

Evasive IPv6 Covert Channels: Design, Machine Learning Detection, and Explainable AI Evaluation

Viet Anh Phan^a and Jan Jerabek^b

*Faculty of Electrical Engineering and Communication, Brno University of Technology,
Technická 3082/12, Brno, Czech Republic*

Keywords: IPv6, Covert Channel, Dataset, Machine Learning, Intrusion Detection, Explainable AI.

Abstract: Adopting a dual approach, this paper presents a framework that integrates two complementary components: CovertGen6, a novel tool for generating realistic IPv6 covert channel attack packets, and a framework of detection system based on multiple machine learning models. CovertGen6 outperforms existing tools by producing diverse, evasive attack scenarios that are captured by Wireshark and converted into CSV datasets for analysis. These authentic datasets are then used to train and evaluate machine learning models for detecting IPv6 covert channels, with the Random Forest classifier achieving a binary classification AuC of 0.985 and a multi-label classification F1-score of 90.3%. Additionally, the explainable AI technique is incorporated to transparently interpret model decisions and pinpoint the specific header fields used for covert injections. This dual approach bridges the gap between theoretical research and practical network security, laying a robust foundation for intrusion detection systems in IPv6 networks.

1 INTRODUCTION


The introduction of IPv6 has brought about new features (Shiranzaei and Khan, 2018). This also includes new challenges in network security, notably the emergence of covert channels that enable malicious actors to embed hidden communications within legitimate traffic. These IPv6 covert channels pose a significant threat as they can evade traditional intrusion detection systems by exploiting unused or under-monitored protocol fields. Prior research has made important strides in this domain; for instance, Zhao and Wang (Zhao and Wang, 2020) introduced a blind network steganalysis model based on convolutional neural networks, demonstrating high detection accuracy in IPv6 environments. Similarly, Dua et al. (Dua et al., 2022) developed DICCh-D, which utilizes deep neural networks to detect IPv6-based covert channels, thereby highlighting the potential of machine learning in addressing these sophisticated threats.


Despite these promising developments, existing methodologies often rely on oversimplified datasets that fail to capture the complexity and diversity of real-world covert communication scenarios. More-

over, many of these approaches focus solely on maximizing detection accuracy without providing transparency into the model's decision-making process. Addressing these gaps, this paper introduces a novel tool called CovertGen6¹ for designing and generating diverse, evasive IPv6 covert channel packets that realistically mimic potential attack scenarios. The generated network traffic is captured by Wireshark, then transformed into CSV datasets and used to train a suite of machine learning models, achieving approximately 0.985 of AuC for binary classification and around 90.3% of F1-score for multi-label classification.

A key contribution of our work is the integration of explainable AI (XAI) techniques. These methods not only enhance the transparency of our detection models by interpreting their decisions but also identify the specific IPv6 header fields exploited for covert injection. Such interpretability is essential for fostering trust among network administrators and for enabling more informed and rapid responses to security threats.

By combining realistic data generation with robust machine learning detection and in-depth XAI evaluation, our framework bridges the critical gap be-

^a <https://orcid.org/0009-0003-1787-8063>

^b <https://orcid.org/0000-0001-9487-5024>

¹<https://github.com/vafekt/CovertGen6.git>

tween theoretical research and practical network security applications. This approach lays the foundation for more robust and adaptive intrusion detection systems, which are vital to safeguarding IPv6 networks.

2 RELATED WORKS

This section reviews literature on IPv6 covert channels, covering design methodologies that inspire our tool CovertGen6 and detection techniques that inform our integrated approach.

2.1 Covert Channel Designs in IPv6

The design of IPv6 covert channels in Mavani et al. (Mavani and Ragha, 2014) and Mazurczyk et al. (Mazurczyk et al., 2019) leverages the flexibility of IPv6 protocol fields and extension headers to embed hidden data. Mavani et al. focused on exploiting the Destination Options Extension Header, a feature allowing optional processing by destination nodes. They manipulated this header by embedding covert data in two ways: (1) defining custom, unrecognized options (e.g., reserved or vendor-specific codes) that evade standard validation, and (2) abusing the PadN option by inserting nonzero padding bytes, which typically serve alignment purposes but can encode secret information.

Mazurczyk et al. expanded this scope by evaluating six distinct data-hiding techniques in real-world IPv6 deployments. Their methods included abusing fields like the Flow Label, which typically ensures packet sequencing but can be repurposed to encode covert bits, and the Traffic Class field for timing-based steganography. They also exploited extension headers (e.g., Fragment, Hop-by-Hop) by embedding data in rarely monitored fields or manipulating fragmentation offsets. For instance, covert channels were established by altering the Flow Label's pseudo-random values to encode messages or embedding data in unused bits of extension headers.

In practice, such covert channels manifest as seemingly ordinary IPv6 traffic. For example, a packet with a Destination Options Extension Header might appear standard but include a custom option type (e.g., a proprietary code point) carrying encrypted data, or a Flow Label field might exhibit nonrandom patterns that encode a hidden message. Mazurczyk et al. emphasized the challenges in detection, as these channels exploit IPv6's inherent complexity and the limited scrutiny of newer protocol features in security tools. Their work underscores the need for advanced detection frameworks, such as ma-

chine learning models or protocol-aware wardens, to identify anomalies in header fields and traffic behavior.

2.2 Detection Approaches

Wang et al. (Wang et al., 2022) introduced CC-Guard, an IPv6 covert channel detection method based on field matching. Their approach involves extracting specific header fields from IPv6 packets and comparing them against predefined normal patterns using a deterministic decision automaton. Although CC-Guard achieves high detection accuracy by pinpointing deviations in fields like Traffic Class and Flow Label, the exhaustive pattern matching process results in a time-consuming analysis that may hinder real-time applicability.

Danyang Zhao and colleagues (Zhao and Wang, 2020) developed BNS-CNN, a blind network steganalysis model based on convolutional neural networks specifically tailored for IPv6 networks. Their method involves generating a dataset of 24,000 packets by embedding covert data into the IPv6 source address, Hop Limit field, and TCP ISN. The CNN automatically extracts relevant features from this dataset to detect covert channels with high accuracy. However, the dataset's narrow focus on only a few covert channel types limits the model's generalizability to other types of covert communications.

Arti Dua and co-authors (Dua et al., 2022) proposed DICCh-D, which employs a deep neural network (DNN) to detect IPv6-based covert channels. Their approach leverages the pcapStego tool (Zupelli and Caviglione, 2021) to generate covert channels by modifying fields such as Flow Label, Traffic Class, and Hop Limit. While this method demonstrates the potential of DNNs for efficient detection, its reliance on pcapStego confines the dataset to only these specific field modifications, thereby limiting its ability to detect covert channels exploiting other parts of the IPv6 header.

This review shows that while early works effectively highlighted various covert channel designs in IPv6, detection methods have evolved from exhaustive statistical analyses, as seen in CC-Guard, to more efficient machine learning techniques in DICCh-D. However, the limited scope of the datasets used in these machine learning approaches remains a critical drawback, underscoring the need for frameworks that can generate a broader range of covert channel scenarios and support more robust detection strategies.

3 COVERTGEN6: A NETWORK TOOL FOR IPV6 COVERT CHANNEL COMMUNICATION

3.1 Software Architecture

Before presenting the methodology, we introduce CovertGen6, a custom-developed command-line tool for covert data transmission over IPv6 networks. Implemented in Python, it enables embedding data within various IPv6 header fields to establish covert channels that bypass traditional detection mechanisms. CovertGen6 supports multiple covert channel types, encryption methods, and transport protocols, offering flexibility in constructing and transmitting covert packets. The main components of CovertGen6 are depicted in Figure 1.

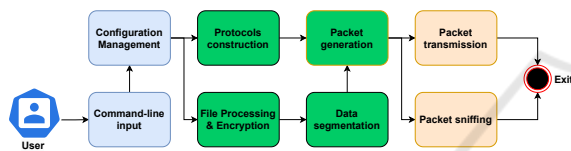


Figure 1: Software architecture of CovertGen6.

As shown in Figure 1, the user first enters the parameter as a command-line input, with its usage explained later in the next section. Once all parameters are provided, the process moves on to the next module. In this phase, **Configuration Management** validates the inserted parameters. The tool verifies the existence of the specified network interface, input file, and cryptographic parameters, ensuring all required inputs are correctly formatted.

Once validated, the tool moves into **Protocol Construction**. This module determines how the packet frame is built based on the selected protocol (IPv6, ICMPv6, UDP, or TCP). The tool structures packet headers accordingly to ensure proper network transmission.

Simultaneously, **File Processing & Encryption** prepare the data for insertion into IPv6 packets. The selected file is read in binary format, and if encryption is enabled, it is processed using AES-CBC or DES-CBC encryption (based on the user's preference). The encryption key can either be user-defined or automatically generated. The encrypted data is then encoded for insertion into IPv6 header fields.

The method of **Data Segmentation** depends on the selected covert channel type:

- For Traffic Class, Flow Label, Next Header, and Hop Limit: Each 1-byte segment of encrypted data is embedded into the respective field of a single IPv6 packet. These fields have limited size,

making them suitable for small-scale covert transmission.

- For IPv6 Extension Headers (Hop-by-Hop, Destination Option, Routing Header, Fragment Header): Data is divided into 256-byte blocks and embedded within Unknown Options of these headers. If the remaining data is less than 256 bytes, zero-padding is applied to maintain block consistency.

This segmentation process ensures that the data is correctly structured within IPv6 packets, avoiding corruption or detection by simple header integrity checks.

With the covert data embedded, **Packet Generation** constructs the final IPv6 packets. These packets follow the structure defined in the protocol construction phase, incorporating the chosen protocol, encryption method, and covert channel technique.

Once generated, the packets are sent via **Packet Transmission**, using raw socket operations to inject them directly into the network interface. The tool ensures proper sequencing of packets, maintaining a structured covert communication flow.

In parallel, CovertGen6 includes a **Packet Sniffing** module that monitors network traffic for responses from the covert receiver. If the receiver acknowledges the packets, the tool logs response metadata, including source and destination addresses, timestamps, and protocol details. This feature enables users to verify whether covert packets successfully bypassed Intrusion Detection System (IDS) or firewall defenses between the sender and receiver.

By analyzing responses, the tool helps assess network security risks and refine covert channel strategies to improve stealth and reliability.

3.2 Usage

The tool is executed with the following inserted parameters:

```
python3 covert_channel.py [interface] [options]
```

- **Interface:** The network interface to use from the sender. Manually specified or automatically resolved.
- **Source and Destination MAC/IP Addresses (-smac/-dmac and -sip/-dip):** Manually specified or automatically resolved.
- **Covert Channel Type (-t):** Currently, only one option may be used at a time; data must be embedded exclusively in one of the following options:
 - Traffic Class (TC)

- Flow Label (FL)
- Next Header (NH)
- Hop Limit (HL)
- Hop-by-Hop (HBH)
- Routing Header (RH)
- Destination Option (DO)
- Fragment Header (FH)
- **Input File (-i):** The file to be transmitted covertly.
- **Encryption (-a):** Data can be encrypted using AES-CBC or DES-CBC, with an optional user-provided encryption key.
- **Protocol (-p):** Users may select one of the following protocols for packet construction: IPv6, ICMPv6, UDP, or TCP. For ICMPv6, an Echo Request packet (used for ping) is employed. For TCP, a TCP SYN packet is used, which is ideal for carrying the user's message to the target. In every case, the hidden data remains unaffected since the chosen protocol merely serves as a carrier, ensuring that the packets appear as legitimate, normal traffic.
- **Key (-k):** The key to encrypt the file using the defined algorithm (automatically generated if skipping).

4 COMPARISON OF CovertGen6 WITH EXISTING IPv6 COVERT CHANNEL TOOLS

To evaluate the advantages of CovertGen6, we compare it with existing tools, including pcapStego², THC-IPv6³, IPv6CC⁴, and IPv6teal⁵. Table 1 highlights key differences in supported covert channel types, encryption capabilities, and protocol compatibility.

Table 1: Comparison of IPv6 Covert Channel Tools.

	pcapStego	THC-IPv6	IPv6CC	IPv6teal	CovertGen6
IPv6 Covert channel types	3	1	3	1	5
Encryption allowed	×	✓	×	×	✓
Encryption algorithms	×	Blowfish	×	×	AES-CBC, DES-CBC
ICMPv6 protocol applied	✓	✓	×	×	✓
UDP protocol applied	×	×	×	×	✓
TCP protocol applied	×	×	×	×	✓

CovertGen6 outperforms existing IPv6 covert channel tools in multiple aspects. Unlike pcapStego, IPv6CC, and IPv6teal, which support only a limited number of covert channel types, CovertGen6 extends

its capability to five different covert channel types (Traffic Class, Flow Label, Hop Limit, Next Header and Extension Headers), providing greater flexibility in data transmission. Furthermore, while some tools such as THC-IPv6 allow encryption, they are limited to a single algorithm (Blowfish). CovertGen6 enhances security by offering AES-CBC and DES-CBC encryption options, significantly increasing resistance against IDS and firewalls. Additionally, CovertGen6 is the only tool that supports covert data transmission over UDP and TCP, whereas other tools primarily rely on ICMPv6. This extended protocol support allows CovertGen6 to operate in more diverse network environments, making it a superior choice for covert communication research and practical applications. By integrating these advanced features, CovertGen6 sets a new benchmark for IPv6 covert channel tools.

5 METHODOLOGY

This section details the proposed framework for designing, generating, and detecting IPv6 covert channels, and is organized into five main phases: Traffic Collection and Packet Generation (5.1), Dataset Creation: PCAP to CSV Conversion (5.2), Data Preprocessing (5.3), Machine Learning-Based Detection and Evaluation (5.4) and Explainable AI-Based Model Validation (5.5), as described in Figure 2.

The proposed methodology provides a structured approach to detecting IPv6 covert channels using machine learning. It begins with traffic collection and packet generation, where covert IPv6 (attack) traffic is sent using our designed tool called CovertGen6 and captured by sniffing tools (Wireshark, in this case). The collected PCAP files are then processed by extracting essential fields, converting them into CSV format with modified features. In the data preprocessing phase, irrelevant features are removed, missing values handled, categorical data encoded, and the dataset is split into training and testing sets with proper normalization. Machine learning models from four main categories, including Gradient Boosting, Neural Network, Ensemble Learning and Probabilistic Classifier are then trained and evaluated based on accuracy, precision, recall, F1-score, and AuC metrics. Finally, XAI using SHAP is applied to interpret model decisions, highlighting the most influential features in covert channel detection. This methodology ensures an effective, data-driven approach for identifying hidden communications within IPv6 networks, enhancing security through AI-driven detection and explainability.

²https://github.com/Ocram95/pcap_injector.git

³<https://github.com/vanhauser-thc/thc-ipv6.git>

⁴https://github.com/Ocram95/IPv6CC_SoftwareX.git

⁵<https://github.com/christophetd/IPv6teal.git>

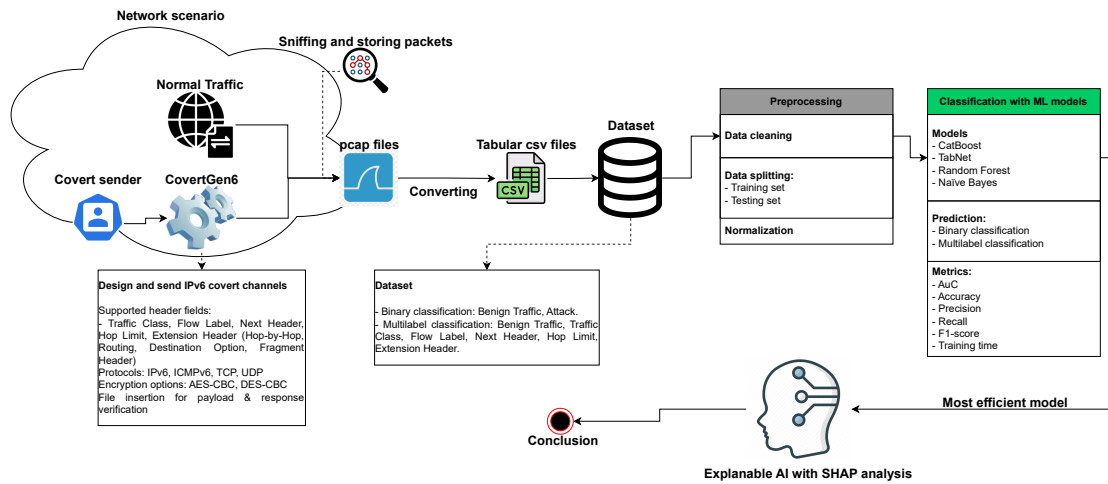


Figure 2: Framework of proposed methodology for IPv6 Covert Channel design, detection and evaluation.

5.1 Traffic Collection and Packet Generation

In the depicted scenario (Figure 3), CovertGen6 operates on the covert sender's side, transmitting various types of covert channel packets to the covert receiver. To ensure a realistic dataset, benign traffic is collected by allowing normal network communication between the sender and other hosts. This background traffic is captured concurrently with the covert channel attacks, ensuring a comprehensive dataset containing both legitimate and malicious activities.

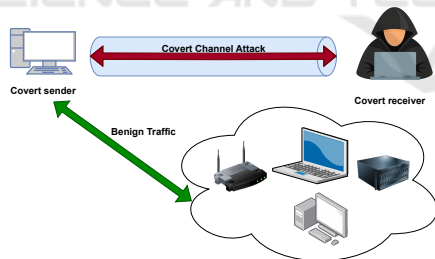


Figure 3: Network Scenario Utilizing CovertGen6 for Attack Traffic Extraction and Benign Traffic Monitoring.

All network traffic, including both benign and covert channel attack packets, is captured and stored using Wireshark in the form of PCAP files. These files serve as the dataset for evaluating classification approaches. We analyze the packet distribution in two scenarios: multi-label classification, where each covert channel type is treated as a distinct class, and binary classification, where all attack types are grouped into a single category.

5.2 Dataset Creation: PCAP to CSV Conversion

Our dataset is created by capturing network traffic with Wireshark into PCAP files, which include both benign and covert channel attack packets. We convert these PCAP files to CSV format, where each row corresponds to a packet with features extracted from its headers. These features are essential for machine learning models to identify IPv6 covert channels.

Several existing approaches have attempted to extract features from PCAP files for similar purposes, but they have notable drawbacks. Wang et al. (Wang et al., 2022) focused on standard IPv6 header fields such as Flow Label, Traffic Class, and Hop Limit, but their method does not adequately handle packets with extension headers, treating them the same as those without. This is problematic for detecting covert channels that exploit extension headers. Another method by Zhao et al. (Zhao and Wang, 2020) constructs a two-dimensional matrix from IP, TCP, and UDP headers, but when multiple extension headers are present, it leads to excessive zero-padding, resulting in sparse matrices that can degrade model performance.

In contrast, our approach extracts a wide range of features from both the main IPv6 header and its extension headers. Key features include Version, Traffic Class, Flow Label, Next Header, Hop Limit, Source and Destination Addresses, as well as specific fields from extension headers like Destination Options, Hop-By-Hop Options, Routing Header, and Fragment Header. Additionally, we extract transport layer information such as source and destination ports for UDP and TCP, and payload-related fields. For each packet, if a particular feature is present, its value

is recorded; otherwise, it is set to zero. This comprehensive feature set allows us to capture the nuances of IPv6 covert channel attacks effectively.

The extracted features are organized into a CSV where each row represents a packet. Figure 4 shows the sample distribution for multi-label classification of different attack types. For binary classification, all attacks are labeled as **Attack**.

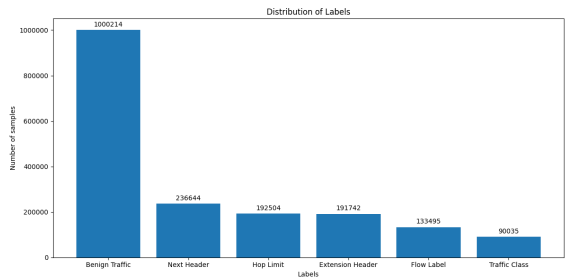


Figure 4: Samples distribution for the case of multi-label classification.

5.3 Data Preprocessing

In this phase, we refine our dataset to enhance the performance and generalization of our machine learning models. First, we clean the dataset by eliminating rows containing non-numerical values. This ensures that our data is both consistent and representative of unique network flows. After cleaning, the dataset is split into training and testing subsets using a widely accepted 70/30 ratio, providing a balanced approach for model training and validation.

Normalization is then applied to the numerical features to ensure uniformity in scale. We use two common normalization techniques: the Standard Scaler and the Min-Max Scaler. The Standard Scaler transforms each feature to have a zero mean and unit variance, which is particularly useful when the data follows a Gaussian distribution. In contrast, the Min-Max Scaler rescales features to a specified range (typically $[0, 1]$), preserving the relationships among the original data values while compressing the scale. These normalization methods help prevent features with larger numerical ranges from dominating the model training process.

Following these steps, the dataset is fully prepared and ready for input into our machine learning models.

5.4 Machine Learning–Based Detection and Evaluation

In our approach, we leverage four distinct categories of machine learning models to detect IPv6 covert channels, each bringing unique strengths and limi-

tations to the task. The models under consideration include Gradient Boosting (CatBoost), Neural Networks (TabNet), Ensemble Learning (Random Forest), and a Probabilistic Classifier (Naive Bayes).

For binary classification, we evaluate model performance using the Area Under the Receiver Operating Characteristic Curve (AuC), a robust metric that provides an aggregate measure of performance across all classification thresholds by effectively capturing the trade-off between true positive and false positive rates. In contrast, for multi-label classification, we employ accuracy, F1-score, recall, and precision to obtain detailed insights into the performance on individual classes. Additionally, training time is recorded as a metric in both cases, allowing for a comprehensive comparison of each model’s efficiency and overall suitability.

5.4.1 Binary Classification

After training and testing using different models, the ROC curves for each model are shown in Figure 5. The AuC values indicate the effectiveness of each classifier in distinguishing between classes:

- **Random Forest:** AuC = 0.985
- **CatBoost:** AuC = 0.976
- **TabNet:** AuC = 0.959
- **Naive Bayes:** AuC = 0.818

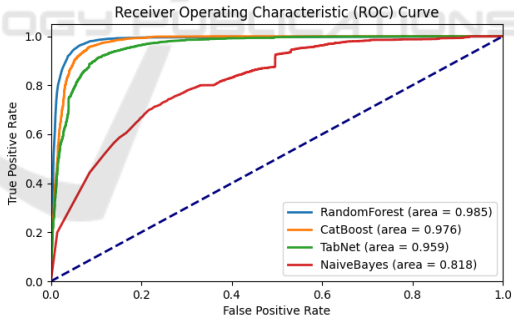


Figure 5: ROC Curve Comparison for Binary Classification.

From these results, it is evident that Random Forest and CatBoost outperform the other models in classification performance, with AuC values very close to 1. TabNet follows with slightly lower accuracy, while Naive Bayes shows the weakest performance due to its assumption of feature independence.

Besides classification performance, training time is a crucial metric, particularly for real-time detection scenarios. The recorded training times for each model are as follows:

- **CatBoost:** 1.042 seconds
- **TabNet:** 76.241 seconds
- **Random Forest:** 5.371 seconds
- **Naive Bayes:** 0.824 seconds

The results indicate that CatBoost provides a near-optimal balance between high classification performance and fast training time, making it a strong candidate for practical deployment. Random Forest also performs well but requires slightly longer training time. TabNet, because of its deep learning architecture, shows a significantly higher computational cost (76.241 seconds), which may not be suitable for time-sensitive applications. Naive Bayes, while the fastest, demonstrates the weakest classification performance.

5.4.2 Multi-Label Classification

Table 2 summarizes the performance of the evaluated models on the multi-label classification task. The metrics include Accuracy, Precision, Recall, F1-score, and Training Time (seconds).

Table 2: Multi-label Classification Results.

Metric	CatBoost	TabNet	Random Forest	Naive Bayes
Accuracy	0.812	0.782	0.902	0.420
Precision	0.815	0.780	0.910	0.675
Recall	0.805	0.784	0.896	0.400
F1-score	0.810	0.782	0.903	0.520
Training Time (s)	1.195	110.234	5.845	0.034

The Random Forest classifier achieves the highest overall performance, with an accuracy, precision, recall, and F1-score of approximately 0.903, which substantially exceeds the results of the other models. Although Naive Bayes offers an extremely short training time, its low accuracy and F1-score make it unsuitable for practical deployment. CatBoost demonstrates competitive performance, yet its results remain inferior to those of Random Forest, and TabNet, despite its deep learning approach, suffers from significantly higher training time and lower performance metrics.

Consequently, after comparing both binary and multi-label classification results, Random Forest emerges as the most efficient model for the detection of IPv6 covert channels.

5.5 Explainable AI-Based Model Validation

The Random Forest algorithm demonstrated superior performance in both binary and multi-label classification tasks for detecting IPv6 covert channel attacks, achieving high accuracy and robustness. To validate its reliability and interpretability, SHAP analysis is employed both binary and multi-label classification.

This approach deciphers the model’s decision logic by quantifying feature contributions, ensuring transparency in a domain where explainability is critical for trust and actionable insights.

As can be seen in Figure 6, the Random Forest model for binary classification identifies Hop Limit (relative importance: almost 0.4) as the top feature aligns with its exploitation patterns in IPv6 covert channels. In benign traffic, the Hop Limit field follows predictable decrement patterns as packets traverse routers. However, frequent or anomalous variations in this field within a single communication session are statistically rare in legitimate traffic. Covert channel attacks often manipulate this field to encode data or evade detection (e.g., maintaining static values or cycling through specific ranges), creating a distinct fingerprint.

Next Header ranks second, despite its prevalence across packet types. While this field naturally varies in legitimate traffic (e.g., indicating TCP, UDP, or extension headers), its importance in detection arises from attackers’ use of non-standard or sequential header types (e.g., chaining Fragment, Destination Options, or Routing headers) to embed covert payloads. However, its lower discriminative power compared to Hop Limit reflects its inherent variability in both benign and malicious traffic, reducing its uniqueness as an attack marker.

Flow Label and Traffic Class show moderate importance. While these fields are occasionally repurposed in attacks (e.g., Flow Label manipulation to mimic legitimate flows), their standardized use in QoS prioritization limits their anomaly potential. Lower-ranked features like Destination Option NH and ICMPv6 Data are less impactful, as their exploitation requires specialized attack logic (e.g., abusing ICMPv6 for tunneling).

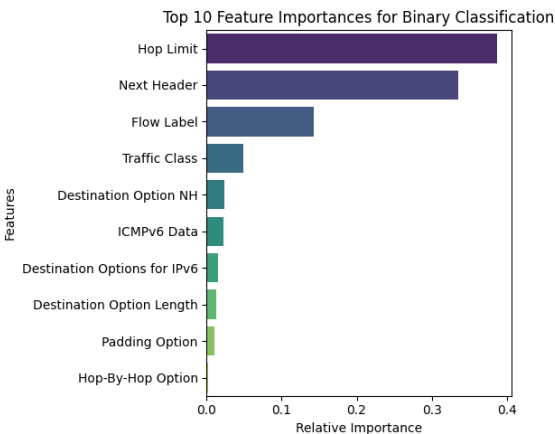


Figure 6: Feature Importance for Binary Classification.

For the multi-label classification (depicted in Figure 7), Next Header emerges as the top feature (almost 0.3) in multi-label classification, as different attack subtypes (e.g., Traffic Class vs. Extension Header attacks) require distinct protocol configurations. For example:

- Extension Header-based attacks (e.g., Destination Options abuse) directly alter Next Header values to chain non-standard headers.
- Traffic Class attacks manipulate QoS bits but retain standard Next Header values (e.g., UDP/TCP).

This necessitates Next Header as a primary discriminator. Hop Limit retains importance but to a lesser degree than in binary classification, as its manipulation is common across multiple attack subtypes, reducing its specificity. Flow Label and Destination Options for IPv6 gain relative importance for distinguishing attacks that modify flow-state metadata or embed payloads in extension headers.

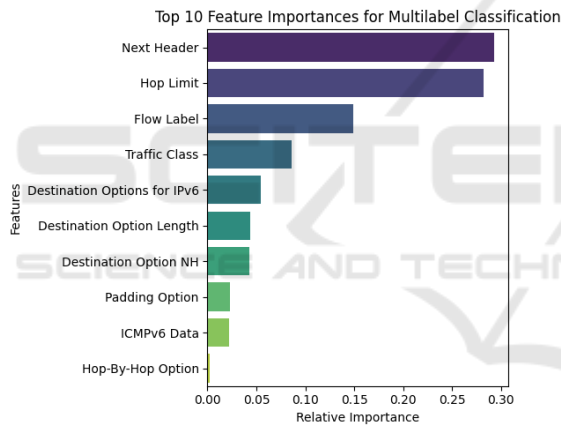


Figure 7: Feature Importance for multi-label Classification.

Next, the SHAP summary plot reveals critical insights into the Random Forest model’s decision logic for binary classification (through Figure 8). The model demonstrates strong logical fidelity to IPv6 covert channel mechanics. Its reliance on Next Header and Hop Limit, which are most frequently manipulated in this dataset, reflects domain-aware feature prioritization. The suppression of Version (SHAP: ± 0.02) and Option Payload (SHAP: ± 0.03) aligns with their irrelevance to header-based attacks. However, the muted impact of Fragment Header suggests potential blind spots in detecting fragmentation-based covert channels, warranting dataset augmentation. Overall, for the case of binary classification, this SHAP analysis validates the model’s efficacy in detecting header manipulation patterns while exposing limitations in payload-based attack detection. Feature

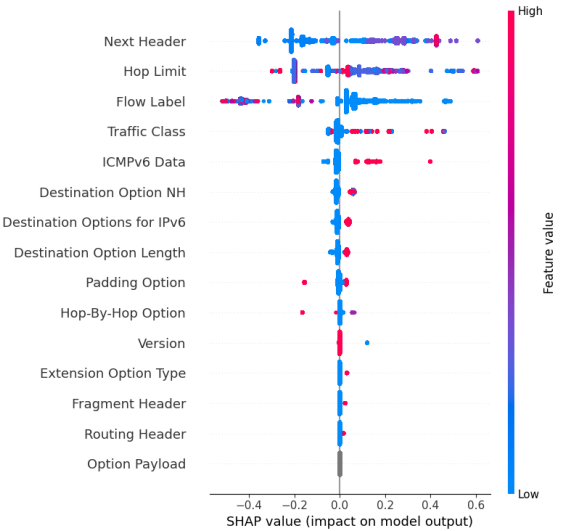


Figure 8: SHAP Summary Plot of Attack class for Binary Classification.

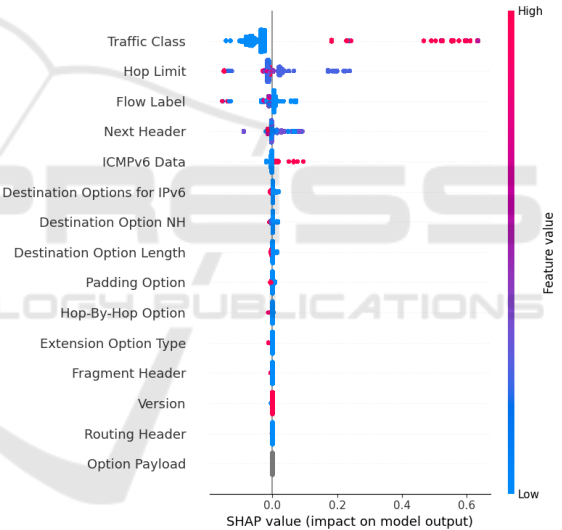


Figure 9: SHAP Summary Plot of Traffic Class for multi-label Classification.

importance hierarchy directly mirrors IPv6 protocol abuse trends, confirming the model’s operational relevance.

In multi-label classification, the SHAP analysis reveals assessment below:

- When detecting a Traffic Class attack (refer to Figure 9), the Traffic Class feature exhibits the highest impact, with a SHAP magnitude exceeding 0.6, reinforcing its role as the primary determinant of the model’s output. Other features, such as Hop Limit and Flow Label, rank second in terms of contribution to detection. However, their influence is more balanced and moderate, as

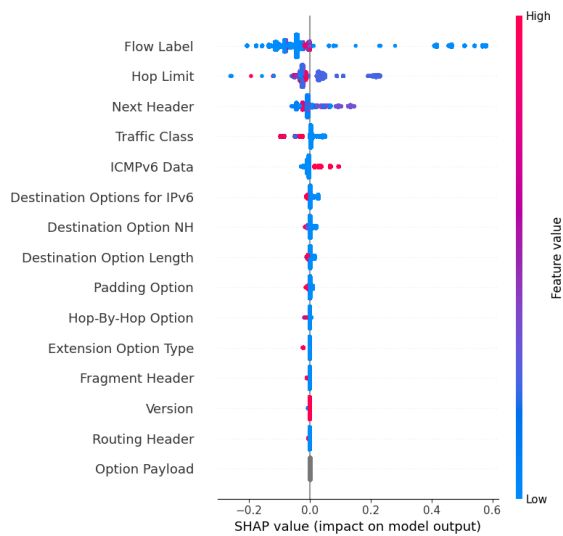


Figure 10: SHAP Summary Plot of Flow Label for multi-label Classification.

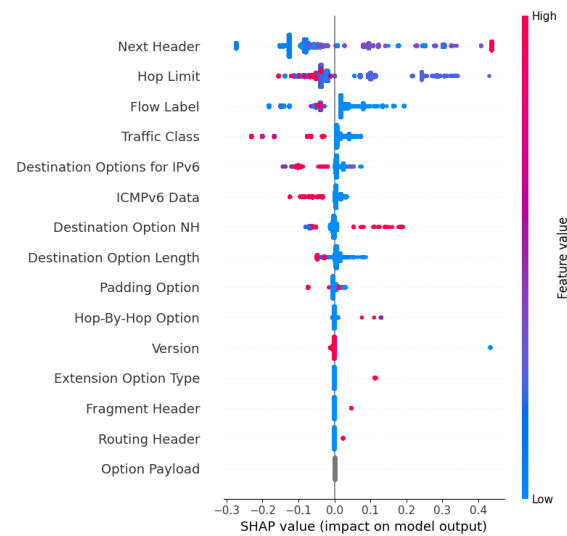


Figure 12: SHAP Summary Plot of Next Header for multi-label Classification.

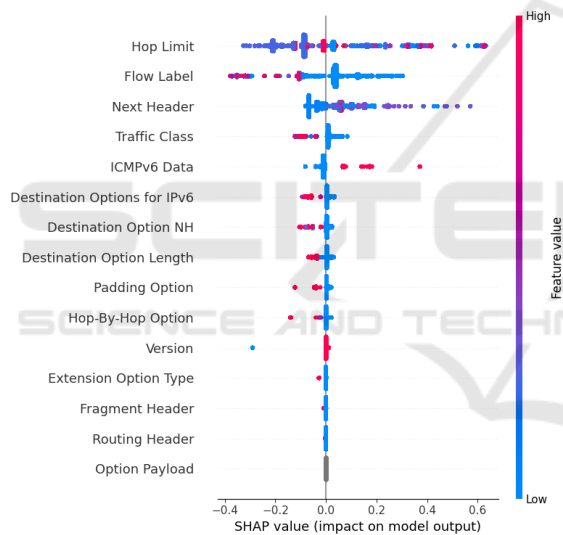


Figure 11: SHAP Summary Plot of Hop Limit for multi-label Classification.

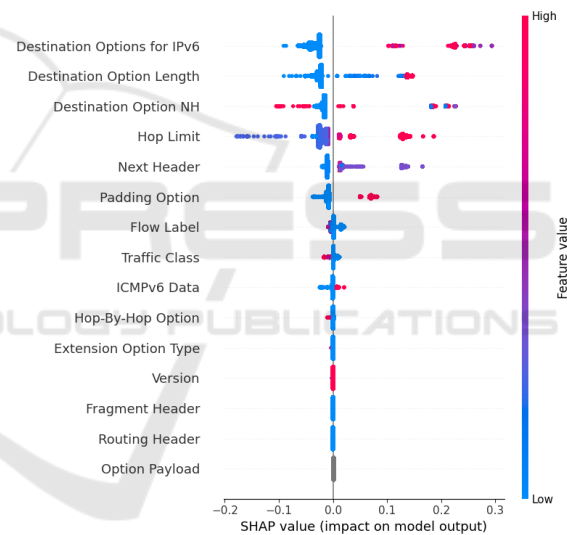


Figure 13: SHAP Summary Plot of Extension Headers for multi-label Classification.

- their positive and negative SHAP magnitudes are approximately equal.
- When detecting Flow Label attack (through Figure 10), Flow Label dominates (SHAP magnitude: 0.6), with secondary contributions from Hop Limit (0.3) and Next Header (0.2). This aligns with Flow Label manipulation tactics that pair altered flow identifiers with header modifications to mimic legitimate traffic. The model effectively isolates Flow Label-specific anomalies but shows minor overreliance on Traffic Class (0.2), a less specific feature.

- When detecting Hop Limit attack (through Figure 11), Hop Limit exhibits the highest impact (SHAP magnitude: 0.6), consistent with its role as a session-level fingerprint. Supporting features like Next Header and Flow Label indicate the model contextualizes HL anomalies with protocol-chain deviations. The narrow SHAP spread for Hop Limit (0.4–0.6) confirms robust discrimination of Hop Limit manipulation from benign traffic.
- When detecting Next Header attack (through Figure 12), Next Header drives predictions (SHAP magnitude: over 0.5), particularly for non-standard values. Secondary features like

Hop Limit (0.35) and Destination Option Next Header for IPv6 (0.2) reflect attacks combining header-chain manipulation with session-level anomalies. The model excels at detecting Next Header-based evasion but shows reduced sensitivity to Routing Header (SHAP: 0.05), a legacy attack vector.

- When detecting Extension Header attack (through Figure 13), the model prioritizes Destination Options for IPv6, Destination Option Length, and Destination Option Next Header (SHAP magnitude: 0.25 to over 0.3) for Extension Header detection. These features directly reflect payload embedding in extension headers, such as padding fields or option overloading. The strong correlation confirms the model's precision in detecting header-structure anomalies, though limited impact from Fragment Header (SHAP: 0.05) suggests gaps in fragmentation-based attack detection.

The SHAP analysis validates the model's effectiveness in detecting IPv6 covert channel attacks across both binary and multi-label classification tasks. The binary classifier reliably distinguishes malicious traffic by prioritizing protocol-layer anomalies in header fields rarely altered in benign traffic. For multi-label classification, the model adapts to subtype-specific patterns, leveraging distinct feature hierarchies to differentiate attack vectors. Minor overlaps in feature influence for edge cases suggest targeted refinements, but the model demonstrates robust alignment with IPv6 attack mechanics, confirming its practical utility for real-world deployment.

6 CONCLUSION AND FUTURE WORK

This paper introduced a dual approach for detecting IPv6 covert channels, combining CovertGen6, a novel attack dataset generator, with machine learning-based detection. CovertGen6 enhances dataset realism by generating five covert channel types, which are captured and processed for model training. The proposed detection system, leveraging multiple machine learning models, achieved high classification performance, with the Random Forest classifier obtaining an AuC of 0.985 in binary classification and an F1-score of 90.3% in multi-label classification. Furthermore, XAI method provided transparent insights into model decisions, identifying critical IPv6 header fields exploited in covert communications.

Future work will focus on expanding the scope of IPv6 covert channels by exploring additional extension headers, IPv6 flags, and other unexplored fields for covert transmissions. Additionally, we aim to refine feature selection methodologies to align with newly discovered covert channel types, ensuring a more adaptive and precise detection framework.

ACKNOWLEDGEMENTS

This work is supported by Ministry of the Interior of the Czech Republic under Grant VK01030019.

REFERENCES

- Dua, A., Jindal, V., and Bedi, P. (2022). Dicch-d: Detecting ipv6-based covert channels using dnn. *Communications in Computer and Information Science*, 1670 CCIS:42 – 53.
- Mavani, M. and Ragha, L. (2014). Covert channel in ipv6 destination option extension header. In *2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)*, pages 219–224.
- Mazurczyk, W., Powójski, K., and Caviglione, L. (2019). Ipv6 covert channels in the wild.
- Shiranzaei, A. and Khan, R. Z. (2018). Ipv6 security issues—a systematic review. *Advances in Intelligent Systems and Computing*, 638:41 – 49.
- Wang, J., Zhang, L., Li, Z., Guo, Y., Cheng, L., and Du, W. (2022). Cc-guard: An ipv6 covert channel detection method based on field matching. page 1416 – 1421.
- Zhao, D. and Wang, K. (2020). Bns-cnn: A blind network steganalysis model based on convolutional neural network in ipv6 network. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12022 LNCS:365 – 373.
- Zuppelli, M. and Caviglione, L. (2021). Pcapstego: A tool for generating traffic traces for experimenting with network covert channels.