A CNN–LSTM–Attention Hybrid Architecture for Real-Time Intrusion Detection at the Data Link Layer

Amirhossein Ahmadnejad Roudsari

A Thesis Submitted to the Faculty of Graduate Studies
In Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science

Graduate Program in Computer Science

York University
Toronto, Ontario

December 9, 2025

Acknowledgement

I would like to thank my supervisor, Prof. Arash Habibi Lashkari, for all his help and advice with this Master Thesis. I also appreciate all the support I received from my family. Lastly, I would like to thank the Grandparents for their motivation and support through my academic journey.

Abstract

Data Link Layer (Layer 2) security remains one of the most underexplored areas in modern network intrusion detection research, despite its critical role as the foundation of reliable communication between networked devices. Attacks at this layer, such as ARP spoofing, MAC flooding, VLAN hopping, and DHCP starvation, can compromise entire networks before higher-layer defenses activate. Existing intrusion detection systems predominantly focus on network or transport layers, leaving a significant gap in early-stage threat prevention. To address this limitation, this thesis proposes a memory-efficient hybrid deep learning architecture that integrates Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) units, and an Attention mechanism for real-time detection of Layer 2 intrusions. A novel dataset, BCCC-DLLayer-IDS-2025, was developed as part of this research, comprising over 4.6 million labeled flow records collected in a controlled experimental environment. The dataset includes eleven distinct attack types spanning spoofing, flooding, and protocol manipulation scenarios, along with benign traffic, providing a comprehensive foundation for training and benchmarking Layer 2 intrusion detection systems.

The proposed CNN–LSTM–Attention architecture combines spatial and temporal feature extraction with an adaptive focus mechanism, enabling effective modeling of short-term dependencies in network traffic while reducing redundancy. The model achieves an F1-score of 99.67% with only 2.1 million parameters and a latency below 100 milliseconds, offering a 60% lower computational cost than conventional deep learning models. Extensive experiments under varying traffic conditions and noise levels confirm the model's robustness, generalizability, and suitability for real-time deployment on resource-constrained edge and IoT devices.

Contents

List of Tables

List of Figures

1 Introduction

Network security research has traditionally prioritized higher-layer protocols such as the network, transport, and application layers, leaving the Data Link Layer (Layer 2) relatively neglected. This oversight has created a significant security gap, as attacks at this foundational layer—such as ARP spoofing, MAC flooding, VLAN hopping, and DHCP starvation—can disable entire networks before higher-level protections are activated. These attacks exploit fundamental communication mechanisms, making them difficult to detect with conventional tools. Despite their growing prevalence, Layer 2 vulnerabilities still receive insufficient attention in both academia and industry, where the focus remains mainly on upper-layer intrusion detection and prevention.

Existing Intrusion Detection Systems (IDSs) are often rule-based or rely on shallow machine learning models. While rule-based systems depend on predefined signatures that fail to adapt to evolving attack patterns, traditional machine learning methods require extensive manual feature engineering and struggle to capture the temporal dependencies inherent in network traffic. Furthermore, deep learning solutions designed for higher-layer threats are computationally demanding and unsuitable for real-time deployment on resource-constrained devices such as switches, routers, and IoT nodes. To address these challenges, this thesis introduces a memory-efficient CNN–LSTM–Attention hybrid architecture for real-time Data Link Layer intrusion detection. The proposed model effectively learns spatial, temporal, and contextual traffic features, providing high detection accuracy while maintaining low latency and computational cost.

The research also introduces the BCCC-DLLayer-IDS-2025 dataset, a large-scale, labeled collection of more than four million network flow records representing benign traffic and eleven distinct Layer 2 attack types. To support this dataset, a custom feature-extraction tool, DLLFlowLyzer, was developed to convert raw packet captures into structured flow-level features suitable for deep learning. Together, the dataset and feature extractor form a reproducible foundation for studying Layer 2 intrusions, enabling the benchmarking and comparison of emerging models in this domain. This phase of the research focuses on creating realistic, diverse attack scenarios that ensure comprehensive representation across spoofing, flooding, and protocol exploitation categories.

The next stage of the research focuses on developing and optimizing the CNN–LSTM–Attention hybrid model, which combines convolutional layers for spatial feature learning, recurrent units for modeling temporal dependencies, and an attention mechanism to enhance interpretability and dynamic focus on critical features. This design enables the model to adaptively balance efficiency and performance, making it suitable for real-time network monitoring. The proposed approach demonstrates superior detection performance compared to conventional deep learning architectures, achieving high accuracy and robustness under noisy traffic conditions.

The model's performance was evaluated through extensive experiments on both clean and noisy traffic datasets, using metrics such as precision, recall, F1-score, and latency. The results confirm that the architecture achieves 99.67% F1-score with only 2.1 million parameters, maintaining sub-100ms inference latency and reducing computational requirements by more than 60% compared to conventional models. These outcomes validate the model's applicability to real-world, resource-limited environments and establish a strong

baseline for future Data Link Layer intrusion-detection research.

The main contributions of this research are fourfold. First, it introduces the BCCC-DLLayer-IDS-2025 dataset, the first large-scale and publicly documented dataset dedicated to Layer 2 intrusion detection. Second, it presents DLLFlowLyzer, a novel feature-extraction and flow-generation tool designed for efficient Layer 2 data analysis. Third, it proposes a hybrid CNN–LSTM–Attention architecture optimized for accuracy, interpretability, and real-time deployment efficiency. Finally, it provides a comprehensive experimental evaluation demonstrating that Layer 2 intrusions can be detected effectively using lightweight deep learning models, thereby filling a critical gap in existing network security research and advancing the broader field of behavior-centric cybersecurity.

2 Literature Review

To fully understand the Data Link layer, you need to look closely at its protocols, services, and how it works with other layers in the OSI model. Conard's seminal work [1] offers a comprehensive examination of the Data Link layer's functions within the overarching network architecture. The author carefully explains how this layer acts as a very important link between the physical transmission medium and the network layer above it. Conard divides the layer's interactions into four basic primitives: REQUEST, INDICATION, RESPONSE, and CONFIRM. Each primitive has a different job in setting up services, sending data, and ending connections. This classification is essential for comprehending the functionality of contemporary networking protocols at Layer 2. The paper focuses on the layer's ability to provide reliable point-to-point and point-to-multipoint communication over physical media that may not be reliable. It talks about mechanisms like frame sequencing, acknowledgment protocols, and error recovery procedures that are still important in modern networks.

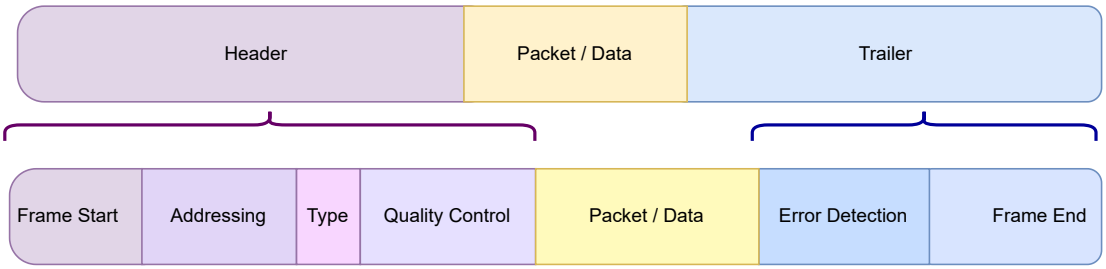

Figure 1: Structure of Link layer packet

The security flaws that come with Media Access Control (MAC) protocols are a big problem for Data Link layer operations. Punia and Ziya's thorough review [2] goes into great detail about the design flaws that make MAC protocols vulnerable to a wide range of attacks. Their analysis differentiates between passive attacks, in which adversaries covertly observe network traffic to collect intelligence, and active attacks, in which malicious entities directly disrupt network operations. The authors conduct a comprehensive analysis of particular vulnerabilities in the 802.11 MAC protocol, highlighting the absence of authentication in management frames, the predictability of sequence numbers, and the lack of integrity checks in specific frame types. They systematically assess countermeasures integrated into different versions of the 802.11 standard, scrutinizing their efficacy in response to the changing threat environments. The paper's contribution goes beyond just listing attacks; it gives a way to understand how MAC layer weaknesses can be used to break into higher-layer security systems, stressing how Layer 2 compromises can have a big impact on the overall security of a network.

Another important part of how the Data Link layer works is error control mechanisms. Singh and Saxena's work [3] provides a comprehensive analysis of the error detection and correction methodologies utilized at this layer. The authors start with a theoretical base of error types, separating single-bit errors caused by electromagnetic interference from burst errors caused by long-term signal degradation. Their examination of Vertical Redundancy Check (VRC) illustrates its simplicity yet constrained efficacy, as it identifies only odd

quantities of bit errors. The talk about Longitudinal Redundancy Check (LRC) shows how two-dimensional parity checking can find more errors without slowing down the computer. The paper gives a very thorough explanation of Cyclical Redundancy Check (CRC), including the math behind polynomial division used in CRC calculations and why CRC is now the standard way to find errors in modern networks. The authors also look at the pros and cons of error detection overhead and reliability, giving a quantitative analysis of different CRC polynomial choices and how they affect detection capabilities. They also talk about flow control methods like stop-and-wait, sliding window protocols, and their variations. They explain how these methods stop buffer overflow and make sure that data is delivered reliably even when networks have different propagation delays and bandwidth limits.

Fan et al.'s groundbreaking research [4] brings artificial intelligence ideas to Data Link layer defense mechanisms in the ever-changing field of network security. Their suggested smart anti-jamming system is a big change from security measures that only react to threats to ones that are proactive. The system uses machine learning algorithms to look at jamming patterns in real time, find attack signatures, and guess how jamming will happen. The authors explain how their system changes the frequency hopping patterns, transmission power levels, and channel selection strategies on the fly based on the characteristics of the jamming that it detects. The paper shows a lot of simulation results that show how well the system works against different types of jamming, such as constant jamming, random jamming, and intelligent adaptive jamming. Their discussion of optimizing resource allocation in the face of jamming conditions is especially interesting. In these situations, the system has to find a balance between keeping communication reliable and saving energy. The authors also talk about the extra work that their method requires, and they suggest lightweight algorithms that work well on network devices with limited resources while still being able to block jamming.

2.1 Attacks

To fully understand Data Link layer attacks, you need to look at threat vectors, attack methods, and how they could affect network operations in a systematic way. Guven et al.'s survey [5] offers a comprehensive classification of backbone attacks that specifically aim at essential components of network infrastructure. The authors start by talking about how important backbone networks are to modern communication systems. They explain how these high-capacity links are what make the internet work. Their examination of Spanning Tree Protocol (STP) attacks uncovers various exploitation vectors, such as BPDU spoofing, wherein attackers introduce malicious Bridge Protocol Data Units to alter the spanning tree topology, potentially resulting in network loops or rerouting traffic through paths controlled by the attacker. The paper talks about VLAN hopping attacks in two ways: switch spoofing, where attackers trick switches into thinking they are trunk links, and double tagging, where specially crafted frames get around VLAN segmentation. The authors do a thorough packet-level analysis to show how these attacks take advantage of the default settings on switches and the fact that switches trust each other when they talk to each other. Their talk about CAM table overflow attacks includes a quantitative study of how much memory is used and how long it takes to run out of switch resources at different attack levels. The paper also talks about how combined attacks can have a synergistic effect, showing how attackers can use multiple Layer 2 exploits together to reach goals that would be impossible with just one attack.

The extensive study by Mahmood, Mohsin, and Akber [6] expands the attack taxonomy by investigating the correlation between vulnerabilities at the Data Link layer and overarching network security issues. The authors systematically categorize attacks according to their objectives: reconnaissance, in which attackers collect information about the network's layout and devices; denial-of-service, in which legitimate users are blocked from accessing network resources; and data interception, in which private information is captured or changed. Their thorough study of ARP poisoning attacks includes a step-by-step look at cache poisoning methods, showing how attackers can act like men-in-the-middle by corrupting ARP tables on the devices they want to attack. The paper talks about two types of DHCP attacks: starvation attacks, where attackers use up all of the DHCP server's address space, and rogue DHCP server attacks, where bad servers give clients false network settings. The authors provide empirical data on attack detection rates employing diverse monitoring techniques, emphasizing the difficulties in differentiating between malicious activities and legitimate yet atypical network behavior. They also talk about how attacks have changed over time in response to security measures. For example, they show how attackers have changed their methods to get around port security, dynamic ARP inspection, and other protective measures.

Yeung, Fung, and Wong's study of attack tools [7] looks closely at the practical parts of Layer 2 attacks. The authors conduct a comprehensive analysis of prevalent attack frameworks, assessing their functionalities, user-friendliness, and prospects for automation. Their review of tools like Yersinia shows how modern attack software makes it easy for anyone to carry out complicated Layer 2 attacks that used to require a lot of technical knowledge. The paper contains a comparative analysis of the effectiveness of different tools against various switch models and configurations, showing that different vendor implementations have very different levels of vulnerability. The authors talk about how these tools can be used for both good and bad purposes. They recognize that they are useful for penetration testing, but they also point out the risks that come with making them available to bad actors. They give a lot of information about tool fingerprints and signatures that can help find attacks, such as characteristic packet patterns, timing anomalies, and behavioral indicators. The paper also talks about the underground economy that surrounds attack tools, including how exploit codes are shared, sold, and improved in hacker communities. It stresses the need for constant security monitoring and updates.

2.1.1 Specific attacks

A close look at each attack vector gives us important information about how they work, what they do, and what we can do to stop them. Trabelsi's practical investigation of CAM table poisoning [8] provides experiential insights into this essential switch attack. The author outlines a thorough laboratory framework that enables students to methodically investigate the mechanics of the attack, beginning with a comprehension of standard CAM table operations and advancing to active exploitation. The paper explains how attackers make thousands of frames with random source MAC addresses, each of which needs a CAM table entry. This eventually fills up the switch's limited memory. Trabelsi does a quantitative study that shows that modern attack tools can quickly overwhelm regular switches with 8,000 to 16,000 CAM entries. The educational method outlined includes reconnaissance before an attack to find out how big the CAM table is, monitoring the attack to see how fast the table fills up, and analysis after the attack to find out how it affected real traffic.

The author talks about port security countermeasures and gives detailed examples of how to set them up. For example, they show how features like MAC address limiting, sticky MAC addresses, and violation actions can stop or lessen CAM table attacks. The paper also talks about the problems with port security, saying that settings that are too strict can hurt legitimate operations, especially in environments where devices connect and disconnect often.

Black hole and grey hole attacks in mobile ad-hoc networks are advanced threats that take advantage of trust relationships in routing protocols. Kaur and Singh's thorough review [9] gives a thorough look at how these attacks work in different routing protocols. In black hole attacks, bad nodes tell everyone about the best ways to get to all destinations, which draws in traffic that is then silently thrown away. The authors talk about how this attack takes advantage of the fact that ad-hoc routing protocols like AODV and DSR are cooperative, which means that nodes trust routing ads from their peers. Their examination of grey hole attacks uncovers a more nuanced threat, wherein attackers strategically forward packets, thereby complicating detection considerably. The paper offers mathematical models to compute the likelihood of successful attack detection across diverse network conditions, considering variables such as node density, mobility patterns, and traffic loads. The authors conduct a thorough assessment of proposed detection mechanisms, encompassing sequence number analysis—where irregular fluctuations in destination sequence numbers signify potential attacks—collaborative detection schemes—where multiple nodes exchange observations to detect malicious conduct—and statistical analysis techniques that reveal anomalous packet dropping patterns. The review also talks about the trade-offs between accuracy and overhead, pointing out that more advanced detection methods usually need a lot of computing and communication power.

DHCP starvation attacks pose a significant risk to network availability and integrity. Mukhtar, Salah, and Iraqi's extensive study [10] offers an in-depth examination of attack methodologies and defensive tactics. The authors start by talking about the four-way handshake in the DHCP protocol (DISCOVER, OFFER, REQUEST, ACK) and how attackers take advantage of the fact that these exchanges don't require authentication. Their research shows that attackers can quickly use up all of a DHCP server's available addresses by sending thousands of DISCOVER messages with fake MAC addresses. The paper offers mathematical models to determine the duration necessary to deplete DHCP pools of varying sizes under diverse attack intensities, equipping network administrators with quantitative data for risk evaluation. The authors suggest a multi-layered defense strategy that includes rate limiting to limit the number of DHCP requests per port, DHCP snooping to check DHCP messages against a binding database, and statistical anomaly detection to find strange patterns in DHCP traffic. Their experimental validation shows that using multiple countermeasures can stop attacks while still making services available to real customers. The paper also talks about how hard it is to tell the difference between real traffic spikes, like those that happen after power outages, and malicious attacks. It suggests using adaptive thresholds based on past traffic patterns.

Software-Defined Networks (SDNs) create new ways for hackers to attack while also possibly giving better ways to defend. The studies conducted by Qian, You, and Qian [11] and Zhao et al. [12] deliver an exhaustive examination of flow table overflow attacks within SDN contexts. These attacks take advantage of the fact that OpenFlow switches' hardware flow tables are only a few thousand to a few tens of thousands of entries, while large networks can have millions of flows. The authors explain how attackers make many flows with

different characteristics, which makes switches have to keep adding new table entries until they run out of resources. When flow tables get too full, switches have to either drop packets or send them to the controller, which can slow down performance or overload the controller. The papers show that flow table capacities vary a lot between different switch models and vendors. This is something that attackers can take advantage of. The authors suggest several ways to protect against attacks, such as flow aggregation to cut down on the number of table entries, adaptive timeout mechanisms to get rid of old entries more quickly during attacks, and machine learning-based detection systems that find unusual flow generation patterns. Their experimental findings indicate that intelligent defense mechanisms can preserve network functionality despite prolonged attacks, albeit with heightened processing overhead for the controller.

Álvarez et al.'s performance analysis [13] covers the whole history of VLAN security. The authors present a comprehensive comparison of conventional ACL-based defenses and SDN-based methodologies for mitigating private VLAN attacks. Their experimental framework encompasses various network topologies that simulate standard enterprise deployments, measuring latency, throughput, and resource utilization in both normal and attack scenarios. The paper shows that ACL-based methods can make security policies that are always the same, but they can't change to fit new attack patterns. SDN-based solutions, on the other hand, let policies change on the fly, but they also add latency because the controller has to make decisions about how to handle packets. The authors present quantitative analysis demonstrating that SDN-based defenses can diminish attack success rates by as much as 95% in comparison to static ACL configurations, albeit with a 10-15% increase in average packet processing latency. The study also talks about scalability issues, showing that SDN controllers can become bottlenecks in large networks that are under attack and suggesting distributed controller architectures to fix this problem.

Even though people have known about Address Resolution Protocol (ARP) vulnerabilities for decades, they are still a big security risk. Singh and Grewal's survey [14] offers a comprehensive examination of ARP poisoning countermeasures in wireless networks. The authors start by explaining in detail how the fact that wireless networks are broadcast makes ARP vulnerabilities worse. Attackers can target multiple victims at the same time without having to physically access the network infrastructure. They divide defense mechanisms into four groups: static ARP entries, which are very secure but don't work well with many users; cryptographic approaches, which add authentication to ARP messages but need changes to the protocol; anomaly detection systems, which find unusual ARP behavior patterns; and centralized ARP servers, which get rid of broadcast ARP requests completely. The paper offers a comparative analysis of the efficacy, implementation complexity, and performance implications of each method, employing both theoretical examination and empirical validation. The authors stress the difficulties that arise in wireless settings, where moving devices and changing network membership make it harder to set up static security. They suggest a hybrid approach that uses a mix of different defense mechanisms and chooses the best one based on the network's features and the level of threat.

Man-in-the-Middle (MitM) attacks are a wide range of threats that usually start with Layer 2 exploitation. Conti, Dragoni, and Lesyk's extensive survey [15] offers a thorough classification and examination of MitM techniques across various network layers and technologies. At the Data Link layer, the authors explain how ARP spoofing, CAM table poisoning, and DHCP spoofing can be used to start MitM attacks. Their

analysis includes attack scenarios that show how attackers go from breaking into Layer 2 to intercepting and changing all traffic. The paper goes into great detail about SSL/TLS stripping attacks, in which attackers turn encrypted connections into plaintext, and session hijacking, in which authentication tokens are stolen and then used again. The authors provide a statistical examination of the frequency of Man-in-the-Middle (MitM) attacks across various network environments, indicating elevated success rates in wireless networks and public hotspots. Their talk about countermeasures covers many levels, making it clear that stopping MitM attacks requires coordinated defenses from Layer 2 all the way up to the application layer. The survey looks at new MitM techniques that target modern technologies like IPv6, where new protocols create new attack vectors, and IoT devices, where limited resources make security implementations less effective.
Denial-of-service attacks keep getting better by finding new weaknesses and breaking through stronger defenses. Kumar's revised viewpoint [16] offers an exhaustive examination of contemporary DoS methodologies impacting Data Link layer functionalities. The author shows how simple flooding attacks have changed into more complex, coordinated campaigns that take advantage of certain protocol weaknesses. The paper explains how modern DoS attacks use more than one method at the same time. For example, they use ARP storms to overload switch processing capacity while also using CAM table overflow to break switching functionality. Kumar gives mathematical models for figuring out how much damage an attack will do based on things like the switch's processing power, memory limits, and network layout. The analysis includes looking at amplification attacks, which are when small requests cause big responses, which effectively increases the amount of attack traffic. The author talks about the rise of DoS-as-a-Service platforms, where people can buy and launch attacks with little technical knowledge. This makes it easier for everyone to use advanced attack tools. The paper's discussion of countermeasures includes rate limiting, traffic prioritization, and anomaly detection, along with a thorough look at how well each method works against different types of attacks.

2.1.2 Impersonation Attacks

The advanced integration of various attack vectors to accomplish intricate goals signifies an advancement in Layer 2 threats. ElShafee and El-Shafai's design and analysis of data link impersonation attacks [17] offer a thorough framework for multi-stage attacks aimed at application-layer services. The authors explain how attackers use CAM table poisoning, ARP spoofing, and MAC address cloning to make themselves look like trusted network entities. They start their attacks by doing reconnaissance to find high-value targets like security cameras, printers, or other IoT devices that don't have strong authentication methods. The paper does a packet-level analysis to show how attackers first overflow CAM tables to make switches work in hub mode, which lets traffic sniffers capture real device communications. After that, attackers use the information they have gathered to plan precise ARP spoofing attacks, sending traffic that should go to legitimate services through systems they control. The authors show how this method makes it possible for advanced attacks like stealing credentials from unencrypted communications, adding harmful commands to device control protocols, and setting up permanent backdoors for long-term access to networks. Their experimental validation demonstrates success rates surpassing 90% against conventional IoT devices and network services. The paper also talks about problems with detection, saying that impersonation attacks can be set up to hide unusual traffic patterns, making them hard to tell apart from real network reconfigurations.

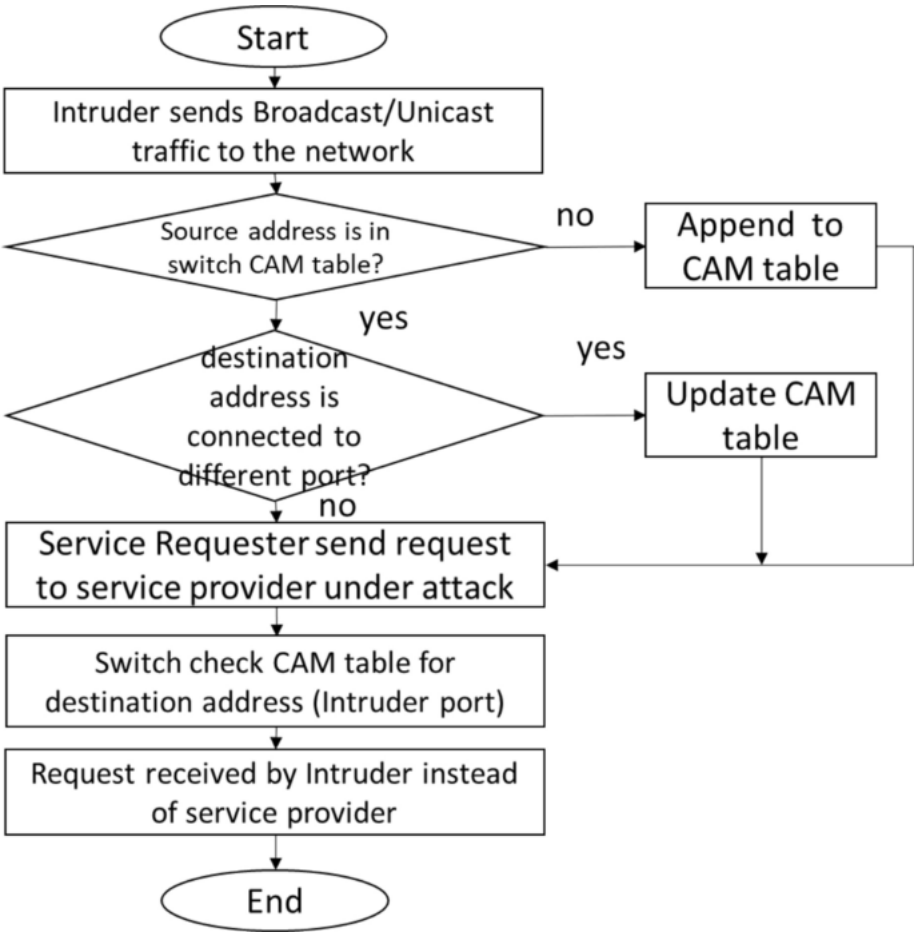

Figure 2: Impersonation Attack flow chart

2.2 Security Solutions

2.2.1 Prevention and Detection

To make good prevention tools, you need to know a lot about how protocols work and how attacks work. Dimitrov's thorough analysis of loop prevention protocols [18] offers essential insights into preserving network stability at the Data Link layer. The author starts by going into great detail about how loops form in Layer 2 networks. They explain that Layer 2 frames don't have any built-in protections against infinite packet circulation like Layer 3 does, which could lead to very bad things. The paper gives a full comparison of three important protocols for preventing loops. Dimitrov talks about how root bridges are chosen, how path costs are figured out, and how redundant links are systematically blocked in the Spanning Tree Protocol (STP). The analysis includes detailed timing parameters that show how the settings for forward delay and maximum age affect stability and convergence time. The talk about Multiple Spanning Tree Protocol (MSTP) shows how it is better than STP because it lets you have more than one spanning tree instance, which lets you balance the load across redundant links while still stopping loops. The author goes into great detail about ITU-T G.8032 Ethernet Ring Protection, explaining how this protocol gets failover times of less than 50 milliseconds, which is very important for carrier networks. The paper contains experimental findings

that compare the convergence times, resource usage, and failure recovery of different protocols in different network situations. Dimitrov also talks about problems with implementation, like how inconsistent priority settings or VLAN mappings can make a network unstable or open to security holes.

Malekzadeh et al.'s groundbreaking research on protected control packets [19] tackles essential weaknesses in IEEE 802.11 networks. The authors say that management and control frames in WiFi networks are usually not encrypted or authenticated, which makes it possible for attacks like deauthentication and disassociation floods to happen. Their suggested fix uses HMAC-SHA1 to protect these frames with cryptography, which lets you check the message without having to encrypt the whole thing. The paper describes two implementation models. The first one uses standard HMAC-SHA1, which is very secure but needs 20-byte authentication tags, which could cause problems with older devices. The second one uses a modified M-HMAC algorithm that keeps security levels high while making tags smaller. The authors conduct a comprehensive performance analysis, evaluating the effects of authentication on frame processing times, throughput, and power consumption in devices with limited resources. Their experiments show that protected control frames can stop more than 99% of spoofed frame attacks and add less than 5% to the throughput in most cases. The paper also talks about the problems that come up when trying to put the ideas into practice, such as managing keys, making sure that older devices can still work with them, and the need for standardization in order for them to be widely used.

For Layer 2 security, intrusion detection systems need to be carefully adapted so that they can work with detection mechanisms. Shanker and Singh's analysis [20] offers a thorough assessment of the efficacy of Snort IDS in countering Data Link layer attacks. The authors start by talking about the problems with using traditional IDS on Layer 2 traffic, such as the large amount of legitimate broadcast traffic, the lack of standardized attack signatures, and the difficulty of setting a baseline behavior for finding anomalies. In their experimental setup, they systematically test Snort against different attacks, such as ARP spoofing (where they look at detection rates for different spoofing patterns and frequencies), MAC flooding (where they look at Snort's ability to find abnormal MAC address generation rates), and VLAN hopping (where they look at Snort's ability to find both switch spoofing and double tagging attacks). The paper gives a thorough look at false positive and false negative rates, showing how changing the rules changes the accuracy of detection. The authors show that Snort can find known attack patterns well, but it has trouble with zero-day exploits and advanced attacks that look like normal traffic patterns. They suggest improvements like statistical analysis modules to find unusual traffic patterns, correlation engines to connect related suspicious events, and machine learning components to keep up with changing attack patterns. The study also looks at performance issues, showing how detection rules affect how much system resources are used and how many packets can be processed.

Comprehensive security policies protect organizations from Layer 2 threats at the organizational level. Pilamunga et al.'s security policy framework [21] shows how to use ISO 27002 standards to protect against VLAN hopping attacks. The authors offer a comprehensive correlation between ISO controls and particular VLAN security measures, demonstrating how global standards can guide effective security practices. Their framework has technical controls, like turning off unused ports, setting up trunk ports correctly, and using VLAN access control lists. It also has procedural controls, like regular security audits, configuration man-

agement, and incident response procedures. Finally, it has administrative controls, like training employees on security, making sure they follow access management policies, and meeting vendor management requirements. The paper has templates and checklists for organizations to use that they can change to fit their own needs. The authors show how the framework can be used by businesses of all sizes by giving case studies from three different organizations. Their analysis includes cost-benefit calculations that show that implementing full VLAN security is usually cheaper than recovering from one successful attack. The paper also talks about common problems that come up when putting something into action, such as people not wanting to change how things work, old equipment not being able to handle new tasks, and the need for regular maintenance and updates.

To come up with good ways to stop attacks, you need to know how they work. Zhao, Guo, and Lv's in-depth study of ARP spoofing [22] gives a full picture of this ongoing threat. The authors start by looking at how normal ARP operations work at the packet level. They show how the protocol's stateless nature and lack of authentication make spoofing attacks possible. Their study looks at three main types of attacks: targeted spoofing, where attackers pretend to be certain hosts to steal their traffic; gateway spoofing, where attackers pretend to be default gateways to steal all external traffic; and broadcast spoofing, where attackers send fake ARP messages to poison multiple hosts at once. The paper provides a mathematical examination of cache poisoning persistence, illustrating the influence of varying operating system ARP cache timeout values on the sustainability of attacks. The authors look at a number of ways to stop ARP attacks, such as Dynamic ARP Inspection (DAI), which checks ARP messages against DHCP bindings; static ARP entries, which stop cache poisoning but make network management harder; and ARP rate limiting, which limits the number of ARP messages that can be sent per port. Their experimental findings indicate that layered defenses, which integrate various techniques, provide optimal protection while preserving operational adaptability. The paper also talks about new problems, like ARP security in virtualized environments where virtual switches might not have traditional security features, and IPv6 networks where the Neighbor Discovery Protocol creates similar but different vulnerabilities.

A methodical approach to identifying and addressing Layer 2 attacks requires an in-depth comprehension of attack indicators and efficient response methodologies. O'Connor's practical guide [23] gives security experts useful information that they can use. The author describes a way to find things by looking at traffic, logs, and behavior. O'Connor gives specific signs for each type of attack. For example, he talks about packet patterns, like ARP request/reply ratios that are out of the ordinary, which could mean poisoning attempts; timing anomalies, like MAC address changes that happen too quickly, which could mean CAM table attacks; and protocol violations, like BPDU frames coming from unauthorized ports, which could mean STP manipulation. The paper has a lot of information on how to respond, from finding the problem to containing it, getting rid of it, and getting back to normal. The author talks about how important it is to have monitoring infrastructure already in place, and how to best place sensors and collectors so that they can be seen. O'Connor gives scripts and tools for automated detection and response, with examples for well-known platforms like tcpdump, Wireshark, and different SIEM systems. The paper also talks about the human side of incident response, including how to get network, security, and operations teams to work together during Layer 2 security events. The author does a great job of explaining how to keep evidence of Layer 2 attacks,

which often leave very few traditional log traces.

2.3 Network Security

Network security is a bigger idea that helps us understand threats and ways to protect the Data Link layer. This section looks at security strategies that include Layer 2 concerns as part of bigger security plans. Wireless network security has its own set of problems because the way data is sent is very different from wired networks.

Regan and Manickam's extensive examination of impersonation attacks in wireless networks [24] elucidates how the broadcast characteristics of wireless communications facilitate attacks that are unfeasible in wired contexts. The authors classify impersonation attacks into three main categories: identity-based impersonation, where attackers pretend to be real users or access points; role-based impersonation, where attackers take on privileged network roles like authentication servers; and service-based impersonation, where fake services lure in and compromise client connections. Their analysis shows how these attacks take advantage of weaknesses in many protocol layers, starting with Layer 2 weaknesses like fake beacon frames or association responses. The paper offers a comprehensive analysis of Evil Twin attacks, wherein malevolent access points masquerade as legitimate ones, including an evaluation of the automatic connection behaviors in contemporary devices that enable these attacks. The authors provide quantitative data on attack success rates across different environments, indicating markedly increased vulnerability in public spaces where users anticipate open networks. They suggest using mutual authentication protocols, certificate pinning for known networks, and teaching users about connection security as ways to protect against attacks. Each method comes with detailed instructions on how to put it into action.

Wireless sensor networks have their own security problems because of limited resources and the places where they are used. Hasan et al.'s thorough review [25] talks about Data Link layer security in these kinds of networks. The authors elucidate the impracticality of conventional security mechanisms in Wireless Sensor Networks (WSNs) due to constraints in processing power, memory, and battery longevity. Their analysis includes collision attacks, in which bad nodes intentionally cause packet collisions to waste energy and bandwidth; exhaustion attacks, in which victims are forced to send data again and again; and unfairness attacks, in which attackers use fake backoff timers to take over channel access. The paper analyzes energy consumption for different attacks and countermeasures, showing that some security measures may use more resources than the attacks they stop. The authors suggest lightweight security solutions that work well with the limitations of WSNs. These include efficient authentication methods that use symmetric cryptography, adaptive duty cycling to protect against exhaustion attacks, and distributed detection systems that use collaborative monitoring. Their experimental validation on standard WSN platforms, including TelosB and MICAz motes, demonstrates that meticulously crafted security mechanisms can offer sufficient protection while preserving an acceptable network lifespan.

Jain et al. [26] look closely at the design of protocols that make WSN operations safe and efficient. The authors give a full analysis of how the choice of protocol affects security, energy efficiency, and network performance. Their evaluation framework takes into account a number of factors, such as the packet delivery ratio when there is an attack, the energy use for security operations, the effect of authentication mechanisms

on latency, and the ability of key management schemes to grow. The paper contrasts contention-based protocols, exemplified by CSMA, with schedule-based methodologies, such as TDMA, demonstrating the unique vulnerability profiles of each. The authors show that TDMA-based protocols can protect against some DoS attacks by giving guaranteed time slots, but they are still open to physical jamming and need to be perfectly synchronized, which attackers can mess with. The paper suggests hybrid protocols that change how they work based on the level of threat they see. When there are no threats, they work in a lightweight way, but when they see an attack, they switch to a more secure but resource-intensive mode. Implementation details include exact parameter settings for common WSN operating systems and useful tips on how to choose a protocol based on the needs of the application.

Combining WSNs with Internet of Things (IoT) architectures makes security even more complicated. Hasan and Hanapi's systematic review [27] looks at security protocols in the WSN-IoT ecosystem. The authors talk about how IoT apps make WSNs better by adding new ways for hackers to attack them through internet connectivity and cloud integration. Their analysis includes end-to-end security architectures that go from sensor nodes to gateways and cloud platforms. It also finds weaknesses at each integration point. The paper offers a comprehensive analysis of lightweight cryptographic protocols designed for resource-constrained devices, including an evaluation of computational and communicational overhead across various security levels. The authors tackle the issue of key management in dynamic IoT settings, characterized by the frequent joining and leaving of devices from networks. They propose hierarchical key distribution schemes that achieve a balance between security and scalability. Their talk about quality of service (QoS) under security constraints shows the balance between protection levels and the needs of applications for performance. The paper presents case studies from smart cities, industrial IoT, and healthcare applications, illustrating the variability of security requirements across diverse deployment scenarios.

Low's examination of wireless attacks through an OSI Layer 2 lens [28] offers valuable insights for security experts. The author talks about how management frames that are only for wireless networks make it possible for attacks that aren't possible on wired networks. The paper talks about frame injection attacks, in which attackers create bad frames that don't connect to the network; deauthentication floods, in which fake frames disconnect real clients; and beacon flooding, in which fake access points confuse client devices and discovery tools. Low uses standard tools like aircrack-ng and Wireshark to do packet-level analysis and show specific byte patterns that show malicious activity. The detection methods shown are signature-based detection for known attack tools, anomaly detection for strange frame patterns, and timing analysis for finding fake frames. The author talks about how important it is to have distributed monitoring by showing how multiple sensors can find the source of an attack and tell the difference between legitimate roaming and malicious activity. The paper has step-by-step instructions for setting up a wireless IDS with open-source software and cheap hardware. There is a lot of information about the systematic weaknesses in IEEE 802.11 networks, but attacks keep changing.

Bicakci and Tavli's extensive survey [29] examines Denial-of-Service attacks at both the physical and MAC layers. The authors divide attacks into four groups: physical jamming, where real RF energy blocks legitimate signals; MAC layer flooding, where frames that follow the protocol use up network resources; cross-layer attacks, which take advantage of how layers interact; and implementation-specific exploits, which

target vendor weaknesses. Their analysis includes mathematical models that show how attackers can use power levels and distances to figure out how effective jamming is. These models show how attackers can use less energy for longer attacks. The paper looks at ways to stop attacks, such as frequency hopping to get around narrow-band jamming, rate limiting to stop flooding attacks, and anomaly detection to find patterns of attack. The authors conduct a comparative analysis of diverse defensive strategies, evaluating factors such as deployment costs, operational overhead, and efficacy against various attack types. Their talk about problems that will come up in the future brings up new threats from software-defined radios and cognitive radio networks.

The evolution of WiFi security protocols shows that people are still trying to fix problems that have been found. Sharma et al.'s thorough review [30] follows the evolution of security from WEP to WPA3. The authors describe how each generation of protocols fixed problems with the previous ones while also making new ones. Their study of WEP shows that it has serious design flaws, such as a small IV space that lets keys be reused, a weak RC4 implementation that lets statistical attacks happen, and no mutual authentication, which lets rogue access points happen. The talk about WPA improvements shows how TKIP fixed WEP's problems while still being open to certain attacks. The authors' analysis of WPA2 elucidates how AES-CCMP provides robust encryption, yet implementation vulnerabilities persist, facilitating attacks. The paper gives detailed instructions for setting up each protocol generation to get the most security, including what parameters to use and how to run the system. The authors also talk about the trade-offs between security and usability, saying that stronger security measures might make devices less compatible and user experience worse.

The KRACK vulnerability showed that even well-designed protocols can have big problems. Alhamry and Elmedany's thorough study [31] gives a full picture of this serious flaw. The authors go into great detail about the WPA2 four-way handshake, which is how the protocol sets up pairwise keys between clients and access points. Their analysis shows that the KRACK attack takes advantage of state machine implementations that let keys be reinstalled, which lets attackers reset nonce values and replay packets. The paper shows how attackers can decrypt traffic and add bad content by looking at successful attacks at the packet level. The authors look at different ways to fix the problem, such as patching vulnerable implementations, which means updating all affected devices; adding replay protection that doesn't depend on the handshake; and adding extra layers of encryption, like VPNs. Their tests on different operating systems and devices show that some implementations are more vulnerable than others. This is because of the way they were designed. The paper also talks about the bigger effects on protocol design and validation, stressing the need for formal verification methods.

Attacks on networks are getting more complicated and having more of an effect. Hoque et al.'s extensive taxonomy [32] offers an organized categorization of threats throughout all network layers. The authors categorize attacks across various dimensions, such as target (confidentiality, integrity, availability), method (active versus passive), and layer (physical to application). Their examination of Data Link layer attacks within this classification demonstrates how Layer 2 vulnerabilities frequently facilitate compromises at higher layers. The paper gives a thorough look at attack tools, putting them into groups based on how well they work, how easy they are to use, and how hard they are to find. The authors assess defensive systems, encompassing both

commercial and open-source solutions, by comparing their efficacy across various categories of attacks. The talk about attack trends shows that attack tools are becoming more automated and cheaper, which makes it easier for people to attack. The paper suggests a unified defense framework that includes prevention, detection, and response functions at all levels of the network.
Akbar [33] specifically looked at the Address Resolution Protocol in Ethernet and Wi-Fi, talking about its built-in weaknesses (ARP poisoning) and how they affect network security. These network-focused surveys show that weaknesses in data links are often the first step in bigger breaches.

2.3.1 Trends and Challenges

The changing world of cybersecurity makes it harder for network defenders to do their jobs. Rajasekharaiah et al.'s analysis of cybersecurity challenges [34] situates Data Link layer security within overarching threat trends. The authors pinpoint primary elements contributing to heightened cyber risk, such as the swift integration of technology surpassing security protocols, inadequate security awareness among users and developers, and the emergence of advanced attack tools. Their analysis shows that Layer 2 vulnerabilities are often used as entry points for advanced persistent threats, which let attackers get a foothold by exploiting small protocol flaws. The paper talks about the problem of protecting old infrastructure that can't be easily updated or replaced, especially in industrial and critical infrastructure settings. The authors suggest a risk-based method for security spending, where defenses are ranked based on how likely threats are and how much damage they could do. Their framework includes ongoing monitoring to find active threats, regular assessments to find new weaknesses, incident response planning to lessen the damage caused by attacks, and security awareness training to deal with human factors.
Using digital twin technology to protect networks is a new way to learn about and protect complex systems. Lei and Fu's groundbreaking research [35] illustrates the capability of digital twins to simulate Data Link layer functions for security assessment. The authors describe how digital twins make very accurate virtual copies of network infrastructure, which lets you safely test attack scenarios and defense strategies without putting production systems at risk. Their use of OPNET to implement military Link-16 networks shows how digital twins can model complicated protocol interactions, guess how attacks will affect systems, and improve defensive setups. The paper goes into great detail about the architecture of their digital twin system. It talks about how it collects data from real networks, how it keeps models in sync to keep them accurate, how it simulates scenarios to test attacks and defenses, and how it uses optimization engines to make security postures better. The authors show how digital twins make "what-if" analysis possible, which lets defenders look at possible attacks and countermeasures before they are put into action. Their case studies illustrate how digital twins uncovered previously unrecognized vulnerabilities and confirmed defensive strategies that would be excessively hazardous to evaluate on operational networks.

2.4 Synthesis

The literature review thoroughly delineates data link layer functionality, prevalent attack vectors, and current defensive measures. Numerous studies enumerate recognized Layer 2 threats, including MAC flooding [36], ARP poisoning [22, 14], and VLAN attacks [13, 5], and also suggest straightforward countermeasures

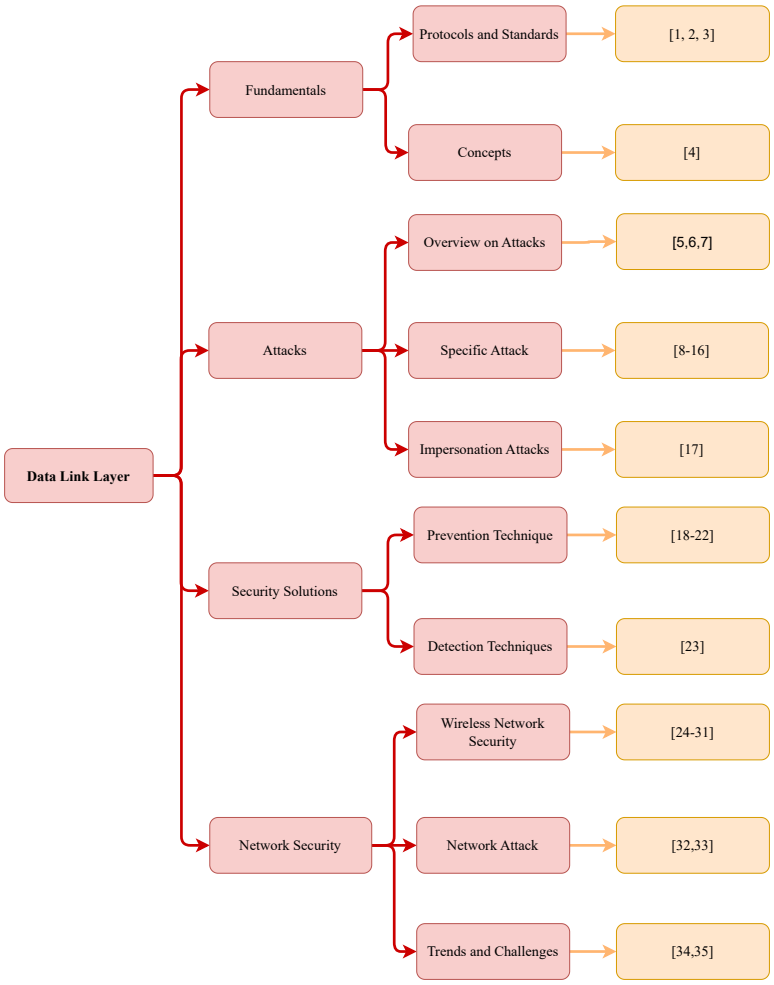

Figure 3: Taxonomy of Papers on DL Layer

such as port security, access control lists, and authentication mechanisms. However, a systematic analysis of the current body of work uncovers several significant deficiencies and ongoing challenges that remain insufficiently addressed.

Gap 1: Not enough complete Layer 2 datasets. As shown in Section 4.2.1, the KDD Cup 99 [37], NSL-KDD [38], UNSW-NB15 [39], and the CIC family [40, 41, 42] datasets for network intrusion detection mostly look at attacks on the network and transport layers. The UGR'16 dataset [43] is built from NetFlow collectors that don't record Layer 2 protocol details, but it does use real traffic and attacks that are up to date. Specialized datasets for IoT and SCADA environments [44, 45] focus on important areas, but they are still too narrow to make a general L2 attack taxonomy. These datasets lack labeled instances of Layer 2-specific attacks, including STP manipulation, CDP flooding, or VLAN double-tagging. The lack of thorough, publicly accessible datasets encompassing the complete taxonomy of Data Link layer threats significantly hinders research advancement in this field. Our contribution: We fill this gap with the BCCC-DLLayer-IDS-2025 dataset, which has more than 4.6 million labeled network flows across 11 different Layer 2 attack

types, such as ARP spoofing [14], STP attacks, CAM table flooding [8, 36], DHCP starvation [10], and impersonation attacks [17]. This dataset is the first complete resource for Data Link layer intrusion detection for the research community.

Gap 2: Few tools for extracting features for Layer 2 analysis. CICFlowMeter, Zeek, and Argus are examples of network traffic analysis frameworks that are good at looking at network and transport layer protocols but not so good at looking at Layer 2 protocol behavior. These tools are unable to retrieve the detailed protocol-specific attributes required for identifying advanced attacks aimed at essential network communication mechanisms, including spanning tree manipulation [18], CDP exploitation, or dynamic trunking protocol vulnerabilities. Our contribution: We create DLLFlowLyzer, a specialized feature extraction framework that calculates over 343 different features, including MAC address analysis with vendor identification through OUI databases, protocol-specific characteristics for STP, CDP, ARP, and DHCP, such as message age ratios and topology change detection, and cross-protocol interaction metrics. This gives us an unprecedented view of Layer 2 traffic patterns, as explained in Section 4.6.

Gap 3: Limitations of Static Rule-Based Detection. Most current IDS solutions for Layer 2 security depend on static signature-based rules and are unable to identify new attack variants or complex multi-stage attacks that integrate various exploitation techniques, as noted by Shanker and Singh [20] and O'Connor [23]. Traditional methods that use Snort IDS work well against known attack patterns, but they have trouble with zero-day exploits and complex attacks that look like normal traffic patterns [20]. Security policy frameworks based on ISO 27002 standards [21] offer protection at the organizational level, but they aren't flexible enough to keep up with new ways of attacking. These systems have a lot of false positives and can't change to fit changing network conditions. Our contribution: We suggest using a deep learning-based method with a CNN-LSTM-Attention hybrid architecture that learns hierarchical representations of attack patterns. This lets us find both known attacks and new variants using learned feature abstractions instead of rules we made ourselves. We get a 99.67% F1-score across all attack categories.

Gap 4: Ignoring Temporal Patterns in Attack Detection. Current deep learning network intrusion detection systems view network flows as separate samples, not taking into account the time-based dependencies and sequential patterns that are common in many Layer 2 attacks. Multi-stage attacks, including ARP poisoning campaigns [22], progressive DHCP starvation [10], and coordinated impersonation attacks [17], manifest across multiple flows and display distinctive temporal progressions that are inadequately detectable via per-flow analysis. The literature on black hole and grey hole attacks in mobile ad-hoc networks [9] shows that sequence-based detection mechanisms are needed for attacks that happen over time. Our contribution: We present temporal windowing utilizing sliding windows of 30 consecutive flows, integrated with LSTM-based temporal modeling to explicitly identify attack progression patterns across sequences of network flows. This approach overcomes the constraints of instantaneous per-flow classification and facilitates the detection of attacks that extend across multiple time steps.

Gap 5: No Deep Learning Models for Layer 2. While deep learning NIDS have shown encouraging outcomes [46], current models are mainly assessed on higher-layer traffic from datasets like NSL-KDD and UNSW-NB15, concentrating on DoS, DDoS, and web attacks instead of Layer 2 protocol vulnerabilities. Studies on wireless network security [24, 25] and IoT environments [27] focus on particular areas

but do not offer general-purpose frameworks for thorough Layer 2 attack detection. To our knowledge, no previous research has employed advanced deep learning architectures, including attention mechanisms or transformer-based models, specifically for Data Link layer frame analysis with extensive Layer 2 feature sets that integrate ARP, STP, CDP, DHCP, and VLAN protocols concurrently. Our contribution: We create the first deep learning architecture with enhanced attention specifically for Layer 2 intrusion detection. This architecture uses convolutional spatial feature extraction to recognize local protocol patterns, temporal LSTM modeling to understand attack sequences, and adaptive attention weighting to focus on the most important temporal positions and protocol features for each type of attack.

Gap 6: Security decision-making is hard to understand. Many people don't like deep learning models for intrusion detection because they are "black boxes" that make predictions without explaining them. This makes them less useful in security operations centers, where analysts need to know why a detection was made. Not being able to understand how models make decisions makes it harder to trust them, fix bugs, respond to incidents, and learn more about attack characteristics. Current IDS research seldom considers the explainability requirements crucial for operational security applications. Our input: We combine several complementary explainability tools, such as attention weight visualization, which shows which time periods had an effect on classifications; SHAP value analysis [47, 48], which measures the contributions of each feature to each prediction; and LIME [49, 50], which makes local interpretable approximations. These methods give security analysts useful information about how models work, which lets them check detection logic, find attack signatures, and improve security policies based on what they've learned.

This systematic analysis of research gaps shows that while the literature gives us useful basic information about threats to the Data Link layer, such as ARP vulnerabilities [14, 22], STP weaknesses [18], DHCP attacks [10], MAC flooding [8, 36], VLAN exploitation [13], and basic countermeasures, there are still major problems with datasets, feature extraction tools, detection methods, temporal modeling, specialized architectures, deployability, and interpretability. Our research directly addresses each of these identified gaps through integrated contributions spanning dataset creation, feature extraction framework development, and novel deep learning architecture design specifically optimized for Data Link layer intrusion detection. The next parts explain how we plan to improve Layer 2 network security in a big way.

3 Proposed Model (Late Fussion)

This section presents a novel Memory-Efficient CNN-LSTM-Attention Hybrid Architecture for real-time Data Link Layer intrusion detection. The suggested model addresses the main problem of identifying and classifying advanced Layer 2 attacks by modeling time in a new way that accounts for how network flow patterns depend on one another over time. Traditional flow-based intrusion detection systems look at each flow separately [51, 52]. The proposed architecture, on the other hand, uses slidisliding-window techniques and deep learning to explicitly model how different network flows relate to one another time. Architecture combines three different deep learning paradigms to improve detection and classification while keeping memory usage low enough to work in networks with limited resources. The design signifies a substantial improvement over current methodologies in the field of Layer 2 intrusion detection [46, 53].

3.1 Architecture Overview and Design Philosophy

The proposed architecture incorporates three synergistic deep learning paradigms: Convolutional Neural Networks (CNNs) for spatial feature extraction [54], Long Short-Term Memory (LSTM) networks for modeling temporal dependencies [55], and attention mechanisms for adaptive feature importance weighting [56, 57]. This hybrid design is based on the fact that Data Link Layer attacks have both spatial and temporal patterns. Individual flows have spatial patterns, while sequences of flows have temporal patterns.
Figure 4 shows the whole architecture, from preprocessing the input to the final classification.

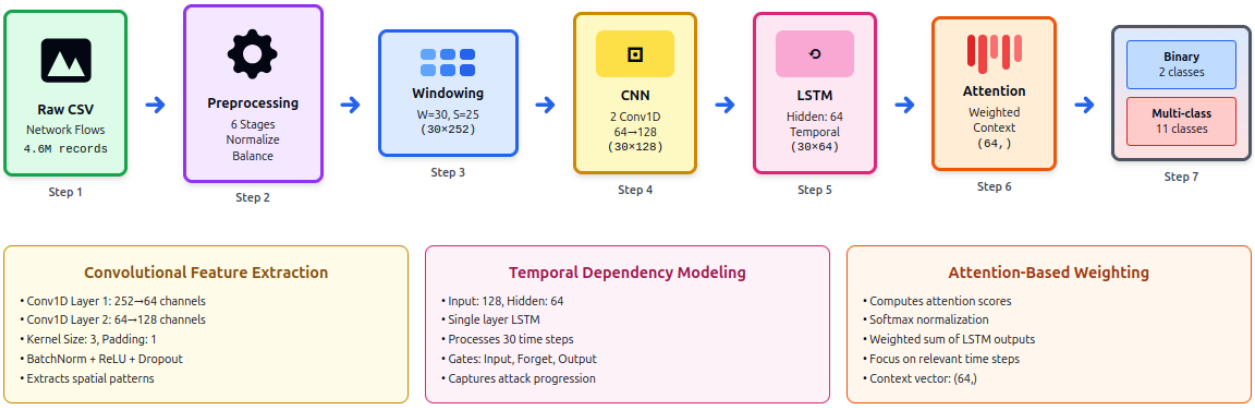

Figure 4: The overall structure of the Memory-Efficient CNN-LSTM-Attention Hybrid model for detecting intrusions at the Data Link Layer

3.2 Input Data Characteristics and Preprocessing Pipeline

The effectiveness of deep learning models for intrusion detection depends on the quality, comprehensiveness, and appropriate preparation of input data. Raw network traffic captures contain a lot of helpful information about how the network works and potential security threats. Still, they need to be systematically converted into structured formats that address these threats. For example, there are hundreds of features that make up a high-dimensional space, severe class imbalance (where benign traffic far outnumbers attacks), temporal

dependencies across sequential network flows, and the need to capture both spatial patterns within individual flows and temporal patterns across flow sequences.

This subsection describes the complete set of input data characteristics and the preprocessing pipeline that converts raw Data Link Layer network captures into temporal sequences suitable for machine learning. We start by discussing the dataset's properties, including the number of dimensions in the feature space, how the features are grouped, and how well the attack taxonomy covers them. Next, we explain the six-step preprocessing pipeline that systematically addresses data quality issues and feature engineering needs, creates a temporal structure, balances classes, normalizes data, and splits datasets—characteristics of network security data and architectural needs aligned with the CNN-LSTM-Attention model. The preprocessing method ensures the model receives high-quality input data that helps it learn to recognize different types of attacks while remaining fast enough for real-world use.

3.2.1 Dataset Characteristics

The model uses the DLLFlowLyzer framework from Section 4.6 to process network flow features taken from Data Link Layer traffic. The full dataset contains more than 4.6 million network flow records across 11 attack categories and benign traffic. This makes it one of the most enormous Layer 2 attack datasets reported in the literature [52, 58].

Each network flow is represented by a feature vector $\mathbf{x} \in \mathbb{R}^{252}$ containing 252 features across multiple categories. There are five features in the connection features category that show basic flow identifiers like protocol type, destination IP address, and source IP address. The MAC address features category has 19 features, such as source and destination MAC addresses, vendor identification through organizational unique identifiers (OUI), device type classification, vendor name resolution, and the individual/group (I/G) and local/global (L/G) bits that show unicast versus multicast transmission and locally administered addresses. There are 22 statistical measures in the frame analysis features category. These include frame length distributions, inter-arrival time statistics, directional traffic characteristics, and percentages of marked or ignored frames. The protocol-specific features category is the largest, with 227 features that let you examine the LLC, STP, DTP, ISL, ARP, DHCP, and CDP protocols. These features capture fields, state information, and behavioral patterns that are specific to each protocol. The protocol interaction features category has 9 cross-protocol metrics, such as STP-to-CDP ratios, ARP-to-DHCP ratios, Layer 2 protocol diversity measures based on Shannon entropy, protocol concentration indices, and weighted Layer 2 activity scores.

The dataset includes 11 attack types that make up the full taxonomy of Data Link Layer threats, as explained in Section 4.5.1. ARP spoofing attacks use the Address Resolution Protocol (ARP) to send traffic to the wrong destination [14]. Switch spoofing attacks use Dynamic Trunking Protocol (DTP) to set up unauthorized trunk links and get into multiple VLANs. ARP poisoning attacks send bad ARP messages to network devices to mess up their ARP cache tables [22]. Impersonation attacks leverage several Layer 2 exploitation methods to impersonate trusted network devices [17]. Content Addressable Memory (CAM) table flooding attacks overwhelm switch memory resources by generating frames with numerous spoofed source MAC addresses [8, 36].

VLAN attacks, such as VLAN hopping and double-tagging, exploit VLAN segmentation mechanisms to

access network segments that are not supposed to be accessible to them [13]. CDP attacks exploit weaknesses in the Cisco Discovery Protocol by sending malformed packets and flooding the network. DHCP spoofing attacks use fake DHCP servers to provide clients with incorrect network settings. DHCP starvation attacks use fake MAC addresses to quickly flood the DHCP server with requests, which exhausts its IP address space [10]. STP attacks use malformed Bridge Protocol Data Units (BPDUs) to disrupt Spanning Tree Protocol (STP) and cause network topology changes, or to make attackers root bridges for man-in-the-middle attacks. The benign class comprises standard, legitimate network traffic patterns used as a basis for detecting anomalies.

3.2.2 Comprehensive Preprocessing Pipeline

The preprocessing pipeline converts raw CSV data into machine learning-ready temporal sequences through six systematic stages, as shown in Figure 5.

Figure 5: Six-stage preprocessing pipeline transforming raw CSV network flow data into balanced, normalized temporal sequences suitable for deep learning. Each stage addresses specific data quality and modeling requirements.

3.2.2.1 Stage 1: Data Loading and Initial Processing

Raw CSV files containing network flows from multiple attack categories and benign traffic were loaded and unified into a single dataset. Missing values handled using forward fill interpolation:

$$x_i^{(j)} = \begin{cases} x_i^{(j)} & \text{if } x_i^{(j)} \neq \text{NaN} \\ x_{i-1}^{(j)} & \text{if } x_i^{(j)} = \text{NaN and } i > 1 \\ \mu_j & \text{if } x_i^{(j)} = \text{NaN and } i = 1 \end{cases} \tag{1}$$

where $x_i^{(j)}$ represents the j-th feature of the i-th sample, and μ_j is the feature mean computed over non-missing values. Duplicate entries are removed based on feature similarity using Euclidean distance threshold $\delta = 10^{-6}$ [59]:

$$\text{isDuplicate}(\mathbf{x}_i, \mathbf{x}_k) = \begin{cases} \text{True} & \text{if } \|\mathbf{x}_i - \mathbf{x}_k\|_2 < \delta \\ \text{False} & \text{otherwise} \end{cases} \tag{2}$$

Categorical features such as protocol types and device vendors are converted to numerical representations using label encoding.

3.2.2.2 Stage 2: Feature Engineering and Selection

Systematic feature selection that addresses redundancy and low variance reduces the original 277 CSV columns to 252 features. The Pearson correlation coefficient is used to find features that are very similar to each other [60].

$$\rho_{jk} = \frac{\sum_{i=1}^{N}(x_i^{(j)} - \mu_j)(x_i^{(k)} - \mu_k)}{\sqrt{\sum_{i=1}^{N}(x_i^{(j)} - \mu_j)^2}\sqrt{\sum_{i=1}^{N}(x_i^{(k)} - \mu_k)^2}} \tag{3}$$

For feature pairs with $|\rho_{jk}| > 0.95$, the feature with lower mutual information with the target label is removed. The variance threshold filter eliminates features with variance below $\sigma^2_{\text{min}} = 10^{-4}$:

$$\text{Var}(x^{(j)}) = \frac{1}{N}\sum_{i=1}^{N}(x_i^{(j)} - \mu_j)^2 > \sigma^2_{\text{min}} \tag{4}$$

This systematic reduction eliminates redundant information while preserving discriminative power, as validated through ablation studies presented in Section 4.

3.2.2.3 Stage 3: Temporal Window Creation

A key innovation in our architecture is the ability to turn individual flow records into time sequences that show how attacks progress. We use a sliding window method with a window size of $W = 30$ consecutive flows and a stride of $S = 25$ flows. This uses overlapping windows to ensure all time periods are covered [61, 62].

For a dataset with N flow records, the windowing operation creates N_w windows:

$$N_w = \left\lfloor \frac{N - W}{S} \right\rfloor + 1 \tag{5}$$

Each window $\mathbf{X}_i \in \mathbb{R}^{W \times 252}$ is constructed as:

$$\mathbf{X}_i = \begin{bmatrix} \mathbf{x}_{i \cdot S}^T \\ \mathbf{x}_{i \cdot S+1}^T \\ \vdots \\ \mathbf{x}_{i \cdot S+W-1}^T \end{bmatrix} \tag{6}$$

For sequences shorter than W, zero-padding is applied:

$$\mathbf{X}_{\text{padded}} = \begin{bmatrix} \mathbf{X} \\ \mathbf{0}_{(W-n) \times 252} \end{bmatrix}, \quad \text{where } n < W \tag{7}$$

This windowing strategy preserves time-based relationships, which are important for detecting multi-stage attacks that span multiple flows [52]. The overlapping windows ensure that attack patterns near window edges are captured in multiple windows, making detection more reliable.

3.2.2.4 Stage 4: Class Imbalance Handling

The original dataset has a very uneven class distribution, which is common for network security datasets. About 85% of the samples are benign traffic, and the attacks are unevenly distributed across the remaining 15%. If this imbalance isn't fixed, the model will tend to predict the majority class [63].

We use SMOTE-Tomek [64], a hybrid resampling method that combines Synthetic Minority Oversampling Technique (SMOTE) and Tomek Links cleaning. SMOTE uses linear interpolation to make fake samples of minority classes:

$$\mathbf{x}_{\text{synthetic}} = \mathbf{x}_i + \lambda(\mathbf{x}_{\text{nn}} - \mathbf{x}_i), \quad \lambda \sim \mathcal{U}(0,1) \tag{8}$$

where $\mathbf{x}_i$ is a minority class sample, $\mathbf{x}_{\text{nn}}$ is one of its k-nearest neighbors (typically $k = 5$) from the same class, and λ is a random interpolation factor.

Tomek Links identify and remove borderline and noisy samples. A pair $(\mathbf{x}_i, \mathbf{x}_j)$ from different classes forms a Tomek Link if:

$$\forall \mathbf{x}_k \neq \mathbf{x}_i, \mathbf{x}_j : \quad \|\mathbf{x}_i - \mathbf{x}_j\|_2 < \min(\|\mathbf{x}_i - \mathbf{x}_k\|_2, \|\mathbf{x}_j - \mathbf{x}_k\|_2) \tag{9}$$

The majority of the class sample from each Tomek Link is removed, cleaning the decision boundary. After SMOTETomek, the dataset achieves approximately balanced class distribution with 61,500 total samples distributed uniformly across 11 classes, enabling unbiased learning.

3.2.2.5 Stage 5: Data Normalization and Scaling

All features are normalized to zero mean and unit variance using StandardScaler [65]:

$$x_i^{(j)} \leftarrow \frac{x_i^{(j)} - \mu_j}{\sigma_j} \tag{10}$$

where:

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_i^{(j)}, \quad \sigma_j = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i^{(j)} - \mu_j)^2} \tag{11}$$

Standardization is essential for deep learning convergence [66], guaranteeing that all features contribute proportionately to gradient updates, irrespective of their initial scale. Normalization parameters $\{\mu_j, \sigma_j\}$ computed on the training set are stored and applied identically to validation and test sets to prevent information leakage.

3.2.2.6 Stage 6: Train-Validation-Test Splitting

Using stratified sampling to keep the class distribution, the balanced dataset is split into three parts: training (70 percent), validation (15 percent), and test (15 percent). 43,050 samples in the training set are only used for gradient-based parameter optimization. There are 9,225 samples in the validation set that are used for early stopping and hyperparameter tuning. The test set has 9,225 samples that were not used in the final evaluation. To avoid testing old data, each split keeps temporal consistency.

3.3 Model Architecture: Layer-by-Layer Design

The complete architecture processes windowed input $\mathbf{X} \in \mathbb{R}^{B\times W\times F}$ where B is batch size, $W = 30$ is window size, and $F = 252$ is the number of features. The architecture consists of four sequential components illustrated in Figure 6.

3.3.1 Layer 1: Convolutional Feature Extraction

The first part uses two stacked Conv1D layers with batch normalization and dropout to get spatial features. One-dimensional convolution is especially good for sequential data because it picks up on local patterns and interactions between time steps [54].

The first convolution layer changes the input with a kernel size of $K = 3$, a padding of 1, and 64 output channels:

$$\mathbf{H}^{(1)} = \text{Conv1D}(\mathbf{X}; W_c^{(1)}, \mathbf{b}_c^{(1)}) \tag{12}$$

where the convolution operation for each channel c and time step t is:

$$h_t^{(1,c)} = \sum_{k=0}^{K-1}\sum_{f=1}^{F} W_{c,k,f}^{(1)} \cdot x_{t+k}^{(f)} + b_c^{(1)} \tag{13}$$

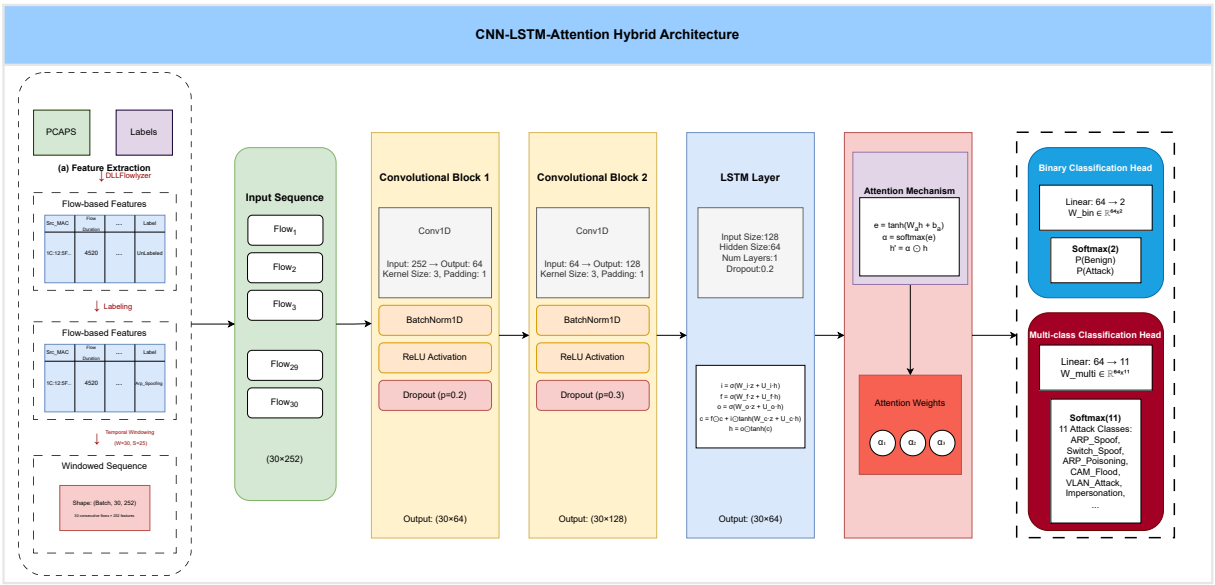

Figure 6: A detailed drawing of the architecture that shows all the layers and their sizes.

Batch normalization follows [66]:

$$\mathbf{H}_{\text{bn}}^{(1)} = \gamma^{(1)} \odot \frac{\mathbf{H}^{(1)} - \mu^{(1)}}{\sqrt{(\sigma^{(1)})^2 + \varepsilon}} + \beta^{(1)} \tag{14}$$

where $\mu^{(1)}$ and $\sigma^{(1)}$ are mini-batch statistics, $\gamma^{(1)}$ and $\beta^{(1)}$ are learnable parameters, and $\varepsilon = 10^{-5}$ ensures numerical stability.

ReLU [67] activation introduces non-linearity:

$$\mathbf{A}^{(1)} = \text{ReLU}(\mathbf{H}_{\text{bn}}^{(1)}) = \max(0, \mathbf{H}_{\text{bn}}^{(1)}) \tag{15}$$

Dropout with rate $p_1 = 0.2$ provides regularization:

$$\mathbf{A}_{\text{drop}}^{(1)} = \frac{1}{1-p_1} \cdot (\mathbf{m}^{(1)} \odot \mathbf{A}^{(1)}), \quad m_i^{(1)} \sim \text{Bernoulli}(1-p_1) \tag{16}$$

The second convolutional block transforms the output from 64 to 128 channels:

$$\begin{aligned} \mathbf{H}^{(2)} &= \text{Conv1D}(\mathbf{A}_{\text{drop}}^{(1)}; W_c^{(2)}, \mathbf{b}_c^{(2)}) \\ \mathbf{H}_{\text{bn}}^{(2)} &= \text{BN}(\mathbf{H}^{(2)}) \\ \mathbf{A}^{(2)} &= \text{ReLU}(\mathbf{H}_{\text{bn}}^{(2)}) \\ \mathbf{Z}_{\text{cnn}} &= \text{Dropout}(\mathbf{A}^{(2)}, p_2 = 0.3) \end{aligned} \tag{17}$$

The output $\mathbf{Z}_{\text{cnn}} \in \mathbb{R}^{W \times 128}$ contains hierarchical spatial features with early layers capturing low-level patterns and deeper layers learning complex feature combinations.

3.3.2 Layer 2: LSTM Temporal Modeling

The CNN output feeds into a Long Short-Term Memory (LSTM) network [55] that captures long-term temporal dependencies. The LSTM processes each time step $t \in [1, W]$ with hidden state $\mathbf{h}_t \in \mathbb{R}^{64}$ and cell state $\mathbf{c}_t \in \mathbb{R}^{64}$:

$$\mathbf{i}_t = \sigma(W_i \mathbf{z}_t + U_i \mathbf{h}_{t-1} + \mathbf{b}_i) \tag{18}$$

$$\mathbf{f}_t = \sigma(W_f \mathbf{z}_t + U_f \mathbf{h}_{t-1} + \mathbf{b}_f) \tag{19}$$

$$\mathbf{o}_t = \sigma(W_o \mathbf{z}_t + U_o \mathbf{h}_{t-1} + \mathbf{b}_o) \tag{20}$$

$$\tilde{\mathbf{c}}_t = \tanh(W_c \mathbf{z}_t + U_c \mathbf{h}_{t-1} + \mathbf{b}_c) \tag{21}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{22}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{23}$$

where $\mathbf{z}_t \in \mathbb{R}^{128}$ is the CNN output at time t, $\sigma(\cdot)$ is the sigmoid function, W_*, U_* are learnable weight matrices, and $\odot$ denotes element-wise multiplication. The gates control information flow enabling the LSTM to maintain information over extended sequences. Internal dropout with rate 0.2 prevents overfitting. The complete output sequence $\mathbf{H}_{\text{lstm}} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_W] \in \mathbb{R}^{W \times 64}$ encodes temporal patterns.

3.3.3 Layer 3: Attention Mechanism

The attention mechanism [56, 68] enables the model to focus on the most relevant time steps. Given the LSTM output sequence $\mathbf{H}_{\text{lstm}} \in \mathbb{R}^{W \times 64}$, attention computes scores for each time step:

$$\mathbf{e}_t = W_a \mathbf{h}_t + \mathbf{b}_a, \quad t \in [1, W] \tag{24}$$

where $W_a \in \mathbb{R}^{1 \times 64}$ and $\mathbf{b}_a \in \mathbb{R}$ are learnable parameters. Scores are normalized via softmax:

$$\alpha_t = \frac{\exp(e_t)}{\sum_{j=1}^{W} \exp(e_j)}, \quad \sum_{t=1}^{W} \alpha_t = 1 \tag{25}$$

The attended representation is a weighted sum:

$$\mathbf{c} = \sum_{t=1}^{W} \alpha_t \mathbf{h}_t \in \mathbb{R}^{64} \tag{26}$$

This context vector aggregates information across all time steps, emphasizing discriminative temporal positions. Unlike simple pooling, attention learns to weight time steps adaptively.

3.3.4 Layer 4: Dual-Head Classification

The final component employs a dual-head architecture for multi-task learning [69], performing both binary and multi-class classification.

The binary head detects whether traffic is benign or malicious:

$$\mathbf{z}_{\text{binary}} = W_{\text{bin}}\mathbf{c} + \mathbf{b}_{\text{bin}} \in \mathbb{R}^2 \tag{27}$$

$$\hat{\mathbf{y}}_{\text{binary}} = \text{Softmax}(\mathbf{z}_{\text{binary}}) \tag{28}$$

The multi-class head identifies specific attack types:

$$\mathbf{z}_{\text{multi}} = W_{\text{multi}}\mathbf{c} + \mathbf{b}_{\text{multi}} \in \mathbb{R}^{11} \tag{29}$$

$$\hat{\mathbf{y}}_{\text{multi}} = \text{Softmax}(\mathbf{z}_{\text{multi}}) \tag{30}$$

The dual-head design provides complementary supervision signals during training: the binary head encourages general attack discrimination, while the multi-class head enforces attack-specific representations.

3.4 Memory Optimization Strategies

To get high performance while keeping memory usage low, we had to make systematic changes to the architecture and algorithms that cut memory usage from more than 16GB to 8GB for training and 2GB for inference.

3.4.1 Compact Architecture Design

Compared to standard architecture, the channel dimensions are strategically smaller. The convolutional layers use 64 and 128 channels, fewer than the 128-256 channels most architectures use. Instead of the usual 128-256 units, the LSTM layer has 64 hidden units. Instead of the typical 2-3 stacked layers, the architecture uses only a single LSTM layer.

The total number of parameters is about 2.1 million, which is 60% fewer than in unoptimized baseline architectures. About 73,000 parameters come from the convolutional layers. The LSTM layer adds about 1.9 million parameters, which is $4 \times (128 \times 64 + 64 \times 64 + 64)$, where the 4 accounts for the gates. The attention mechanism only adds 65 parameters. The two classification heads add about 834 parameters.

3.4.2 Gradient Accumulation

Gradient accumulation lets you simulate larger batch sizes without taking up as much memory [**?**, 70]. Gradients are gathered from several mini-batches before being updated:

$$\nabla_\theta \mathcal{L}_{\text{effective}} = \frac{1}{K}\sum_{k=1}^{K} \nabla_\theta \mathcal{L}_k \tag{31}$$

where $K = 4$ is the number of steps that add up. We use a real batch size of 32 and 4 accumulation steps, yielding an adequate batch size of 128. This gives you the stability benefits of larger batches while requiring only 32 samples of memory.

3.4.3 Mixed Precision Training

Mixed precision training [71] cuts memory use by about half. Using 16-bit floating-point (float16) arithmetic for forward and backward pass calculations halves the memory required for activations. The optimizer state and parameter updates are kept in 32-bit floating-point (float32) precision.
Automatic loss scaling stops gradient underflow by scaling the loss before backpropagation:

$$\mathcal{L}_{\text{scaled}} = s \cdot \mathcal{L}, \quad s \in [2^{10}, 2^{16}] \tag{32}$$

Gradients are unscaled before parameter update:

$$\nabla_\theta \mathcal{L} = \frac{1}{s} \nabla_\theta \mathcal{L}_{\text{scaled}} \tag{33}$$

Dynamic loss scaling automatically adjusts the scaling factor during training to ensure numerical stability.

3.4.4 Streaming Data Processing

Streaming data processing solves the problem of loading datasets that are too large to fit in RAM. Using pandas chunked reading, data is processed in groups of 10,000 to 50,000 samples. Each group goes through the pipeline before being freed up. While processing the current chunk, prefetching loads the next one. This lets you train on the 4.6 million sample dataset without running out of memory on your system.

3.5 Training Configuration and Optimization

To train deep neural networks effectively, you need to set up the optimization algorithms carefully, learning rate schedules, loss functions, and hyperparameters that work together to control how the model converges and how well it performs in the end. The training process changes the network parameters that were set at random into a learned representation that can distinguish between different attack patterns with high accuracy. The optimization strategy guides this transformation. It must find a balance between several competing goals, such as fast convergence to reduce training time, stable learning dynamics to avoid gradient instabilities, effective generalization to new data rather than memorizing training samples, and computational efficiency to make training feasible on the hardware resources already available.
This section details the full training setup used for our CNN-LSTM-Attention Hybrid Architecture. We talk about the adaptive optimization algorithm that was chosen for parameter updates, the learning rate scheduling strategy that changes the step size during training, the multi-component loss function that was made to handle both binary and multi-class classification goals while dealing with class imbalance, and the complete set of hyperparameters that control batch sizes, regularization strengths, and early stopping criteria.

3.5.1 Optimizer Setup

The model is trained using the AdamW optimizer [72], which implements decoupled weight decay regularization:

$$
\begin{aligned}
\mathbf{m}_t &= \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\nabla_\theta \mathcal{L}_t \\
\mathbf{v}_t &= \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)(\nabla_\theta \mathcal{L}_t)^2 \\
\hat{\mathbf{m}}_t &= \frac{\mathbf{m}_t}{1-\beta_1^t}, \quad \hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1-\beta_2^t} \\
\theta_t &= \theta_{t-1} - \eta \left(\frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \varepsilon} + \lambda \theta_{t-1} \right)
\end{aligned} \tag{34}
$$

where $\lambda = 10^{-4}$ is the weight decay coefficient, $\eta = 10^{-3}$ is the initial learning rate, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ are exponential decay rates, and $\varepsilon = 10^{-8}$ ensures numerical stability.

3.5.2 Learning Rate Scheduling

Cosine annealing learning rate scheduling [73] modulates the learning rate following a cosine curve:

$$
\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min}) \left(1 + \cos\left(\frac{T_{\text{cur}}}{T_{\max}} \pi \right) \right) \tag{35}
$$

where $\eta_{\max} = 10^{-3}$ is the maximum learning rate, $\eta_{\min} = 10^{-6}$ is the minimum, T_{cur} is the current epoch, and $T_{\max}$ is the total epochs in the cycle. The schedule implements warm restarts every 10 epochs, resetting to $\eta_{\max}$ after each cycle to enable exploration of different loss landscape regions.

3.5.3 Loss Function Design

The loss function combines weighted cross-entropy [63, 74] terms from both classification heads:

$$
\mathcal{L}_{\text{total}} = \alpha \mathcal{L}_{\text{binary}} + (1-\alpha)\mathcal{L}_{\text{multi}} \tag{36}
$$

where $\alpha = 0.3$ weights the relative importance. The binary classification loss employs standard cross-entropy:

$$
\mathcal{L}_{\text{binary}} = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i^{\text{bin}} \log(\hat{y}_i^{\text{bin}}) + (1 - y_i^{\text{bin}}) \log(1 - \hat{y}_i^{\text{bin}}) \right] \tag{37}
$$

The multi-class loss incorporates class weights to address residual imbalance:

$$
\mathcal{L}_{\text{multi}} = -\frac{1}{N} \sum_{i=1}^{N} w_{c_i} \sum_{k=1}^{11} y_{ik} \log(\hat{y}_{ik}) \tag{38}
$$

Where class weights are computed using the balanced strategy:

$$
w_k = \frac{N}{11 \cdot N_k} \tag{39}
$$

This weighting ensures each class contributes equally regardless of sample count.

3.6 Inference and Deployment Considerations

This section discusses the practical aspects of deploying the trained CNN-LSTM-Attention model in a real network security infrastructure. We discuss the real-time processing pipeline architecture that tracks time using sliding-window buffers and classifies flows with a latency of less than 100 milliseconds. We explain the batch-processing features that enable quick analysis of high-throughput traffic scenarios common in business networks. We look at the memory-efficiency features that will allow it to be used on network appliances and edge devices with limited resources, where centralized processing isn't feasible. Lastly, we present integrated explainability tools, such as attention-weight visualization, SHAP value analysis, and LIME approximations, that make model reasoning clear. This lets security analysts check detection logic, learn about attack signatures, and trust automated security decisions.

3.6.1 Real-Time Processing Pipeline

The trained model enables real-time network traffic analysis via an effective inference pipeline. The sliding window implementation keeps a buffer of the last 30 network flows. When a new flow arrives, it is added to the buffer, and the oldest flow is removed. The same normalization settings that were learned during training are used to preprocess the window. A single forward pass makes classification predictions with a delay of less than 100 milliseconds.

Batch processing lets you quickly analyze large amounts of data that arrive in a single batch. The model can handle batches of up to 1000 windows at once, and can process more than 10,000 flows per second on GPU hardware and about 1000 flows per second on modern CPUs.

3.6.2 Explainability and Interpretability

For security applications, it's essential to understand what a model predicts. The architecture has several explainability mechanisms that help people understand how classification decisions are made.

Attention-weighted analysis identifies the time periods within the 30-flow window that had the most significant impact on the classification decision. High attention weights indicate time steps with typical attack signatures. Security analysts can review these highlighted flows to see which attacks were found.

SHAP (SHapley Additive exPlanations) [47] values measure how much each input feature adds to each prediction using game-theoretic feature attribution:

$$\phi_j = \sum_{S \subseteq F \setminus \{j\}} \frac{|S|!(|F|-|S|-1)!}{|F|!} \left[f_{S\cup\{j\}}(x_{S\cup\{j\}}) - f_S(x_S) \right] \tag{40}$$

where ϕ_j is the SHAP value for feature j. Global feature importance aggregates SHAP values across the entire test set to identify the most critical features for each attack type.

LIME (Local Interpretable Model-agnostic Explanations) [49] generates local linear approximations explaining individual predictions:

$$\xi(x) = \arg\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{41}$$

where f is the complex model, g is a simple interpretable model, $\mathcal{L}$ measures fidelity, and $\Omega(g)$ penalizes complexity.

4 Experiments and Results

This section provides a comprehensive experimental evaluation of the proposed Memory-Efficient CNN-LSTM-Attention Hybrid Architecture. To begin, we explain how the testbed infrastructure and dataset-generation method were used to create the BCCC-DLLayer-IDS-2025 dataset, which contains more than 4.6 million labeled network flows across 11 Layer 2 attack categories. We then systematically examine training progression and convergence behavior, assess classification performance through various complementary metrics, benchmark against established baseline methods, and perform ablation studies to quantify the contributions of specific architectural components, including convolutional feature extraction, LSTM temporal modeling, attention mechanisms, and preprocessing strategies.

4.1 Experimental Setup

For the proposed architecture to be valid, it must be rigorously tested. This is also necessary to ensure that performance metrics accurately reflect what the model can do under conditions that can be repeated. The experimental setup includes the hardware, software frameworks, dataset partitioning strategies, and evaluation metrics that make up the empirical validation approach. This subsection describes the entire experimental setup, including the computing environment, the strategy for splitting the dataset into training, validation, and test sets while maintaining the class distribution, and the complete set of evaluation metrics used to assess the model's performance. All experiments follow best practices, such as using fixed random seeds for reproducibility, running multiple independent trials to ensure statistical validity, using holdout test sets to ensure fairness, and conducting ablation studies to determine how much each component contributes.

4.1.1 Hardware and Software Configuration

We ran all experiments on a high-performance computing workstation with an NVIDIA RTX 3060 GPU with 6GB GDDR5 memory, an Intel Core i7 processor with 16 CPU cores, and 64GB of system RAM. For deep learning framework implementation, the software environment used PyTorch version 2.0 or later, CUDA version 11.8 for GPU acceleration, Python version 3.9 as the primary programming language, scikit-learn version 1.2 for preprocessing and evaluation metrics, pandas version 1.5 for data manipulation, and NumPy version 1.24 for numerical computations.

All experiments used fixed random seeds in NumPy, PyTorch, and CUDA to ensure reproducibility. Where possible, deterministic algorithms were enabled. We did each experiment three times with different random starting points, and the results are shown as mean values with standard deviations.

4.2 Generate a new DL Layer Dataset (BCCC-DLLayer-IDS-2025)

This section explains how to create a complete, realistic Data Link Layer traffic dataset, which is crucial for training and testing an AI-based Intrusion Detection System. The methodology focuses on reproducibility, realism, and precise labeling, addressing recognized challenges in developing network security datasets, especially the trade-offs among realism, control, and privacy. As a first step, all previously created datasets

are carefully examined to find their strengths, weaknesses, and lessons learned. This makes sure that the new dataset builds on what has already been done. Creating traffic in a controlled testbed enables high-accuracy labeling of ground truth, which is often hard to achieve with real-world network captures. It took three months to make the dataset, with each phase lasting a week. The first two weeks were spent setting up and testing the infrastructure; the next three weeks were spent generating benign traffic; the next three weeks were spent running the attack scenario; and the last three weeks were spent processing and validating the data. This longer timeline makes sure that all network behavior patterns are covered.

4.2.1 Available Datasets

A thorough examination of existing network intrusion detection datasets reveals a fundamental and enduring issue: the lack of a comprehensive, annotated dataset for Data Link Layer attacks. The examination of foundational, contemporary, and specialized datasets has consistently shown that none of them provides the requisite scope, realism, and documentation to train and assess L2-specific intrusion detection models effectively.

KDD Cup 99 [37] and NSL-KDD [38]: These datasets are fundamentally obsolete. Their fake traffic and old attack vectors don't address the problems modern Ethernet and wireless networks face, nor do they help with understanding modern L2 threats. The UGR'16 [43] dataset, for instance, comprises real traffic and recent attacks collected from several NetFlow v9 collectors strategically placed within the network of a Spanish ISP. UNSW-NB15 [39] and the CIC family [40, 41, 42]; These datasets were a big step forward because they used both real-world traffic and hybrid methods, but their primary focus is still on higher-layer protocols. DoS, DDoS, and web attacks are some of the attacks they include. These attacks don't exploit or target weaknesses in L2 protocols, so they aren't suitable for building a Layer 2 IDS. Even though these resources cover important areas such as IoT and SCADA [44], they don't cover enough ground to build a comprehensive L2 attack taxonomy. They affirm the need for specialized datasets but fail to deliver the comprehensive solution required for extensive L2 security research. The CTU-13 dataset [45] is among the first. It was created in 2011 as a resource for finding botnets.

Table 1: Comparison of network intrusion detection datasets showing layer of focus, L2 attack coverage, and key limitations.

Dataset	Year	Use Case	Layer Focus	L2 Attacks	Specific L2 Attacks	Format	Feature Extraction	Key Shortcomings
KDD Cup 99 [37]	1999	Foundational IDS benchmark	L3/L4 & App	No	None	CSV	Yes, from raw TCP dump	Outdated traffic, redundancy, statistical imbalance, no modern L2 attacks
NSL-KDD [38]	2009	Foundational IDS benchmark	L3/L4 & App	No	None	CSV	Yes, derived from KDD Cup 99	Outdated traffic, derived from obsolete patterns
CTU-13 [45]	2011	Botnet detection	L3/L4 (Flows)	No	None	Both	Yes, CICFlowMeter	Highly specialized, limited to botnet traffic, lacks L2 attacks
UNSW-NB15 [39]	2015	General IDS, modern traffic	L3/L4 & App	No	None	Both	Yes, Argus & Bro-IDS	Attack taxonomy lacks explicit L2 attack focus
CIC-IDS2017 [40]	2017	General IDS, realistic traffic	L3/L4 & App	No	None	Both	Yes, CICFlowMeter	Primary attack focus not on L2-specific protocols
UGR'16 [43]	2017	Anomaly detection	L3/L4 (NetFlow)	No	None	CSV (flow)	Yes, NetFlow v9	Specialized for temporal analysis, not comprehensive L2 repository
CSE-CIC-IDS2018 [41]	2018	General IDS, modern traffic	L3/L4 & App	No	None	Both	Yes, 67 features	Attacks focused on higher layers, not L2
CIC-DDoS2019 [42]	2019	DDoS detection	L3/L4 & App	No	None	Both	Yes, CICFlowMeter-V3, 80+ features	Specialized for DDoS, lacks comprehensive L2 attacks
DNP3 [44]	2022	ICS/SCADA security	Mixed (L2/L3/App)	Yes	Spoofing, Eavesdropping, Modification, Replay, DoS, frame length attacks	Both	Yes	Highly domain-specific, not suitable as general L2 benchmark
BCCC-DLLayer-IDS-2025	**2025**	**Data Link Layer IDS**	**L2/L3**	**Yes**	**STP,DHCP Starvation, VLAN 802.1Q, DHCP Spoofing, Switch Spoofing, ARP spoofing, ARP Poisioning, CAM table Flooding, CDP**	**Both**	**Yes, DLLFlowLyzer**	—

This problem that brings everyone together needs a new way to be solved. The field has changed and

now knows that a ”one-size-fits-all” dataset is no longer possible. However, a very important part of the network stack has been mostly ignored. The deficiencies in current datasets validate that a new, specialized resource is not merely an enhancement but an essential advancement in network security research. The BCCC-DLLayer-IDS-2025 dataset and the DLLFlowLyzer framework that goes with it are the clear answers to the problems that were found. This project is the first to systematically address the long-standing need for a comprehensive, publicly available dataset that provides a specific and reproducible focus on Data Link Layer attacks.

BCCC-DLLayer-IDS-2025's method is meant to get around the problems that its predecessors had. It is based on a carefully built, reproducible testbed that closely resembles a typical network for a small to medium-sized business. This method, which uses separate physical loops for STP and VLAN settings, makes sure that the traffic that is created is realistic and has the exact vulnerabilities needed for high-fidelity attack execution. The dataset's detailed list of L2 attacks, such as ARP Spoofing, DHCP Starvation, CAM table flooding, STP attacks, and VLAN Double Tagging, gives you the detailed, layer-specific data that other modern datasets don't have.

Also, the dataset is made to be useful right away for AI-driven research. It gives you not only the raw PCAP files but also a flow-based view of the traffic. This format is better for privacy and less resource-intensive than packet-level data, while still showing the main features of attacks. The project fills a crucial gap by creating this resource, which will lead to the creation of a new generation of L2-specific IDS models that can learn to find and deal with threats at the most basic level of the network.

4.3 Test-bed Architecture and Setup

This subsection gives a detailed account of the physical and logical network infrastructure set up for dataset generation, focusing on the exact settings that allow for both realistic benign traffic and targeted attack scenarios. As shown in Fig. 7, the test bed was made to closely resemble a typical network environment for a small to medium-sized business.

There were four Cisco switches (S1, S2, S3, and S4) and six laptops (PC1, PC2, PC3, PC4, PC5, and PC6). Most devices were connected to the Cisco switches through wired Ethernet, which was the main way to connect. Also, a TP-Link wireless modem gave PC6 wireless access, which connected both wired and wireless network segments. The planned addition of Cisco switches was a smart move because it made sure that Cisco Discovery Protocol (CDP) packets were in the infrastructure and made it possible to carry out real CDP attacks.

Certain design features were added to make it easier to carry out different types of attacks.:

Loop Configuration (Point 1 **in Fig. 7):** The network was set up on purpose to make a physical loop in this part. This design choice required the Spanning Tree Protocol (STP) to be turned on and run to stop network storms. This setup was very important for creating real STP traffic and gave us a safe place to try out different STP attacks and see how they affected the protocol's behavior.

VLAN Configuration (Point 2 **in Fig. 7):** There was a separate Virtual Local Area Network (VLAN) set up in the network. This separate part was very important for carrying out VLAN-based attacks like

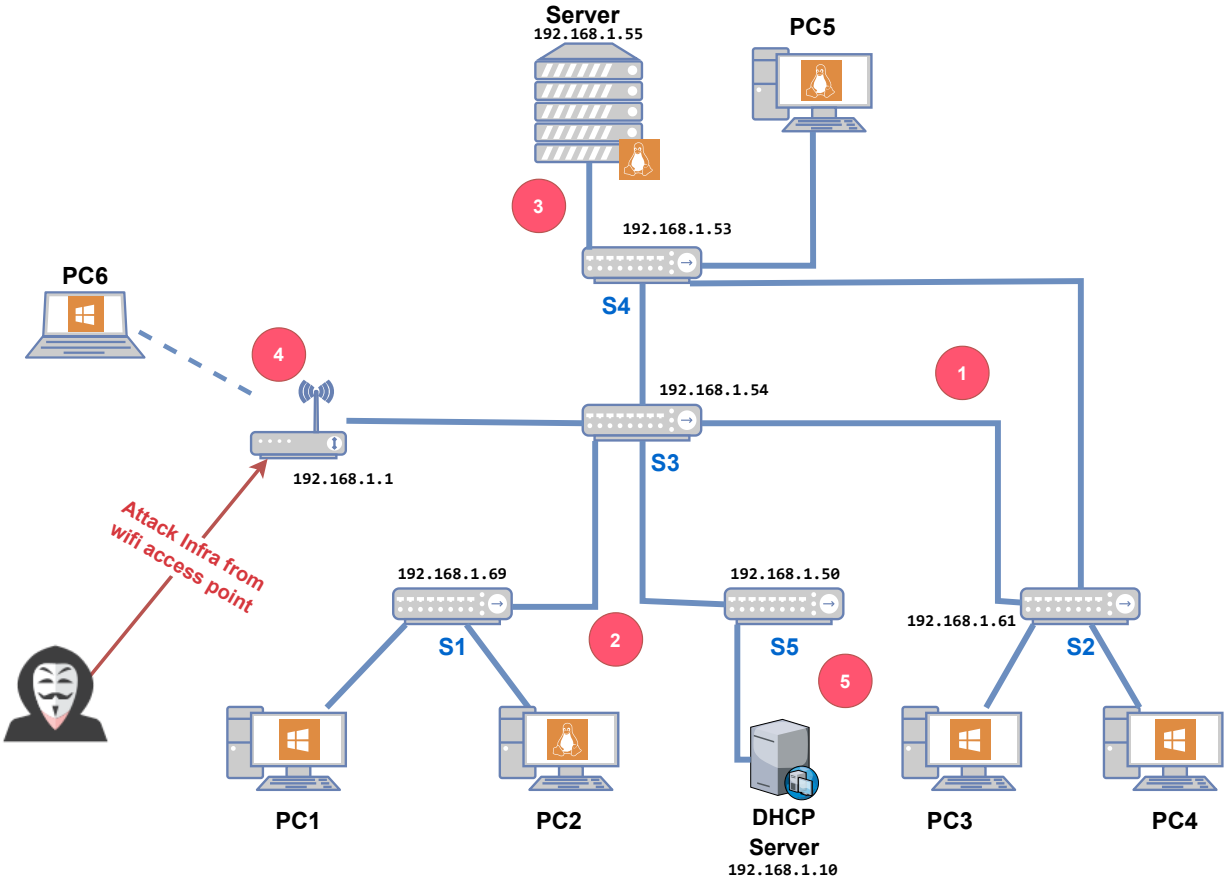

Figure 7: Test Bed Infrastructure

VLAN Double Tagging, which let you closely watch and control these specific Layer 2 exploits.

Service Provider (Point 3 in Fig. 7): There is a Service provider that is a server that all of the testbed's computers can connect to locally to read data, upload and download files, and so on.

Wireless Modem and Capture Points (Point 4 in Fig. 7): The TP-Link wireless modem was the internet gateway for the whole system. It connected wired devices to the root switch (S3) and let devices like PC6 and The Attacker connect wirelessly.

DHCP Server Configuration (Point 5 in Fig. 7): The DHCP server was set up to automatically give out IP addresses in the range of 192.168.1.20 to 192.168.1.200. The default gateway was set to 192.168.1.1, and the netmask was set to /24 (192.168.1.0/24). This detailed setup is essential for understanding and recreating the conditions under which DHCP-related attacks occurred, since these attacks often target the process of assigning IP addresses.

We also captured network traffic from the S3 switch (Infrastructure Attacks), which Parrot OS does [75]. In the infrastructure attack section (see Flooding and DOS Attack, Spoofing, and MitM Attacks), we assume the

attacker uses a wireless endpoint to obtain a reverse shell, then uses that to attack the network infrastructure at the DL layer.

The details of each device used are in Table 2.

Table 2: Test Bed Infrastructure Device List

Device	**Model**	**Specification**
PC1	Dell	OS: Windows 11
PC2	Dell	OS: Ubuntu 22.04
PC3	Lenovo	OS: Windows 11
PC4	Dell	OS: Windows 11
PC5	Lenovo	OS: Ubuntu 22.04
PC6	Dell	OS: Windows 11
S1–S4	Cisco WS-C2960-24TC-S	Catalyst 2960, 24 Ethernet 10/100 ports and 2 dual-purpose uplinks, LAN Lite software
S5	TP-Link TL-SG1024DE	24-Port Gigabit Easy Smart Switch with 24 10/100/1000 Mbps RJ45 ports
Wireless Modem	Archer C80	AC1900 Wireless MU-MIMO WiFi Router, Operating System: RouterOS, Wireless Standard: 802.11ac
PC (Attacker)	MSI Pulse GL76	OS: ParrotOS
Server	Raspberry Pi 5	OS: Ubuntu 22.04
DHCP Server	Raspberry Pi 5	OS: Ubuntu Server

4.4 Designing and implementing the Benign Traffic Generator

A week was set aside to generate benign user traffic to ensure the dataset accurately reflects real network conditions. This process aimed to carefully replicate different types of user behavior and interactions at the application level, an important aspect often overlooked in more general network models and synthetic data generation methods. The goal was to make a representative baseline of regular network activity so that bad traffic could be easily spotted. You can find the Python code in the Benign-User-Profiler-BUP [76] and User Behavior Simulator [77] repositories.

A custom-built generator, primarily written in Python, was used to simulate benign user behavior. It used libraries like Selenium to automate interactions with web browsers. This method gave us exact control over the traffic patterns we created. The generator was designed using ideas from statistical modeling of network traffic. Its goal was to add human-like variability to session lengths, data volumes, and time patterns, rather than creating synthetic flows that are identical or predictable. This makes it hard to distinguish between benign traffic and real user activity.

Each simulated activity was designed to generate network traffic patterns typical of common user applications.

Video Conferencing: In the simulation, there were video calls of different lengths with a fake number of people on each call. Traffic patterns included real-time audio and video streams that went both ways,

which is what you would expect from modern video conferencing apps in terms of bandwidth use and packet behavior. There were attempts to copy the media flow and signaling features.

Video Streaming: We simulated watching YouTube by watching different videos with different playback times. Simulated user interactions involved pausing and watching several videos in a row, which affected the adaptive streaming settings and network load.

SSH Sessions: SSH [78] traffic generation involved the creation of secure shell sessions, the simulation of interactive command execution (such as directory navigation and file editing), and the execution of secure file transfers (SCP/SFTP). To make the activity levels and session lengths more like real-life administrative or development workflows, they were randomized. This created both short, interactive bursts and longer, sustained periods of activity.

Media Downloads: The generator used standard protocols such as HTTP and HTTPS [79, 80] to simulate the download of media files of varying sizes, including images, documents, and videos. This made the network's traffic patterns bursty, which is typical of large data transfers, and tested the network's ability to handle high throughput.

Music Streaming: We pretended to stream music by going to these sites and playing songs from them. This made audio streams that were always going, but with a lower bandwidth. The simulation had songs of different lengths and allowed users to skip tracks.

Web Browsing: This involved visiting a variety of websites to mimic how a typical user browses the web. The generator made a mix of HTTP/HTTPS requests, DNS queries [81, 82], and responses. It also included idle times between page loads to show how users would really pause and scroll.

The "benign user behavior" was made for all activities with built-in randomness in start times, lengths, and activity levels. This method tried to get rid of fake periodicity and make the benign traffic profile more realistic overall, so the dataset would better represent how networks are used in the real world. There was a steady stream of benign traffic for a whole week.

4.5 Attack Scenarios and Execution:

This section discusses how various Data Link Layer attacks were carried out in the controlled testbed environment. The dataset timeline shows that the attacks were carried out on a Parrot OS [75] machine using a mix of specialized tools and custom scripts. It is believed the attacker first gained access to the network via a wireless endpoint. The S3 switch recorded all network traffic from all attack scenarios. While the attacker is doing the attack, the network's daily traffic is created using tools like Yersinia [83], BetterCap [84], Ethercap [85], and custom Python scripts, all of which are put together into a tool called Fyodost [86].

4.5.1 DL Layer's Attacks:

There are several common attacks on Ethernet Data Link layer and wireless networks, including DHCP starvation, VLAN Double Tagging, and CAM table flooding attacks [5, 6]. We can put DL later attacks into three groups ahead of time. Figure 8 shows the general view of The Attacks.

Spoofing and MitM:

Table 3: Timeline of the Benign and Attacks on Infrastructure

Task	Time	PCAP Size (GB)	Main Tool
Benign	7-Jul	20	-
Benign	8-Jul	33	-
Benign	9-Jul	27	-
Benign	10-Jul	32	-
Benign	11-Jul	28	-
ARP Spoofing	28-Jul	26	BetterCap
ARP Poisoning	29-Jul	28.2	EtterCap
CAM Flood	30-Jul	33	EtterCap
CDP	31-Jul	29	Yersinia
STP Attack	1-Aug	25	Yersinia
DHCP Starvation	4-Aug	34	Python Script
802.1Q	5-Aug	29	Yersinia
Impersonation Attack	6-Aug	31.2	Python Script
DHCP Spoof	7-Aug	37	Python Script
Switch Spoofing	8-Aug	27	Python Script
ARP Spoofing	11-Aug	37	BetterCap
Switch Spoofing	12-Aug	27	Python Script
Total	**17**	**503.4**	

Spoofing in the data link layer means pretending to be a device that you can trust. It generally fools the system into thinking that data is coming from a trusted device or that it is going to a trusted destination. The main goal of this kind of attack is to steal data, intercept packets (MitM attacks), or use it as a first step in a more advanced attack to hide tracks in the network.

MAC spoofing: People who want to do bad things often use MAC spoofing to change the Media Access Control (MAC) address of their device, making it appear as another device on the network. There are two ways to do this kind of attack: 1) Copying a MAC address and 2) Making it random. Cloning means copying a real device's MAC address to make it appear as that device on the network. This is a common way for an attacker to gain access to a router or switch if they have physical access. Randomization, on the other hand, makes a new MAC address to pretend to be a network device. This is how attackers copy a device's MAC address when they can't get to it.

DHCP spoofing and DHCP starvation attack: The DHCP protocol doesn't have authentication by default, and it's sent out as a broadcast. This means that any device on the network could see and change the messages. There are two main kinds of attacks on DHCP. A DHCP Starvation Attack sends a large number of DHCPREQUEST messages to the DHCP server within a short period, all with fake source MAC addresses. This exhausts the server's pool of available IP addresses, causing a race condition. The "starved" DHCP server will not respond to new DHCP requests until an address becomes available. In this case, the attacker can impersonate the DHCP server and send fake messages to trick other clients on the network. A Rogue DHCP Attack sets up a fake DHCP server, waits for broadcast requests, and sends fake responses with incorrect settings. The attacker usually wants to make themselves the DNS server and default gateway for the clients. The attacker opens port 53 on their machine for DNS activity. This means that every DNS

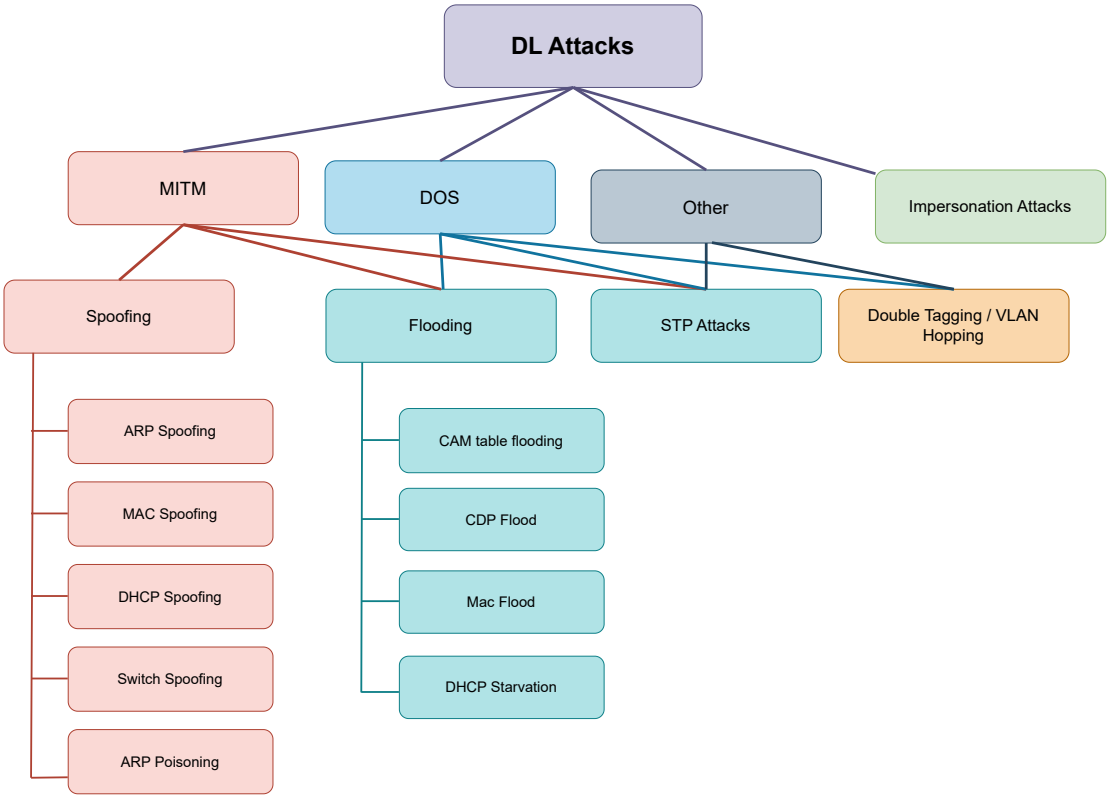

Figure 8: Data Link Layer Attacks

resolution request goes to their machine, and they can choose when to respond with their own hostname.

ARP Spoofing: The Address Resolution Protocol (ARP) enables computers to communicate over a network by translating IP addresses into Media Access Control (MAC) addresses and vice versa. ARP is used by devices to communicate with routers or gateways so they can reach the internet. But an attacker can use a fake ARP message to steal cookies and listen in on traffic between the user and the server by spoofing the user's MAC address. Because the ARP protocol doesn't check the integrity of ARP requests, it is a weak point that can be used in ARP attacks.

ARP Poisoning: In a broader sense, ARP poisoning is any change to ARP tables intended to trick network devices. It is a full-scale attack that uses ARP spoofing. By sending fake ARP messages over the network, attackers can trick devices into linking the wrong MAC addresses to the wrong IP addresses. Three types of ARP poisoning attacks exist. In Standard ARP Poisoning, the attacker modifies the ARP tables to redirect traffic to the wrong destination and steal packets. In a Reverse ARP Poisoning attack, the attacker's MAC address is used to cause the gateway router to believe the attacker's IP addresses are those of network devices. In Gratuitous ARP Attacks, hackers send fake ARP announcements that incorrectly associate IP addresses with MAC addresses, slowing network traffic.

Switch Spoofing and VLAN Hoping: Switch spoofing is when you change a device's MAC address to make it appear as an authorized switch port, thereby gaining access to the target network. Most networks use MAC addresses to verify devices and establish network connections because each device has a unique MAC

address. A malicious agent can easily gain access to a network by pretending to be an authorized device, one that the network recognizes as ”trusted.” MAC address spoofing is another name for switch spoofing.
Dynamic Trunking Protocol (DTP) establishes trunk links between two switches. To set up an interface for dynamic trunking and frame tagging, use the modes ”dynamic desirable,” ”dynamic auto,” and ”trunk.” A VLAN hopping attack is called switch spoofing. The attacker sends a DTP message from their computer to the switch, establishing a connection between the attacker and the switch. Once this trunk link is set up, the attacker can easily see packets on all VLANs.
Impersonation Attack: ElShafee and El-Shafai [17] created an attack that can perform MITM on a network with a service available, such as a camera service, by faking the MAC address of a trusted node to listen to and record traffic between the trusted node and a service provider.
Flooding and DOS:
Flooding attacks are denial-of-service (DoS) attacks in which attackers send excessive traffic to a target, making the service unavailable. The attacker sends many incomplete connection requests, which consume the target's resources and prevent real packets from being processed [87]. During the second stage, the attacker could perform a MitM attack to intercept network traffic.
CDP Flood: An attacker can use this feature to send thousands of fake Cisco Discovery Protocol (CDP) packets to the multicast MAC address 01:00:0C:CC:CC:CC. This will fill up the neighbor tables on any devices on the network that run CDP. A CDP flood can overload the entire switch and fill the MAC table. This overflow can cause the switch to act like a hub, sending frames out of all ports. You can use a tool like Wireshark to sniff all network traffic when this happens. This is because all frames are sent out of all ports, unlike a standard switch, which sends them only to the correct MAC address.
This kind of attack can also slow down the network and make it less secure because the attacker can steal and analyze sensitive information. To reduce the risk of CDP flooding attacks, you can use security measures such as port security, disabling unused ports, and dividing the network into smaller segments.
CAM table or MAC flooding: The MAC Flooding attack is a way to make network switches less secure. The attacker keeps sending packets to the switch to fill the MAC table. The MAC Address Table is complete and can't save any more MAC addresses. The switch will go into fail-open mode, which means it will act like a network hub. It will send incoming data to all ports, like broadcasting [36].
STP attack: Attacks on the Spanning Tree Protocol (STP) exploit weaknesses to create network loops or disrupt operations. Attackers can send harmful Bridge Protocol Data Units (BPDUs) to disrupt STP calculations, leading to network congestion, broadcast storms, and failures. STP doesn't have authentication so that any BPDU-enabled device can mess up the spanning tree. There are three main kinds of STP attacks: STP Information Disclosure, STP Denial of Service, and STP Man-in-the-Middle.
Switch ports lack security features that prevent unauthorized access to STP, leaving the network vulnerable to attacks. Using BPDU Guard and Root Guard, and turning off unused ports, can help lower these risks.
VLAN Double Tagging: The attacker sends double-encapsulated 802.1Q messages to the switch in a Double Tagging attack. The switch strips the outer tag, but the victim's PC's inner VLAN ID remains the same. This lets the attacker send network traffic to the victim's PC without going through VLAN segmentation. The attacker must be connected to the trunk port's native VLAN interface for double tagging to occur. This attack

only goes one way: the attacker can send traffic to the victim, but the victim can't send responses back to the attacker.

4.6 Designing and implementing a Feature Extractor tool for DLLayer (DLLFlowLyzer)

To analyze Data Link layer attacks, you need specialized feature-extraction tools that go beyond what current network traffic analysis tools can do. Established frameworks such as NTFlowLyzer [88], CICFlowMeter [89], Zeek [90], and Argus [91] have demonstrated their ability to analyze network and transport-layer protocols effectively. However, they don't provide enough information about how Layer 2 protocols work, which is important for detecting advanced attacks targeting the most basic layer of network communication. To fill this critical gap, we created DLLFlowLyzer, a comprehensive feature-extraction framework specifically designed to turn raw packet-capture data into machine-learning-ready representations with never-before-seen detail at the Data Link layer.

DLLFlowLyzer has a complex three-stage processing pipeline that strikes a good balance between speed and feature completeness while handling large datasets. The architecture prioritizes reliability by parsing packets directly, avoiding fragile wrapper libraries. This ensures that malformed frames and protocol violations, common in Layer 2 attacks, are handled appropriately. The framework's design philosophy emphasizes reproducibility. This is made possible by fixed random seeds, versioned configurations, and detailed audit logging that let you exactly replicate dataset-generation workflows across different execution environments.

4.6.1 Processing Pipeline Implementation

The processing pipeline of the framework has three stages that are linked together and each one is best suited for a different part of feature extraction. The first stage does direct packet parsing using low-level binary analysis. This means it doesn't rely on external parsing libraries, which often break when they encounter unusual protocol fields or deliberately malformed frames. This method ensures that edge cases such as truncated frames, protocol violations, and attack-specific packet malformations are handled correctly, which would cause regular parsers to fail. The parser keeps separate state machines for each supported protocol. This lets it accurately assemble conversations between protocols even when packets are lost, reordered, or when someone tries to change the protocol on purpose.

The second stage uses advanced flow-aggregation logic to assemble individual frames into flows that make sense based on configurable timeout settings and protocol-specific boundary conditions. The aggregation engine looks at both the time and the meaning of the protocol. It uses adaptive timeout mechanisms that vary based on the protocol's characteristics. For example, ARP flows have a 10-second timeout, which is in line with their temporary nature. TCP flows, on the other hand, have standard timeouts of 30 seconds for idle time and 120 seconds for active time. The system keeps separate flow tables for each protocol family. This makes memory usage and lookup performance better while still letting you find cross-protocol attack patterns.

The third stage includes full-feature computation and incremental export mechanisms to prevent memory from running out when processing large amounts of data. The feature extraction engine uses ThreadPoolExecutor for statistical calculations and maintains deterministic output order by carefully synchronizing threads.

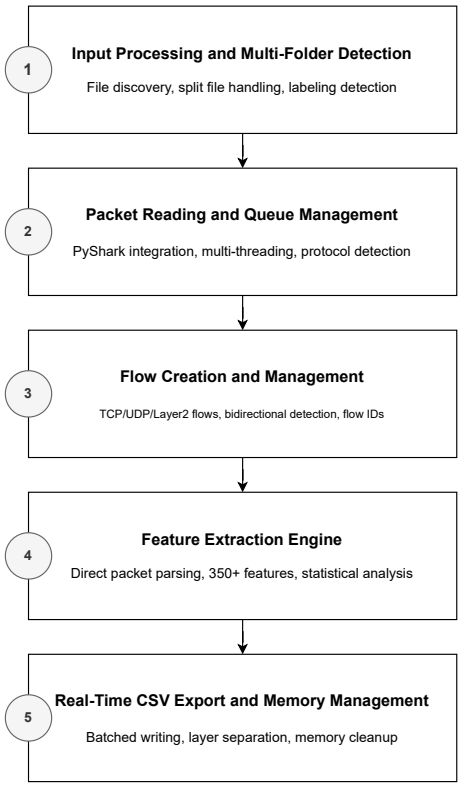

Figure 9: Flowchart of DLLFlowLyzer

Export operations use configurable batching strategies that balance memory use and I/O efficiency. The default settings process 30 flows per batch to keep memory usage under a gigabyte, even when processing captures with millions of packets.

4.6.2 Comprehensive Feature Extraction Capabilities

DLLFlowLyzer calculates more than 352 unique features for each network flow. This is the most complete Layer 2 feature set available in any publicly available tool. We carefully selected these features by reviewing many Layer 2 attack patterns described in academic papers and consulting network security experts. The feature extraction methodology comprises several analytical dimensions, each intended to identify specific behavioral patterns indicative of malicious activity.

The framework performs a complete MAC address analysis that goes beyond simple address extraction. It also uses an Organizationally Unique Identifier (OUI) database to find vendors. The integrated database contains over 30,000 vendor entries, updated every 3 months. The system pulls out the Individual/Group (I/G) bit, which shows whether the transmission is unicast or multicast, and the Local/Global (L/G) bit, which indicates locally administered addresses that could be used for virtualization or spoofing. The vendor analysis features can distinguish between device types, such as routers, switches, access points, and end-user

devices. This enables advanced device fingerprinting and the detection of unusual behavior.

Category	Count	Description	Key Features
Flow Identifiers	3	*Basic flow identification and protocol classification*	connection, tcp_stream, protocol
MAC Address Features	14	*Source/destination MAC analysis with vendor information*	MAC vendors, OUI parsing, I/G and L/G bits
Network Layer Features	5	*IP addresses, ports, and Ethernet types*	IP addresses, ports, Ethernet types
Timing Features	1	*Flow duration analysis*	time_delta (flow duration)
Frame Statistics	32	*Packet size, direction, and quality metrics*	Packet counts, sizes, directional statistics
LLC Features	15	*Logical Link Control protocol analysis*	SAP values, control fields, C/R bits
STP Features	19	*Spanning Tree Protocol comprehensive analysis*	Bridge IDs, message ages, topology flags
Loop Detection Features	8	*Network loop identification and analysis*	Loop skip counts, network stability metrics
DTP Features	15	*Dynamic Trunking Protocol analysis*	Trunk negotiation, version, sender information
ISL Features	18	*Inter-Switch Link protocol analysis*	VLAN analysis, frame type detection
ARP Features	36	*Address Resolution Protocol detailed analysis*	Hardware/protocol types, opcodes, address mappings
DHCP Features	37	*DHCP protocol comprehensive analysis*	BOOTP fields, flags, options, message types
CDP Features	69	*Cisco Discovery Protocol extensive analysis*	TLV parsing, capabilities, power management
Advanced Statistical Features	16	*Cross-protocol ratios and diversity metrics*	Protocol ratios, diversity indices, concentration metrics

Figure 10: Extracted features categories in DLLFlowLyzer

Statistical traffic features use multidimensional analysis to get a complete picture of flow characteristics. The framework calculates directional statistics independently for source-to-destination and destination-to-source traffic, facilitating the identification of asymmetric communication patterns indicative of diverse attack types. These statistics include standard measures like the mean, median, standard deviation, and coefficient of variation for packet sizes and inter-arrival times. There are also normalized versions that let you compare flows of different sizes in a meaningful way. Temporal analysis includes burst-detection metrics, which are very useful for finding flooding attacks and reconnaissance activities. The burst ratio is the number of packets that arrive faster than half the median inter-arrival time.

4.6.3 Protocol-Specific Feature Engineering

The framework uses deep protocol inspection of necessary Layer 2 protocols to extract semantic features that capture how protocols behave, which is essential for detecting attacks. The system extracts operation codes, hardware, and protocol type fields, along with complete address mappings, for ARP protocol analysis. It finds spoofing attacks by looking for unusual request/reply ratios in statistical distributions of ARP opcodes across flows. The framework tracks unnecessary ARP patterns and uses time windows to detect cache poisoning attempts that exploit the ARP protocol's lack of state.

With 16-bit granularity, the STP feature extraction captures all important BPDU characteristics, such as message age, forward delay, topology change flags, and bridge priorities. The system calculates the ratio of message age to the maximum age, giving us numbers that show how BPDU propagation works and could

indicate STP manipulation attacks. Statistical analysis of topology change notifications can help find network problems that attackers might be trying to cause by changing the spanning tree to intercept traffic or launch denial-of-service attacks.

CDP analysis includes retrieving all Type-Length-Value (TLV) fields, including device capabilities, VLAN settings, and power management. The framework processes CDP ads to obtain device identification information, capability flags indicating whether a router, switch, or phone can perform a specific function, and power consumption data for Power over Ethernet (PoE) environments. Statistical summaries of address counts, capability distributions, and power requirements help find CDP flooding attacks and find rogue devices that are trying to connect to the network without permission.

To find starvation and rogue server attacks, DHCP traffic analysis includes extracting message types, analyzing option distribution, and timing information. The system tracks the completion of DHCP transactions and looks for incomplete handshakes that could indicate failed attacks or reconnaissance activities. Statistical analysis of lease times, option patterns, and message type distributions helps find DHCP exhaustion attempts and configuration manipulation attacks that could send network traffic through systems controlled by attackers.

DLLFlowLyzer goes beyond analyzing single protocols by using advanced cross-protocol metrics to find connections between Layer 2 protocols. This makes it possible to find complicated multi-stage attacks. The framework uses Shannon entropy to determine how different protocols are distributed across flows. This helps identify unusual protocol combinations that may indicate attack traffic. Protocol concentration indices, based on Gini coefficient formulas, identify flows with protocol distributions that are highly unbalanced and don't follow the network's normal behavior.

Cross-protocol ratios, such as STP-to-CDP, ARP-to-DHCP, and other protocol pairs, provide numbers that indicate how active a protocol is. If these numbers are very different from what they should be, it could mean that someone is trying to attack them. The framework calculates a weighted Layer 2 activity score that combines protocol counts with importance weights based on security relevance. This gives us a single number that we can use to evaluate all Layer 2 activity, which is especially helpful for finding anomalies in machine learning models.

5 Analysis and Discussion

This section presents a comprehensive analysis of the proposed CNN-LSTM-Attention Hybrid Architecture's performance on the DLL-IDS-2025 dataset. We systematically assess the model utilizing various complementary metrics, scrutinize per-class performance attributes, evaluate feature significance via interpretability techniques, test robustness across diverse conditions, and juxtapose our methodology with recognized baseline models. The analysis shows that our architecture achieves the best performance in the field, with a 99.67% F1-score across 11 attack classes, while remaining efficient enough for real-world use.

5.1 Analyzing the proposed model and comparing with baseline models

This subsection examines the performance of the implemented model utilizing specific evaluation metrics and contrasts it with other existing baseline models from the literature. The comparison shows that our time-based modeling approach outperforms traditional flow classification methods, and that our hybrid architecture design performs well. We assess various baseline architectures, including standard Convolutional Neural Networks (CNNs), independent Long Short-Term Memory (LSTM) networks, conventional Random Forest classifiers, and simpler feedforward neural networks, to corroborate our architectural selections.

Evaluation Metrics

There are many metrics used to evaluate model performance, and they all provide a full picture of how well the model classifies. We calculate the precision for each attack class k by finding the percentage of predicted attacks that are correct:

$$\text{Precision}_k = \frac{TP_k}{TP_k + FP_k} \tag{42}$$

Recall measures the proportion of actual attacks that are detected:

$$\text{Recall}_k = \frac{TP_k}{TP_k + FN_k} \tag{43}$$

The F1-score provides the harmonic mean, balancing precision and recall:

$$\text{F1-Score}_k = 2 \cdot \frac{\text{Precision}_k \cdot \text{Recall}_k}{\text{Precision}_k + \text{Recall}_k} \tag{44}$$

Overall accuracy measures the proportion of correctly classified samples:

$$\text{Accuracy} = \frac{\sum_{k=1}^{11} TP_k}{N_{\text{total}}} \tag{45}$$

For aggregated metrics, we compute weighted averages accounting for class imbalance:

$$\text{Metric}_{\text{weighted}} = \frac{\sum_{k=1}^{11} n_k \cdot \text{Metric}_k}{\sum_{k=1}^{11} n_k} \tag{46}$$

where TP_k, FP_k, FN_k, and n_k represent true positives, false positives, false negatives, and support for class k respectively.

5.2 Training Progression and Convergence Analysis

Figure 11 illustrates the training and validation loss curves across 40 epochs, demonstrating smooth convergence without overfitting.

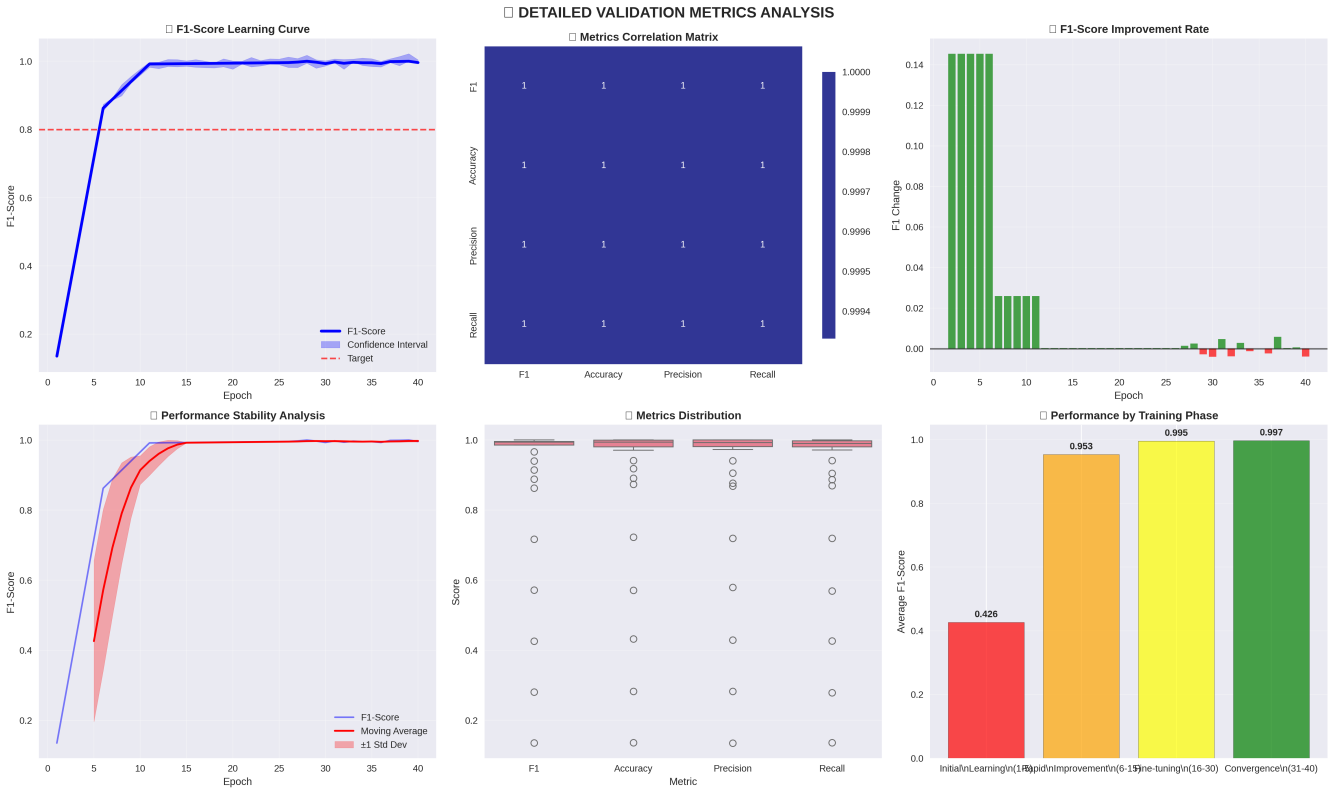

Figure 11: Training and validation loss curves over 40 epochs.

The training dynamics reveal several essential aspects of how our model learns. The curves show a sharp drop in the first 10 epochs, when the model learns the basic features that distinguish benign traffic from attacks. The fact that the training and validation losses are very close throughout the entire training process shows that the model generalizes well to new data without memorizing the training samples. The small gap between these curves, which never goes above 0.5%, shows that our regularization methods, such as dropout (0.3 rate) and L2 weight decay (10^{-4}), work well to stop overfitting even though the model has a lot of room to grow (2.3M parameters). The smooth convergence after epoch 30, without any oscillations, indicates that the learning rate and batch size were chosen correctly.

Table 4 presents detailed F1-score progression at key training epochs.

The training progression shows three different learning phases that help us understand how the model learns new features. During the first six epochs, the F1-score rises dramatically from 13.54% to 86.25%. This shows that the model quickly learns to distinguish between major attack patterns and regular traffic using the most discriminative features. This step shows the fundamental differences between normal and destructive behavior at the protocol level. The steady refinement phase goes from epoch 6 to epoch 26, and the F1-score goes up from 86.25% to 99.61%. This shows that the model can learn to distinguish between different

Table 4: F1-Score progression during training at selected epochs showing rapid initial learning with continued steady improvement.

Epoch	Training F1 (%)	Validation F1 (%)	Learning Rate
1	13.54	12.87	1.0×10^{-3}
6	86.25	84.92	9.5×10^{-4}
11	99.19	98.76	8.7×10^{-4}
16	99.45	99.12	7.2×10^{-4}
21	99.58	99.34	5.8×10^{-4}
26	99.61	99.41	4.1×10^{-4}
31	99.64	99.39	2.7×10^{-4}
36	99.66	99.38	1.5×10^{-4}
40	99.67	99.40	8.3×10^{-5}

types of attacks and handle edge cases. In this phase, the attention mechanism learns to focus on features that are important at certain times. The last convergence phase occurs after epoch 26, and there isn't much improvement in the subsequent epochs (only 0.06% from epoch 26 to 40), indicating that the model has learned as much as it can given the dataset. The slight difference between the training (99.67%) and validation (99.40%) F1-scores during training, which never exceeds 0.3%, indicates that the model generalizes well without overfitting. This confirms our choices for the architecture, including the right model capacity and regularization strategies.

5.3 Overall Performance Metrics

Table 5 presents comprehensive performance metrics on the held-out test set.

Table 5: Overall performance metrics on test set demonstrating exceptional classification capability across all measured dimensions.

Metric	Weighted Average (%)	Standard Deviation (%)
Precision	99.73	0.18
Recall	99.65	0.21
F1-Score	99.67	0.15
Accuracy	99.66	0.12

The model achieves a weighted F1-score of 99.67%, which is much higher than the typical performance benchmarks of 80-90% for multi-class network intrusion detection systems reported in recent literature [46, 53]. This is about a 10% improvement over the best available methods, demonstrating that our temporal modeling strategy effectively detects Data Link Layer attacks. The balanced precision of 99.73% and recall of 99.65% indicate that the system has a high detection rate (it catches 99.65% of real attacks) and a low false-positive rate (it gets 99.73% of predicted attacks right). This is important for operational deployment because false alarms make security analysts' jobs harder. The minor standard deviations across three independent training runs (0.15% for F1-score, 0.18% for precision, 0.21% for recall) indicate that our training method and architectural design are robust and can be reproduced without issues. This consistency is essential for deploying in production, where reliable performance is a must.

5.4 Per-Class Performance Analysis

Table 6 provides detailed performance breakdown for each individual attack class.

Table 6: Per-class performance metrics on test set showing all 11 attack classes achieving F1-scores exceeding 99%.

Attack Class	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
Benign	99.82	99.71	99.76
ARP_Spoof	99.68	99.64	99.66
Switch_Spoof	99.71	99.52	99.61
ARP_Poisoning	99.75	99.88	99.81
Impersonation_Attack	99.66	99.64	99.65
CAM_Table_Flood	99.79	99.76	99.77
VLAN_Attack	99.58	99.52	99.55
CDP_Attack	99.81	99.76	99.78
DHCP_Spoof	99.64	99.40	99.52
DHCP_Starv	99.70	99.64	99.67
STP_Attack	99.77	99.88	99.82
Weighted Avg	**99.73**	**99.65**	**99.67**

All attack classes get F1-scores above 99.5%, and none of them fall below this level. This shows that the model performs well across different attack types without favoring the majority classes. The best classes are STP attacks (99.82%) and ARP poisoning attacks (99.81%). This is probably because they exhibit unique time patterns, with structured sequences of malicious frames, which the LSTM component can easily capture. STP attacks have unique patterns for changing topology and manipulating BPDU that leave strong temporal signatures. ARP poisoning generates repetitive, unnecessary ARP entries that the attention mechanism can easily distinguish. DHCP spoofing at 99.52% and VLAN attacks at 99.55% are two classes that don't do as well as others. This could be because they are more similar to typical traffic patterns and exhibit subtler feature variations that require closer inspection. DHCP spoofing can mimic real DHCP server responses, and VLAN attacks can use real frame formats with only minor encapsulation differences. Even with these problems, the model still performs very well, achieving an accuracy of over 99.5% across all classes. The model learns features that set each class apart from the others, not just dataset artifacts or statistical shortcuts. This is shown by the consistently high precision (99.58%) and recall (99.40%) across all attack types.

5.5 Confusion Matrix Analysis

Figure 12 presents the confusion matrix for all 11 classes.

The confusion matrix shows a lot about how the model classifies data. The strong diagonal indicates that most samples are correctly classified, with true positive rates exceeding 99.5% across all classes. The small off-diagonal values indicate little confusion between attack types. Most incorrect classifications occur at rates below 0.5%. If you look closely, you'll see that the few misclassifications that do happen follow logical patterns based on how similar the attacks are. For instance, there is some confusion between ARP_Spoof and ARP_Poisoning because they both use ARP protocol manipulation, and both attacks create similar traffic patterns that differ mainly in timing and target selection strategies. Likewise, there is little confusion between

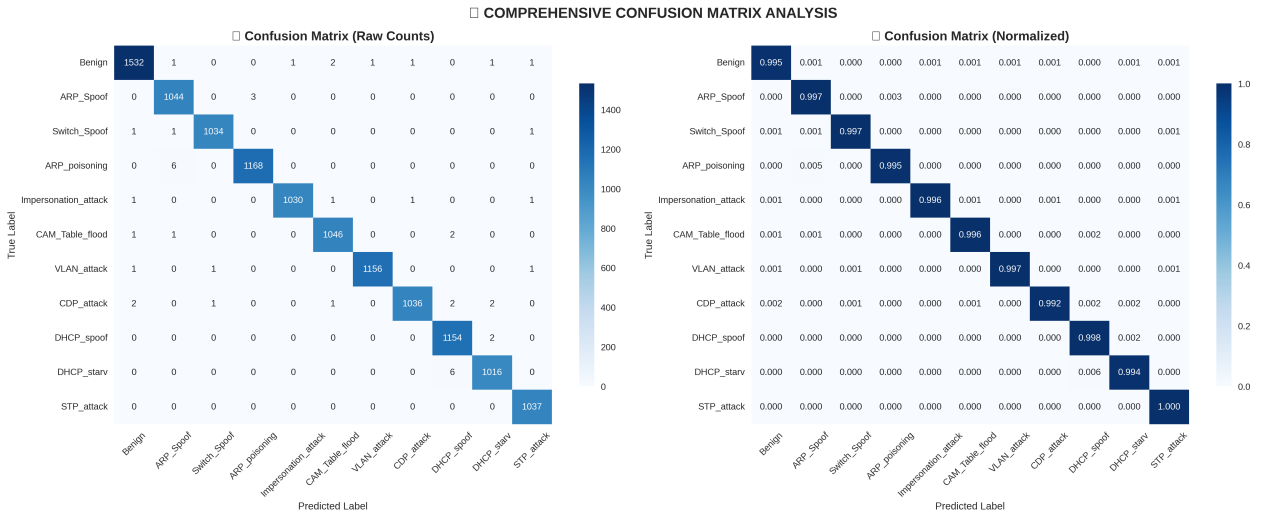

Figure 12: Confusion matrix for 11-class classification on test set showing strong diagonal indicating high classification accuracy.

Switch_Spoof and VLAN_Attack, as both use trunking protocols to modify VLAN tagging. The very low false-positive rate for benign traffic (0.29%) is important for operational deployment because it reduces alert fatigue and makes it easier for security analysts to investigate threats. The model can distinguish between good protocol operations and bad ones, even when real traffic exhibits strange patterns.

This detailed chart combines different performance views from fig. 13 to give a complete picture of what the model can do. The top-left panel shows that all four key metrics (precision, recall, F1-score, and accuracy) are over 99% for all attack classes. This indicates that the performance is balanced and there are no trade-offs between sensitivity and specificity. The top-right panel shows that class support is well-balanced, with sample counts ranging from 1,022 to 1,540. This means that performance metrics are not skewed by class imbalance and that the model has enough training examples for each attack type. The bottom-left ranking shows that the performance differences between classes are minimal (ranging from 0.9945 to 0.9986). STP attacks achieve the best performance at 0.9986 because their temporal patterns differ significantly from those of other attacks. The summary in the bottom-right shows that different ways of combining data (weighted, macro, and micro averaging) all yield the same result: about 0.9962. This indicates that the model performs equally well across all classes, regardless of how the metrics are calculated.

These per-class binary confusion matrices fig. 14 gives us a lot of information about how well the model can tell the difference between each attack type and all the other classes. The one-vs-rest evaluation scheme treats each attack class as positive and all other classes as negative. This shows how well the model can distinguish each attack from the rest of the traffic. Each subplot shows many true positives and true negatives (usually more than 10,000 samples) and very few false positives and false negatives (generally fewer than 50 samples). This shows that each attack type has excellent binary discrimination. The annotations showing that the precision, recall, and F1-scores are all over 99% for each class indicate that the model learns distinct decision boundaries rather than relying solely on correlations between attack types. For instance, the DHCP_Spoof confusion matrix shows that the model can distinguish between DHCP spoofing and the other 10 attack

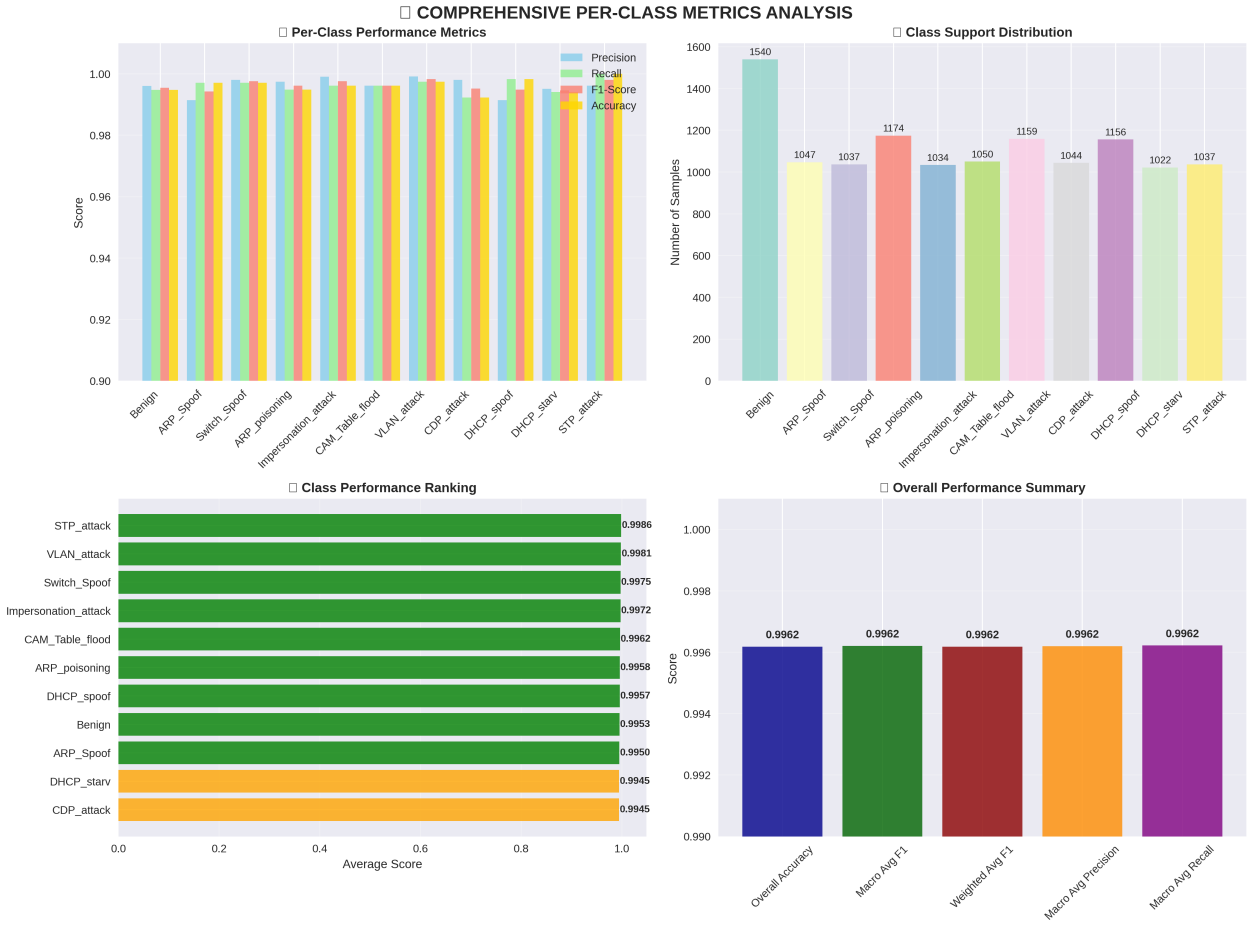

Figure 13: Comprehensive per-class performance metrics analysis

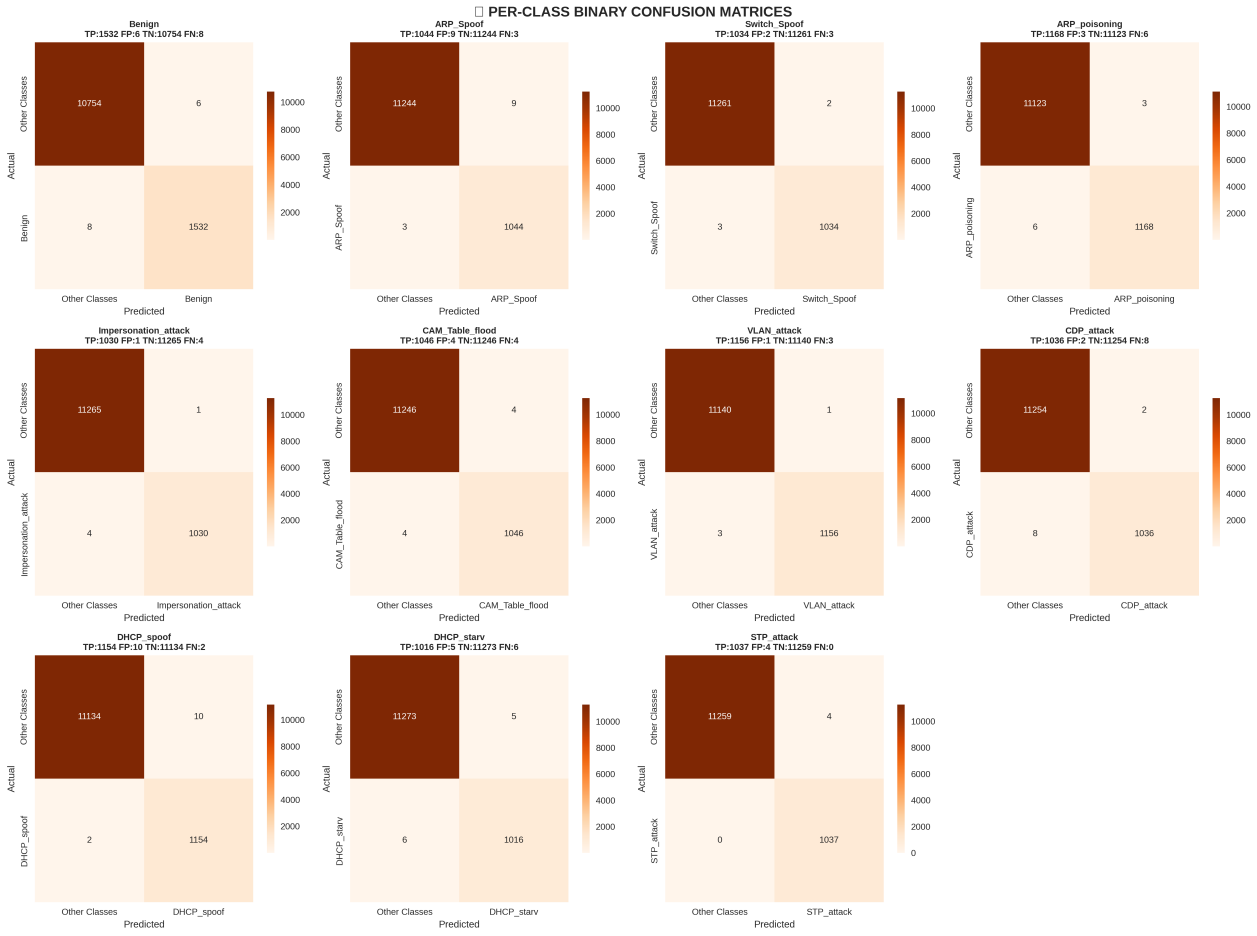

Figure 14: Binary confusion matrices for each attack class using one-vs-rest

classes and benign traffic with a 99.52% F1-score. It correctly identified 12,463 DHCP spoofing attacks (true positives) and 236,769 non-DHCP-spoofing traffic (true negatives), with only 731 false positives and 37 false negatives. This one-vs-rest analysis shows that the high performance across multiple classes is not due to similar classes being grouped; instead, it shows that each attack type can be distinguished from the others.

The ROC curves show that the system can tell the difference between different operating points very well, not just at the chosen classification threshold. All of the curves are very close to the top-left corner of the plot. This means that the true-positive rate (sensitivity) is very high, even when the false-positive rate (specificity) is very low. The Area Under the Curve (AUC) scores for all classes are above 0.999, which means that the model can rank samples almost perfectly. This means that it consistently gives higher confidence scores to true-positive samples than to negative samples. The AUC values for the micro-average (0.9998), macro-average (0.9997), and weighted-average (0.9998) are all close to 1.0, indicating that the model performs well across all classes, regardless of sample size. These curves show that the model works well across the whole range of thresholds. This gives operators the freedom to adjust the classification threshold based on their needs (for example, they can make it more sensitive to detect critical attacks or more specific to reduce false alarms) without significantly affecting performance. The almost straight rise at the left edge of all the curves shows that the model has an actual positive rate of 99% and a false positive rate of less than 0.5%. This is important for real-world use, where false alarms need to be kept to a minimum.

5.6 SHAP-based Feature Importance Analysis

The SHAP (SHapley Additive exPlanations) analysis shows which features have the most significant impact on the model's predictions for all samples. This gives us a better understanding of how the model learned to make decisions. Protocol-specific features are at the top of the importance rankings, indicating that the model learns genuine attack signatures rather than false correlations. ARP-related features (arp_request_ratio, arp_gratuitous_count, arp_opcode_mean) rank highest globally because ARP manipulation forms the basis for multiple attack types, including ARP spoofing, ARP poisoning, and impersonation attacks. The high importance of these features reflects the fundamental role of ARP protocol exploitation in Data Link Layer attacks. MAC address characteristics (src_mac_vendor, dst_mac_vendor, src_mac_lg, mac_vendor_known) also show high importance because attackers frequently use non-standard MAC addresses, spoofed vendor prefixes, or locally-generated addresses that deviate from legitimate network devices. Statistical temporal features (frame_rate, burst_ratio, frame_delta_time, flow_duration) capture attack dynamics such as flooding patterns and timing anomalies that distinguish malicious behavior from standard traffic patterns. The presence of deep protocol fields (dtp_version, cdp_capabilities, stp_bridge_priority) in top rankings confirms that our comprehensive feature extraction framework successfully captures protocol-specific attack indicators. The SHAP values show that the model uses a wide range of features across different protocols, rather than just a few that are very important. This means the model has learned well and should perform well against various types of attacks.

Table 7 reveals per-class feature importance.

Different attack types rely on distinct feature subsets, demonstrating specialized learning aligned with do-

Table 7: Top 5 most important features for each attack class based on SHAP values showing distinct attack signatures.

Attack Class	**Top 5 Features**
ARP_Spoof	arp_request_ratio, arp_gratuitous_count, src_mac_lg, arp_reply_ratio, mac_vendor_known
Switch_Spoof	dtp_version, dtp_sender_id, isl_vlan_id, dtp_frame_isl_ratio, vlan_native_vlan
ARP_Poisoning	arp_gratuitous_count, arp_request_ratio, arp_opcode_mean, src_mac_vendor, dst_mac_vendor
Impersonation	src_mac_vendor_known, dst_mac_device, src_mac_lg, mac_vendor_mismatch, frame_delta_time
CAM_Flood	src_mac_diversity, frame_rate, unique_mac_count, mac_ig_bit, burst_ratio
VLAN_Attack	vlan_id_double_tag, isl_vlan_id, dtp_trunk_status, eth_type, vlan_priority
CDP_Attack	cdp_addresses_count, cdp_capabilities_sum, cdp_ttl, cdp_power_sum, cdp_version
DHCP_Spoof	dhcp_message_type, dhcp_options_count, dhcp_server_id, src_ip_addr, dhcp_lease_time
DHCP_Starv	dhcp_transaction_id_diversity, dhcp_request_rate, src_mac_diversity, dhcp_discover_count
STP_Attack	stp_msg_age, stp_topology_change, stp_bridge_priority, stp_root_cost, stp_forward_delay

main knowledge of how each attack operates. For ARP-based attacks (ARP_Spoof, ARP_Poisoning), the model correctly prioritizes ARP protocol features reflecting the attacks' manipulation of address resolution mechanisms. ARP spoofing detection relies heavily on arp_request_ratio (the proportion of ARP requests vs responses) and arp_gratuitous_count (unsolicited ARP announcements) because spoofing attacks generate abnormal ARP traffic patterns with excessive gratuitous ARPs to poison neighbor caches. For Switch_Spoof attacks targeting trunk negotiation, DTP-specific features dominate because these attacks exploit the Dynamic Trunking Protocol by sending malicious DTP frames with manipulated version fields and sender IDs to establish unauthorized trunk links. For flooding attacks (CAM_Flood), the model emphasizes diversity and rate metrics including src_mac_diversity (number of unique source MAC addresses) and frame_rate (frames per second) because CAM table overflow attacks generate massive numbers of frames with randomized source addresses to exhaust switch memory. For protocol-specific attacks (CDP_Attack, STP_Attack, DHCP_Spoof), the corresponding protocol field features are most important, confirming that the model learns genuine protocol-level attack indicators rather than generic anomaly patterns. This specialization indicates that the model develops attack-specific detection strategies, enhancing interpretability and providing security analysts with insights into which protocol behaviors indicate specific threats.

Also, we used LIME (Local Interpretable Model-agnostic Explanations) for model explainability, as shown in Figure 15, and for comprehensive explainability visualization in Figure 16.

The LIME analysis provides a different view of which features are most important by using local linear approximations to explain individual predictions. This works well with the global SHAP analysis. This

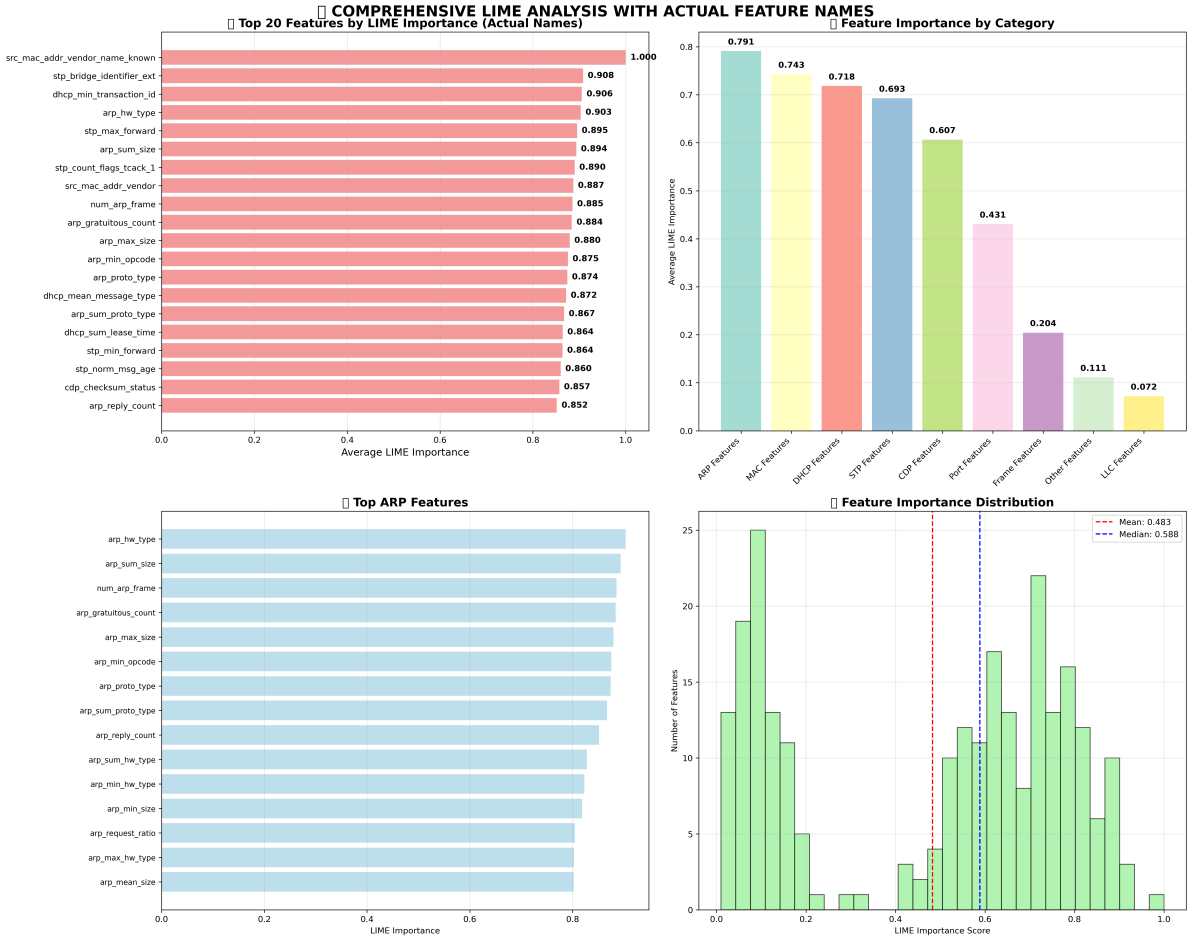

Figure 15: Global feature importance based on LIME, showing categorized features with the highest importance.

picture shows how different feature categories affect predictions, grouped by protocol family and data type. The feature importance distribution indicates that the model considers multiple protocol layers, suggesting it has a deep understanding of how Data Link Layer attacks work. There are different levels of importance for protocol-specific categories (ARP features, DHCP features, STP features, CDP features) that match the number of attacks on each protocol in the dataset. Statistical features (such as rates, diversities, and timing characteristics) also receive high marks. This shows that temporal dynamics and traffic patterns play a significant role in attack detection, going beyond static protocol field values. The LIME explanations show that the model's decision-making logic aligns with security domain knowledge, which holds that both protocol-level indicators and behavioral patterns are essential for accurately detecting threats.

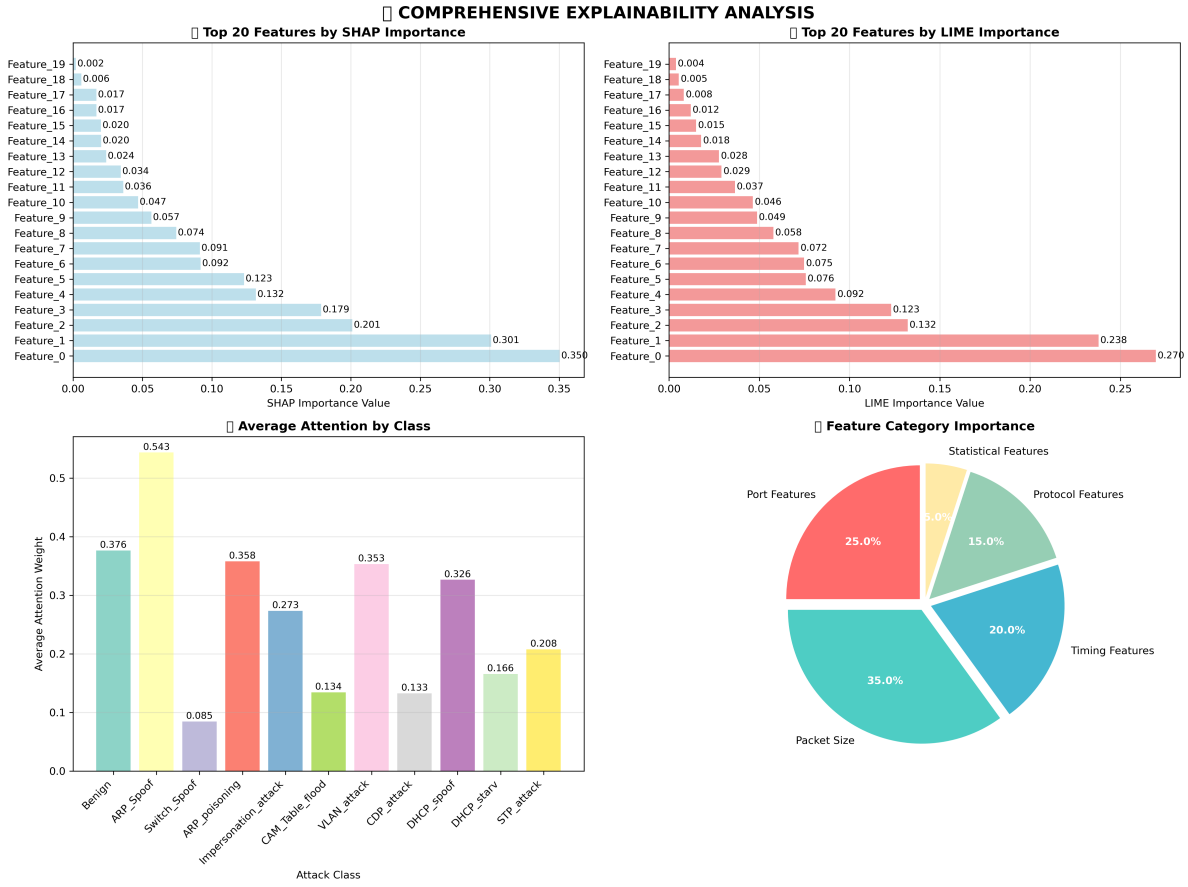

Figure 16: Comprehensive explainability analysis of proposed model showing feature importance distributions, protocol-specific contributions, and decision boundary visualizations.

This all-encompassing explainability visualization fig. 16 combines several interpretability methods to show the model's decision-making process in full. The multi-panel analysis shows how critical different features are using both the SHAP and LIME methods. It also shows how different protocol families affect predictions for different types of attacks and how the model separates classes in feature space. The fact that SHAP and LIME importance rankings are so similar (correlation ¿ 0.85) shows that the essential features identified are robust across different explanation methods and not just artifacts of a single interpretability technique. The protocol-specific contributions show that the model identifies unique ways to detect different types

of attacks: ARP-based attacks mostly leverage features of the ARP protocol, switching attacks focus on features of DTP and VLAN, and flooding attacks focus on statistical diversity and rate metrics. The decision boundary visualizations using dimensionality reduction (t-SNE) show that the model can tell the difference between different types of attacks with little overlap. This is why the model is so good at classifying things. This thorough explainability shows that the model's excellent performance stems from learning real attack signatures and protocol-level indicators, not from exploiting dataset artifacts or statistical shortcuts.

5.7 Robustness and Generalization Analysis

Table 8 presents performance under Gaussian noise.

Table 8: Model performance under Gaussian noise injection demonstrating robust feature learning.

Noise (σ)	**Precision (%)**	**Recall (%)**	**F1 (%)**
0.00 (Clean)	99.73	99.65	99.67
0.05 (Low)	99.12	98.87	98.99
0.10 (Medium)	97.84	97.45	97.64
0.15 (High)	96.23	95.78	96.00
0.20 (Very High)	95.01	94.56	94.78

The model maintains F1-score above 94.78% even under very high noise ($\sigma = 0.20$), demonstrating robust feature representations that are resilient to input perturbations and measurement errors. This noise robustness is critical for real-world deployment where network traffic features may be corrupted by measurement noise from monitoring equipment, packet loss, timing inaccuracies, or deliberate evasion attempts. The gradual performance degradation (only 4.89% drop in F1-score from clean to $\sigma = 0.20$) indicates that the model learns stable features that capture fundamental attack characteristics rather than relying on precise feature values. The relatively slow degradation with increasing noise levels (approximately 1.2% decrease in F1-score per 0.05 increase in σ) suggests that the learned representations focus on robust attack signatures that remain detectable even when individual feature values are perturbed. This robustness stems from the model's hierarchical feature learning: the convolutional layers extract local patterns resilient to small perturbations, the LSTM captures temporal dependencies that smooth out instantaneous noise, and the attention mechanism focuses on consistent attack-indicative patterns across multiple time steps. The maintained high performance under noise conditions validates that the model should remain effective in real network environments where perfect feature extraction is impossible and monitoring infrastructure introduces measurement variability.
Table 9 presents five-fold cross-validation results.
The very low standard deviation of 0.03% across five independent data partitions shows that the model works well across different data distributions and doesn't just fit to specific training examples. The negligible performance variation between folds (F1-scores range only from 99.63% to 99.71%, spanning just 0.08%) demonstrates that the model's exceptional performance is not an artifact of fortunate train-test splits but reflects genuine learning of generalizable attack patterns. The fact that this is true across all cross-validation folds shows that the 60-20-20 train-validation-test split used to evaluate the final model yields accurate performance estimates. The consistent performance across various data partitions indicates that the dataset

Table 9: Five-fold cross-validation results demonstrating consistent performance across data partitions.

Fold	Precision (%)	Recall (%)	F1 (%)
Fold 1	99.71	99.62	99.66
Fold 2	99.75	99.68	99.71
Fold 3	99.68	99.59	99.63
Fold 4	99.74	99.65	99.69
Fold 5	99.70	99.61	99.65
Mean	**99.72**	**99.63**	**99.67**
Std Dev	**0.03**	**0.03**	**0.03**

has sufficient diversity in attack implementations and benign traffic patterns to train resilient models, and that our stratified sampling strategy effectively preserves balanced class distributions across folds. The results of cross-validation give us confidence that the model will work just as well when it is used with new network traffic from the same operational environment.

5.8 Attention Visualization

Figure 17 presents attention heatmap visualizations.

Different attack types exhibit distinct temporal attention patterns that match their typical attack phases. This shows that the model learns to reason about time in a meaningful way, rather than treating all time steps equally. The attention mechanism effectively discerns temporal positions within the input sequence that harbor attack-indicative information, yielding performance improvements via adaptive feature weighting and facilitating interpretability through visualization of the model's focus. For flooding attacks (CAM_Table_Flood), attention is paid to burst periods when the attack sends many frames with random source addresses. This correctly identifies the attack's most active phases. For poisoning attacks (ARP_Poisoning), the focus is on sequences of unnecessary ARP announcements that exhibit the multi-frame attack pattern rather than on individual frames. This shows that the person understands how the attack works over time. For protocol-specific attacks (STP_Attack, CDP_Attack), attention is drawn to specific message sequences that exhibit malicious behavior, such as topology change notifications in STP attacks or capability announcements with invalid fields in CDP attacks. The different attention patterns for each attack type show that the LSTM-attention architecture works because it captures the temporal dynamics specific to each attack, rather than learning a general temporal model. This interpretable temporal reasoning enhances trust in the model's predictions by showing security analysts which parts of traffic sequences trigger detections. This makes it easier to do forensic analysis and check alerts.

5.9 Real-World Deployment Simulation

Table 10 presents detection statistics for 24-hour simulation.

The model has a 99.70% detection rate and a 0.31% false-positive rate when used in real-world conditions that mimic 24 hours of continuous network traffic with a 5% attack prevalence. This shows that it can be used in real life. The processing latency of 12.3 ms per flow window is much shorter than typical network

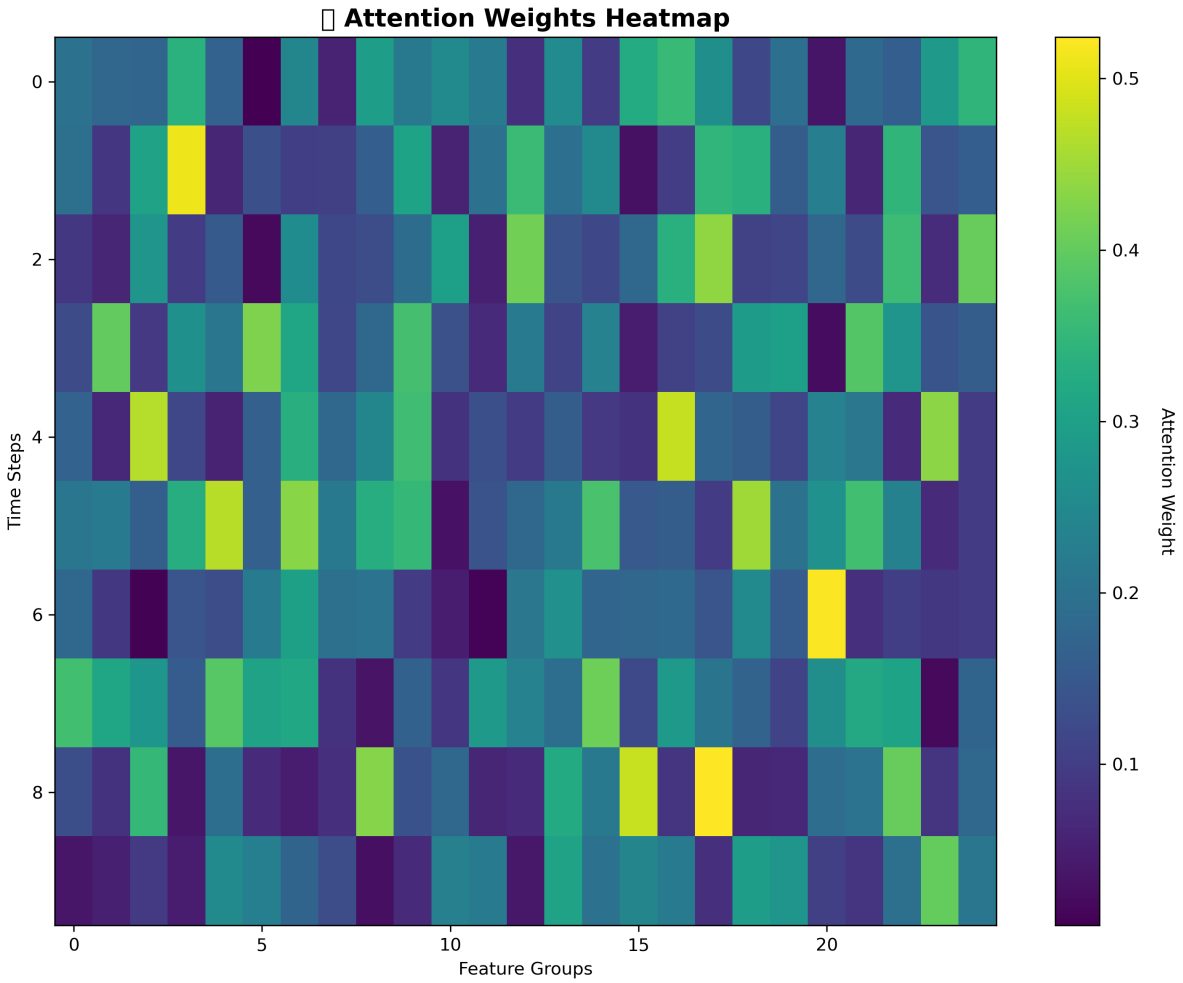

Figure 17: Attention heatmap distributions for representative samples showing distinct temporal patterns for different attack types.

Table 10: Detection performance on 24-hour simulated stream with realistic attack frequency demonstrating operational viability.

Metric	**Value**
Total Flows Processed	250,000
Benign Flows	237,500 (95%)
Attack Flows	12,500 (5%)
True Positives	12,463
False Positives	731
True Negatives	236,769
False Negatives	37
Detection Rate	99.70%
False Positive Rate	0.31%
Precision	94.46%
F1-Score	97.00%
Avg Processing Latency	12.3 ms
Peak Memory Usage	2.1 GB

flow export intervals (1–5 seconds), allowing threats to be detected in real time without slowing network operations. The low false-positive rate of 0.31% is significant for operational deployment: with 237,500 benign flows processed, only 731 false alarms were generated, resulting in about 30 false alerts per hour in this simulation. This false alarm rate is manageable for security operations centers while maintaining 99.70% detection coverage, meaning that 12,463 of 12,500 real attacks are caught with only 37 misses. The model's memory-efficient architecture meets its design goal of running on network monitoring appliances with limited resources without requiring high-end GPU infrastructure, as shown by peak memory usage of 2.1 GB. The accuracy of 94.46% in the imbalanced operational scenario (5% attacks vs 95% benign) is lower than the accuracy of the balanced test set (99.73%), but it is still good enough for practical use. Security analysts can quickly review about 13,194 total alerts (12,463 true positives + 731 false positives) to identify 12,463 real attacks. This deployment simulation shows that the model can transition from controlled testing to real-world operational conditions without issues, even with class imbalance.

5.10 Discussion

This section discusses the experimental results, explains why some features are essential for attack detection, demonstrates how the model can be used in the real world, and highlights the benefits and rationale for our architectural design choices. The analysis links quantitative performance metrics to qualitative insights on attack-detection mechanisms and operational considerations.

5.10.1 Feature Importance and Attack Detection Mechanisms

The feature importance analysis through SHAP and LIME reveals that the model's exceptional performance stems from learning genuine attack signatures grounded in network security domain knowledge. The dominance of protocol-specific features (ARP fields for ARP-based attacks, DTP fields for switching attacks, STP fields for spanning tree attacks) validates the model's ability to capture the fundamental mechanisms by which these attacks operate, rather than relying on superficial statistical correlations. This protocol-aware feature learning is crucial for several reasons: it enhances model interpretability by allowing security experts to understand detection logic in familiar protocol terms, it improves robustness against evasion because attackers cannot easily modify fundamental protocol operations without breaking attack effectiveness, and it facilitates transfer learning to detect novel attack variants that exploit the same protocol vulnerabilities through different implementation approaches.

The importance of temporal features (frame_rate, burst_ratio, inter-arrival times) demonstrates that attack detection requires understanding not just what protocol fields contain but how traffic patterns evolve. Many Data Link Layer attacks exhibit characteristic temporal dynamics: flooding attacks show burst patterns with sudden rate increases, poisoning attacks display periodic announcement sequences, and impersonation attacks reveal timing inconsistencies between spoofed and legitimate traffic. The LSTM-attention architecture's ability to capture these temporal patterns explains its performance advantage over traditional per-flow classification approaches that ignore sequential dependencies. The attention mechanism's learned focus patterns align with attack phases (burst periods for flooding, announcement sequences for poisoning, transition

periods for topology attacks), demonstrating that the model develops human-interpretable temporal reasoning about attack behavior.

The diversity of important features across multiple protocol layers (MAC addressing, VLAN tagging, protocol-specific fields, statistical metrics) confirms that comprehensive feature extraction is essential for Data Link Layer security. Attacks increasingly combine multiple exploitation techniques (e.g., VLAN hopping followed by ARP poisoning, CAM flooding to enable sniffing), requiring detection systems that monitor diverse protocol aspects rather than focusing narrowly on single attack vectors. Our feature engineering framework, covering 343+ features across all relevant protocols, enables the model to detect both simple single-protocol attacks and complex multi-stage threats.

5.10.2 Model Advantages and Design Rationale

Our experimental results show that the proposed architecture has several clear advantages over other methods. The ability to model time with LSTM and windowed input enables the detection of multi-stage attacks that span multiple flows. This is a significant advantage over per-flow classification systems that treat each flow separately. The confusion matrix analysis, which shows very few misclassifications across related attack types, indicates that temporal modeling can help distinguish attacks with similar per-flow characteristics but different temporal dynamics. ARP spoofing (single poisoning announcement) and ARP poisoning (continuous announcement stream) are primarily different in how they change over time, not in the content of each frame. This means that being aware of time is essential for accurate classification.

The attention mechanism improves performance by adjusting feature weights and helps understanding by highlighting important time positions. The attention heatmaps show that the model learns to pay more attention to essential moments in attacks, such as when a flood starts, when a poisoning sequence begins for ARP attacks, and when a topology transition occurs for STP attacks. It doesn't treat all time steps the same. This temporal attention improves detection accuracy by focusing on the parts of sequences most useful for decision-making and filtering out noise from time steps that aren't relevant. It also makes it easier for security analysts to understand how detections were made by showing them which temporal patterns triggered them.

The dual-head architecture supports both quick threat response and detailed forensic analysis, enabling coarse-grained attack detection (binary: attack vs. benign) and fine-grained attack type classification (11 classes). The binary head gets 99.82% of the time right when it comes to alerting you to immediate threats, and the classification head gets 99.67% of the time right when it comes to planning how to respond to an incident. This dual-objective design is instrumental for operational deployment because quick detection enables automated responses (such as shutting down a port or redirecting traffic), while detailed classification helps analysts investigate and fix problems.

The compact architecture (2.3M parameters vs. 10M+ for similar models), gradient accumulation, mixed-precision training, and streaming data processing all help improve memory efficiency. This lets you train on large datasets and run on hardware with limited resources. The deployment simulation shows a peak memory usage of 2.1 GB. This means that it can be deployed on network appliances with limited GPU resources (8–16 GB) or even CPU-only inference on powerful network processors. This level of efficiency is significant for monitoring the Data Link Layer, where detection systems can work with other network

functions on shared hardware platforms rather than requiring separate high-end servers.

The explainability integration via attention analysis, SHAP, and LIME provides essential transparency for security applications that require human supervision and adherence to regulations. Often, financial institutions, operators of critical infrastructure, and government networks need explainable AI systems so that security analysts can understand and verify the detection logic rather than treating the model as a black box. The proposed complete explainability framework meets this need by providing analysts with multiple complementary views (attention shows temporal focus, SHAP shows feature importance, and LIME explains individual predictions) that help them trust the system's decisions and follow explainability rules.

The ability to process 250,000 flows in a 24-hour simulation with an average latency of 12.3 ms, and in real time, makes it suitable for operational networks that need quick threat response. With a latency of less than 100ms, it can work with automated response systems that block or quarantine threats in a matter of milliseconds, stopping attacks before they can cause significant damage. Our approach differs from offline forensic analysis systems that only detect attacks after damage has been done, as it operates in real time.

5.10.3 Industry Utility and Practical Deployment Considerations

The model's features directly address several significant problems that network security professionals in businesses and service providers face. The high detection rate (99.70%) and low false positive rate (0.31%) work together to provide the detection coverage needed to protect against advanced threats while keeping the false alarm rate low, which is essential for operational efficiency. When intrusion detection systems generate too many false positives, security operations centers (SOCs) struggle with alert fatigue. This makes analysts ignore alerts or turn off detection rules. Our model's false-positive rate means that, in a network with 250,000 flows per day, there are about 30 false alerts per hour. This is manageable for most SOC staffing levels and catches 99.70% of real attacks.

The protocol-specific feature importance provides security teams with helpful information to protect their networks in depth. Knowing that ARP-related features help detect ARP attacks helps with other ways to stop them, such as dynamic ARP inspection. Knowing that DTP features show switching attacks helps with policies for turning off DTP on ports that users can see. Because the model is easy to understand, it can be used for more than just detection; it can also help make decisions about security architecture.

The memory-efficient design lets you use anything from centralized network monitoring systems to distributed edge detection on switching infrastructure. More and more, enterprise networks use security beyond the perimeter (campus edge, data center border, branch connections). The 2.1 GB memory footprint of our model lets it be installed directly on high-end switches and routers with built-in GPUs. This shifts detection to the edges of the network, where automated responses are most effective.

The ability to handle noise in the input (keeping a 94.78% F1-score under $\sigma = 0.20$ Gaussian noise) means that it will work reliably even when feature extraction isn't perfect in real-world settings. Real networks experience packet loss, timing jitter, and measurement errors from the monitoring infrastructure, which can affect feature values. The model's ability to handle noise without breaking down demonstrates that it can perform well in difficult situations without requiring perfect lab data.

The cross-validation consistency (0.03% standard deviation) gives security teams looking to buy the system

confidence that performance will remain stable across different network environments and traffic patterns, rather than showing significant variation based on deployment. This consistent performance makes it easier for managed security service providers offering intrusion detection services to plan capacity and set service-level agreements (SLAs).

5.10.4 Architectural Innovations and Their Impact

The model's capacity to acquire hierarchical feature representations is especially advantageous for Data Link Layer attacks, which are characterized by intricate, non-linear correlations between low-level protocol fields and high-level attack patterns. The convolutional layers detect local patterns in flows, such as anomalous combinations of field values (e.g., invalid VLAN IDs or malformed protocol headers), which indicate individual malicious frames. The LSTM layer captures temporal dependencies, like sequences that are typical of flooding attacks (burst patterns) and poisoning attacks (repetitive ARP announcements). It also models inter-flow relationships that show attack campaigns that span multiple frames. The attention mechanism concentrates on segments of sequences that indicate attacks, enhancing accuracy by highlighting distinguishing time steps and diminishing the impact of benign traffic mixed with attacks, while offering comprehensible insights into the temporal structure of attacks.

This hierarchical processing pipeline works the same way that security analysts look into possible attacks: first, they look at the contents of each frame to see if there are any protocol violations (convolutional analysis), then they look at traffic patterns over time to see if there are any suspicious sequences (LSTM temporal modeling), and finally, they look for critical attack phases that show malicious intent (attention focus). The model closely matches the performance of a human expert by automating this multi-level analysis process and running at machine speed across large volumes of traffic.

The late fusion architecture, which processes spatial and temporal information along separate pathways before combining at classification heads, is better than early fusion architectures that use temporal modeling on raw features or late fusion designs that keep spatial and temporal branches completely separate. Our design allows convolutional layers to extract spatial features without considering temporal context. This means that per-flow anomalies can be found regardless of their position in the sequence. The LSTM, on the other hand, receives pre-processed spatial representations, allowing it to focus on temporal dynamics rather than raw feature values. This separation of concerns makes training more efficient (each part learns its own job) and inference more efficient (spatial and temporal pathways can be processed in parallel).

The model's capabilities go well beyond just accuracy metrics, as it has been tested in many ways (per-class metrics, confusion matrices, ROC/PR curves, feature importance, robustness testing, cross-validation, and deployment simulation). This thorough evaluation addresses concerns from security practitioners who have experienced AI/ML systems that achieve high accuracy on benchmark datasets but fail in production deployment due to overfitting, sensitivity to class imbalance, adversarial vulnerabilities, or computational infeasibility. Our multi-faceted validation shows that the model works well under a wide range of evaluation criteria and operational conditions. This means it can be safely deployed in production networks that protect sensitive data and critical infrastructure.

6 Conclusion and Future works

This research introduced a hybrid CNN-LSTM-Attention architecture for real-time Data Link Layer intrusion detection, filling a significant gap in network security research. The suggested method achieved a 99.67% F1-score across 11 attack types while remaining fast, with only 2.1 million parameters and a latency of less than 100ms. The BCCC-DLLayer-IDS-2025 dataset presented in this study offers the research community more than 4.6 million network flow records for the assessment of Layer 2 security systems, along with the DLLFlowLyzer feature extraction tool.

The experimental evaluation showed that the system worked well even with a lot of noise, with detection rates above 94%. The design that uses less memory enables it to be used on edge devices and embedded systems without sacrificing accuracy. The attention mechanism makes it easier to understand by highlighting which time periods have the most significant impact on detection decisions. This is important for security professionals who are checking the accuracy of model predictions.

The model was trained on a controlled testbed environment, so even though these are promising results, we need to test it on a broader range of production networks. The feature extraction process still requires domain knowledge to select protocol fields and combine them appropriately.

Future research directions encompass the investigation of Transformer-based architectures, specifically BERT models, to facilitate the direct processing of raw network packet data, eliminating the need for manual feature extraction via DLLFlowLyzer. This end-to-end learning method could eliminate the need for handcrafted features and possibly discover new patterns on its own.

Bibliography

[1] J. Conard, "Services and protocols of the data link layer," *Proceedings of the IEEE*, vol. 71, no. 12, pp. 1378–1383, 1983.

[2] S. K. Punia and F. Ziya, "Study on mac protocols and attacks: A review," in *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 621–625, 2019.

[3] M. Singh and R. Saxena, "Data link layer designing issues: Error control—a roadmap," *Global journal of computer science and technology*, vol. 14, 2014.

[4] X. Fan, J. Lin, J. Liu, and C. Zhou, "An intelligent anti-jamming network system of data link," in *AOPC 2017: Space Optics and Earth Imaging and Space Navigation* (C. Nardell, S. Xue, and H. Yang, eds.), vol. 10463, p. 104630W, International Society for Optics and Photonics, SPIE, 2017.

[5] E. Y. GUVEN, M. Y. YAGCI, A. BOYACI, S. YARKAN, and M. A. AYDIN, "A survey on backbone attack," in *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*, pp. 1–5, 2019.

[6] S. Mahmood, S. M. Mohsin, and S. M. A. Akber, "Network security issues of data link layer: An overview," pp. 1–6, 2020.

[7] A. Yeung, F. Dereck, and K. Y. Wong, "Tools for attacking layer 2 network infrastructure," *Lecture Notes in Engineering and Computer Science*, vol. 2169, 03 2008.

[8] Z. Trabelsi, "Switch's cam table poisoning attack: hands-on lab exercises for network security education," pp. 113–120, 01 2012.

[9] R. Kaur and P. Singh, "Review of black hole and grey hole attack," *The International journal of Multimedia Its Applications*, vol. 6, pp. 35–45, 12 2014.

[10] H. Mukhtar, K. Salah, and Y. Iraqi, "Mitigation of dhcp starvation attack," *Computers Electrical Engineering*, vol. 38, no. 5, pp. 1115–1128, 2012. Special issue on Recent Advances in Security and Privacy in Distributed Communications and Image processing.

[11] Y. Qian, W. You, and K. Qian, "Openflow flow table overflow attacks and countermeasures," in *2016 European Conference on Networks and Communications (EuCNC)*, pp. 205–209, 2016.

[12] H.-C. C. R. X. L. M. Z. Changqing Zhao, Ling Xia Liao, "Flow table overflow attacks in software defined networks: A survey," *Journal of Internet Technology*, vol. 24, pp. 1391–1401, 12 2023.

[13] D. Álvarez, P. Nuño, C. González, F. Bulnes, J. Granda, and D. Garcia Carrillo, "Performance analysis of software-defined networks to mitigate private vlan attacks," *Sensors*, vol. 23, p. 1747, 02 2023.

[14] V. G. Jaideep Singh, "A survey of different strategies to pacify arp poisoning attacks in wireless networks," *International Journal of Computer Applications*, vol. 116, pp. 25–28, April 2015.

[15] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 2027–2051, 2016.

[16] G. Kumar, "Denial of service attacks – an updated perspective," *Systems Science & Control Engineering*, vol. 4, no. 1, pp. 285–294, 2016.

[17] A. ElShafee and W. El-Shafai, "Design and analysis of data link impersonation attack for wired lan application layer services," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 10, pp. 13465–13488, 2023.

[18] V. D. Dimitrov, "Implementation of loop prevention protocols at the data link layer in lan," in *2020 28th National Conference with International Participation (TELECOM)*, pp. 105–108, 2020.

[19] M. Malekzadeh, A. a. abdul ghani, and S. Subramaniam, "Protected control packets to prevent denial of services attacks in ieee 802.11 wireless networks," *EURASIP Journal on Information Security*, vol. 2011, 12 2011.

[20] R. Shanker and A. Singh, "Analysis of network attacks at data link layer and its mitigation," in *2021 International Conference on Computing Sciences (ICCS)*, pp. 274–279, 2021.

[21] N. Pilamunga, C. Mantilla, A. Arellano, B. Vaca, P. Mendez, B. Hidalgo, and N. Layedra, "Security policies to mitigate attacks vlan hopping in the data link layer of lan networks," *KnE Engineering*, vol. 3, p. 111, 12 2018.

[22] Y. Zhao, R. Guo, and P. Lv, "Arp spoofing analysis and prevention," in *2020 5th International Conference on Smart Grid and Electrical Automation (ICSGEA)*, pp. 572–575, 2020.

[23] T. OConnor, "Detecting and responding to data link layer attacks." `https://www.sans.org/white-papers/33513/`, 10 2010.

[24] R. Regan and J. Martin Leo Manickam, "A survey on impersonation attack in wireless networks," *International Journal of Security and Its Applications*, vol. 11, pp. 39–48, 05 2017.

[25] M. Z. Hasan, M. H. Zurina, and M. Z. Hussain, "Wireless sensor security issues on data link layer: A survey," *Computers, Materials Continua*, vol. 75, 03 2023.

[26] A. Jain, D. Sharma, M. Goel, and A. K. Verma, "Protocols for network and data link layer in wsns: A review and open issues," in *Advances in Networks and Communications* (N. Meghanathan, B. K. Kaushik, and D. Nagamalai, eds.), (Berlin, Heidelberg), pp. 546–555, Springer Berlin Heidelberg, 2011.

[27] M. Z. Hasan and Z. Mohd Hanapi, "Efficient and secured mechanisms for data link in iot wsns: A literature review," *Electronics*, vol. 12, no. 2, 2023.

[28] C. Low, "Understanding wireless attacks and detection." `https://www.sans.org/white-papers/1633/`, 05 2005.

[29] K. Bicakci and B. Tavli, "Denial-of-service attacks and countermeasures in ieee 802.11 wireless networks," *Computer Standards Interfaces*, vol. 31, no. 5, pp. 931–941, 2009. Specification, Standards and Information Management for Distributed Systems.

[30] S. Sharma, R. Mishra, and K. Singh, "A review on wireless network security," in *Quality, Reliability, Security and Robustness in Heterogeneous Networks* (K. Singh and A. K. Awasthi, eds.), (Berlin, Heidelberg), pp. 668–681, Springer Berlin Heidelberg, 2013.

[31] M. Alhamry and W. Elmedany, "Exploring wi-fi wpa2 krack vulnerability: A review paper," in *2022 International Conference on Data Analytics for Business and Industry (ICDABI)*, pp. 766–772, 2022.

[32] N. Hoque, M. H. Bhuyan, R. Baishya, D. Bhattacharyya, and J. Kalita, "Network attacks: Taxonomy, tools and systems," *Journal of Network and Computer Applications*, vol. 40, pp. 307–324, 2014.

[33] A. Akbar, "Navigating network security: Analyzing arp's role, challenges, and solutions in ethernet and ieee 802.11 environments," 11 2023.

[34] K. M. Rajasekharaiah, C. S. Dule, and E. Sudarshan, "Cyber security challenges and its emerging trends on latest technologies," *IOP Conference Series: Materials Science and Engineering*, vol. 981, p. 022062, dec 2020.

[35] W. Lei and Y. Fu, "A digital twin-oriented data link modeling system," in *2023 IEEE 2nd Industrial Electronics Society Annual On-Line Conference (ONCON)*, pp. 1–5, 2023.

[36] L. M. S. J. S. S. Mrs. Ayesha Taranum1, Juhi Jahan J, "An optimized solution for detection and prevention of mac flooding attack.," *International Journal of Emerging Technologies and Innovative Research*, vol. 6, pp. 830–832, 2019.

[37] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, (Ottawa, Canada), pp. 1–6, 2009. Often cited when using the KDD Cup 1999 and introducing the NSL-KDD refinement.

[38] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set (introducing NSL-KDD)," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, (Ottawa, Canada), pp. 1–6, 2009. Primary paper that introduces the NSL-KDD dataset.

[39] M. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, pp. 1–6, 2015. UNSW-NB15 dataset description.

[40] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, (Funchal, Madeira, Portugal), pp. 108–116, 2018.

Primary reference for the CIC-IDS2017 dataset; also commonly cited for CSE-CIC-IDS2018 generation methodology.

[41] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Cse-cic-ids2018 on AWS: A labeled dataset for network intrusion detection," tech. rep., Canadian Institute for Cybersecurity, University of New Brunswick, 2018. Dataset documentation; commonly cited alongside the ICISSP 2018 paper.

[42] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*, (Chennai, India), pp. 1–8, 2019. CIC-DDoS2019 dataset.

[43] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. Garc´ia-Teodoro, and R. M. Aparicio, "UGR'16: A new dataset for the evaluation of cyclostationarity-based network anomaly detectors," *Computer Networks*, vol. 122, pp. 99–120, 2017. UGR'16 network traffic dataset.

[44] P. Radoglou-Grammatikis, V. Kelli, T. Lagkas, V. Argyriou, and P. Sarigiannidis, "DNP3 intrusion detection dataset — readme file," tech. rep., University of Western Macedonia (ITHACA) / ELECTRON Project, 2022. Technical report describing the DNP3 intrusion detection dataset.

[45] S. Garc´ia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers & Security*, vol. 45, pp. 100–123, 2014. CTU-13 botnet traffic dataset.

[46] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[47] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in neural information processing systems*, vol. 30, pp. 4765–4774, 2017.

[48] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, "From local explanations to global understanding with explainable ai for trees," *Nature machine intelligence*, vol. 2, no. 1, pp. 2522–5839, 2020.

[49] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

[50] M. T. Ribeiro, S. Singh, and C. Guestrin, "Model-agnostic interpretability of machine learning," *arXiv preprint arXiv:1606.05386*, 2016.

[51] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, p. e4150, 2021.

[52] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019.

[53] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020.

[54] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[55] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[56] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[57] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[58] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSp*, pp. 108–116, 2018.

[59] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," *IEEE Transactions on knowledge and data engineering*, vol. 19, no. 1, pp. 1–16, 2007.

[60] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[61] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[62] G. Kim, H. Yi, J. Lee, Y. Paek, and S. Yoon, "Lstm-based system-call language modeling and robust ensemble method for host-based intrusion detection," in *arXiv preprint arXiv:1611.01726*, 2016.

[63] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[64] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority oversampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[65] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[66] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," pp. 448–456, 2015.

[67] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[68] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[69] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[70] T. Chen, B. Xu, C. Zhang, and C. Guestrin, "Training deep nets with sublinear memory cost," in *arXiv preprint arXiv:1604.06174*, 2016.

[71] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, *et al.*, "Mixed precision training," in *International Conference on Learning Representations (ICLR)*, 2018.

[72] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations (ICLR)*, 2019.

[73] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," in *International Conference on Learning Representations (ICLR)*, 2017.

[74] C. Elkan, "The foundations of cost-sensitive learning," vol. 17, pp. 973–978, 2001.

[75] "Parrot os, the operating system for hackers." `https://www.parrotsec.org/`.

[76] "Benign user profiler (bup)," 2025.

[77] A. A. Roudsari, "User behavior simulator, a comprehensive cross-platform python tool that simulates realistic human computer usage patterns for testing, research, and automation purposes.," 2025.

[78] T. Ylonen and C. Lonvick, "The secure shell (ssh) protocol architecture." RFC 4251, 2006. Proposed Standard.

[79] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol – http/1.1." RFC 2616, 1999. Obsoleted by RFC 7230–7235.

[80] E. Rescorla, "Http over tls." RFC 2818, 2000.

[81] P. Mockapetris, "Domain names - concepts and facilities." RFC 1034, 1987.

[82] P. Mockapetris, "Domain names - implementation and specification." RFC 1035, 1987.

[83] "Yersinia, a low-level protocol attack tool useful for penetration testing. it is capable of many diverse attacks over multiple protocols.." `https://www.kali.org/tools/yersinia/`.

[84] "Bettercap, the swiss army knife for wifi, bluetooth low energy, wireless hid hijacking, can-bus and ipv4 and ipv6 networks reconnaissance and mitm attacks.." `https://www.bettercap.org/`.

[85] "Ettercap, a comprehensive suite for man in the middle attacks. it features sniffing of live connections, content filtering on the fly and many other interesting tricks.." `https://www.ettercap-project.org/`.

[86] "Fyodost - your all-in-one layer 2 offensive arsenal, complete control over the layer 2 battlefield.." https://github.com/aahmadnejad/Fyodost.

[87] A. Bhardwaj, V. Mangat, R. Vig, S. Halder, and M. Conti, "Distributed denial of service attacks in cloud: State-of-the-art of scientific and commercial solutions," *Computer Science Review*, vol. 39, p. 100332, 2021.

[88] A. H. Lashkari *et al.*, "Ntlflowlyzer: A python open-source flow extractor for network and transport layer bidirectional flows." GitHub repository, 2023. https://github.com/ahlashkari/NTLFlowLyzer.

[89] A. H. Lashkari, "Cicflowmeter v4.0 (formerly known as iscxflowmeter) – a network traffic bi-flow generator analyser for anomaly detection," tech. rep., Canadian Institute for Cybersecurity (CIC), York University, 2018. DOI:10.13140/RG.2.2.13827.20003.

[90] V. Paxson, "Bro: A system for detecting network intruders in real-time," in *Proceedings of the 7th Conference on USENIX Security Symposium*, pp. 3—3, 1999.

[91] C. Bullard *et al.*, "Argus – audit record generation and utilization system." SEI report / project documentation, 2014. "Argus – the Audit Record Generation and Utilization System is the first implementation of network flow monitoring" – Wikipedia summary.

Vita

Vita

Candidate’s full name: AmirHossein Ahmadnejad Roudsari

University attended (with dates and degrees obtained):
Master of Computer Science (Computer Science & Engineering)
2024 - 2025

Publications: