of the software application.
b. Implement access controls and strong authentication mechanisms to prevent unauthorized access.
c. Remove all debug options before the software is released to the production environment.
d. Monitor system logs and user activity to detect any suspicious behavior.
e. Stay up-to-date with the latest security patches and updates.

3) **Broken authentication:** This is a type of attack that targets the authentication mechanisms of a software application, and exploits vulnerabilities in the authentication process to gain unauthorized access to the application or the data it stores. In the context of SaaS (Software-as-a-Service), broken authentication can be a serious threat, as it can allow attackers to access sensitive customer data and other valuable information stored in the cloud.

Some common examples of broken authentication attacks in SaaS include:
Password attacks: In this type of attack, an attacker tries to guess or crack a user's password to gain access to the SaaS application. This can be done through brute-force attacks, where the attacker tries a large number of password combinations, or through phishing attacks, where the attacker tricks the user into revealing their password.

Session hijacking: This type of attack involves stealing a user's session ID, which is used to authenticate the user's identity during a session. The attacker can then use the stolen session ID to act like the user and gain access to the SaaS application.

Cross-site scripting (XSS): In this type of attack, the attacker injects malicious code into a web page or form within the SaaS application, which can then be executed by other users who visit the page. This can allow the attacker to steal session IDs or other sensitive data from other users.

To prevent broken authentication attacks, SaaS providers should follow the best practices for secure authentication, including:

a. Implementing strong password policies, such as requiring users to use complex passwords and enforcing regular password changes.

b. Implementing multi-factor authentication (MFA) requires users to provide an additional form of authentication, such as a fingerprint or one-time code, in addition to a password.
c. Encrypting all sensitive data, including passwords and session IDs, to prevent them from being intercepted by attackers.
d. Implementing secure session management, including timeouts and logouts, to prevent session hijacking attacks.
e. Regularly testing the authentication mechanisms of the SaaS application, including conducting penetration testing and vulnerability assessments.

4) **Buffer Overflow:** A buffer overflow attack is a type of security vulnerability that occurs when an application attempts to store more data in a buffer than it was designed to hold. Buffers are temporary storage areas in a computer's memory used to store data while it is being transferred between different parts of an application.

When an overflow of the buffer occurs, the extra data is written to adjacent memory locations, corrupting the contents of these locations and potentially leading to system crashes or other security issues.

Ways to prevent buffer overflow attacks:
a. Developers must ensure that their applications are designed to handle input data correctly and that buffers are properly sized and validated before data is stored in them.
b. Security testing and code review should be performed regularly to identify and address buffer overflow vulnerabilities before they can be exploited.
c. System administrators can also use various security measures such as intrusion detection systems, firewalls, and access controls to detect and prevent buffer overflow attacks.

5) **Code Injection**: Code Injection is a security vulnerability that allows an attacker to insert and execute malicious code into a software application. In the cloud where resources are shared, attackers can exploit this vulnerability to access sensitive data from other users or the system itself. There are several code injection techniques like code injection, Cross-site Scripting (XSS), Command injection, etc.

One of the most common techniques for code injection is SQL injection, where an attacker inserts malicious SQL commands through web forms or URLs to access a database. If successful, the attacker can view, modify, or delete data from the database, or even take over the entire system. If a user clicks on an infected URL, the malicious code can be executed on the user's machine, giving the attacker access to the user's information.

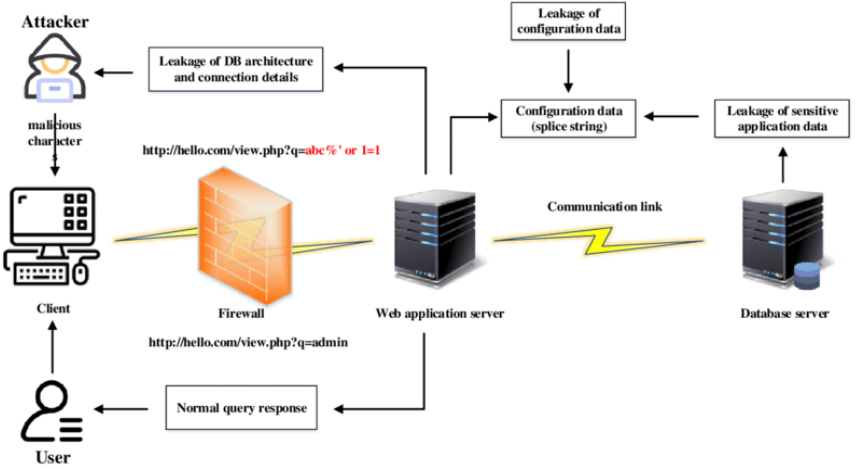

Attacker
Leakage of DB architecture and connection details
Leakage of configuration data
Configuration data (splice string)
Leakage of sensitive application data
malicious character
http://hello.com/view.php?q=abc%' or 1=1
Communication link
Client
Firewall
Web application server
Database server
http://hello.com/view.php?q=admin
Normal query response
User

Fig 2.1.2 SQL Injection
source: https://ieeexplore.ieee.org/ielx7/6287639/8600701/08854182.pdf

To prevent code injection attacks, active content filtering is often used. This technique involves analyzing the content of incoming requests to identify and block any potentially malicious code. When SQL injection is detected, dynamic SQL code is generated in the application to prevent the attacker's commands from being executed on the database.

6) **Cookie Poisoning**: Cookie poisoning is a type of attack where an attacker modifies the contents of a cookie that is used to store information about a user's session on a website.

This can allow the attacker to hijack the user's session and perform actions on their behalf, such as making unauthorized purchases, changing their account settings, or accessing sensitive information.

Ways to prevent cookie poisoning attacks:

a. Use HTTPS: HTTPS encrypts the communication between the user's browser and the web server, which makes it much more difficult for an attacker to intercept and modify the contents of a cookie.
b. Use the HttpOnly and Secure flags: The HttpOnly flag ensures that the cookie can only be accessed by the web server and not by JavaScript, which can help prevent cross-site scripting (XSS) attacks. The Secure flag ensures that the cookie is only transmitted over a secure HTTPS connection.
c. Use session tokens instead of user IDs: Session tokens are randomly generated values that are stored in cookies and used to authenticate the user's session. This makes it much more difficult for an attacker to guess or intercept the user's session ID.
d. Validate cookie contents on the server: Before accepting a cookie, the server should validate its contents to ensure that it has not been tampered with or modified.

7) **Cross-Site Request Forgery (CSRF):** Cross-Site Request Forgery (CSRF) is a type of web attack when malicious website tricks a user into making an unintended request to a vulnerable web application in which the user has an active session. The attack takes advantage of the trust between the user and the vulnerable web application to execute unauthorized actions.

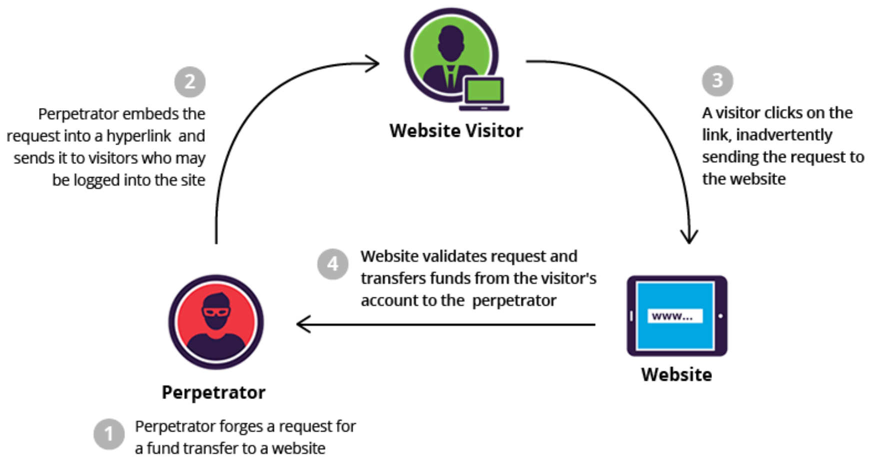

Website Visitor
2
Perpetrator embeds the request into a hyperlink and sends it to visitors who may be logged into the site
3
A visitor clicks on the link, inadvertently sending the request to the website
4
Website validates request and transfers funds from the visitor's account to the perpetrator
www...
Website
Perpetrator
1
Perpetrator forges a request for a fund transfer to a website

Fig 2.1.3 Cross-Site Request Forgery
source: www.imperva.com

Ways to prevent CSRF attacks:

a. Secret Tokens: One effective way to prevent CSRF attacks is to use secret tokens, which are unique values generated by the web application and embedded in forms or URLs that perform sensitive actions. When the user submits the form or clicks the URL, the web application checks that the secret token matches the one that was generated for the user's session.

b. Reference and Origin Headers: When the web application receives a request, it can check the origin and reference headers to ensure that the request is coming from a legitimate source.

c. CSRF Cookies: In addition to session cookies, web applications can use CSRF cookies, which are stored in a separate cookie that is not accessible from JavaScript code. The web application can then use this cookie to validate requests and ensure that they are coming from a legitimate source.

d. Captchas: A Captcha is a challenge-response test that is designed to be easy for humans to pass but difficult for bots. By requiring the user to complete a Captcha before submitting a form, the web application can ensure that the request is coming from a human and not a bot.

e. HTTP-only cookies: It is a good practice to set the HttpOnly flag on all cookies. This flag ensures that the cookie is only accessible through HTTP requests, and not through JavaScript code. This prevents any malicious code injected into the web application from accessing sensitive cookies.

8) **DNS Poisoning:** DNS poisoning is an attack that involves corrupting the DNS (Domain Name System) cache of a computer or a network device. DNS translates names into IP addresses that computers can understand, and if a hacker can manipulate the DNS cache, they can redirect traffic to malicious websites.

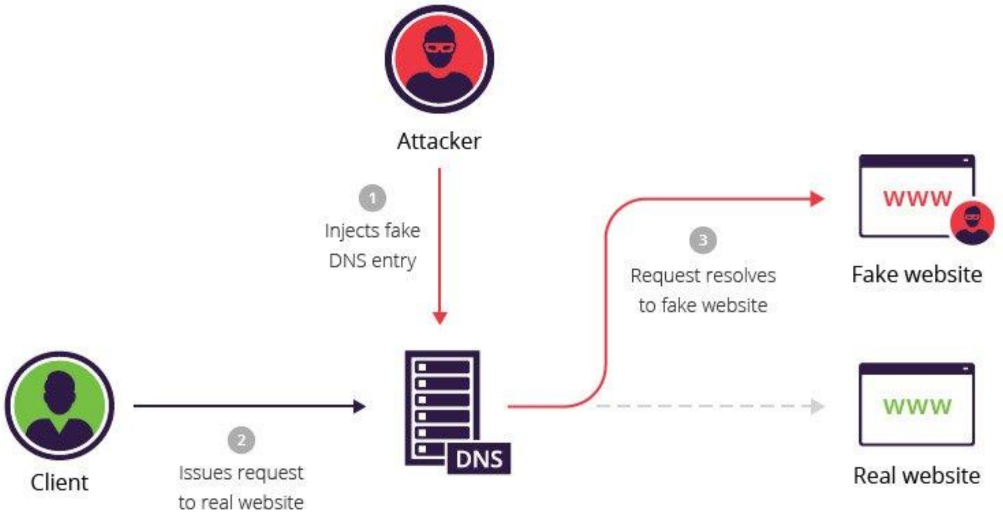


Fig 2.1.4 DNS Poisoning
source: www.imperva.com

One common method of DNS poisoning involves the attacker injecting fake DNS data into a DNS server, which can then be propagated to other DNS servers and clients. The attacker may also use a technique known as a "man-in-the-middle" (MITM) attack, which involves intercepting communication between a client and a DNS server and replacing legitimate DNS responses with fake ones.

To prevent DNS poisoning attacks, you can take the following measures:

a. Keep your software up-to-date: Make sure to install security patches and updates for your operating system, web browser, and other applications regularly.
b. Use a reputable DNS server: Use a reliable DNS server, such as those provided by your ISP or a well-known public DNS provider like Google or Cloudflare.
c. Implement DNSSEC: DNS Security Extensions (DNSSEC) is a security protocol that adds an extra layer of security to the DNS system, making it harder for attackers to manipulate DNS data.
d. Using VPN: A Virtual Private Network (VPN) can help protect your online privacy and security by encrypting your internet traffic and hiding your IP address, making it more difficult for attackers to intercept your DNS queries.
e. Enable firewalls: Firewalls can help block malicious traffic from entering your network, preventing attackers from accessing your devices and DNS servers.

9) **Dumpster Diving**: Dumpster diving is a type of attack in which an attacker searches through trash or other discarded materials to find information that can be used to compromise the security of a system or organization. This information can include passwords, sensitive documents, personal information, or any other confidential data.

Fig 2.1.5 Dumpster Diving
source: Real eLearning YouTube Channel

To prevent dumpster diving attacks, organizations can take the following measures:

a. Shred all important information before disposing of it.
b. Use locks or security cameras to ensure only authorized personnel have access.
c. Encrypt sensitive information to make it useless if it falls into the wrong hands.

10) **Eavesdropping**: An eavesdropping attack is a type of security breach in which cybercriminal intercepts and monitors communication between two parties to obtain sensitive information. This can include confidential messages, passwords, credit card numbers, or other personal data.

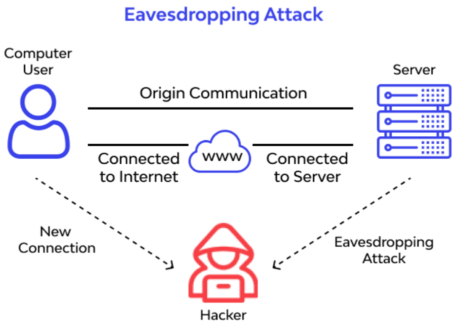


Fig 2.1.6 Eavesdropping Attack
source: www.wallarm.com

Ways to prevent the attacks:

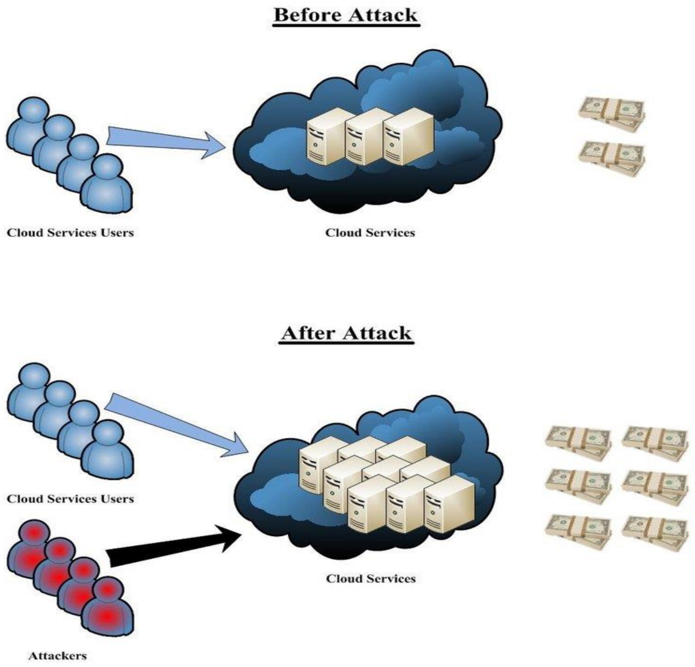
a. Encryption: One of the most effective ways to prevent eavesdropping attacks is to use encryption. Encryption involves scrambling the information sent between two parties so that it can only be read by someone who has the decryption key. This ensures that even if an attacker intercepts the communication, they will not be able to read the sensitive information.

b. Secure communication protocols: Another way to prevent eavesdropping attacks is to use secure communication protocols such as HTTPS, SSH, or SSL.

c. Physical security: Restricting physical access to servers or computers, using secure passwords and access controls, and monitoring physical access to sensitive areas.

d. Network security: Using firewalls and intrusion detection systems that can detect and block suspicious activity on a network, including attempts to intercept or monitor communication.

11) EDoS. Customer billing is the target of this assault. It comprises raising the prices for the services provided to users.


Before Attack

After Attack

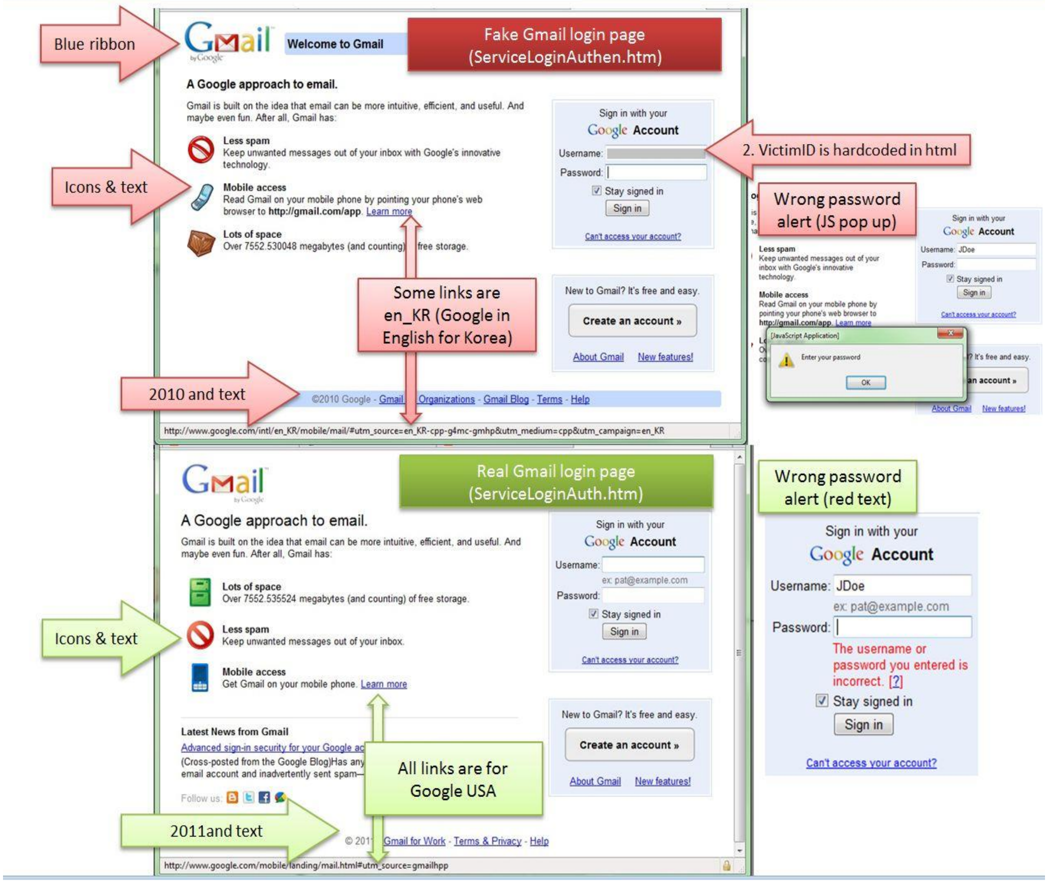
Fig 2.1.7 EDoS Attack
source: http://dx.doi.org/10.13140/RG.2.2.18274.07368

Targeting various components of the cloud infrastructure, EDoS attacks against cloud systems can be launched using a variety of approaches, such as DDoS assaults, brute-force attacks, and vulnerability exploits. Because it might be difficult to tell legitimate traffic patterns from malicious traffic patterns, detecting EDoS attacks in a cloud environment can be difficult.

DoS attacks increase the consumption of bandwidth, CPU, and storage on services that charge for use. Cloud providers and consumers can put protections like resource limiting, traffic filtering, and load balancing into place to stop EDoS assaults. The combination of technical and non-technical safeguards, such as robust security procedures, efficient monitoring, and proactive incident response planning, is necessary to avoid EDoS assaults in a cloud context. To make their systems durable and able to survive EDoS attacks, users and cloud providers must work together.

12) Google Hacking. To find certain sorts of information that may be utilized to obtain illegal access to cloud resources, Google hackers employ sophisticated search operators. Search engines like Google are a suitable choice for an attacker to utilize to steal sensitive information from a person or company. On Google, they often look for security gaps that they may exploit after gathering essential knowledge about the target system.

Default login pages, exposed data, and other flaws that may be leveraged to get into cloud services can all be found with Google hacking. Security risks may be initiated at the application level, resulting in system outages that prevent even authorized users from using the program.


Blue ribbon
Welcome to Gmail
Fake Gmail login page (ServiceLoginAuthen.htm)
2. VictimID is hardcoded in html
Icons & text
Wrong password alert (JS pop up)
Some links are en_KR (Google in English for Korea)
2010 and text
Real Gmail login page (ServiceLoginAuth.htm)
Wrong password alert (red text)
Icons & text
All links are for Google USA
2011and text
Sign in with your Google Account
Username:
Password:
☑ Stay signed in
Sign in
Can't access your account?
New to Gmail? It's free and easy.
Create an account »
About Gmail
New features!
[JavaScript Application]
Enter your password
OK
Follow us:

A Google approach to email.

Gmail is built on the idea that email can be more intuitive, efficient, and useful. And maybe even fun. After all, Gmail has:

Less spam
Keep unwanted messages out of your inbox with Google's innovative technology.

Mobile access
Read Gmail on your mobile phone by pointing your phone's web browser to http://gmail.com/app. Learn more

Lots of space
Over 7552.530048 megabytes (and counting) of free storage.

©2010 Google - Gmail for Organizations - Gmail Blog - Terms - Help

http://www.google.com/intl/en_KR/mobile/mail/#utm_source=en_KR-cpp-g4mc-gmhp&utm_medium=cpp&utm_campaign=en_KR

A Google approach to email.

Gmail is built on the idea that email can be more intuitive, efficient, and useful. And maybe even fun. After all, Gmail has:

Lots of space
Over 7552.535524 megabytes (and counting) of free storage.

Less spam
Keep unwanted messages out of your inbox.

Mobile access
Get Gmail on your mobile phone. Learn more

Latest News from Gmail
Advanced sign-in security for your Google ac
(Cross-posted from the Google Blog)Has any email account and inadvertently sent spam—

© 2011 Gmail for Work - Terms & Privacy - Help

http://www.google.com/mobile/landing/mail.html#utm_source=gmailhpp

Username: JDoe
ex: pat@example.com
Password:
The username or password you entered is incorrect. [?]

Fig 2.1.8 Google Hacking using Fake Gmail Login Page
source: www.indexoncensorship.org

Using Google hacking, a group of hackers in China steal the login information from many Gmail subscribers.

Detecting Google hacking attacks requires monitoring search engine results for keywords and search operators commonly used by attackers. Implementing best security practices, such as strong passwords, two-factor authentication, frequent security audits, and a Software solution such as Web Vulnerability Scanner are some of the countermeasures for Google hacking.

13) Hash Value Manipulation. Attackers can affect the integrity of cloud data covertly by using the technique of hash value manipulation. Some methods, including collision attacks, in which an attacker creates two distinct files with the same hash value, can be used to manipulate hash values.

By altering a file's hash value without altering its content, an attacker can change cloud data without being noticed with this technique. The server associates the file with the hash value if the database contains the updated hash value. On the other hand, if the database does not contain the updated hash value, the server asks the user for a file. In situations when the server utilizes OpenSSL with the Ncrypto class, this vulnerability may exist.

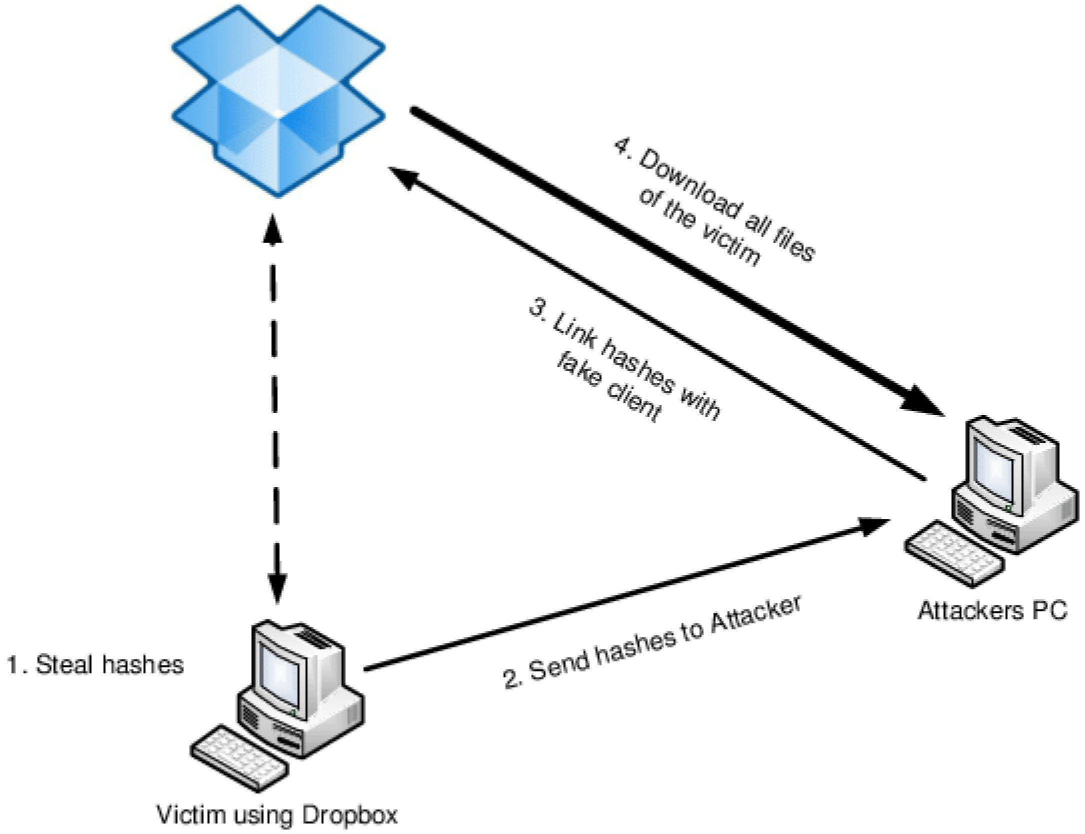

4. Download all files of the victim
3. Link hashes with fake client
Attackers PC
1. Steal hashes
2. Send hashes to Attacker
Victim using Dropbox

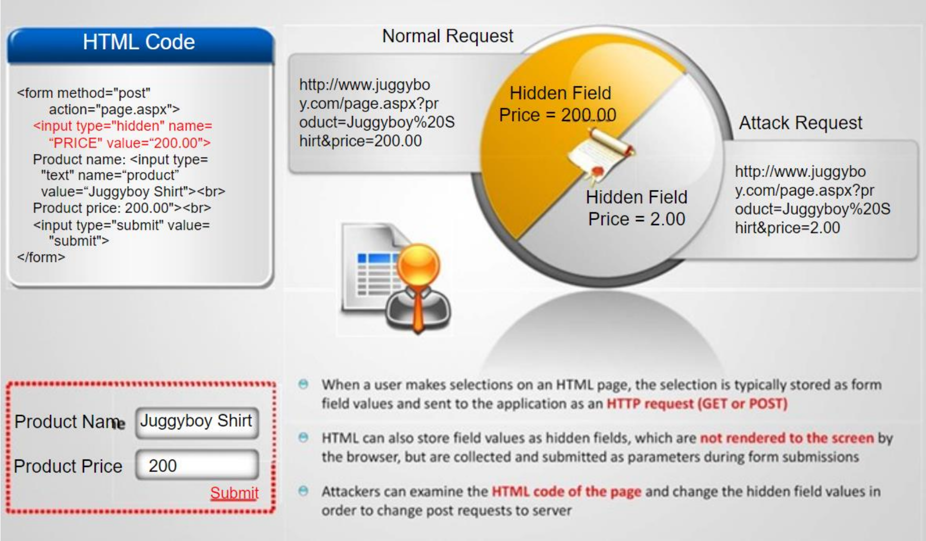
Fig 2.1.9 Hash Value Manipulation in Virtual Box
source:
https://www.researchgate.net/publication/230035795_Dark_Clouds_on_the_Horizon_Using_Cloud_Storage_as_Attack_Vector_and_Online_Slack_Space

Detecting hash value manipulation attacks requires implementing monitoring and auditing tools that detect changes in hash values, as well as implementing security controls that restrict access to data and prevent unauthorized modifications.

The use of secure hash algorithms, the implementation of access restrictions and permissions, regular data backups, and testing to assure data integrity are all examples of countermeasures for hash value manipulation. Additionally, cloud providers and users can use intrusion detection systems and other security measures like the implementation of strong communication protocols using encryption and probabilistic tests required to prevent or quickly detect and respond to hash value manipulation attacks.

14) Hidden Field Manipulation. This is a sort of attack that takes the use of flaws in online forms to modify data and send it to cloud apps. This attack may be carried out either by directly changing the HTML code or via tools that manipulate hidden form fields. Attackers can modify hidden fields to bypass security measures, gain access to cloud resources, or steal sensitive data.

More specifically, certain fields are concealed and utilized by the developer when the user is browsing the website. In HTML forms, hidden fields provide crucial data like price, user ID, etc. The catalog page may be saved, and the attacker can edit the concealed field's value and publish it online. As an illustration, it can be used to save money, but an attacker may use this to make purchases at a higher cost.



HTML Code
Normal Request
http://www.juggyboy.com/page.aspx?product=Juggyboy%20Shirt&price=200.00
Hidden Field Price = 200.00
Hidden Field Price = 2.00
Attack Request
http://www.juggyboy.com/page.aspx?product=Juggyboy%20Shirt&price=2.00
Product Name
Juggyboy Shirt
Product Price
200
Submit

```
<form method="post"
     action="page.aspx">
  <input type="hidden" name=
     "PRICE" value="200.00">
  Product name: <input type=
    "text" name="product"
    value="Juggyboy Shirt"><br>
  Product price: 200.00"><br>
  <input type="submit" value=
     "submit">
</form>
```

- When a user makes selections on an HTML page, the selection is typically stored as form field values and sent to the application as an HTTP request (GET or POST)
- HTML can also store field values as hidden fields, which are not rendered to the screen by the browser, but are collected and submitted as parameters during form submissions
- Attackers can examine the HTML code of the page and change the hidden field values in order to change post requests to server

Fig 2.1.10 Hidden Field Manipulation in HTML Code
source: EG-Council

Detecting hidden field manipulation attacks can be challenging, as attackers can use various techniques to conceal their activities. Implementing security controls like input validation and sanitization, encrypting sensitive data, and employing secure communication protocols like HTTPS are some countermeasures for hidden field manipulation.

15) Malware injection and steganography attacks. To steal data or launch additional assaults, malicious code must be injected into cloud apps or virtual machines. An attacker can add code to the files that go over the network in this fashion. Security tools can disregard this assault since it seems to be a regular file being transferred. Steganography assaults hide malware inside files that appear to be safe, making them hard to find.

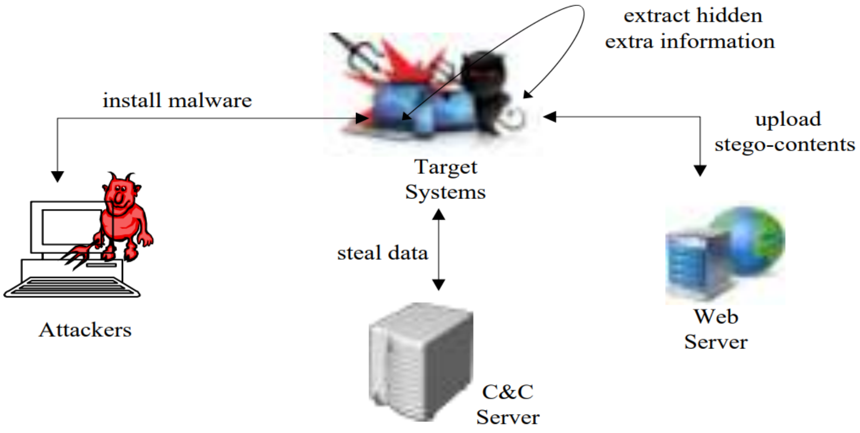


Fig 2.1.11 Malware Injection and Steganography Attack

Detecting these types of attacks requires implementing security controls that monitor for suspicious activity, such as unusual network traffic, unexpected file modifications, or unauthorized access attempts. Strong access restrictions, firewalls, intrusion detection systems, and routine security audits are all examples of defenses against malware injection and steganography assaults. To further prevent unwanted access to data, cloud service providers and customers can use anti-malware software, carry out routine software upgrades and patches, and utilize encryption and other security measures.

16) Man-in-the-middle Attack. A man-in-the-middle attack (MITM) is a kind of cyberattack in which a hacker intercepts and modifies communications between two parties using a cloud computing system to steal data, pose as trustworthy users, or insert harmful code. This attack can be carried out via eavesdropping on network traffic or by pretending to be a trustworthy entity to get private data and also can occur due to weak password management and weak authentication.

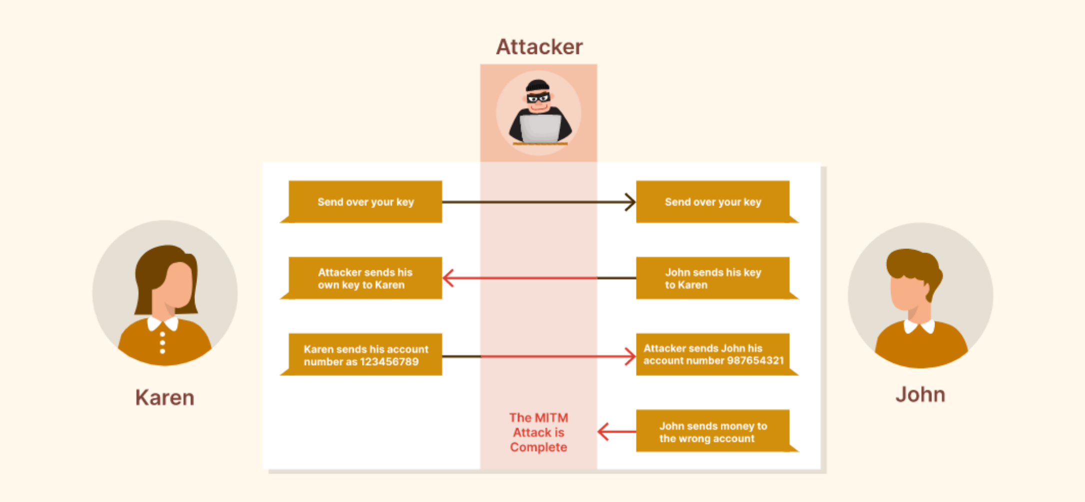
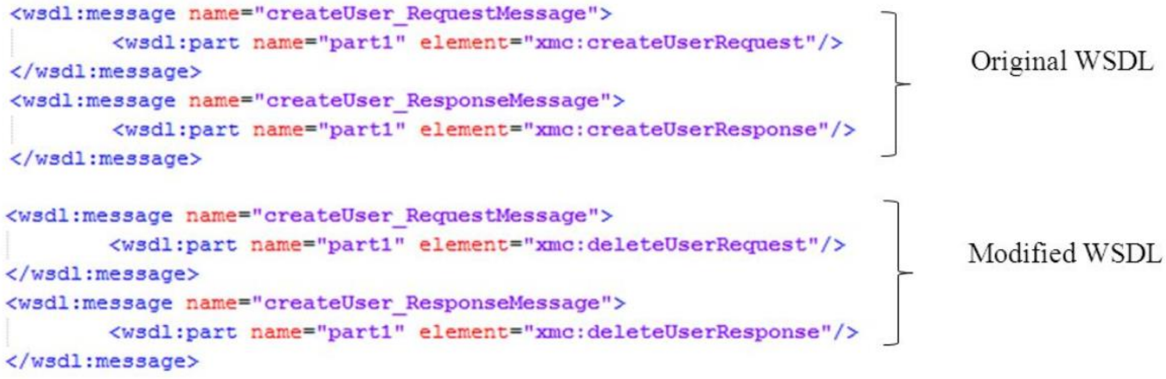

Attacker
Send over your key
Send over your key
Attacker sends his own key to Karen
John sends his key to Karen
Karen sends his account number as 123456789
Attacker sends John his account number 987654321
The MITM Attack is Complete
John sends money to the wrong account
Karen
John

Fig 2.1.12 Man-in-the-Middle Attack
source: www.cloudpanel.io

Detecting a MITM attack requires monitoring for unusual activity, such as unexpected network traffic or changes in data integrity. Strong encryption, the use of secure communication protocols like HTTPS, and the implementation of access restrictions and authentication methods are all countermeasures for MITM attacks.

To prevent illegal access, cloud providers and consumers can also utilize intrusion detection systems, routine security audits, and two-factor authentication. We advise using SSL-secured communications and providing authentication and authentication methods to the nodes as security precautions. Airjack, Cain, Dsniff, and Ettercap are some effective defenses against these assaults.

17) Meta Data Spoofing Attack. A cyberattack called metadata spoofing enables an attacker to get around security measures and access confidential information by changing the metadata of files kept in a cloud computing environment. This technique may be used to disguise the presence of harmful files or misrepresent the genuine contents of a file by altering file properties like creation date, author, or access rights. In this way, the attacker gains access to sensitive applications and data.



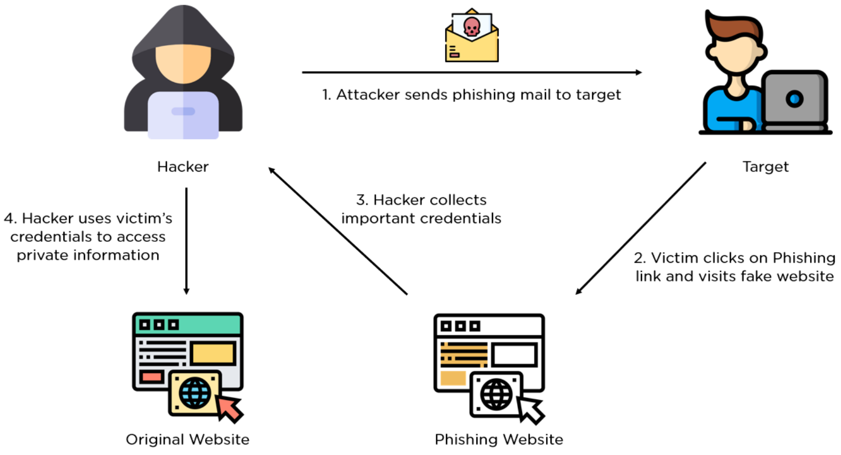
```
<wsdl:message name="createUser_RequestMessage">
        <wsdl:part name="part1" element="xmc:createUserRequest"/>
</wsdl:message>
<wsdl:message name="createUser_ResponseMessage">
        <wsdl:part name="part1" element="xmc:createUserResponse"/>
</wsdl:message>
```

```
<wsdl:message name="createUser_RequestMessage">
        <wsdl:part name="part1" element="xmc:deleteUserRequest"/>
</wsdl:message>
<wsdl:message name="createUser_ResponseMessage">
        <wsdl:part name="part1" element="xmc:deleteUserResponse"/>
</wsdl:message>
```

Fig 2.1.13 Meta Data Spoofing Attack

Detecting metadata spoofing requires implementing monitoring and auditing tools that detect metadata changes, as well as implementing security controls that restrict access to data and prevent unauthorized modifications. Implementing access limits and permissions, encrypting sensitive data, and utilizing security tools to monitor and react to suspicious activities are some of the countermeasures for metadata spoofing. The integrity of files saved in the cloud can also be protected by routine data testing and backups.

18) Phishing Attack. Phishing attacks are a frequent kind of cyber assault in which the attacker pretends to be a reliable entity to fool a victim into disclosing sensitive information, including login credentials or financial information. Phishing attacks can be employed in the context of cloud computing to obtain confidential information held in cloud-based apps or to download malware onto user devices.



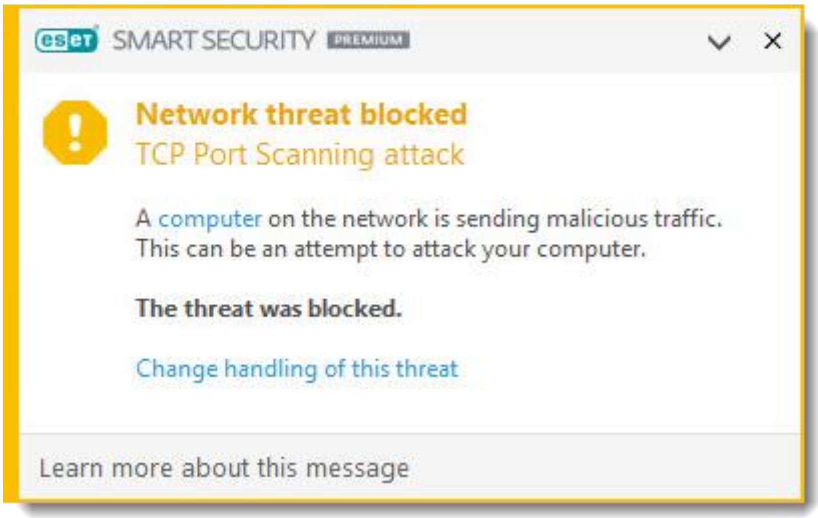
1. Attacker sends phishing mail to target
Hacker
Target
4. Hacker uses victim's credentials to access private information
3. Hacker collects important credentials
2. Victim clicks on Phishing link and visits fake website
Original Website
Phishing Website

Fig 2.1.14 Phishing Attack via Mail
source: www.simplilearn.com

Simply put, it is a tactic that entails sending a genuine person to a phony website. Users submit their credentials thinking it is a trustworthy website a consequence, which compromises the user account.

Users must be made aware of the warning signals of phishing, such as dubious links or demands for personal information, to recognize phishing assaults. Using two-factor authentication, utilizing email filters to prevent questionable emails, and regularly educating users about security issues are all phishing assault defenses. To improve cloud security, encryption must be implemented, as well as the use of TLS for applications. Finally, the use of certificates through HTTPS is essential.

19) Port Scanning. In a cyber-attack known as port scanning, a target's network is scanned to find open, closed, or filtered ports and security holes that may be used to access cloud-based resources without authorization. A target's network is often thoroughly probed by the attacker using port scanning tools to find possible entry holes. If the customer configures the security group to allow traffic from any source to a specific port, then that specific port will be vulnerable to a port scan.



Network threat blocked
TCP Port Scanning attack

A computer on the network is sending malicious traffic.
This can be an attempt to attack your computer.

The threat was blocked.

Change handling of this threat

Learn more about this message

Fig 2.1.15 Port Scanning Alert
source: www.eset.com

Detecting port scanning attacks requires implementing network monitoring tools that detect unusual activity or attempts to access restricted resources and when Port scanning is detected it should be stopped and blocked. Implementing access controls and permission settings to restrict access to sensitive data, implementing firewalls and intrusion detection systems to block unauthorized access to cloud-based resources, and routinely updating software and patches to stop known vulnerabilities from being exploited are all examples of countermeasures for port scanning attacks.

20) Race Condition. A race condition attack is a type of cyber-attack where an attacker exploits the timing of events in a cloud-based system to gain unauthorized access or cause a denial of service. This attack happens when several processes access the same data simultaneously. This can result in unintended or unauthorized behavior, such as the deletion of files, unauthorized access to data, or the introduction of malware.

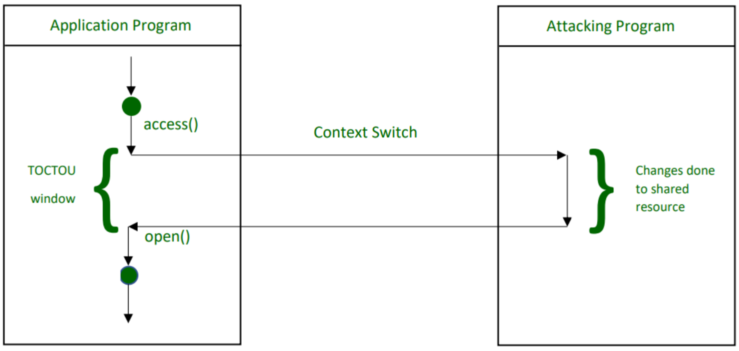


Fig 2.1.16 Race Condition Attack
source: www.geeksforgeeks.org

Detecting race condition attacks requires implementing access controls and monitoring tools that track system events and anomalies. The use of appropriate synchronization techniques, implementation of access controls and permissions to restrict access to sensitive data, frequent security audits and testing, and vulnerability identification and mitigation are all examples of countermeasures for race condition attacks.

21) Replay Attack. The data or answer sent over the network is the target of a security attack. The attacker who has gained access without authorization intercepts the communication and transmits the contents to the target while posing as the original sender. The fundamental justification for calling this assault a replay attack is because the client would get the message twice, therefore the name.

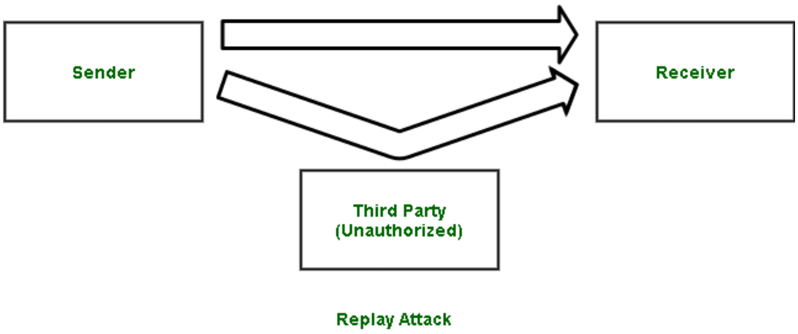


Fig 2.1.17 Replay Attack
source: www.geeksforgeeks.org

Countermeasures for replay attack:

a. Timestamp method: We can prevent this type of attack by using timestamps along with the data. if the timestamp is more than a certain limit, it can be discarded, and the sender can ask to send the again.
b. Session key method: This key can only be used once(by the sender and receiver) per transaction, and cannot be reused.

22) Reused IP address. A security issue arises if a user switches networks and the same IP address is given to a new network. This is due to the user's misconception that their resources are inaccessible after they leave the network. Nevertheless, if another user learns the first user's

IP address, they may be able to access these resources, which would breach the first user's right to privacy.

Countermeasures for reused IP address: To avoid this security problem, the elimination of cache in ARP tables (Address resolution protocol, the table is used to maintain the correlation between each MAC address and its corresponding address) is proposed.

23) Service Injection Attack. The user's requests are automatically sent to malicious services in this attack when an attacker is successful in injecting a malicious service. Data integrity is compromised as a result of this attack, and account or service theft is made possible.

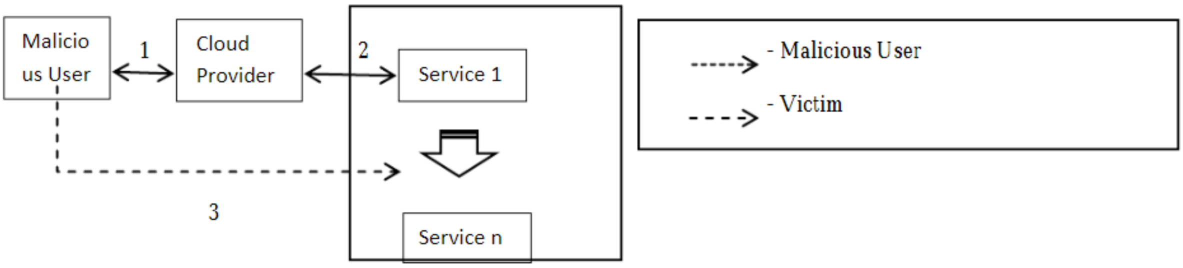


Fig 2.1.18 Service Injection Attack

Countermeasures for service injection attack: As a security measure, we propose the use of a robust attack detection mechanism, mechanisms for identifying virtual machines, and using integration services.

24) Sniffing: it is a technique that uses any software tool to capture the packets that are traveling through the network. In this attack, the attacker can have access to sensitive data, such as credentials and credit card data.

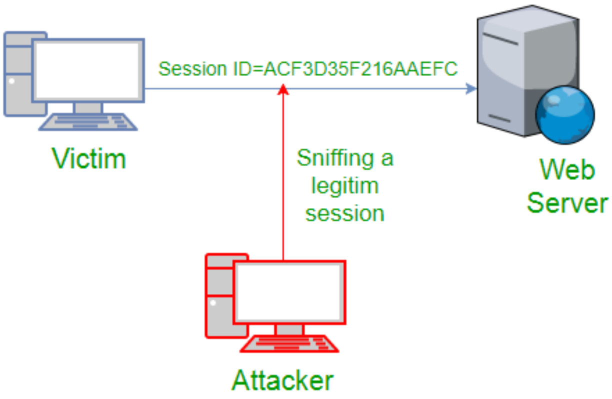


Fig 2.1.19 Sniffing a Legitim Session
source: www.geeksforgeeks.org

Countermeasures for sniffing attack:

a. Use of cryptographic protocols like SSL (secure sockets layer – used for establishing a secure link between client and server) and TLS (Transport layer security)
b. Encryption of each IP packet is recommended.

25) Social Engineering Attack: These assaults take place when a hostile individual tricks trustworthy users into disclosing their private information, including passwords, emails, and credit card numbers among other things. This is accomplished, among other things, by using phony emails, phone calls, and online sites.

Fig 2.1.20 Social Engineering Attack
source: www.easydmarc.com

Countermeasures of Social Engineering attack:

a. This type of attack can be reduced if the implementation of strict security policies takes place.
b. Users should be trained about the important credentials and are consequences of it. So that users can understand what not to share.

26) Sybil attack: To establish a connection with a valid user and perhaps gain access to higher privileges within the system, the attacker takes the identity of a user. Peer-to-peer networks are where this assault is most common. To carry out illegal operations in the system, the attacker seeks to exert the most power within the network.

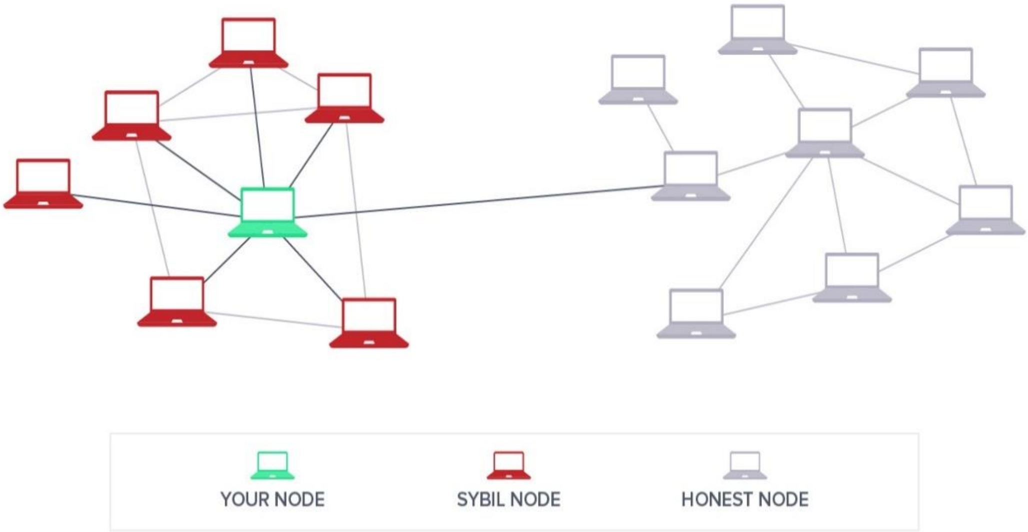


Fig 2.1.21 Sybil Attack

Countermeasures for Sybil attack:

a. Giving different power to different members.
b. Cost to create an Identity – to prevent multiple fake identities in the network, we can put a cost for every identity that aims to join the network.
c. Direct validation: An established member verifies the new joiner of the network.

27) User-to-Root Attack: When a hacker has access to an administrator user's privileges, this kind of attack can occur. By flooding an application with data, this is accomplished. In this attack initially, the attacker accessed a normal user account, and later gain access to the root privileges.

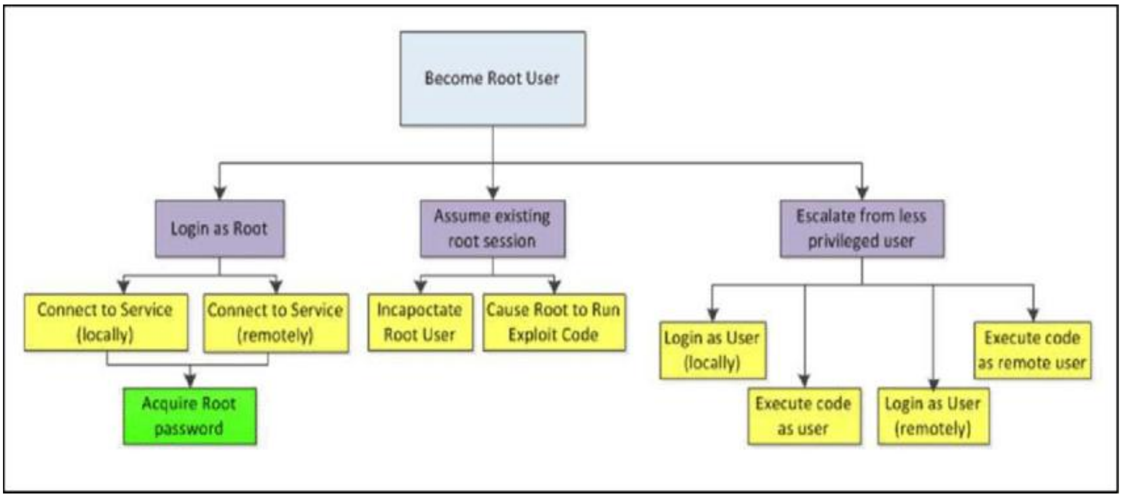

Become Root User
Login as Root
Assume existing root session
Escalate from less privileged user
Connect to Service (locally)
Connect to Service (remotely)
Incapoctate Root User
Cause Root to Run Exploit Code
Login as User (locally)
Execute code as remote user
Acquire Root password
Execute code as user
Login as User (remotely)

Fig 2.1.22 User-to-Root Attack
source:
https://www.researchgate.net/publication/313814425_Data_Loss_Prevention_Management_and_Control_Inside_Activity_Incident_Monitoring_Identification_and_Tracking_in_Healthcare_Enterprise_Environments

Countermeasures for User to Root attack:

a. we can have serious implications for confidentiality and integrity, we recommend using strong passwords and a better authentication mechanism.
b. it is necessary to adopt a privilege separation mechanism to manage privilege access control between different platforms.

28) XML signature wrapping attack: This attack takes place when SOAP communications have a flaw. The server creates a SOAP message when the user submits a request through the browser. The data required to initiate contact between the client and the server may be located using this message. If the attacker is successful in compromising the message, they might be recognized as a trusted user.

*SOAP = Simple Object Access protocol, this protocol is defined to exchange information in the form of XML.

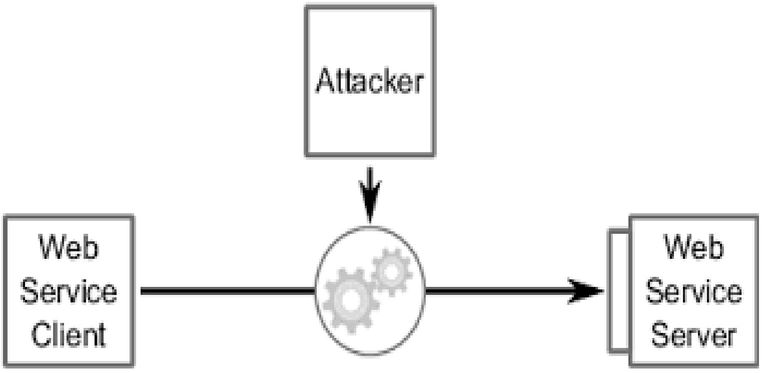


Fig 2.1.23 SOAP-Simple Object Access Protocol

Countermeasures for XML signature wrapping attack:

a. we can use tools for vulnerability detection.
b. Also, we can do manual verification.

2.2 Our Main Focus

After carefully examining numerous SaaS-based issues, we decided to focus on DDoS attacks. This choice was made after analyzing several factors, including the rising frequency and severity of DDoS attacks, and the potential negative effects these attacks could have on SaaS applications and services, As enterprises and individuals increasingly rely on SaaS-based technologies, we also took this into account. To guarantee the availability and security of these services, we needed solid, scalable solutions. We are aiming to make a fast and efficient machine learning solution that can aid in defending SaaS applications and services against this evolving danger by concentrating on DDoS attack mitigation.

2.3 Detection of DDoS Attacks in Cloud-based Services Using Machine Learning Techniques

2.3.1 DDoS Attack

A Distributed Denial of Service (DDoS) attack is a type of cyberattack in which an attacker floods the target system with traffic, making it unreachable to normal users. DDoS attacks on cloud computing services based on software as a service (SaaS) can target the service provider's infrastructure, interrupting access to the SaaS application for all customers.

Severe downtime, slow response times, service interruptions, financial loss, reputational harm, and complete outages are examples of DDoS attack symptoms that can have a big impact on a company or organization.

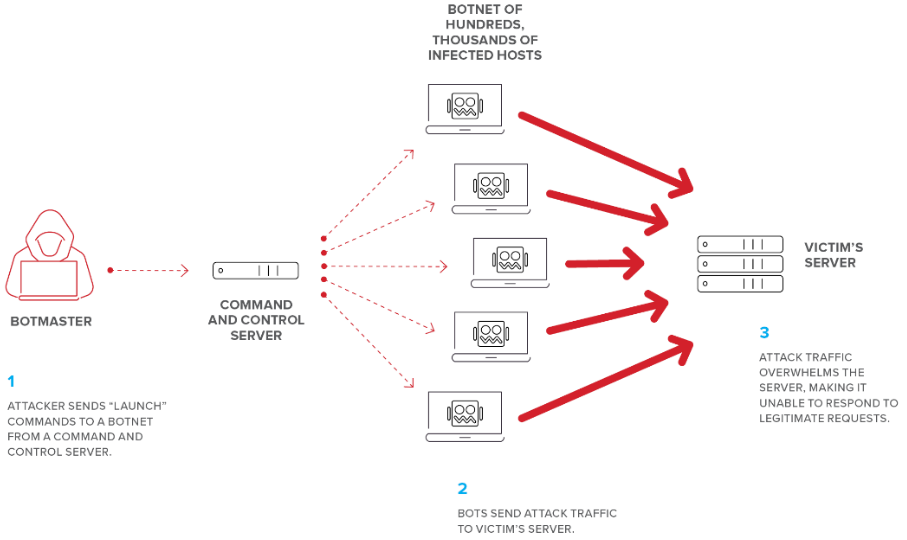

BOTNET OF HUNDREDS, THOUSANDS OF INFECTED HOSTS
BOTMASTER
COMMAND AND CONTROL SERVER
VICTIM'S SERVER
1
ATTACKER SENDS "LAUNCH" COMMANDS TO A BOTNET FROM A COMMAND AND CONTROL SERVER.
2
BOTS SEND ATTACK TRAFFIC TO VICTIM'S SERVER.
3
ATTACK TRAFFIC OVERWHELMS THE SERVER, MAKING IT UNABLE TO RESPOND TO LEGITIMATE REQUESTS.

Fig 2.3.1 DoS Attack

source: F5 Networks

Note: The above-shown example is not a distributed attack, since it only targets one organization

DDoS attacks can be conducted by individuals or groups with a variety of motives, including extortion, activism, or personal revenge. Attackers may use more complex methods to avoid detection and magnify the attack. DDoS assaults can be launched using a variety of approaches, including botnets, amplification attacks, and application-layer attacks.

2.3.2 Problem Statement

The growing popularity of cloud computing services based on Software as a Service (SaaS) has made them an appealing target for cyber assaults, notably Distributed Denial of Service (DDoS) attacks. DDoS assaults may create major disruptions and financial losses for both service providers and users. The difficult part is figuring out the attack, reducing its impact, and putting precautions in place to stop it from happening again.

Therefore, there is an urgent need to create reliable and strong detection techniques for DDoS assaults in cloud computing services based on SaaS.

2.3.3 Why is it critical to resolve this issue?

Stakeholder Impact: A DDoS assault might cause significant interruptions to customers, workers, partners, and vendors. When a website or service goes down, it can have an impact on sales and the brand's reputation. Both national security and public safety can be harmed by critical infrastructure or government services.

Problem-related risks: DDoS assaults are becoming more common, as are the dangers associated with them. Attackers are becoming more competent, and attacks are becoming larger and longer in duration. Organizations that do not take adequate protection against DDoS assaults suffer several issues.

Consequences of failing to take appropriate steps to combat DDoS attacks: The repercussions of ignorance can be severe, including lost income, brand image damage, and cleaning and recovery costs. Secondary costs connected with outages and interruptions may include diminished personnel productivity and lost business opportunities.

2.3.4 Proposed Solution

Machine learning models have demonstrated the potential in recognizing abnormal network traffic patterns, which might be used to detect and mitigate DDoS assaults. The detection performance of machine learning models, on the other hand, is dependent on the quality of the training data and the features utilized for classification. Therefore, we propose to develop a robust machine learning-based approach that can accurately detect DDoS attacks in SaaS-based cloud computing services by leveraging appropriate features and training data.

2.3.5 Survey of DDoS Detection Systems

This section discusses the recently developed DDoS Detection Systems employed using the datasets namely ISCX2012, CIC2017, CSECIC2018, KDD'99, and several others. We have researched and analyzed the recently developed DDoS detection systems and made a survey to discuss these detection systems with a table consisting of columns that describe the Name of the paper, the Name of the publisher & Year of Publication, the Datasets used in the model, the Features used in the model, the accuracy of the model, and the Evaluation Parameters performed in the model.

Name of the Paper	Name of the Publisher & Year of Publication	Dataset(s) Used	Algorithm(s) Used	Feature(s) Used	Accuracy	Evaluvation Parameters
LUCID: A Practical, Lightweight Deep Learning Solution for DDoS Attack Detection	IEEE, 2020	ISCX2012 CIC2017 CSECIC2018	CNN with Network traffic preprocessing algorithm	Time, Packet Length, IP Flags, TCP Len, TCP Ack, TCP Flags, TCP Window Size, UDP Len ICMP Type	99.67%	Accuracy, False Positive Rate, Precision, Recall, and F1 Score.
Protocol Specific Multi-Threaded Network Intrusion Detection System (PM-NIDS) for DoS/DDoS Attack Detection in Cloud	IEEE, 2018	KDD'99	ID3 decision tree, Random Forest	protocol, port numbers, traffic, HTTP, telnet, flag, source bytes, destination bytes	99.4%	True Positive Rate (TPR), False Positive Rate (FPR), True Negative Rate (TNR), False Negative Rate (FNR), and Accuracy.

Table 2.3.1 A Survey of DDoS Detection Systems-1

The first paper describes LUCID, a CNN-based DDoS detection architecture that aims for a practical, lightweight implementation with minimal processing overhead and attack detection time. LUCID eliminates the requirement for threshold configuration and feature engineering, making it simpler to use in practice. The study shows that LUCID matches state-of-the-art performance, delivers consistent detection results across a variety of datasets, and is suitable for deployment in resource-constrained contexts.

The report also provides an activation analysis, which explains how LUCID learns to detect DDoS traffic and is missing from previous efforts. Using the ISCX2012, CIC2017, and CSECIC2018 datasets, the suggested model achieves 99.67% detection accuracy in identifying DDoS assaults in cloud computing.

The second research article provides PM-NIDS, an efficient security framework for detecting DoS/DDoS threats in the cloud. To detect intrusions, the framework isolates packets that arrive by protocol and use protocol-specific classifiers. A framework thread handles each queue, extracting pertinent information and applying protocol-specific classifiers to each packet in the queue, as well as generating alarms for identified intrusions.

Using the KDD'99 dataset, the proposed model detects DDoS assaults in cloud computing and achieves 99.4% detection accuracy.

Name of the Paper	Name of the Publisher & Year of Publication	Dataset(s) Used	Algorithm(s) Used	Feature(s) Used	Accuracy	Evaluvation Parameters
An SVM-based framework for detecting DoS attacks in virtualized clouds under changing environment	Journal of Cloud Computing: Advances, Systems, and Applications (JoCCASA), 2018	CAIDA “DDoS Attack 2007”, 1998 FIFA World Cup	SVM classifier	source IPs, packet size, No. of sources, packet rate, Linux kernel features, and Number of requests. HTTP-GET requests. Packets containing 200, 206, and 400 HTTP status codes, Linux kernel features, network bandwidth, system memory, and CPU time	97.4 %	false positive, false negative, attack detection, and accuracy rate
Predicting DOS-DDOS Attacks: Review and Evaluation Study of Feature Selection Methods based on Wrapper Process	IJACSA) International Journal of Advanced Computer Science and Applications , 2021	KDD’99	SVM, Rep tree, BFS, Genetic algorithm.	protocol, port numbers, traffic, HTTP, telnet, flag, source bytes, destination bytes	99.72%	Accuracy parameter, True Positive Rate, and False Positive Rate for all; training, testing, and validation.

Table 2.3.2 A Survey of DDoS Detection Systems-2

The third study introduces an SVM-based framework for identifying DoS assaults in a virtualized cloud environment with changing infrastructure. The solution collects system measurements to train the SVM classifier to distinguish between normal and malicious VM activity. To maintain a filter of resource adjustments effect, the hypervisor analyses and evaluates the effect of changes to resources on the collected data.

The model outperforms classic SVM and Decision-Tree techniques in the presence of infrastructure changes, retaining accuracy rates of up to 97.96% under such alterations using the CADIA DDoS Attack 2007, 1998 FIFA World Cup datasets. Under granting changes, the accuracy was reduced by only 1.43%, which has no substantial influence and may be ignored.

The fourth journal publication emphasizes the significance of feature selection, specifically the employment of Wrapper methods, in optimizing DOS/DDOS cybersecurity intelligence models. They examine and assess various datasets, algorithms, and wrapper techniques typically employed in DOS/DDOS ML models, and present the results in three dashboards.

According to the findings, wrapper techniques can considerably improve the choice of relevant DOS/DDOS features and increase the performance of existing ML systems. Using the KDD'99 dataset, the suggested model detects DDoS attacks with a detection accuracy of 99.72%.

Name of the Paper	Name of the Publisher & Year of Publication	Dataset(s) Used	Algorithm(s) Used	Feature(s) Used	Accuracy	Evaluvation Parameters
Detection System of HTTP DDoS Attacks in a Cloud Environment Based on Information Theoretic Entropy and Random Forest	John Wiley & Sons, Inc 2018	CIDDS-001	Recursive Feature Elimination with Cross-Validation	Source/destination ports, source/destination IP	97%	ROC and AUC curves, Processing time, Accuracy
Feature selection for DDoS detection using classification machine learning techniques	IAES International Journal of Artificial Intelligence (IJ-AI), 2020	Used CICFlowMeter-V3 online application to generate the dataset.	Naive Bayes, Random forest, Neural Network, SVM, K-nearest neighbor.	Source address, Destination Address, packet Id, packet size, Number of packets, Number of Bytes.	98.4%	Accuracy parameter, false positive rate, attack detection.

Table 2.3.3 A Survey of DDoS Detection Systems-3

The fifth study provides a detection system for HTTP DDoS attacks in the cloud based on three techniques: network traffic entropy measurement, preprocessing, and classification with Random Forest ensemble classifiers. The suggested approach outperforms each classifier tested on the dataset as well as modern HTTP DDoS detection algorithms. The authors propose that future work should include practical deployment of the technique and testing against various HTTP DDoS tools. The suggested model has been used to detect DDoS attacks, and it yields 97% detection accuracy when utilizing the CIDDS-001 dataset.

The sixth research paper describes a new dataset created with the CICFlowmeter-V3 online program that covers contemporary assaults and includes 25 features and five classes. The dataset was used to train and evaluate five classification algorithms: Random Forest, Support Vector Machine (SVM), Neural Network, Naive Bayes, Random Forest, and KNN. Random Forest scored the greatest accuracy of 98.70% and a Weighted Average of 98.4%, confirming its capacity to detect DDoS attacks on the to-be-developed application.

Following an examination of the recently developed detection techniques for DDoS attacks discussed above, we propose to create a machine learning model capable of accurately detecting DDoS attacks in SaaS-based cloud computing services by training the model with the CSE-CIC-IDS2018, CICIDS2017, and SDN-DDoS (ICMP, TCP, UDP) 2020 datasets.

3 Data and Methods

3.1 Overview of Implementation and Design of the Model

Firstly, DDoS raw data is collected from open sources. A total of three datasets are collected. After collection, data is processed to construct the final datasets for modeling. Data processing includes data cleaning, the transformation of data types, and dataset splitting. This is followed by the model selection process. The models are trained with all tree datasets using five different algorithms; logistic regression, k-nearest neighbor, decision tree, random forest, and gradient boost. Finally, the model is tested with unseen data. The results are evaluated using several performance metrics.

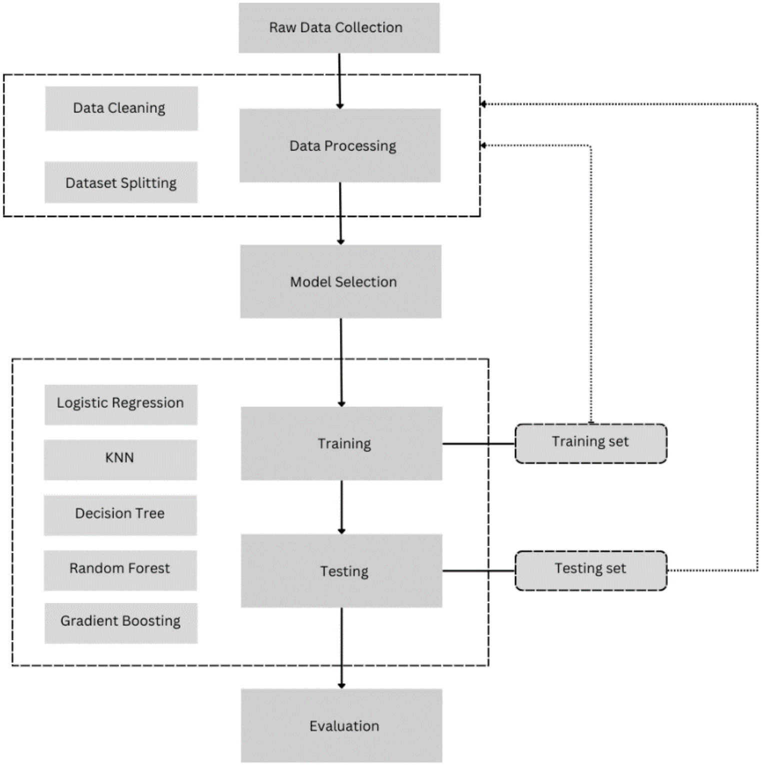


Fig 3.1.1 An illustration of the proposed solution shows the flow of the supervised learning method used in this model

3.2 Datasets

3.2.1 CICIDS2017

The CICIDS2017 dataset is the result of a 5-day simulation from Monday to Friday, and it includes network traffic in both packet and bidirectional flow forms. It is worth mentioning that for every one of the flows, 80 characteristics were gathered, including more information about the IP addresses and attacks of the simulated multiple attackers. Within usual limits, scripts are used to mimic default and user behavior. The first day is considered typical, and traffic is only moderate. The simulated assaults used DDoS data, and the attack scenario was acquired from CICIDS 2017.

Assault Situation	Target	Intruder IP
DDoS Low Orbit Ion Cannon (LOIC)	Ubuntu 16 205.174.165.68	205.174.165.69
		205.174.165.71
		205.174.165.70

Table 3.2.1 Details of DDoS Attack for CICIDS2017

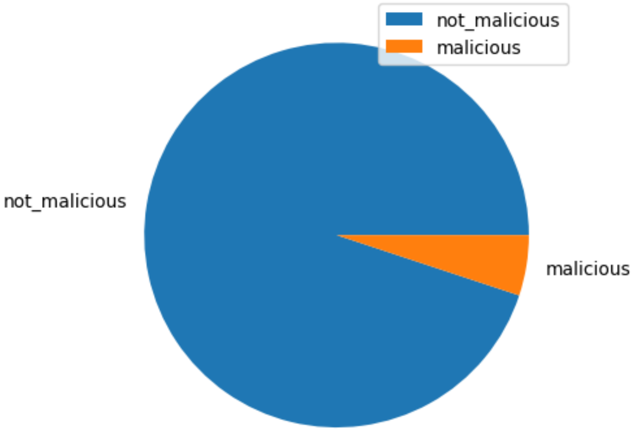


Fig 3.2.1 Visualization of the dataset CICIDS2017

3.2.2 CSE-CIC-IDS2018

The CSECIC2018 dataset is a network traffic dataset derived from a 24-hour online traffic trace on Wednesday. The dataset comprises 44 characteristics for each flow and includes both packet and flow-based data.

The dataset was compiled using the open-source program SiLK and contains both legitimate and malicious traffic. Malicious traffic encompasses a variety of assaults, including Distributed Denial of Service (DDoS), scanning, and brute-force attacks.

The data was gathered in a controlled environment, and scripts were utilized to mimic typical user and attacker behavior.

Tools	Period	Intruder	Target
DDOS Low Orbit Ion Canon (LOIC) for HTTP, UDP, or TCP requests	24 hours	Kali Linux	Windows Vista 7, 8.1, 10 (32-bit), and 10 (64-bit)

Table 3.2.2 Details of DDoS Attack for CSE-CIC-IDS2018

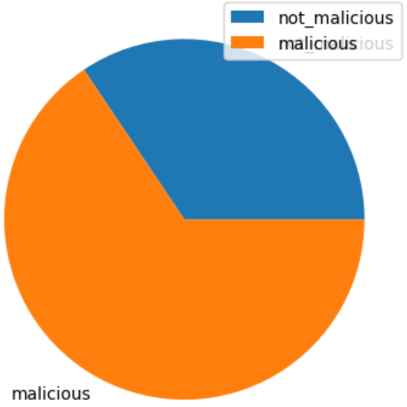


Fig 3.2.2 Visualization of the dataset CSE-CIC-IDS2018

3.2.3 SDN-DDoS (ICMP, TCP, UDP) 2020

The SDN-DDoS (ICMP, TCP, UDP), 2020 dataset is made up of network traffic created by a DDoS assault simulation. The simulation was run with scripts that mimicked default and user behavior within usual restrictions. The dataset comprises 10,000 flows of traffic for ICMP, TCP, and UDP assaults. Each flow has 17 characteristics, such as source and destination IP protocol types, addresses, and port numbers. The dataset is intended to aid in the development and testing of machine learning algorithms for DDoS detection in Software Defined Networking (SDN) settings.

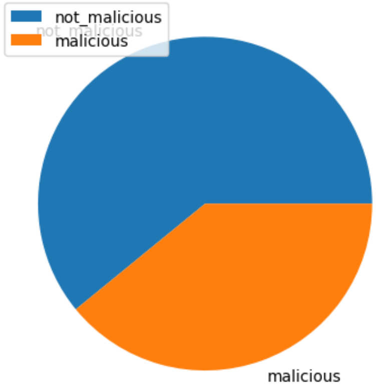


Fig 3.2.3 Visualization of the dataset SDN-DDoS (ICMP, TCP, UDP) 2020

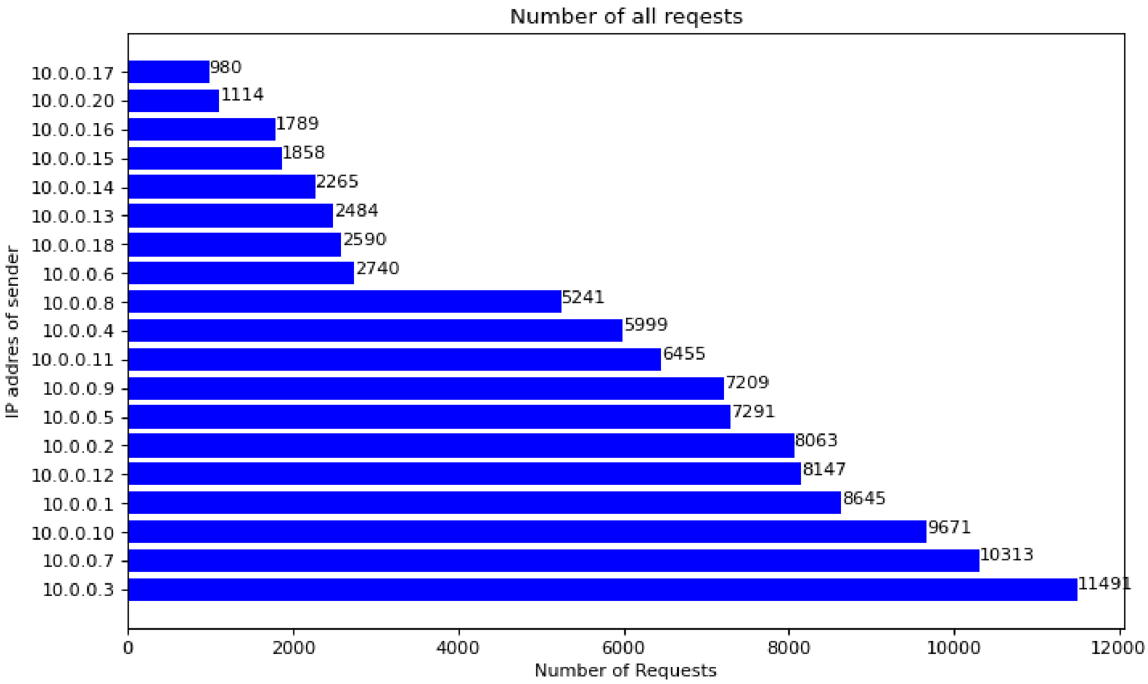


Fig 3.2.4 Representation of the Total Number of requests versus the IP address of the sender

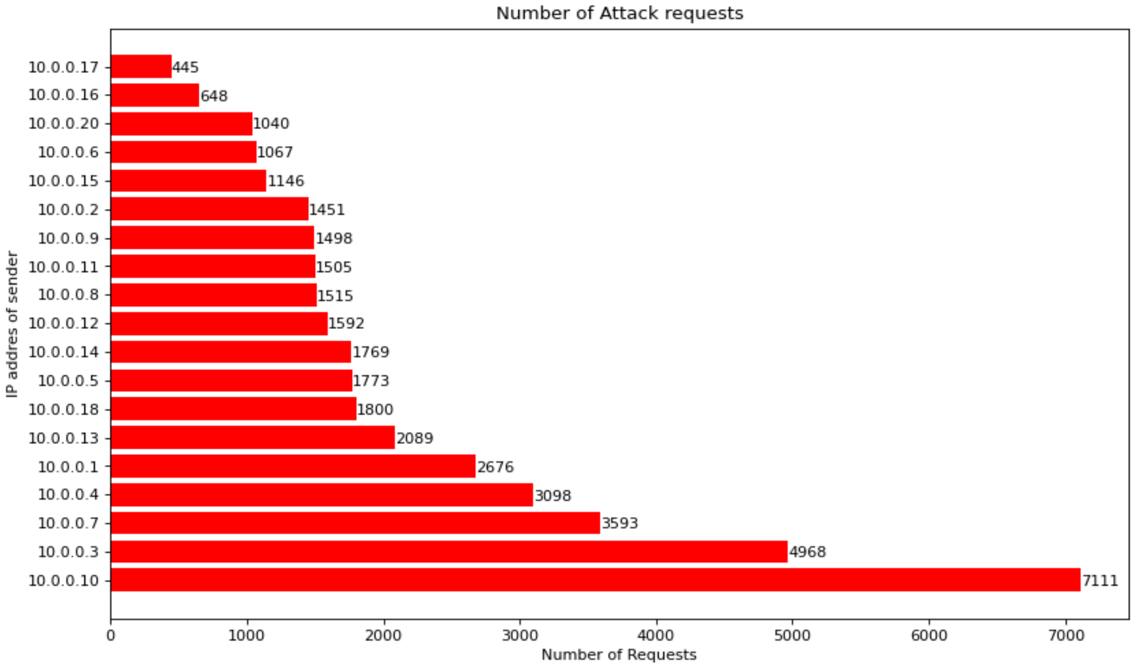


Fig 3.2.5 Representation of the Number of Attack requests versus the IP address of the sender

3.3 Data Preprocessing

We have labeled all the ‘DDOS’ attacks as 1 and all other attacks or intrusions as 0. Since our datasets have many redundant features we removed them as those features make it difficult to understand and analyze the dataset. Also, we have preprocessed our dataset after visualization to reduce any kind of Overfitting of the models.

We have identified and removed all the irrelevant features as it increases complexity and decreases the performance of the model, removing redundant features decreases the computational resources as well as computation time. So, for CICIDS – 2017,2018, the number of features or columns used for training has been reduced from 78 to 39, and for the SDN dataset we have reduced the features from 23 to 16, This reduction in the feature set is a result of the preprocessing operations, and it increases the performance and overall computation time of our model.

3.4 Model Selection

Once the data is processed, the next step involves selecting the most appropriate algorithm for DDoS detection and classification. The chosen algorithms, including Logistic Regression, K-Nearest Neighbors (KNN), Decision Tree, Random Forest, and Gradient Boosting, are specifically tailored for their suitability in detecting DDoS attacks. The selection process takes into account factors such as the algorithms' performance, computational requirements, interpretability, and robustness.

3.4.1 Logistic Regression

Logistic regression is a statistical analysis technique that is specifically designed for situations where the dependent variable is binary. It helps in understanding the relationship between a binary outcome and other independent variables that can have non-binary values.

Table 3.4.1 Pseudo Code for the Logistic Regression Algorithm

Algorithm 1 Logistic Regression

```
given α, {(x i , y i )}
step 1: initialize a = <1,……,1>T.
step 2: perform feature scaling on the examples’ attributes
repeat until convergence.
Step 3: for each j = 0, .., n:
a`j = aj+ α Σi(yi − ha(xi ))Xj^i
for each j = 0, .., n:
aj = aj, output a
```

3.4.2 Decision Tree

The root node of a decision tree is used to evaluate observations and classify them based on their associated attributes. Each node in the tree represents a distinct property, with possible values corresponding to various categories or outcomes.

Table 3.4.2 Pseudo Code for the Decision Tree Algorithm

```
Algorithm 2 Decision Tree

∀ attributes a1, a2, …, an.
Step 1: Find the attribute that best divides the training data using the information gained.
Step 2:  a_best ← the attribute with the highest information gain.
Step 3: Create a decision node that splits on a_best.
Step 4: Recursively add the nodes from the sub-lists created by splitting on a_best to the node.
```

The approach estimates the information gained for each feature in the training dataset starting from the root node. The discriminatory strength of features in distinguishing between target groups is measured by information gain. Higher information gain features are more useful for effectively identifying observations. The root node is then replaced with the feature that provides the most information gain, and the algorithm continues to partition the dataset into subsets based on the feature that was chosen.

3.4.3 K-Nearest Neighbours (KNN)

It is a classifier based on instances. When the k-NN algorithm is used, examples in a dataset are stored in a dimensional space, and new instances are tagged based on their similarity to existing instances. These are referred to as neighbors. If x is the most similar class for the surrounding observations, a new instance of x is created. A distance function is used to determine the similarity of two instances. The distance function utilized in this model is Euclidean.

Table 3.4.3 Pseudo Code for the k-NN Algorithm

```
Algorithm 3 K-Nearest Neighbours

Step 1: Let S = {a1, a2, …, an}, where S represents the training set and a represents article documents
Step 2:  k ← the desired number of nearest neighbors
Step 3: Compute d(x,y) between new instance i and all a ∈ S
Step 4: Select the k closest training samples to i
Step 5: Classi ← best voted class end
```

3.4.4 Random Forest

Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. It improves the accuracy and robustness of the model by reducing overfitting and increasing generalization.

Table 3.4.4 Pseudo Code for the Random Forest Algorithm

Algorithm 4 Random Forest

```
Require IDT (a decision tree inducer), T (the number of iterations), S (the training set), μ (the subsample size), N (the number of attributes used in each node)
t ← 1
repeat
St ← Sample μ instances from S with replacement.
Build classifier Mt using IDT(N) on St
t++
until t > T
```

3.4.5 Gradient Boosting

Gradient Boosting is another ensemble method that builds an additive model by sequentially adding weak learners, such as decision trees, to correct the mistakes made by previous models. It is known for its high predictive power and ability to handle complex relationships in the data.

Table 3.4.5 Pseudo Code for the Gradient Boost Algorithm

Algorithm 5 Gradient Boost

```
Step 1: Initialize the residuals to be the target variable.
Step  2: Initialize the model to be the mean of the target Variable.
Step 3: Loop over the number of trees to train.
        a. Train a decision tree on the residuals.
        b. Use the decision tree to predict the residuals.
        c. Update the model by adding a fraction of the prediction multiplied by learning rate.
        d. Update the residuals by subtracting the prediction from the target variable.
Return the final model.
```

3.5 Training and Testing

After the algorithm selection, the datasets are divided into a training set and a testing set. This division allows for the evaluation of the model's performance on unseen data, providing a realistic assessment of its effectiveness in real-world scenarios. The models are then trained using the training set and subsequently tested using the testing set to measure their ability to accurately identify and classify DDoS attacks.

For the SDN-DDoS2020 dataset, we have split the dataset as 80:20 for training and testing respectively. And for CICIDS2017 and CSE-CIC-IDS2018 we have split the dataset as 70:30 for training and testing respectively.

3.6 Evaluation

Following the training and testing phase, the trained models are evaluated using various performance metrics, including Accuracy, Precision, Recall, F1-score, and AUC-ROC.

3.6.1 Accuracy

Accuracy measures the overall correctness of the model's predictions and is defined as the ratio of the number of correct predictions to the total number of predictions.

Accuracy = (Number of Correct Predictions) / (Total Number of Predictions)

3.6.2 Precision

Precision measures the proportion of correctly predicted positive instances (true positives) out of all instances predicted as positive (both true positives and false positives).

Precision = (True Positives) / (True Positives + False Positives)

3.6.3 Recall (Sensitivity or True Positive Rate)

Recall measures the proportion of correctly predicted positive instances (true positives) out of all actual positive instances (true positives and false negatives).

Recall = (True Positives) / (True Positives + False Negatives)

3.6.4 F1 Score

The F1 score combines precision and recall into a single metric and is useful when you want to find a balance between precision and recall. It is the harmonic mean of precision and recall.

F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

3.6.5 AUC-ROC

AUC-ROC is a metric used for binary classification problems. It measures the model's ability to discriminate between positive and negative instances.

It calculates the area under the Receiver Operating Characteristic (ROC) curve, which shows the trade-off between the true positive rate (TPR) and false positive rate (FPR) at different classification thresholds. The ROC curve plots TPR against FPR, and the AUC-ROC represents the area under this curve. A higher AUC-ROC value indicates better model performance.

These metrics provide valuable insights into the models' overall effectiveness in detecting DDoS attacks and help in assessing their performance across different evaluation criteria. By comparing the performance of different algorithms and analyzing the evaluation results, a detailed report can be generated.

This report provides valuable information on the effectiveness of each algorithm in detecting DDoS attacks, enabling informed decisions on the selection of the most appropriate algorithm for future DDoS detection efforts.

4 Results and Discussion

4.1 Overview

The goal of this evaluation is to analyze the performance of the different DDoS datasets in terms of their capacity to detect intrusion (via a DDoS attack). From the results, we can conclude that all three dataset performance is best overall, achieving an accuracy rate of 99% across all models, and an F1-Score of 99%, denoting that a model trained with all three datasets are performing well, as it is correctly predicting threats (precision) and captures all relevant cases of malicious traffic (recall) at a 99% rate across all models.

From a model point of view Random Forest, Decision Tree, and Gradient Boost are performing best for all the datasets (CICIDS-2017,2018 and SDN dataset) using this model we are achieving accuracy up to 99%, and we have also achieved precision and F1-score of 100%. Our Logistic Regression performance is the lowest among all datasets, there the lowest accuracy we are achieving is 59%, an F1-Score of 42%, and a precision of 47% using the SDN dataset. So, using all these models we are achieving decent evaluation metrics(accuracy, precision, F1-score).

Datasets	Evaluation Parameters	Logistic Regression	K-NN	Random Forest	Decision Tree	Gradient Boost
CICIDS2017	Accuracy	0.989	0.999	0.999	0.999	0.999
	Precision	0.979	0.999	0.999	0.999	0.999
	Recall	0.998	0.999	0.999	0.999	0.999
	F-Measure	0.989	0.999	0.999	0.999	0.999
	Computation Time	9.446 sec	0.044 sec	26.951 sec	1.351 sec	80.521 sec
CSE-CIC-IDS2018	Accuracy	0.999	0.999	0.999	0.999	1.000
	Precision	0.998	0.999	0.999	0.999	1.000
	Recall	1.000	1.000	1.000	1.000	1.000
	F-Measure	0.999	0.999	0.999	0.999	1.000
	Computation Time	9.017 sec	0.031 sec	4.333 sec	0.304 sec	36.978 sec
SDN DDoS 2020	Accuracy	0.596	0.891	1.000	1.000	0.993
	Precision	0.477	0.887	1.000	1.000	0.987
	Recall	0.378	0.825	1.000	1.000	0.996
	F-Measure	0.422	0.855	1.000	1.000	0.992
	Computation Time	0.570 sec	0.020 sec	12.337 sec	0.479 sec	26.863 sec

Table 4.1.1 Performance Metrics of Each Dataset

4.2 CICIDS2017

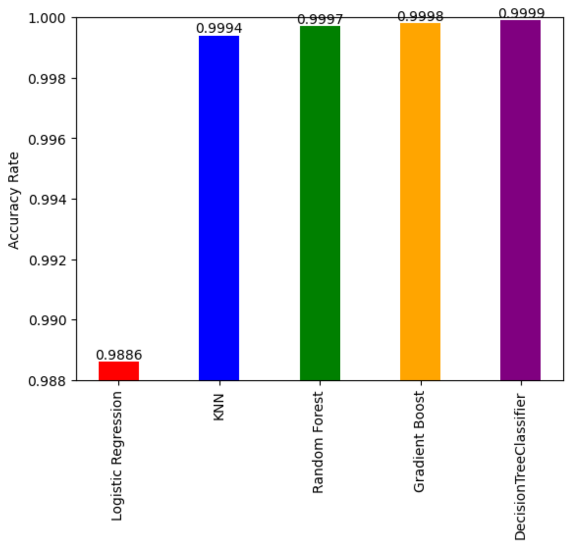


Fig 4.2.1

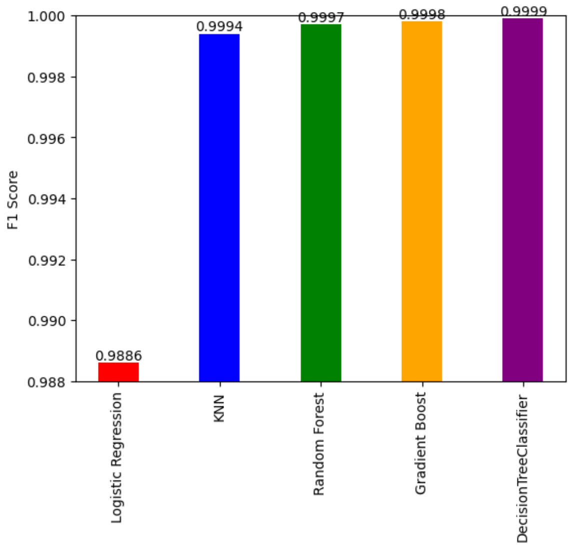


Fig4.2.2

Fig 4.2.1 Accuracy Rate for CICIDS2017

Fig 4.2.2 F1-Score for CICIDS2017

4.3 CSE-CIC-IDS2018

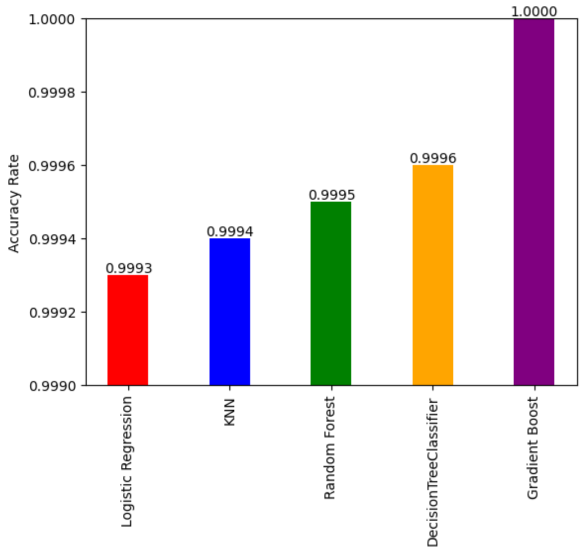


Fig 4.3.1

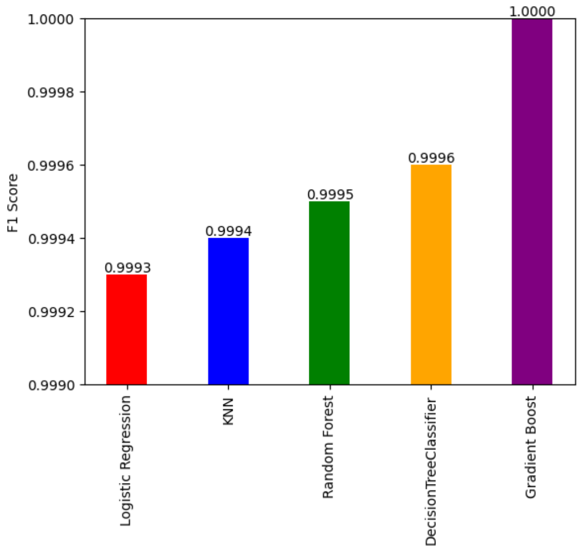


Fig4.3.2

Fig 4.3.1 Accuracy Rate for CSE-CIC-IDS2018

Fig 4.3.1 F1-Score for CSE-CIC-IDS2018

4.4 AUC-ROC for SDN-DDoS (ICMP, TCP, UDP) 2020

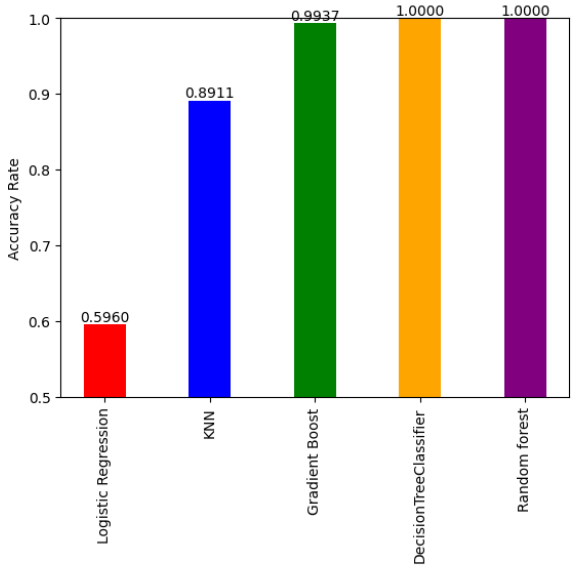


Fig 4.4.1

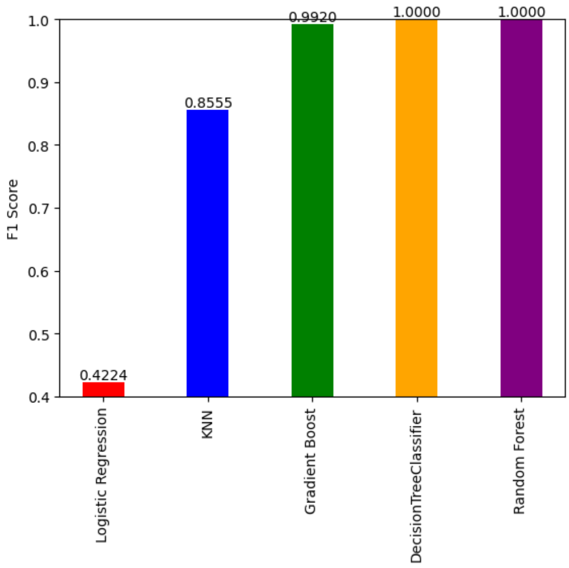


Fig 4.4.2

Fig 4.4.1 Accuracy Rate for SDN-DDoS 2020

Fig 4.4.2 F1-Score for SDN-DDoS 2020

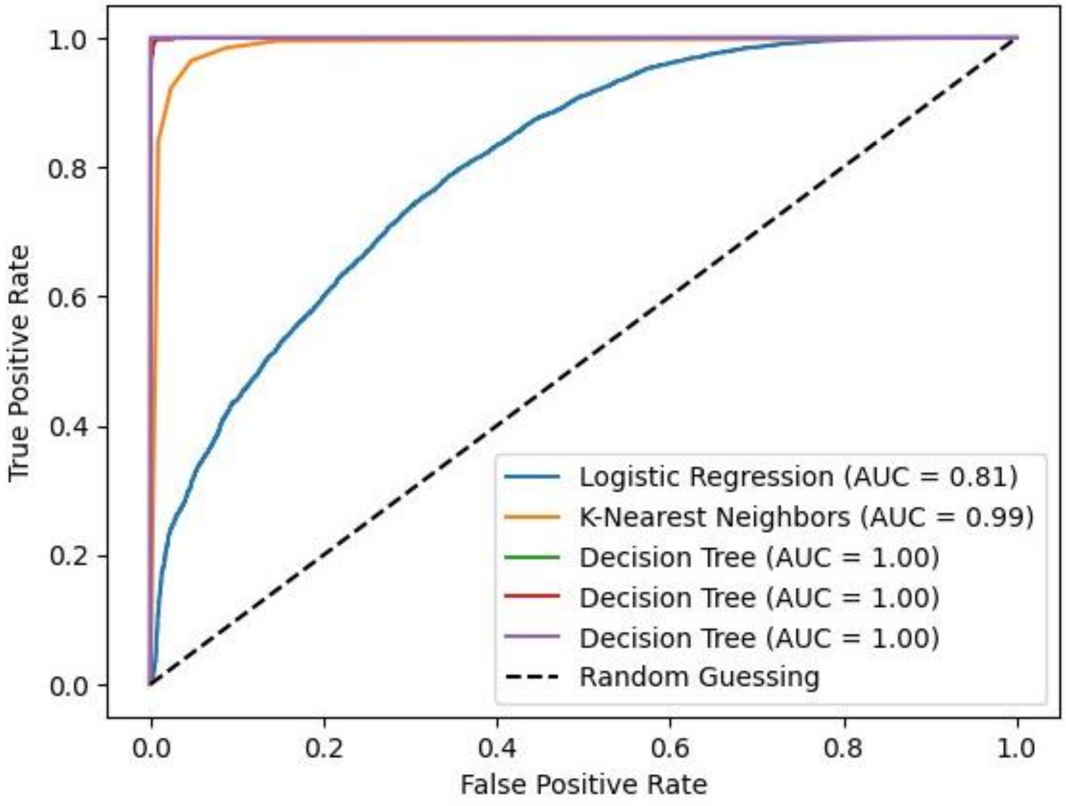


Fig 4.4.3 AUC-ROC for SDN-DDoS 2020

5 Conclusion

Firstly, we have discussed the SaaS services and the security challenges in SaaS Services. In the related work, the first part is the review of attacks and their possible countermeasures in SaaS Services. After conducting this, we have decided to focus on DDoS attacks due to the rising frequency, the potential negative effect, and the severity of these attacks on SaaS applications and services. So the second part discusses the survey of recently developed DDoS Detection Systems on some well-known datasets.

Our Report proposes an approach for detecting DDoS attacks in cloud environments through the utilization of machine learning techniques. We have developed a robust DDoS attack detection system. Our evaluation is based on three prominent datasets CICIDS2017, CSE-CIC-IDS2018, and SDN-DDoS (ICMP, TCP, UDP) 2020. The models we made are trained with all tree datasets using five different algorithms; logistic regression, k-nearest neighbor, decision tree, random forest, and gradient boost.

The utilization of multiple datasets, including CICIDS2017, CSE-CIC-IDS2018, and SDN-DDoS (ICMP, TCP, UDP) 2020 provides a comprehensive evaluation of our system's performance across various attack scenarios. This enhances the reliability and generalizability of our findings, reinforcing the effectiveness of our proposed solution. By using logistic regression, KNN, Random Forest, Decision tree, and Gradient Boost machine learning algorithms our model can effectively analyze network traffic patterns, identifying anomalies associated with DDoS attacks. Among these; Random Forest, Decision Tree, and Gradient Boost are performing the best and Logistic Regression performance is the lowest for all the datasets using this model.

Through our experiments, we have achieved highly promising results in accuracy rates, with performance exceeding 99%. This demonstrates the effectiveness of our proposed system in accurately identifying and preventing DDoS attacks within cloud environments.

In summary, our research contributes to DDoS attack prevention in cloud environments by introducing a machine learning-based detection system. With accuracy rates surpassing 99% and utilizing diverse datasets, our approach proves to be highly reliable and capable of providing robust security measures against DDoS attacks at the source side in the cloud.

5.1 Further Works

After training all the models using all the datasets, we have made a web app using the Python libraries such as Streamlit and Pickle, using the web app any user who feels his system is under any kind of DDOS attack can use the web app to determine whether the traffic was malicious or non-malicious. The user gives captures the web traffic from wireshark or some other tools and gives that file to our app where it pulls the data from the file and determines the result.

In the future, we are open to making a Chrome extension based on this idea, which actively observes the web traffic and determines the assault.

References

1. Díaz de León Guillén, Miguel Ángel, Morales-Rocha, Víctor, and Fernández Martínez, Luis Felipe. ‘A Systematic Review of Security Threats and Countermeasures in SaaS’. 1 Jan. 2020 : 635 – 653.

2. Mohamad Mulham Belal, Divya Meena Sundaram, Comprehensive review on intelligent security defenses in the cloud: Taxonomy, security issues, ML/DL techniques, challenges and future trends, Journal of King Saud University - Computer and Information Sciences, Volume 34, Issue 10, Part B, 2022, Pages 9102-9131, ISSN 1319-1578, https://doi.org/10.1016/j.jksuci.2022.08.035.

3. Thabit, Fursan & Al-ahdal, Abdulrazzaq & Alhomdy, Sharaf & Jagtap, Sudhir. (2020). Exploration of Security Challenges in Cloud Computing: Issues, Threats, and Attacks with their Alleviating Techniques. Journal of Information and Computational Science. 10. 35-59.

4. Sail, Soufiane & Halima, Bouden. (2017). SaaS Cloud Security: Attacks and Proposed Solutions. Transactions on Machine Learning and Artificial Intelligence. 5. 10.14738/tmlai.54.3194.

5. Bachelor’s Thesis, Bachelor of Engineering, Information and Communication Technology, Turku University of Applied Sciences, Prabin Pandey, Security Attacks in Cloud Computing.

6. Chouhan, Pushpinder & Yao, Feng & Yerima, Suleiman & Sezer, Sakir. (2015). Software as a Service: Analyzing Security Issues.

7. S. Alam, M. Muqeem and S.A. Khan, Review on security aspects for cloud architecture, International Journal of Electrical and Computer Engineering 8(5) (2018), 3129–3139.

8. V. Casola, A. De Benedictis, M. Rak and E. Rios, Security-by-design in clouds: A security-SLA driven methodology to build secure cloud applications, Procedia Computer Science 97 (2016), 53–62. doi:10.1016/j.procs.2016.08.280.

9. R. Charanya, M. Aramudhan, K. Mohan and S. Nithya, Levels of security issues in cloud computing, International Journal of Engineering and Technology 5(2) (2013), 1912–1920.

10. L. Coppolino, S. D’Antonio, G. Mazzeo and L. Romano, Cloud security: Emerging threats and current solutions, Computers & Electrical Engineering 59 (2017), 126–140. doi:10.1016/j.compeleceng.2016.03.004.

11. P. Derbeko, S. Dolev, E. Gudes and S. Sharma Security and privacy aspects in MapReduce on clouds: A survey, Computer Science Review 20 (2016), 1–28. doi:10.1016/j.cosrev.2016.05.001.

12. J.B. Hong, A. Nhlabatsi, D.S. Kim, A. Hussein, N. Fetais and K.M. Khan, Systematic identification of threats in the cloud: A survey, Computer Networks 150 (2019), 46–69. doi:10.1016/j.comnet.2018.12.009.

13 S. Iqbal, M.L.M. Kiah, B. Dhaghighi, M. Hussain, S. Khan, M.K. Khan, and K.-K.R. Choo, On cloud security attacks: A taxonomy and intrusion detection and prevention as a service, Journal of Network and Computer Applications 74 (2016), 98–120. doi:10.1016/j.jnca.2016.08.016.

14 J. Jang-Jaccard and S. Nepal, A survey of emerging threats in cybersecurity, Journal of Computer and System Sciences 80(5) (2014), 973–993. doi:10.1016/j.jcss.2014.02.005.

15 A. Joshi, S.T. King, G.W. Dunlap, and P.M. Chen, Detecting past and present intrusions through vulnerability-specific predicates, in ACM SIGOPS Operating Systems Review, Vol. 39, ACM, 2005, pp. 91–104.

16. M.A. Khan, A survey of security issues for cloud computing, Journal of Network and Computer Applications 71 (2016), 11–29. doi:10.1016/j.jnca.2016.05.010.

17. N. Khan and A. Al-Yasiri, Identifying cloud security threats to strengthen cloud computing adoption framework, Procedia Computer Science 94 (2016), 485–490. doi:10.1016/j.procs.2016.08.075.

18. K. Krombholz, H. Hobel, M. Huber and E. Weippl, Advanced social engineering attacks, Journal of Information Security and Applications 22 (2015), 113–122. doi:10.1016/j.jisa.2014.09.005.

19. P.R. Kumar, P.H. Raj, and P. Jelciana Exploring data security issues and solutions in cloud computing, Procedia Computer Science 125 (2018), 691–697. doi:10.1016/j.procs.2017.12.089.

20. P. Mishra, E.S. Pilli, V. Varadharajan and U. Tupakula, Intrusion detection techniques in a cloud environment: A survey, Journal of Network and Computer Applications 77 (2017), 18–47. doi:10.1016/j.jnca.2016.10.015.

21. C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, A survey of intrusion detection techniques in the cloud, Journal of Network and Computer Applications 36(1) (2013), 42–57. doi:10.1016/j.jnca.2012.05.003.

22. M. Mulazzani, S. Schrittwieser, M. Leithner, M. Huber and E. Weippl, Dark clouds on the horizon: Using cloud storage as an attack vector and online slack space, 2011.

23. A. Philpott, Identity theft – Dodging the own-goals, Network Security 2006(1) (2006), 11–13. doi:10.1016/S1353-4858(06)70323-3.

24. A. Singh and K. Chatterjee, Cloud security issues and challenges: A survey, Journal of Network and Computer Applications 79 (2017), 88–115. doi:10.1016/j.jnca.2016.11.027.

25. G. Somani, M.S. Gaur, D. Sanghi, M. Conti and R. Buyya, DDoS attacks in cloud computing: Issues, taxonomy, and future directions, Computer Communications 107 (2017), 30–48. doi:10.1016/j.comcom.2017.03.010.

26. N. Srinivasu, O.S. Priyanka, M. Prudhvi and G. Meghana, Multilevel classification of security threats in cloud computing, International Journal of Engineering and Technology (UAE) 7(1.5) (2018), 253–257.

27. A. Vasudeva and M. Sood, Survey on sybil attack defense mechanisms in wireless ad hoc networks, Journal of Network and Computer Applications 120 (2018), 78–118. doi:10.1016/j.jnca.2018.07.006.

28. M. Wu and Y.B. Moon, Taxonomy of cross-domain attacks on CyberManufacturing system, Procedia Computer Science 114 (2017), 367–374. doi:10.1016/j.procs.2017.09.050.

29. D. Ye, T.-Y. Zhang and G. Guo, Stochastic coding detection scheme in cyber-physical systems against replay attack, Information Sciences 481 (2019), 432–444. doi:10.1016/j.ins.2018.12.091.

30. Kiourkoulis, Stefanos. (2020). DDoS datasets: Use of machine learning to analyze intrusion detection performance.

31. R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martínez-del-Rincón, and D. Siracusa, "Lucid: A Practical, Lightweight Deep Learning Solution for DDoS Attack Detection," in IEEE Transactions on Network and Service Management, vol. 17, no. 2, pp. 876-889, June 2020, doi: 10.1109/TNSM.2020.2971776.

32. Alkasassbeh, Mouhammd & Al-Naymat, Ghazi & Hassanat, Ahmad & Almseidin, Mohammad. (2016). Detecting Distributed Denial of Service Attacks Using Data Mining Techniques. International Journal of Advanced Computer Science and Applications. 7. 10.14569/IJACSA.2016.070159.

33. Bouzoubaa, Kawtar & Taher, Youssef & Nsiri, Benayad. (2021). Predicting DOS-DDOS Attacks: Review and Evaluation Study of Feature Selection Methods based on Wrapper Process. International Journal of Advanced Computer Science and Applications. 12. 10.14569/IJACSA.2021.0120517.

34. Idhammad, Mohamed & Karim, Afdel & Belouch, Mustapha. (2018). Detection System of HTTP DDoS Attacks in a Cloud Environment Based on Information Theoretic Entropy and Random Forest. Security and Communication Networks. 2018. 10.1155/2018/1263123.

35. R. Patil, H. Dudeja, S. Gawade, and C. Modi, "Protocol Specific Multi-Threaded Network Intrusion Detection System (PM-NIDS) for DoS/DDoS Attack Detection in Cloud," 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Bengaluru, India, 2018, pp. 1-7, doi: 10.1109/ICCCNT.2018.8494130.

36. Abusitta, Adel & Bellaiche, Martine & Dagenais, Michel. (2018). An SVM-based framework for detecting DoS attacks in virtualized clouds under changing environments. Journal of Cloud Computing. 7. 10.1186/s13677-018-0109-4.