SMART INDIA HACKATHON 2022

Basic Details of the Team and Problem Statement

Ministry/Organization Name/Student Innovation: **Ministry of power**

PS Code: **SIH1451**

Problem Statement Title : **Develop a AI/ML tool to detect whether a system firewall router network is compromised**

Software Team Name: **Byte Tech**

Team Leader Name: **Atharva Katurde**

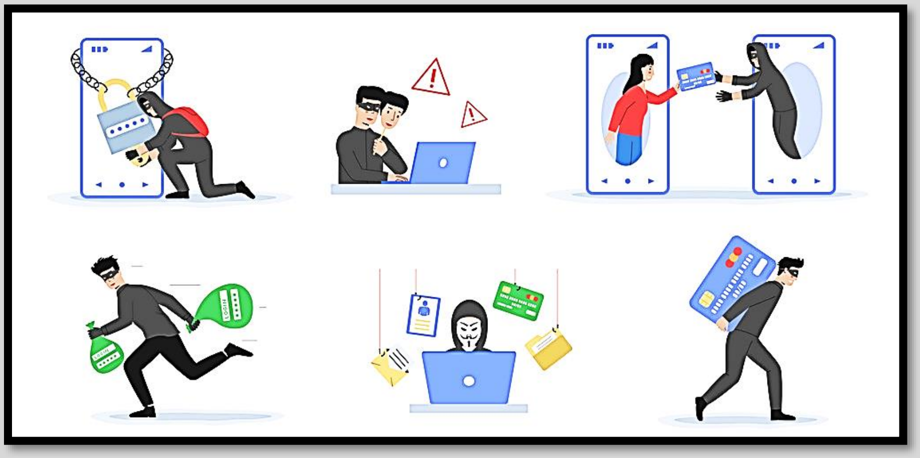
Institute Code (AISHE): **C-41484**
Institute Name: **Anantrao Pawar College of Engineering & Research**

Theme Name: **Blockchain and Cybersecurity**

Idea/Approach Details

Idea/Solution:

- Develop a Behavior-Based Anomaly Detection System
- Data Collection: Gather comprehensive data from various sources, including logs, network traffic, system performance metrics, security events, and user behavior data
- Data Preprocessing: Clean and preprocess the data to handle missing values, outliers, and noise. Normalize or standardize data as needed.
- User Training: Provide training to security personnel on the tool's capabilities and usage to ensure effective utilization.
- Real-time Monitoring: Implement a real-time monitoring system to continuously collect and analyze data from monitored systems and networks.



Objectives:

- ✓ Threat Detection
- ✓ Reduced False Positives
- ✓ Behavior Analysis
- ✓ Cost-Efficiency
- ✓ Reduced Response Time

Use-Cases:

- ✓ Intrusion Detection
- ✓ Malware Detection
- ✓ Anomaly Detection
- ✓ Insider Threat Detection
- ✓ Threat Intelligence Integration
- ✓ Network Segmentation and Access Control

Technology Stack:

- ✓ Programming Languages
- ✓ Machine Learning Frameworks
- ✓ Data Processing
- ✓ Data Preprocessing Libraries
- ✓ Model Training Data

Show Stoppers:

- ✓ Data Quality and Availability
- ✓ Imbalanced Data
- ✓ Scalability and Performance
- ✓ Adversarial Attacks
- ✓ Resource Constraints

ABSTRACT

The "Behavior-Based Anomaly Detection System" presented herein stands at the forefront of modern cybersecurity, offering a proactive and adaptive defense against evolving threats. This comprehensive solution is built upon a structured framework encompassing data collection, preprocessing, user training, and real-time monitoring to ensure the detection and mitigation of security anomalies effectively. Central to the system's capability is its advanced behavioral analysis, driven by machine learning and AI algorithms. By establishing baselines of normal behavior for users and systems, it identifies deviations that may indicate security threats or breaches. Continuous real-time monitoring guarantees that anomalies are detected and addressed promptly, minimizing potential risks. This system's adaptability and customization options allow organizations to tailor their security strategies to meet specific needs and objectives, all while seamlessly integrating with existing security infrastructure. By providing real-time alerts and comprehensive reporting, it empowers security teams to respond decisively and mitigate threats swiftly. In an era of heightened cybersecurity concerns, our Behavior-Based Anomaly Detection System offers a robust and proactive defense, safeguarding critical data, systems, and networks. Join us in fortifying your cybersecurity posture and staying ahead of the dynamic threat landscape with this powerful and adaptive solution

Data Collection:Gather network log data from the system.Collect information about normal system behavior, user activities, and network traffic patterns.Include metadata such as timestamps, source and destination IP addresses, protocols, and any other relevant details.

Feature Engineering:Transform raw log data into meaningful features that can be used for machine learning.Extract relevant information such as frequency of log events, temporal patterns, and anomalies.Consider user-specific patterns, deviations from historical norms, and unusual access patterns.

Labeling:Annotate the dataset with labels indicating whether the log analyzer is compromised or not.Use known instances of compromise or abnormal behavior as training examples.

Model Selection:Choose appropriate machine learning algorithms for anomaly detection. Unsupervised learning methods like Isolation Forests, One-Class SVM, or Autoencoders are common for anomaly detection.Consider ensemble methods for improved performance.

Training:Train the selected model using the labeled dataset.Fine-tune hyperparameters to optimize the model's performance.

Cross-Validation:Validate the model using cross-validation techniques to ensure it generalizes well to new data.

Real-Time Monitoring:Implement the model in a real-time monitoring system for continuous analysis of incoming log data.Establish a feedback loop to update the model based on new data and evolving threats.

Thresholds and Alerts:Set appropriate thresholds for anomaly scores to trigger alerts.Establish a tiered alerting system based on the severity of detected anomalies.

INTRODUCTION

1.0 IDEA OF THE PROJECT:

The idea of the project involves developing an Anomaly-Based Intrusion Detection System (AB-IDS) using machine learning techniques to detect network compromises. Unlike traditional methods that rely solely on predefined Indicators of Compromise (IoCs), this approach focuses on learning the normal behavior of a network and identifying anomalies that may indicate a compromise

Anomaly Detection:Use the trained model to detect anomalies in incoming log data. Anomalies could indicate potential compromise or abnormal activities in the log analyzer.

1.1MOTIVATION OF THE PROJECT:

The motivation for developing an Anomaly-Based Intrusion Detection System (AB-IDS) using machine learning for network compromise detection arises from the limitations of traditional intrusion detection methods, particularly those relying solely on Indicators of Compromise (IoCs).

The motivation for an Anomaly-Based Intrusion Detection System using machine learning from the evolving nature of cyber threats, the limitations of IoC-based detection, and the necessity for a proactive and adaptive defense mechanism to safeguard against emerging risks and unknown attack vectors.

1.2 OVERVIEW:

objectives:threat detection, early warning system , reduced false positives behavior analysis cost-efficiency technology stack: programming languages,machine learning frameworks,data processing,data preprocessing labraries, model traning data use-cases:intrusion detection, malware detection, anomly detection, threat intelligence integration, network segmentation show stoppers: data quality and availabiliy,imbalance data,scalability and performance,adversarial attacks,resource constraints

1.3 Brief Description:

Early detection of a compromise of any compute device is critical for security of critical information infrastructure. While most of infections on ICT are detected using IoCs (Indicators of Compromises), the objective of this problem is to explore techniques for detection of compromise on devices using AI / ML models when the IoC of the compromise is not known. The developer should employ innovative models for non-IoCs based detection of compromise on devices. The evaluation of the solution will be based on the following:

(a) Innovation and ruggedness of the method of detection of compromise.

(b) Utility of the method developed over various types of devices including system network.

(c) Ease of deployment and method of reporting of detected compromise.

(d) Ability to minimize false alarms of compromise.

Technology Bucket:Blockchain & Cybersecurity:In the ever-evolving landscape of cybersecurity, the need for advanced threat detection mechanisms is paramount. Our Behavior-Based Anomaly Detection System is a cutting-edge solution designed to identify and mitigate security threats by monitoring user behavior and system activities. This system is built upon a robust foundation, encompassing data collection, preprocessing, user training, and real-time monitoring to provide comprehensive protection against evolving threats.

Key Components and Features:

1. Data Collection.

2. Data Preprocessing

3.User Training

4 Behavioral Analysis

5. Real-time Monitoring

The system is designed to scale with the evolving needs and complexity of an organization's IT landscape. Our Behavior-Based Anomaly Detection System represents a critical component of modern cybersecurity. By leveraging advanced technologies and user behavior analysis, it empowers organizations to proactively identify and mitigate security threats, safeguarding data, systems, and networks. Join us in enhancing your cybersecurity posture and staying ahead of evolving threats with our robust and adaptive solution.

Analysis:

2.1software requirements:

- coding language
- python
- machine learning libraries
- artificial intelligence libraries

2.2Problem Definition:

Title: Develop a AI/ML tool to detect whether a system network is compromised. The technique should not rely only on IoCs (Indicators of Compromises) detection.

2.3Scope:

The scope of the AI/ML-Based Compromise Detection for System Network Log Analyzers project outlines the specific boundaries and components that the project will cover. The scope helps define what aspects will be addressed and sets the expectations for the project's deliverables.

Data Collection and Preprocessing: Collect network log data from the system, ensuring the inclusion of essential metadata such as timestamps, source/destination IP addresses, protocols, and log types.Implement preprocessing steps to clean and format the data for further analysis.

Feature Engineering: Extract relevant features from the log data to represent system behavior effectively.Features may include the frequency of log entries, time intervals between entries, user-specific patterns, and other behavioral indicators.

Model Selection: Evaluate and select appropriate unsupervised learning algorithms for anomaly detection.Consider algorithms such as Isolation Forest, One-Class SVM, or Autoencoders based on their suitability for network log analysis.Continuous Learning MechanIntegration with IoCsism:Design mechanisms for the model to adapt to changes in the network environment and evolving threat landscapes in real-time.Explore techniques like online learning to enable continuous model updates without disrupting system operations.

Alerting System: Implement an alerting system that categorizes detected anomalies based on their severity.Alerts should be designed to provide actionable information for security teams to investigate potential compromises.

Human-in-the-Loop Validation:Integrate a user interface or system for security analysts to validate and investigate alerts.Allow analysts to provide feedback to improve the model and reduce false positives.

Privacy Measures: Implement privacy-preserving measures for handling sensitive log data.Ensure compliance with privacy regulations and protect user information during the analysis process.

Incorporate IoCs into the anomaly detection model as additional features.Balance IoC-based detection with behavioral analysis to enhance the overall detection capability.

Scalability and Performance: Design the tool to scale with the volume and complexity of network log data in large-scale systems.Optimize performance to handle real-time analysis without compromising accuracy.

Forensic Analysis Support: Develop tools or features for detailed forensic analysis, allowing security teams to trace the origin and impact of potential compromises.Provide contextual information for a more thorough understanding of detected anomalies.

This scoped outline provides a detailed overview of the components and functionalities to be included in the AI/ML-Based Compromise Detection for System Network Log Analyzers project. Adjustments to the scope may be made based on project constraints, resources, and specific organizational requirements.

Detail of Design:

1. Data Collection and Preprocessing:

Data Sources:Collect log data from various sources within the system, including network devices, servers, and applications. Include metadata such as timestamps, source/destination IP addresses, protocols, and log types.

Preprocessing:Clean and filter log entries to remove irrelevant or redundant information.Standardize log formats for consistency.Handle missing or incomplete data through imputation or removal.

2. Feature Engineering:

Feature Selection:Identify relevant features for anomaly detection, such as frequency of log entries, time intervals between entries, and user-specific patterns. Use domain knowledge and statistical analysis to determine feature importance. Normalization and Scaling:Normalize and scale features to ensure uniformity and prevent biases in the model.

Dimensionality Reduction (Optional): Apply dimensionality reduction techniques, such as Principal Component Analysis (PCA), to handle high-dimensional data and improve model efficiency.

3. Model Selection:

Algorithm Selection:Evaluate unsupervised learning algorithms suitable for anomaly detection, such as Isolation Forest, One-Class SVM, or Autoencoders.Consider ensemble methods for improved robustness.

Training:Train the selected model on historical log data to establish a baseline for normal system behavior.Fine-tune hyperparameters to optimize model performance.

4. Continuous Learning Mechanism:

Online Learning:Implement online learning techniques to allow the model to adapt to new data in real-time.Schedule regular model updates to incorporate recent information.

5. Alerting System:

Threshold Setting:Define dynamic thresholds for anomaly detection based on the model's output.Categorize anomalies into severity levels.

Alert Notifications:Implement an alerting system to notify security teams in real-time when anomalies are detected.Include contextual information to aid in the investigation process.

6. Human-in-the-Loop Validation:

User Interface:Develop a user-friendly interface for security analysts to view alerts, investigate anomalies, and provide feedback.

Feedback Loop:Establish a feedback loop to incorporate human validation into the model updating process. Allow analysts to confirm or refute detected anomalies, improving the model over time.

7. Privacy Measures:

Data Encryption:Implement encryption techniques to protect sensitive log data during analysis. Ensure compliance with privacy regulations and organizational policies.

Explore anonymization methods to further protect user identities while still extracting meaningful insights.

8. Integration with IoCs:

IoC Incorporation:Include IoCs as additional features in the model for enhanced detection capabilities.Balance IoC-based detection with behavioral analysis.

9. Scalability and Performance:

Parallel Processing:Design the system for parallel processing to handle large volumes of log data efficiently.Optimize algorithms and infrastructure for scalability.Real-time Analysis:Implement streaming processing capabilities for real-time analysis of incoming log data.

10. Forensic Analysis Support:

Contextual Information:Provide detailed information on detected anomalies, allowing for effective forensic analysis.Include features for tracing the origin and impact of potential compromises.

This detailed design encompasses the key elements of an AI/ML-based compromise detection system for system network log analyzers. The actual implementation would involve the integration of these components and continuous refinement based on feedback and emerging requirements.

Working and Processes:

The working and processes for an AI/ML-Based Compromise Detection system for System Network Log Analyzers involve several steps. Below is a detailed breakdown of the key processes involved:

1. Data Collection: Collect network log data from various sources within the system, including servers, network devices, and applications. Ensure the inclusion of metadata such as timestamps, source/destination IP addresses, protocols, and log types. Regularly update the dataset to include new log entries.

2. Data Preprocessing: Clean and filter log entries to remove irrelevant or redundant information.Standardize log formats for consistency.Handle missing or incomplete data through imputation or removal.

3. Feature Engineering: Identify relevant features for anomaly detection, such as frequency of log entries, time intervals between entries, and user-specific patterns.Normalize and scale features to ensure uniformity and prevent biases in the model. Optionally, apply dimensionality reduction techniques like Principal Component Analysis (PCA).

4. Model Training: Choose and implement unsupervised learning algorithms suitable for anomaly detection, such as Isolation Forest, One-Class SVM, or Autoencoders. Train the model on historical log data to establish a baseline for normal system behavior. Fine-tune hyperparameters to optimize model performance.

5. Continuous Learning: Implement online learning techniques to enable the model to adapt to new data in real-time. Schedule regular model updates to incorporate recent information.Monitor the model's performance after updates.

6. Anomaly Detection: Apply the trained model to incoming log data for real-time anomaly detection. Set dynamic thresholds for anomaly detection based on the model's output. Categorize detected anomalies into severity levels.

7. Alerting System: Implement an alerting system to notify security teams in real-time when anomalies are detected. Include contextual information in alert notifications to aid in the investigation process.Define response protocols based on the severity of detected anomalies.

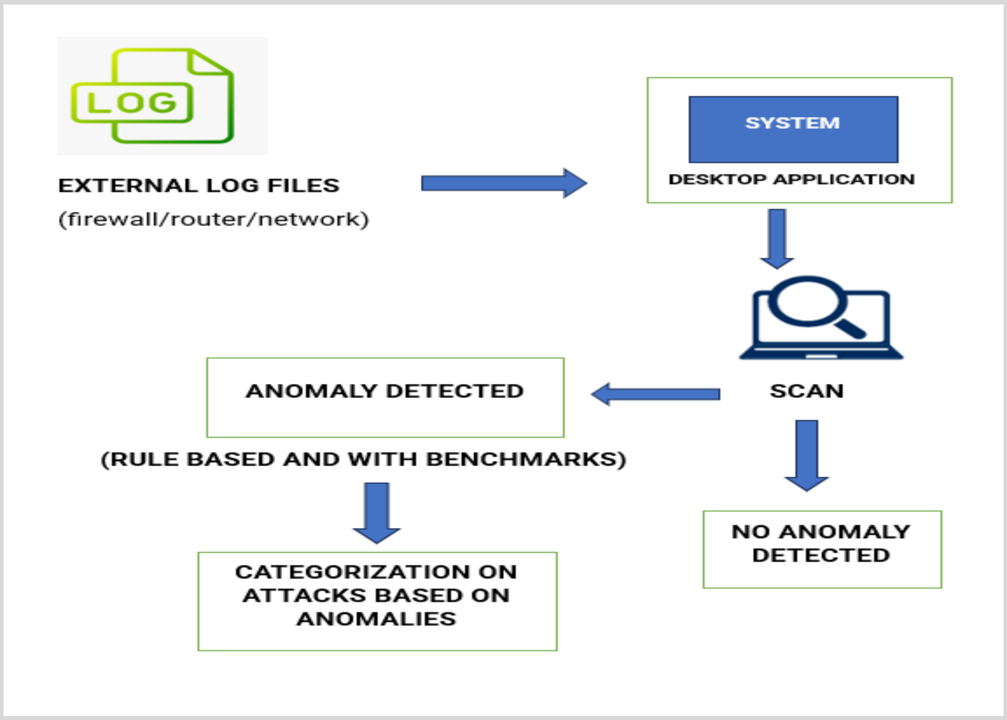
8. Human-in-the-Loop Validation: Develop a user-friendly interface for security analysts to view alerts, investigate anomalies, and provide feedback.Establish a feedback loop to incorporate human validation into the model updating process. Allow analysts to confirm or refute detected anomalies, improving the model over time.

9. Privacy Measures: Implement encryption techniques to protect sensitive log data during analysis.Ensure compliance with privacy regulations and organizational policies.Explore anonymization methods to further protect user identities while still extracting meaningful insights.

10. Integration with IoCs: Incorporate IoCs as additional features in the model for enhanced detection capabilities. Balance IoC-based detection with behavioral analysis to improve overall detection capability.

These processes collectively enable the AI/ML-Based Compromise Detection system to continuously monitor network log data, adapt to changes, detect anomalies, and provide actionable insights for security teams. Regular updates, feedback loops, and continuous learning mechanisms are critical for the system's effectiveness over time.

<u>SYSTEM ARCHITECTURE</u>:


LOG
EXTERNAL LOG FILES
(firewall/router/network)
SYSTEM
DESKTOP APPLICATION
SCAN
ANOMALY DETECTED
(RULE BASED AND WITH BENCHMARKS)
NO ANOMALY DETECTED
CATEGORIZATION ON ATTACKS BASED ON ANOMALIES

- **External Log Files (Firewall/Router/Network**):
  Log files generated by external devices such as firewalls, routers, and network appliances. These logs contain records of events, connections, and activities, providing valuable information for security monitoring and analysis.
- **Desktop Application Scan:**
  The process of scanning desktop applications for vulnerabilities or malicious code. This helps identify security weaknesses that could be exploited by attackers to compromise the integrity or confidentiality of the application and its data.
- **Anomaly Detected (Rule-Based and with Benchmark):**
  Anomaly detection involves identifying patterns or behaviors that deviate from established norms. Rule-based anomaly detection uses predefined rules to flag deviations, while benchmark-based methods compare current behavior against historical data or established benchmarks to detect anomalies in real-time.
- **Categorization on Attacks Based on Anomalies:**
  After detecting anomalies, security systems categorize the identified deviations to understand the nature of potential threats. This categorization helps in assigning appropriate responses or countermeasures based on the type of attack or anomaly detected, enhancing the efficiency of the security system.

Supervised Learning:

The process begins with a robust collection of training data. Supervised learning is a type of machine learning where the algorithm is trained on a labeled dataset, meaning that each input in the training data is associated with a corresponding output or target label. The goal of supervised learning is for the algorithm to learn a mapping from inputs to outputs, so that it can make accurate predictions or classifications on new, unseen dataHere's a breakdown of the key components and process of supervised learning:

Labeled Data: In supervised learning, you start with a dataset that includes both input features and their corresponding correct output labels.

The input features represent the characteristics or attributes of the data, while the output labels are the desired predictions or classifications.

Training Phase: The algorithm is trained on the labeled dataset to learn the patterns and relationships between the input features and output labels.

During training, the algorithm adjusts its internal parameters based on the input-output pairs in the training data, with the goal of minimizing the difference between its predictions and the actual labels.

Model Building: The algorithm builds a model that captures the learned patterns and relationships in the training data.

The nature of the model depends on the specific supervised learning task. For example, it could be a linear regression model for predicting numerical values or a classification model for predicting categories.

Testing and Evaluation: Once the model is trained, it is tested on a separate set of data called the test set, which it has not seen during training.

The performance of the model is evaluated by comparing its predictions on the test set to the actual labels.

Algorithm Used:

Isolation Forest Algorithm:

Isolation: The algorithm works by recursively partitioning the dataset.

It randomly selects a feature and a random split point for that feature.

The selected feature and split point create an isolation (or separation) of the data.

Recursive Partitioning: The process is repeated recursively on the subsets formed by the isolation until individual data points are isolated.

Anomaly Score: The anomaly score of a data point is determined by how quickly it is isolated. Anomalies are expected to be isolated more quickly than normal points.

Decision Function: The isolation process results in a tree structure, and the height of the tree for a particular data point is indicative of its anomaly score.

Points that are isolated with fewer splits or closer to the root of the tree are considered more anomalous.

Thresholding: Anomalies are identified by setting a threshold on the anomaly scores. Data points with scores above the threshold are considered anomalies.

Key advantages of the Isolation Forest algorithm include its ability to handle high-dimensional data, its relatively low computational cost, and its effectiveness in detecting global anomalies (anomalies that differ from the norm in multiple dimensions).

After successful training and evaluation, the trained model can be used to make predictions on new, unseen data.

Anomaly detecting attributes:

- **Timestamps**: Timestamps are fundamental in anomaly detection because they allow algorithms to consider the temporal context of the data. This enables the detection of anomalies that may manifest as deviations from expected patterns over time, helping to build more effective and context-aware anomaly detection systems
- **Source and destination IP address**: Source and Destination IP addresses are fundamental attributes in anomaly detection for network security. By analyzing the communication patterns between these addresses, anomaly detection systems can identify deviations from normal behavior, helping to detect and mitigate potential security threats.
- **Source port and Destination port**: Source port and destination port information is commonly used in network anomaly detection, particularly in the context of network traffic analysis. In networking, communication between devices is facilitated through the use of ports. Ports are numerical identifiers associated with specific processes or services running on a device. In the context of anomaly detection, monitoring source and destination ports helps identify unusual or suspicious patterns of network behavior.
- **Protocol**: In the context of anomaly detection, the term "source protocol" is not a standard or widely used term. However, I'll provide information based on the assumption that you might be referring to the protocol or source of the data in the context of anomaly detection systems. Anomaly detection systems often process data from various sources, and the source protocol could refer to the type of communication or data format used to transmit this data. Here's how the concept of "source protocol" might be relevant:
- **Packet Size:** Source packet size refers to the size of packets transmitted by a source (originating entity) in a network communication. In the context of anomaly detection, analyzing source packet size can be valuable for identifying unusual or suspicious network behavior. Here's how source packet size works in the context of anomaly detection
- **Traffic Category**: Source packet size refers to the size of packets transmitted by a source (originating entity) in a network communication. In the context of anomaly detection, analyzing source packet size can be valuable for identifying unusual or suspicious network behavior. Here's how source packet size works in the context of anomaly detection
- **Application Service**: In the context of anomaly detection, the term "source application service" generally refers to the application or system generating the data that is being monitored for anomalies. The source application service provides the raw data that is analyzed by an anomaly detection system. Here's how the source application service typically works within the broader context of anomaly detection:
- **Duration**: If your dataset involves events or occurrences with associated start and end times, the duration of these events can be used as a feature. Unusual event durations or unexpected changes in duration may indicate anomalies.

Attacks Based on Outputs:

1. **Zero Day Exploit**: A zero-day exploit refers to a cyberattack that takes advantage of a previously unknown or "zero-day" vulnerability in software or hardware. The term "zero-day" implies that the developers have had zero days to address and patch the vulnerability because it is either newly discovered or, in some cases, intentionally kept secret by the attacker
2. **DOS(Denial of Service)**: A Denial-of-Service (DoS) attack is a malicious attempt to disrupt the normal functioning of a network, service, or website by overwhelming it with a flood of illegitimate traffic or requests. The primary goal of a DoS attack is to render the target inaccessible or unusable for its intended users. There are different variations of DoS attacks, including Distributed Denial-of-Service (DDoS) attacks, which involve multiple sources to amplify the impact.
3. **Brute Force attack**: A brute force attack is a cyberattack method where an attacker systematically tries all possible combinations of usernames and passwords until the correct one is found. The goal of a brute force attack is to gain unauthorized access to a system, network, application, or account. This type of attack is based on the assumption that the password is weak or can be easily guessed through exhaustive trial and error.
4. **Port Scanning**: A port scanning attack is a technique used by attackers to discover open ports on a target system. Ports are communication endpoints used by networked applications to exchange data. A port scanning attack involves sending requests to a range of ports on a target system to identify which ports are open and potentially vulnerable to exploitation. Understanding the open ports helps attackers identify services running on the system and potential entry points for further attacks.
5. **Resource Exhaustion**: Long-lived connections, if not properly managed, can consume significant server resources such as memory or CPU over time. Attackers may attempt to exploit this by establishing numerous long-lived connections to exhaust available resources.