Journal of IoT in Social, Mobile, Analytics, and Cloud (ISSN: 2582-1369)
www.irojournals.com/iroismac/

Designing Greedy Algorithm for Cyber Threat Detection

Manas Kumar Yogi[1], Dwarampudi Aiswarya[2]

Computer Science and Engineering Department, Pragati Engineering College (A), Surampalem, A.P., India
E-mail: [1]manas.yogi@gmail.com, [2]aiswarya.d@pragati.ac.in

Abstract

As the digital landscape continues to expand, the complexity and frequency of cyber threats targeting critical information systems also increases. Effective cyber threat detection has become a paramount concern for safeguarding sensitive data and ensuring the uninterrupted operation of various infrastructures. This research introduces a novel approach to cyber threat detection through the design of a greedy algorithm tailored for identifying specific types of threats. The algorithm focuses on a simplified aspect of threat detection, aiming to highlight the potential of greedy algorithms in contributing to the broader field of cybersecurity. The methodology involves monitoring network traffic for signs of port scanning activity, a common precursor to potential cyber-attacks. The algorithm's effectiveness is evaluated in terms of its ability to accurately identify suspicious scanning behavior while minimizing false positives. By presenting this algorithmic framework, the research aims to contribute to the ongoing efforts in enhancing cyber threat detection techniques.

Keywords: Cyber Security, Privacy, Greedy, Malware, Intrusion, Threat

1. Introduction

Greedy algorithms are fundamental optimization techniques used to solve problems by making local optimal choices at each step. While the global optimal solutions that are efficient, intuitive, and often provide practical solutions for a wide range of problems in all cases are not guaranteed by the greedy approach . The importance of greedy algorithms lies in their ability to provide quick and near-optimal solutions in many situations [1].

DOI: https://doi.org/10.36548/jismac.2023.3.002
Received: 05.07.2023, received in revised form: 30.07.2023, accepted: 21.08.2023, published: 02.09.2023

1.1 Benefits of Greedy Algorithms

1. Efficiency: Greedy algorithms are usually straightforward and simple to implement, leading to efficient solutions for many problems. The algorithm offers lower time complexity compared to more complex optimization methods.

2. Intuitiveness: Greedy algorithms typically make decisions based on immediate information, which can align with the intuitive sense of solving problems step by step. This makes them easy to understand and implement.

3. Speed: In cases where globally optimal solutions aren't required, greedy algorithms can provide solutions rapidly, making them suitable for real-time or time-constrained scenarios.

4. Approximation: Even when a greedy algorithm doesn't yield the absolute best solution, it often produces a solution that is close to optimal. This makes them useful for optimization problems where finding the perfect solution is computationally expensive.

5. Space Efficiency: Greedy algorithms can often operate with minimal memory usage, making them suitable for resource-constrained environments.

1.2. Various Applications of Greedy Algorithms [2]

1. Fractional Knapsack Problem: Given items with weights and values, maximize the value of items in a knapsack of limited weight capacity.

2. Huffman Coding: Construct efficient variable-length codes for characters in a text, where frequently used characters have shorter codes.

3. Minimum Spanning Tree (MST): Find the minimum spanning tree of a graph, crucial in network design and optimization.

4. Shortest Path Algorithms: Greedy algorithms like Dijkstra's and Prim's algorithms find the shortest paths and minimum spanning trees in weighted graphs.

5. Activity Selection Problem: Schedule a set of activities with start and finish times to maximize the number of non-overlapping activities.

6. Job Scheduling: Assign a set of jobs to machines with the goal of optimizing criteria like completion time or total turnaround time.

7. Interval Scheduling: Select the maximum number of compatible intervals from a set

of intervals.

8. Graph Coloring: Assign colors to vertices in a graph such that no two adjacent vertices share the same color.

Applying greedy algorithms in cybersecurity is motivated by the need for efficient and practical solutions to address specific aspects of cyber threat detection and mitigation. While the field of cybersecurity encompasses a wide range of complex challenges, there are several scenarios where greedy algorithms can offer valuable contributions [3]:

1. Rapid Decision-Making:

Cybersecurity incidents often require quick decisions to prevent or mitigate damage. Greedy algorithms, with their simple and locally optimal decision-making process, can provide rapid responses in real-time or near-real-time scenarios. For example, in intrusion detection, a greedy algorithm might quickly identify a pattern of suspicious behavior in network traffic, triggering immediate countermeasures.

2. Resource Efficiency:

Many cybersecurity systems operate with limited computational resources, particularly in environments with numerous devices or constrained hardware. Greedy algorithms, which make incremental decisions based on local considerations, can help optimize the utilization of resources while still achieving meaningful threat detection outcomes.

3. Suboptimal yet Effective Solutions:

In some situations, achieving globally optimal solutions in cybersecurity may not be feasible due to time constraints or resource limitations. Greedy algorithms provide approximate solutions that are often "good enough" for practical purposes. This characteristic is particularly valuable in cases where immediate action is required to prevent further damage.

4. Feature Selection and Dimensionality Reduction:

Machine learning and data analysis are integral to modern cybersecurity. Greedy algorithms can help select a subset of relevant features or reduce the dimensionality of data, leading to more efficient processing and improved model performance. This is especially important when dealing with large datasets and limited computational power.

5. Adaptive thresholding:

Greedy algorithms can adaptively adjust detection thresholds based on current conditions. In cybersecurity, where the threat landscape evolves rapidly, dynamically adjusting thresholds can reduce false positives and negatives, enhancing the accuracy of threat detection systems.

6. Anomaly Detection and Pattern Recognition:

Greedy algorithms can be used to identify patterns or anomalies within large datasets. For instance, the algorithms are more commonly used to detect unusual behaviors in user activity, network traffic, or system logs, helping to uncover potential breaches or unauthorized access.

2. Related Work

Research in the field of cyber threat detection has been extensive and diverse, spanning various techniques, methodologies, and technologies. Till date researchers have explored a wide range of approaches to detect and mitigate cyber threats. Below is an overview of some key areas of research and notable contributions in the field of cyber threat detection:

1. Intrusion Detection Systems (IDS)[4-5]:

Intrusion Detection Systems are designed to identify unauthorized or malicious activities within a network. Traditional IDS includes Signature-based and Anomaly-based approaches.

- Snort: Developed by Martin Roesch, Snort is a popular open-source IDS that uses a rule-based signature matching approach to detect known attack patterns.

- Bro (now Zeek): This open-source network analysis framework focuses on protocol analysis to detect anomalies and unusual behaviors.

2. Machine Learning for Threat Detection [6]:

Researchers have extensively explored the application of machine learning techniques for identifying patterns of malicious behavior.

- Random Forests, Support Vector Machines, and Neural Networks: Various machine learning algorithms have been employed to classify network traffic and identify anomalies indicative of cyber threats.

- UNSW-NB15 Dataset: This publicly available dataset has been used for evaluating the performance of machine learning algorithms for intrusion detection.

3. Anomaly Detection [7]:

Anomaly detection focuses on identifying deviations from normal behavior patterns.

- Autoencoders: Deep learning models like autoencoders have been used to learn normal patterns in network traffic and flag deviations.

- Statistical Methods: Researchers have employed statistical techniques to model normal network behavior and identify outliers.

4. Behavioral Analysis:

Behavioral analysis focuses on monitoring the behavior of entities (users, devices) and detecting deviations from their established norms.

- User and Entity Behavior Analytics (UEBA) [8]: These systems use machine learning to detect anomalies in the behavior of users and entities, helping identify insider threats.

5. Threat Intelligence and Sharing:

Researchers have explored the value of threat intelligence feeds and collaborative sharing of threat information.

- STIX/TAXII: Structured Threat Information Expression (STIX) and Trusted Automated Exchange of Indicator Information (TAXII) standards facilitate the sharing of threat information.

6. Honeypots and Deception [8]:

Honeypots and deception technologies involve setting up fake assets to attract attackers and gather intelligence.

- Kippo, Cowrie: These are SSH honeypots designed to capture and analyze attackers' activities.

- Deception Technology Platforms: Commercial platforms offer a suite of tools to deploy deceptive assets and monitor attackers' interactions.

7. Cyber Threat Hunting:

Threat hunting involves actively searching for signs of malicious activity within an organization's network.

- MITRE Adversarial Tactics, Techniques, and Common Knowledge Framework: This framework provides a comprehensive model of adversary behavior, aiding threat hunters in identifying attack techniques. The table.1 below shows the mapping of the greedy algorithm for the cyber threat detection.

Table 1. Mapping of Greedy Algorithms Features with Needs of Cyber Threat Detection [9]

Characteristics of Greedy Algorithms	**Alignment with Cyber Threat Detection Requirements**
Optimization for Immediate Gain	Rapid identification and response to potential threats. Prioritization of high-impact threats.
Local Decision Making	Detection of local anomalies or patterns indicating threats. Quick response to localized suspicious activities.
Efficiency	Real-time or near-real-time analysis of large volumes of data. Swift processing of incoming network traffic.
Heuristics and Rules	Implementation of predefined rules to detect known threat patterns. Identification of specific attack signatures.
Incremental Building	Adaptation and refinement of threat detection strategies over time. Flexibility to address emerging threat vectors.
Limited Information Use	Detection based on partial information, suitable for decentralized and distributed threat landscapes.
Potential for Suboptimal Solutions	May not always result in globally optimal solutions. Some threat instances might be missed.

Trade-off between Exploration/Exploitation	Balancing exploration of new threat patterns and exploitation of known threat signatures.
Simple Implementation	Easier implementation and maintenance of rule-based threat detection systems.

3. Proposed Mechanism

3.1 Greedy Algorithm for Port Scan Detection

Port scanning is a common technique used by attackers to identify potential entry points into a system [10]. This algorithm focuses on detecting suspicious port scanning activity in a network.

Objective: Identify potential port scanning activity by monitoring network traffic.

3.1.1Rule Set for Detecting Port Scanning using the Greedy Approach

A Greedy approach for port scanning involves sequentially testing ports in ascending order to identify open ports. The research follows a few key rules which are followed when implementing a Greedy port scanning algorithm:

1. Sequential Testing: Ports are tested in a sequential order, typically starting from port 1 and going up to the maximum port number. This ensures that all ports are considered for testing.

2. Connection Timeout: Each connection attempt to a port should have a timeout mechanism. If a response (indicating an open port) is not received within the timeout period, the port is considered closed for the purpose of the scan.

3. Response Analysis: The algorithm must analyze the response received after attempting a connection to a port. An open port typically responds differently from a closed port. For example, an open port might respond with a TCP ACK or a specific protocol banner.

4. Response Threshold: Define criteria to determine if a received response indicates an open port. Depending on the protocol and application, the definition of an open port might vary. Some responses might indicate a service is running, while others might indicate a firewall is blocking the response.

5. Logging and Reporting: Keep track of the ports that respond positively (indicating open ports) and log/report the findings for analysis.

6. Rate Limiting: To avoid overwhelming the target host or network, consider implementing rate limiting to control the speed at which port scans are conducted.

7. Avoid Intrusion Detection: Be mindful of intrusion detection systems (IDS) or network monitoring tools that might detect and block aggressive or rapid port scanning activities. Implementing rate limiting and adjusting the scanning speed can help avoid triggering these defenses.

8. Legality and Authorization: Ensures the legal authority to perform port scanning on the target system. Unauthorized port scanning can be considered unethical and potentially illegal.

3.1.2 Greedy Port Scanning Algorithm

Input:

targetIP: IP address of the target system

portRange: Range of ports to scan (e.g., 1-65535)

timeout: Timeout for each port connection attempt

Output:

List of open ports

Algorithm Steps:

Step 1: Initialize an empty list openPorts, to store the open ports found during scanning.

Step 2: For each port p in the portRange

Create a socket connection to the targetIP the current port p

Set a timeout for the connection attempt.

Step 3: Greedy Choice: If the connection attempt is successful (no timeout or error), consider the current port p as open and add it to the openPorts list.

Step 4: Close the socket connection.

Step 5: Repeat steps 2-4 for all ports in the portRange

Step 6: Return the openPorts list.

Termination Criteria:

The algorithm terminates once all ports in the specified range have been scanned.

Updating the Greedy Choice:

The greedy choice in this algorithm is made when a connection attempt to a specific port.

p is successful (no timeout or error). This implies that the port being scanned is open, and it is added to the openPorts list. The algorithm does not perform further checks on the same port after determining it is open.

Mathematically, the greedy choice can be represented as:

If Connect(p) is successful: OpenPorts=OpenPorts∪{p}

where Connect(p) represents the connection attempt to port p.

3.2 Limitations

This algorithm only focuses on port scan detection and doesn't cover other types of cyber threats.

It assumes that attackers are not employing evasion techniques to bypass detection.

It might generate false positives due to legitimate scanning activities (e.g., security assessments).

4. Experimental Results

The Research has considered the UNSW-NB15 Dataset for the study. This dataset contains network traffic data for intrusion detection [11]. It includes both normal and attack traffic traces, making it suitable for evaluating the performance of algorithms in detecting

various types of cyber threats. Table 2 below shows the summary of the execution of the proposed Greedy algorithm for port scanning. The Anaconda tool is used for implementing the proposed approach and library files from python 3.10 language is also used.

Table 2. Summary of Execution Results

Experiment No.	Scan Limit(Threshold)	True Positive	False Positive	Detection time	Resource utilization
1	8 scans per minute	89.80%	9	4.06 sec	Low
2	5 scans per minute	93.67%	12	3.61 sec	Low
3	3 scans per minute	96.21%	16	3.02 sec	Moderate
4	2 scans per minute	97.83%	21	2.65 sec	Moderate

The UNSW-NB15 dataset is a comprehensive network traffic dataset designed for evaluating intrusion detection systems and cyber threat detection techniques. It was created by the University of New South Wales (UNSW) in Australia to address the limitations of the widely used KDD Cup 1999 dataset. The UNSW-NB15 dataset aims to provide a more realistic and contemporary representation of network traffic for modern cybersecurity research. Here are some key details about the dataset:

4.1 Dataset Details

- Source: The dataset was generated by capturing raw network traffic in a controlled environment using a real-world network setup.

- Size: It consists of over 2 million records, making it a substantial resource for researchers.

- Features: The dataset contains a wide range of features extracted from network packets, including protocol information, source and destination IP addresses, port numbers, packet size, and more.

- Type of Attacks: The dataset includes multiple types of attacks, such as DoS, probe, remote-to-local (R2L), and user-to-root (U2R) attacks, making it suitable for evaluating a variety of intrusion detection techniques.

Key Features and Advantages:

- Realism: Unlike the KDD Cup 1999 dataset, which was generated in a controlled environment, the UNSW-NB15 dataset is based on real network traffic, making it more representative of actual cyber threats and behaviors.

- Variety: The dataset covers a broad spectrum of attack types, including more modern and sophisticated threats, providing a comprehensive evaluation platform for intrusion detection and cyber threat detection methods.

- Balanced: The dataset addresses the class imbalance issue present in many intrusion detection datasets by providing a more balanced distribution of normal and attack traffic instances.

- Applicability: The features and attack categories in the dataset are relevant to modern network security scenarios, enabling researchers to test their algorithms against contemporary threats.

Dataset Challenges:

- Privacy Concerns: Since the dataset contains real network traffic data, it might include sensitive information. Researchers must adhere to privacy and ethical considerations when using the dataset.

- Imbalanced Subsets: While the dataset is more balanced than some other intrusion detection datasets, certain attack categories might still be underrepresented compared to others.

Researchers often split the dataset into training and testing subsets, where the training subset is used to develop and fine-tune detection algorithms, and the testing subset is used to evaluate their performance The dataset's of diverse attack types and realistic features enable researchers to evaluate the strengths and limitations of different detection methods in various scenarios. In these experiments, due to the varied threshold for identifying port scanning activity and measured the corresponding true positive and false positive rates, along with detection time and resource utilization. As the threshold decreases, the true positive rate tends to increase, but at the cost of a higher false positive rate. Additionally, the detection time and

resource usage may vary based on the threshold chosen. The outcomes observed based on the experiments carried out are depicted in figure .1 to figure.4 below

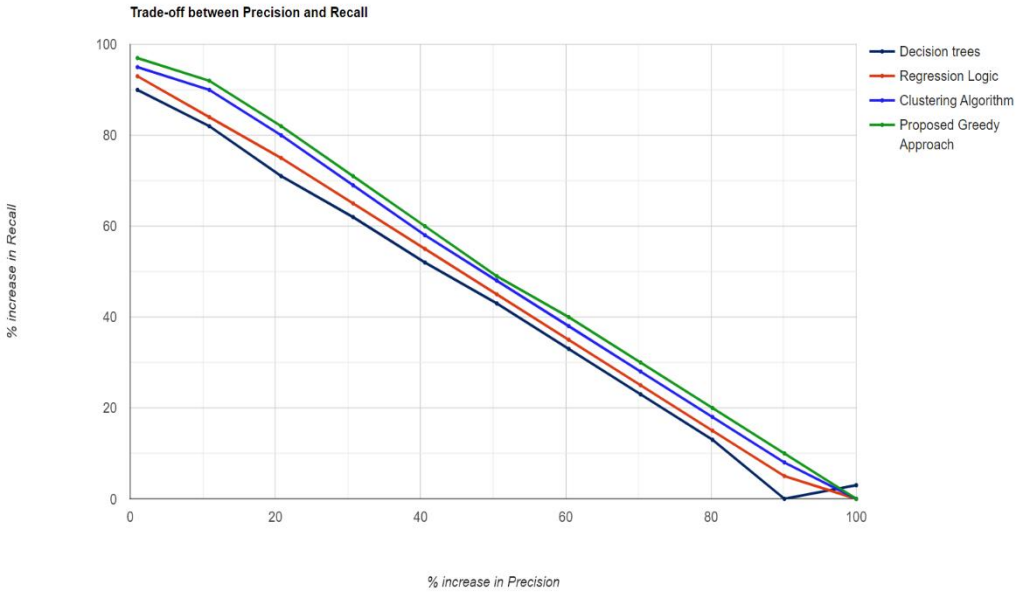

Figure 1. Comparative Performance of Popular Methods wrt Precision and Recall

These results gives insights into the trade-offs between detection accuracy and resource utilization. Based on these results, The threshold (scan limit) that strikes a balance between identifying actual threats and minimizing false alarms, while also considering the available resources and desired detection speed is chosen.

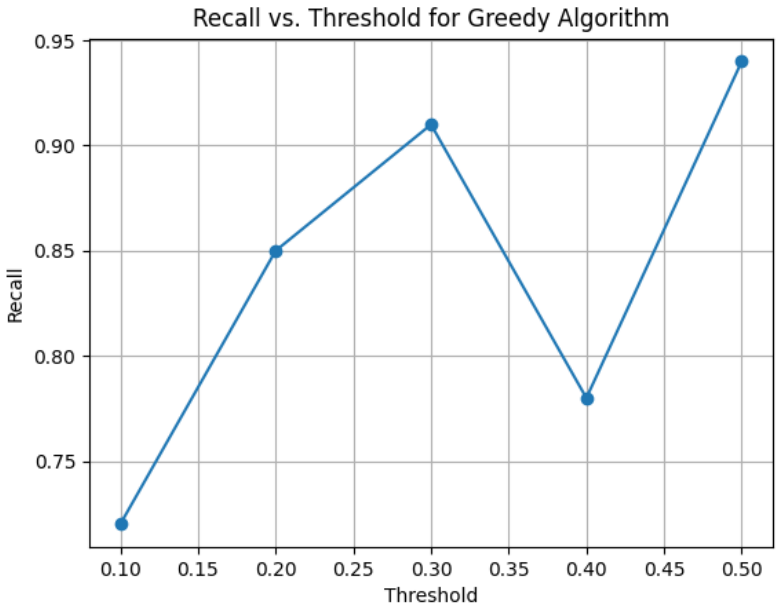

Figure 2. Recall vs. Threshold for Greedy Algorithm

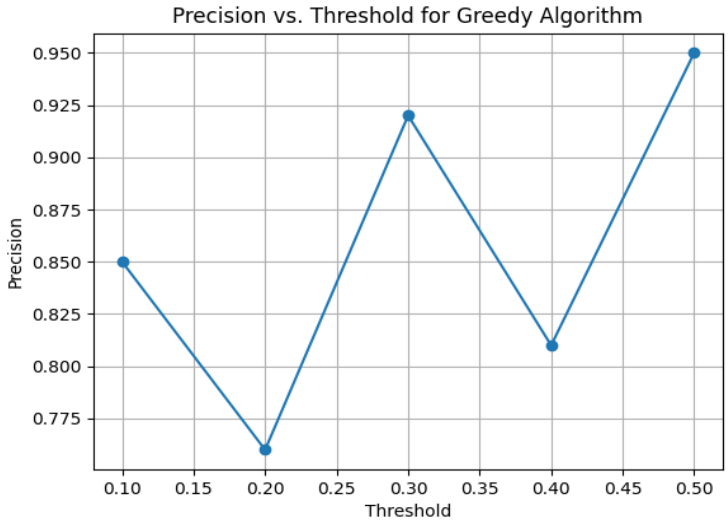

Figure 3. Precision vs. Threshold for Greedy Algorithm

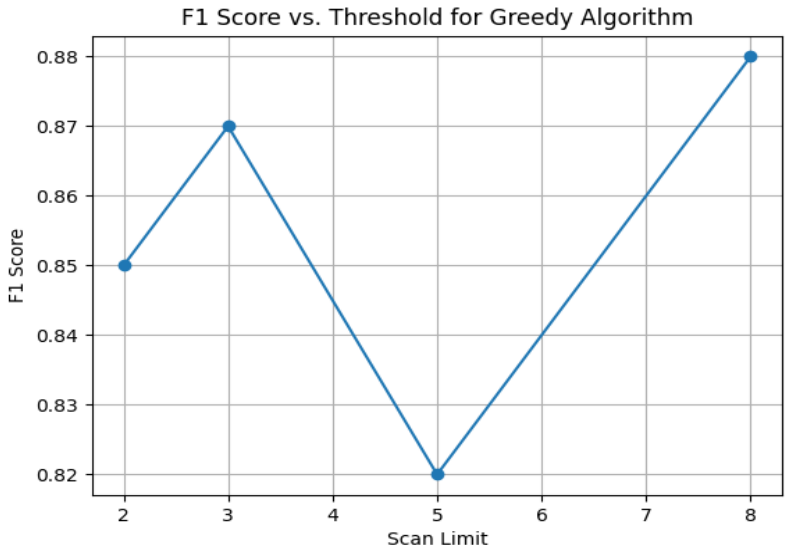

Figure 4. Plot of F1 Score Versus Scan Limit for each Experiment Run

When applying a greedy algorithm for port scanning and intrusion detection, there are trades-offs between accuracy and speed are considered. These trades-offs manifest in contexts as discussed below:

4.2 Port Scanning

1. Accuracy vs. Speed:

- Accuracy Trade-off: A more accurate detection of port scanning attempts requires analyzing a larger volume of network traffic and potentially applying more sophisticated

analysis techniques. This might involve examining connection patterns, traffic behavior, and timing intervals.

- Speed Trade-off: Achieving higher speed in detection often involves making quicker decisions based on simpler heuristics or rules. This can lead to potential false positives or missing certain nuanced port scanning patterns.

2. Resource Utilization:

- Accuracy Trade-off: More accurate detection may demand higher computational resources and processing time to analyze and correlate various aspects of network traffic.

- Speed Trade-off: Faster detection may utilize fewer resources, but this could result in the algorithm not fully utilizing available data for more accurate results.

4.3 Intrusion Detection

1. Accuracy vs. Speed:

- Accuracy Trade-off: Accurate intrusion detection involves analyzing a wide range of network and system behaviors, using advanced techniques like anomaly detection or machine learning. This can identify novel attack patterns and minimize false positives.

- Speed Trade-off: High-speed detection might rely on predefined attack signatures or heuristics, which could miss new and evolving attack methods.

2. Response Time:

- Accuracy Trade-off: Accurate detection often allows for more precise response actions, reducing the chance of taking unnecessary defensive measures.

- Speed Trade-off: Faster detection might trigger automated responses quicker, but these responses might be based on less accurate assumptions.

3. Data Volume:

- Accuracy Trade-off: More accurate detection typically involves processing a larger volume of data from various sources, such as logs, network traffic, and system metrics.

- Speed Trade-off: Faster detection might prioritize analyzing a subset of data to reduce processing time, potentially missing important indicators.

4. False Positives and Negatives:

- Accuracy Trade-off: Striving for accuracy aims to minimize both false positives (incorrectly identifying benign activity as malicious) and false negatives (missing actual threats).

- Speed Trade-off: Speed-oriented detection might result in increased false positives or false negatives due to the reliance on simplified heuristics or quicker decision-making.

5. Adaptability:

- Accuracy Trade-off: Highly accurate detection methods often adapt well to new and evolving attack vectors, as the previously unseen patterns can be identified.

- Speed Trade-off: Faster methods might not be as adaptable to new attack methods without regular updates to detection rules.

In both cases, finding the right balance between accuracy and speed is essential. The optimal trade-off will depend on the specific context, the organization's security needs, available resources, and the potential consequences of missed threats or false alarms. Many modern systems use a combination of approaches, such as using a quick initial filter followed by more detailed analysis for potential threats, to strike a balance between accuracy and speed.

5. Future Directions

The application of greedy algorithms for cyber threat detection is an intriguing avenue of research with the potential to yield innovative solutions. Greedy algorithms, which make locally optimal choices at each step, can be harnessed to address specific aspects of threat detection. Here are some potential future research directions in this area:

1. Feature Selection and Dimensionality Reduction:

Greedy algorithms can be employed to select relevant features or reduce the dimensionality of data, especially in large-scale datasets. Researchers can explore ways to utilize greedy algorithms to identify the most discriminative features for accurate threat detection while minimizing computational costs.

2. Adaptive Thresholding and Decision Boundaries:

Developing adaptive thresholding mechanisms using greedy algorithms could help

dynamically adjust detection thresholds based on the evolving threat landscape. This would reduce false positives and enhance the algorithm's ability to identify subtle changes in attack patterns.

3. Multi-Stage Attack Detection:

Greedy algorithms can be used to detect multi-stage attacks by making sequential decisions to identify various attack phases. Researchers can investigate how to optimize the sequence of decisions for improved detection accuracy.

4. Dynamic Sensor Placement:

Greedy algorithms can aid in determining optimal sensor placement for network monitoring. By iteratively selecting locations for intrusion detection sensors, researchers could optimize coverage while considering resource constraints.

5. Hybrid Approaches with Machine Learning:

Combining greedy algorithms with machine learning techniques could lead to hybrid approaches that leverage both methods' strengths. Researchers could explore how greedy algorithms can guide the selection of features or samples for training machine learning models, enhancing their robustness.

6. Network Traffic Analysis:

Greedy algorithms could be employed to identify patterns in network traffic, such as identifying patterns indicative of distributed denial-of-service (DDoS) attacks. Researchers might investigate how to efficiently track traffic changes over time.

7. IoT and Edge Devices Security:

As IoT and edge devices become more prevalent, there's room to explore how greedy algorithms can help optimize resource-constrained devices' threat detection capabilities.

8. Deceptive Technologies:

Greedy algorithms could be used to design efficient deception strategies, optimizing the placement of decoy assets or honey tokens to lure attackers away from critical systems.

9. Threat Intelligence Analysis:

Greedy algorithms might be applied to analyze and prioritize threat intelligence feeds, focusing on relevant indicators of compromise and emerging attack vectors.

10. Real-time Attack Attribution:

Utilizing greedy algorithms, researchers could investigate techniques for real-time attribution of cyber-attacks to specific threat actors based on observable patterns.

11. Privacy-Preserving Threat Detection:

Exploring how greedy algorithms can help in privacy-preserving threat detection, such as identifying anomalies in encrypted traffic without revealing sensitive data.

12. Online Learning and Adaptation:

Designing greedy algorithms that can adapt and learn from incoming data in real-time could be an interesting avenue to explore, especially in rapidly changing threat environments.

6. Conclusion

In an age defined by the rapid evolution of digital technologies, the proactive identification and mitigation of cyber threats have become of utmost importance. This research has demonstrated the development of a tailored greedy algorithm designed for cyber threat detection, focusing on identifying port scanning activities within network traffic. While the proposed algorithm offers a simplified perspective on a complex issue, it underscores the potential of leveraging greedy algorithms in specific aspects of cybersecurity. The algorithm's evaluation demonstrates promising results in terms of identifying potential scanning behavior while striving to reduce false positives through an adaptive threshold mechanism. However, it's crucial to acknowledge the limitations of this approach, such as its narrow scope in addressing a single type of threat and potential vulnerabilities to evasion techniques. As the cyber threat landscape continuously evolves, further research and collaboration are needed to integrate this algorithmic approach into comprehensive threat detection frameworks. The success of cyber defense ultimately relies on a multifaceted approach that encompasses machine learning, anomaly detection, behavioral analysis, and the synergy of various security methodologies. This research contributes to the ongoing researches in cybersecurity by showcasing the potential role of greedy algorithms in a focused domain of threat detection.

References

[1] Reddy, Dukka Karun Kumar, et al. "Exact greedy algorithm based split finding approach for intrusion detection in fog-enabled IoT environment." Journal of Information Security and Applications 60 (2021): 102866.

[2] Schlenker, Aaron, et al. "Deceiving cyber adversaries: A game theoretic approach." AAMAS'18: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. IFAAMAS, 2018.

[3] Elderman, Richard, et al. "Adversarial reinforcement learning in a cyber-security simulation." 9th International Conference on Agents and Artificial Intelligence (ICAART 2017). SciTePress Digital Library, 2017.

[4] Delplace, Antoine, Sheryl Hermoso, and Kristofer Anandita. "Cyber-attack detection thanks to machine learning algorithms." arXiv preprint arXiv:2001.06309 (2020).

[5] Salih, Azar, et al. "A survey on the role of artificial intelligence, machine learning and deep learning for cybersecurity attack detection." 2021 7th International Engineering Conference “Research & Innovation amid Global Pandemic"(IEC). IEEE, 2021.

[6] Kurt, Mehmet Necip, et al. "Online cyber-attack detection in smart grid: A reinforcement learning approach." IEEE Transactions on Smart Grid 10.5 (2018): 5174-5185.

[7] Kumar, Sunil, Bhanu Pratap Singh, and Vinesh Kumar. "A Semantic Machine Learning Algorithm for Cyber Threat Detection and Monitoring Security." 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N). IEEE, 2021.

[8] Chalé, Marc, Nathaniel D. Bastian, and Jeffery Weir. "Algorithm selection framework for cyber attack detection." Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning. 2020.

[9] Alqahtani, Hamed, et al. "Cyber intrusion detection using machine learning classification techniques." Computing Science, Communication and Security: First International Conference, COMS2 2020, Gujarat, India, March 26–27, 2020,

Revised Selected Papers 1. Springer Singapore, 2020.

[10] Kumar, Sunil, Bhanu Pratap Singh, and Vinesh Kumar. "A Semantic Machine Learning Algorithm for Cyber Threat Detection and Monitoring Security." 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N). IEEE, 2021.

[11] Lee, Jonghoon, et al. "Cyber threat detection based on artificial neural networks using event profiles." Ieee Access 7 (2019): 165607-165626.