electronics

MDPI

Article

Explainable Firewall Penetration Testing Method Employing Machine Learning

Algimantas Venčkauskas, Jevgenijus Toldinas * and Nerijus Morkevičius

Department of Computer Science, Kaunas University of Technology, 44249 Kaunas, Lithuania; algimantas.venckauskas@ktu.lt (A.V.); nerijus.morkevicius@ktu.lt (N.M.)
* Correspondence: eugenijus.toldinas@ktu.lt

Abstract

Cyber adversaries are becoming more sophisticated, creating complex security challenges as digital services expand. The reliability of the firewall is of the utmost importance in the context of network security since it serves as the first line of protection. Penetration testing is an approach used to evaluate the reliability of a firewall and improve security by uncovering exploitable flaws. Frequently, penetration testing solutions are developed using machine learning, and it is of the utmost importance to explain the obtained results during the penetration testing. The emergence of explainable AI (XAI) addresses transparency in ML models, which is essential for informed cybersecurity decisions. Additionally, effective penetration testing reports are crucial for organizations, helping them comprehend and address vulnerabilities with tailored mitigation strategies. This study contributes to firewall security by developing an explainable penetration testing method, which includes two machine learning classification models: a binary model for detecting attacks and a multiclass model for identifying attack types with an explainability feature. This research introduces a novel explainability method that emphasizes significant features related to attack types based on multiclass predictions and proposes an approach using the extended System Security Assurance Ontology (SSAO) to clarify vulnerabilities and suggest alternative mitigation strategies. After evaluating numerous ML algorithms for the CIC-IDS2017 dataset, the Fine Tree model was considered to have the greatest performance. For the binary model, it achieved a validation accuracy of 99.7%, while for the multiclass model, it achieved a validation accuracy of 99.6%. Both models were used to test the firewall for vulnerabilities. Firewall penetration testing using the binary model achieves an accuracy of 82.1%, while the multiclass model achieves an accuracy of 78.7%.

Keywords: penetration testing; firewall; machine learning; explainability; ontology application

Academic Editors: Jianhua Yang, Hyrum Carroll, Linqiang Ge and Lixin Wang

Received: 3 February 2026
Revised: 24 February 2026
Accepted: 25 February 2026
Published: 1 March 2026

1. Introduction

Cyber adversaries are becoming increasingly sophisticated, and, ultimately, many novel threats are launched within the digital domain. Security challenges become increasingly intricate and significant as the number of services and subscribers grows, driven by advanced technologies. To maintain a secure environment, cybersecurity professionals recommend conducting regular assessments of the organization's information security posture through expert-led penetration testing [1]. When it comes to the management of vital data, the objective of network penetration testing is to protect data and maintain overall security. Some examples of this type of attack include SQL injections, firewalls that are not properly contabled, and classic viruses or malware [2]. Additionally, to guarantee

the safety of the network over the long term, several requirements mandate that it undergo continuous maintenance and penetration testing.

A firewall is a network security device that analyzes incoming and outgoing network traffic and decides whether to allow or block certain traffic based on a defined set of security rules [3]. The process of filtering packets through a firewall is an essential component in the preservation of the entire network and the upkeep of the internalized security of a local network. In the majority of networks, network firewalls continue to be the primary line of defense. The traditional version of a firewall comprises a single dedicated machine, similar to a router, which applies the rules in a sequential manner to each received packet.

As a result of their widespread use and the sensitive information that they manage, web applications are currently becoming increasingly popular targets for cyberattacks. Traditional web application firewalls that are focused on rules have proven to be insufficient as the complexity and volume of these threats continue to increase. Because these conventional systems rely on predefined signatures or rules that are manually curated, they are less effective against threats that are novel or disguised. A viable alternative is provided by machine learning (ML) techniques, which enable systems to learn from prior traffic patterns and identify assaults that have been observed before, as well as those that have not been seen before [4].

The issue with some cutting-edge ML models is a lack of transparency, trustworthiness, and explainability. To address this issue, explainable artificial intelligence (XAI) emerged [5]. It is a study area dedicated to making black-box models more intelligible to humans. Recent research on this topic has led to the development of many methodologies, including LIME (Local Interpretable Model-Agnostic Explanations) and SHAP (Shapley Additive Explanations). XAI has a wide range of applications, including ML. However, the majority of research on model interpretation focuses on other domains such as computer vision, natural language processing, biology, and healthcare. This presents a problem for penetration testing experts entrusted with assessing penetration test results, limiting their ability to make informed decisions.

A customer-centric view on usability gaps in penetration testing reports, with a particular emphasis on the issues that companies confront when it comes to comprehending, prioritizing, and acting on the offered insights studied is reported in [6]. Penetration test reports play a crucial role in assisting organizations in identifying and mitigating security vulnerabilities. The efficacy of these reports is contingent on the degree to which clients are able to convert the results into actions that can be implemented. This was accomplished by avoiding generic mitigations, offering numerous mitigation options, evaluating the impact of each option, and stating preferred solutions when applicable.

The main contributions of this study are as follows:

- A novel explainable firewall penetration testing method, utilizing two types of ML classification models:
    - A binary classification model to identify the existence of an attack;
    - A multiclass classification model to determine the attack type and provide an explainability feature.
- An innovative explainability method that utilizes the list of the most significant features associated with the identified attack type, picked from the multiclass model prediction.
- An explainability approach, which makes use of the extended System Security Assurance Ontology (SSAO) in order to explain vulnerabilities in firewalls and to provide alternative strategies for mitigating those vulnerabilities.

The remainder of this article is organized as follows: Section 2 discusses the related works, Section 3 presents and explains the methodology, Section 4 presents and discusses the experimental results, Section 5 presents the discussion, and Section 6 presents the conclusion.

2. Related Works

In this section, an overview of the related work is provided. In the field of cybersecurity, penetration testing is a process for assessing vulnerabilities and finding issues that could otherwise go undetected. To streamline testing, find problems that are hard to uncover with manual testing, and automate specific operations penetration testing frameworks and tools are utilized [7]. The term "penetration testing" can refer to various types of tests, including network penetration testing, web penetration testing, Internet of Things (IoT) penetration testing, Android penetration testing, and iOS penetration testing [8].

A deep reinforcement learning-based automated penetration testing methodology was proposed in [9]. To collect actual host and vulnerability data for creating realistic training and validation scenarios, the suggested method creatively blends the Shodan search engine with a range of vulnerability datasets. The deep Q-learning network model is then trained using the attack information that is generated for each training scenario using the attack tree methodology. Paper [10] offers a novel hybrid architecture of ontology-based cognitive belief-desire-intention (BDI) agent and reinforcement learning (RL) methodologies. This strategy is based on the philosophy that acknowledges the need to develop an intelligent agent that is equipped with the prior knowledge base of penetration specialists. Additionally, by utilizing the algorithmic power of RL, the agent is able to cooperate with environments that are unknown and dynamic.

Class probability random forest (CPRF) is a novel method that was proposed in [11] for the purpose of improving the performance of network attack detection. The CPRF method makes predictions about the class probabilities based on the network attack dataset. These predictions are subsequently utilized as features in the construction of applied machine learning methods. A high-performance accuracy of 99.9% was demonstrated by the random forest approach, which outperformed the state-of-the-art approach, according to the findings of the extensive investigation. Each technique's performance is validated with a k-fold approach and optimized with hyperparameter tuning.

The model that is based on autoencoders and has the ability to detect and explain anomalies in the traffic transmitted over computer networks is proposed in [12]. Using the most recent CIC-IDS2017 dataset, two models were developed, respectively. The first model utilizes all characteristics, while the second model employs selected features identified through the kernelSHAP method, which is a model-agnostic approach that incorporates XAI. To construct the OPT_Model, which performed better than the initial model, the top 40 contributing features are chosen based on Shapley values.

Research study [13] suggests the implementation of an intrusion detection system that makes use of machine learning methods. These algorithms include decision tree (DT), random forest (RF), and support vector machine (SVM) experiments. LIME is utilized for the purpose of picking between models, evaluating trustworthiness, enhancing models that are not trustworthy, and getting insights into predictions for both system experts and anybody else who is not an expert. The work in question suggests employing a collection of ML models, followed by the implementation of a LIME explainable framework, in order to comprehend the prediction made by the model. The ensemble of ML models demonstrated an enhanced accuracy of 96.25 percent for the IDS prediction, and the LIME explanation graphs demonstrated the performance of the DT, RF, and SVM algorithms in terms of prediction.

The work in [14] tackles the significant issue of deep neural network (DNN) limitations in cybersecurity applications, attributed to its opaque nature and absence of interpretability. Through the dynamic incorporation of prediction confidence weighting into a gradient-weighted framework for quantifying feature importance, significant attack signatures, including anomalous target port access frequency and packet size deviation, were effectively recognized. Cross-model validation demonstrated that the sensitivity of the proposed enhanced DNN model to time-series dynamic features was markedly superior to that of the conventional RF model, establishing a comprehensive connection from feature quantization to behavior analysis. The dynamic residual structure and adaptive regularization technique resulted in a detection accuracy of 99.32% on the CIC-IDS2017 dataset.

The research in [15] introduces an explainable AI framework for intrusion detection that prioritizes interpretable machine learning models over opaque alternatives. Models such as explainable boosting machines (EBMs), DT, and SHAP-augmented RF can achieve robust performance while ensuring transparency, as evidenced by extensive testing on benchmark datasets like NSL-KDD and CIC-IDS2017. The authors assert that these models not only enhance threat identification but also provide analysts with significant insights into model behavior. The incorporation of explainability into intrusion detection systems (IDS) offers numerous concrete advantages. It fosters confidence in automated judgments, facilitates adherence to regulatory mandates, and augments the efficacy of cybersecurity experts through enhanced clarity of insights. The user survey validates that analysts place a high value on explanations, which can substantially influence decision-making processes, particularly in time-sensitive or high-stakes situations. Although black-box models such as neural networks demonstrate significant accuracy, their lack of transparency restricts their use in fields where decision-making requires comprehension and justification.

In [16], the authors propose a novel IDS (integrated multi-classifier model) for detecting and classifying cyber threats, called Hybrid Adaptive Ensemble for Intrusion Detection (HAEnID). Experimental findings demonstrate that the model exhibits good accuracy, achieving elevated detection and classification rates for various cyber threats, alongside a moderate false positive rate across all network states. The integrated model also preserves an effective balance among the many dimensions. The interpretability of the novel machine learning technique is further augmented by the modeling of the SHAP and LIME algorithms.

The critical necessity for transparent elucidations of AI methodologies in intrusion detection is underscored by the introduction of an XAI framework tailored for network intrusion detection tasks in [17] The proposed approach offers both global and local explanations for AI models, highlighting significant model-specific and intrusion-specific aspects across diverse AI models and types of network intrusions. The referenced research entails the evaluation of seven black-box AI models against three authentic network intrusion datasets: RoEduNet-SIMARGL2021, NSL-KDD, and CIC-IDS2017. It first reviews their performance, followed by an assessment of the XAI approaches, revealing insights into the overall significance of features, their contributions to model decisions, the most critical characteristics for each type of intrusion, and common features across many models. The authors demonstrate that employing XAI approaches incurs minimal additional time overhead.

A comprehensive examination of the applications of Large Language Models (LLMs) in cybersecurity and program analysis, providing a more focused and domain-specific examination, is presented in [18]. LLMs can be used at several levels of penetration testing. For example, LLM-driven frameworks make reconnaissance easier by automating tool output interpretation and intelligence gathering. Although LLMs demonstrate considerable promise in tackling cybersecurity issues, many intrinsic limitations impede their wider implementation and efficacy in security operations [19]. A significant disadvantage is the lack

of interpretability, as the opaque nature of LLMs hinders users from comprehending how models make security-critical decisions. The absence of transparency is especially alarming given the heightened risks associated with AI-generated content, including privacy violations, the dissemination of misinformation, and the creation of susceptible code.

Based on the summary presented in Table 1, we can draw the following conclusions:

- In firewall penetration testing, one of the most significant shortcomings is the situation in which an attack is discovered and may be prevented without a comprehensive explanation of why it occurred and what should be done to mitigate it.
- The penetration testing techniques proposed by the other authors are not directed toward XAI; rather, the methods that were proposed are oriented toward automating penetration testing.
- A new approach must be introduced utilizing explainable machine learning for firewall penetration testing.

Table 1. A list of state-of-the-art research papers on improving firewall security.

Reference	Method	Dataset	Knowledge Presentation	XAI	Application Domain
Hu et al. [9]	Deep Q-learning network, the Shodan search engine	Custom dataset	CVE and Microsoft vulnerability file	No	Penetration testing
Qian et al. [10]	Cognitive BDI agent and RL-based Q-learning	Prior knowledge base of penetration experts	Ontology-based BDI agent	No	Penetration testing
Raza et al. [11]	The CPRF approach predicts the class probabilities from the network attack dataset	CIC-IDS2017	Not used	No	IDS
Roshan et al. [12]	The model that is based on autoencoders	CIC-IDS2017	Not used	kernelSHAP	Network anomaly detection
Patil et al. [13]	DT, RF, and SVM	CIC-IDS2017	Not used	LIME	IDS
Chen et al. [14]	An enhanced DNN architecture	CIC-IDS2017	Not used	Interpretability	IDS
Ikram et al. [15]	EBM, DT, and SHAP-augmented RF	NSL-KDD CIC-IDS2017	Not used	SHAP-augmented	IDS
Ahmed et al. [16]	Stacking ensemble, Bayesian model averaging, and conditional ensemble	CIC-IDS2017	Not used	SHAP and LIME	IDS
Arreche et al. [17]	ADA, DNN, LightGBM, SVM, MLP, RF, and KNN	RoEduNet-SIMARGL2021, NSL-KDD, and CIC-IDS2017	Not used	SHAP and LIME	IDS

These assumptions guided us in developing the proposed method for explainable firewall penetration testing using machine learning.

3. Materials and Methods

We propose a method for firewall penetration testing using ML and XAI. In the real world, the firewall is unable to filter all malicious flows with attack packets; some attack packets pass through the firewall and could be obtained in the LAN. The main goal of a penetration test is to identify if all attack packets are filtered by the firewall. The packets that

passed through the firewall are captured from the LAN traffic and sent to the explainable firewall penetration testing engine, as shown in Figure 1.

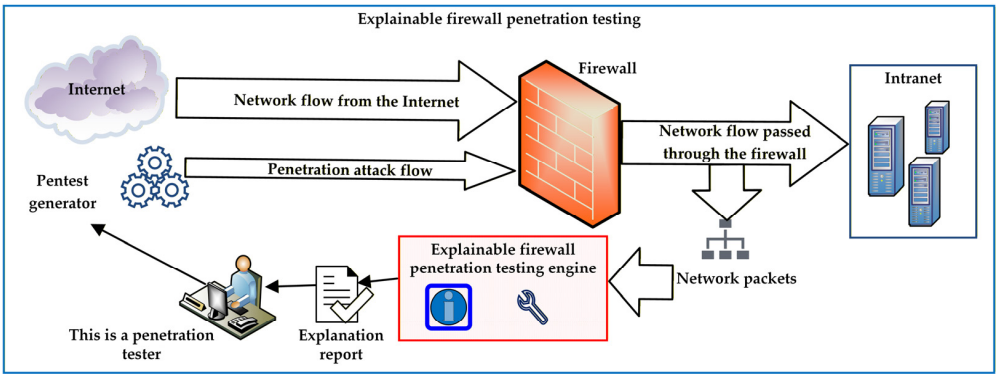

Figure 1. The process of firewall penetration testing using explainable firewall penetration testing engine.

A penetration tester uses the pentest generator to generate network flow for penetration attacks. A firewall receives network traffic from the Internet, as well as penetration attack flow generated by the pentest generator. Only network packets that were not blocked by the firewall and captured in the LAN were sent to the explainable firewall penetration testing engine. The explainable firewall penetration testing engine finds potential security vulnerabilities, detects types of attacks, and offers strategies for mitigating these difficulties, which are summarized in the report delivered to a penetration tester.

The explainable firewall penetration testing (EFPT) engine of the proposed method is depicted in Figure 2.

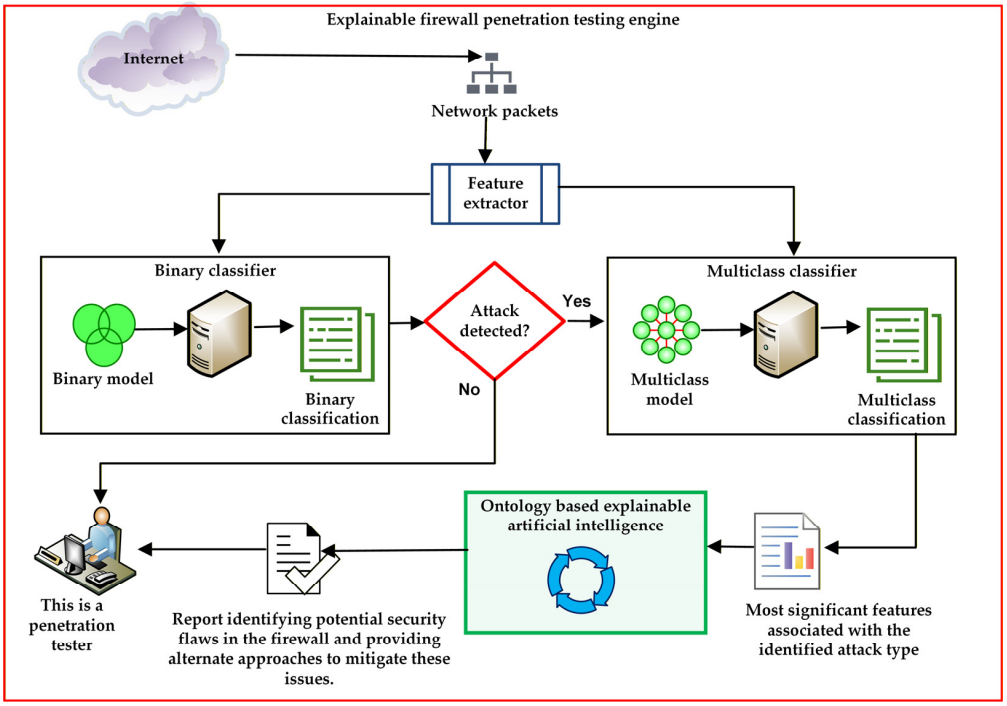

Figure 2. The explainable firewall penetration testing engine.

There are four main modules in the EFPT: the feature extractor, the binary classifier, the multiclass classifier, and the ontology-based explainable artificial intelligence. The feature extractor analyzes captured raw network packets, generates bidirectional flows, and extracts features from them. The feature extraction step dramatically reduces the total

amount of data and enables the derivation of statistical time-related features calculated separately in the forward and backward network flow directions. Extracted features are transmitted to the binary and multiclass classifiers. The first evaluation produces a binary classifier that determines whether an attack is present or not. If an attack is not detected, experts are informed about classification results. If an attack is detected, multiclass classification is then performed to more precisely identify the type of attack, explain the vulnerabilities in the firewall, and suggest alternative strategies for mitigating those vulnerabilities.

Figure 3 depicts the ontology-based explainable artificial intelligence module that uses ontology to explain vulnerabilities in firewalls and provide alternative strategies for mitigating those vulnerabilities.

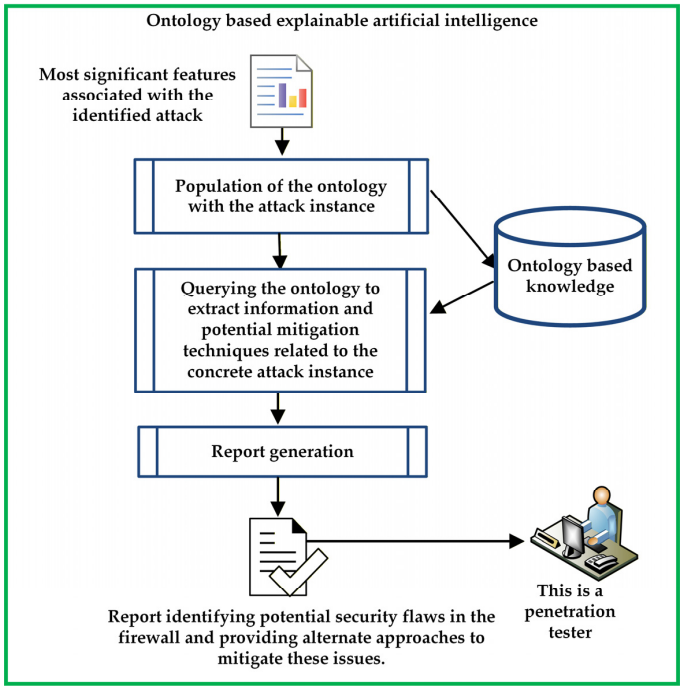

Figure 3. The ontology-based explainable artificial intelligence module.

We propose to incorporate the usage of an ontology-based approach for explaining the results of the detected attacks, as well as for providing further information on the nature of the attack and potential vulnerabilities' mitigation steps. For this purpose, we have extended the readily available System Security Assurance Ontology [20], which, in turn, is based on the well-established Unified Cybersecurity Ontology (UCO) [21]. System Security Assurance Ontology (SSAO) includes concepts and relations representing information collected in widely used vulnerability and weakness databases, such as Common Vulnerabilities and Exposures (CVE) [22], Common Weakness Enumeration (CWE) [23], Common Attack Pattern Enumeration and Classification (CAPEC) [24], etc. The added part of the extended ontology is presented in Figure 4.

In this figure, the new concepts are presented using blue-colored circles, while the original SSAO concepts are presented using red circles. The extended ontology was developed using Protégé v. 5.6.3, an open-source software providing a user-friendly interface for creating, querying, and editing ontologies. Three base concepts necessary for integration with the XAI block of the EFPT process were added to the SSAO. These include *Attack*, *Feature* and *MitigationTechnique* concepts with corresponding object relations. The remaining added concepts are subclasses of the *Attack* concept, providing further details on the types of possible attacks. The *Attack* concept is a direct generalization for eight concepts representing more

detailed types of the possible attacks: *Heartbleed*, *DoS*, *DDoS*, *WebAttack*, *PortScan*, *Patator*, *Infiltration*, and *Bot*. The *WebAttack* concept is further detailed by classes describing the exact nature of the web attack: *SQLInjection*, *BruteForce*, and *XSS*. In the same way, the *Patator* concept is detailed by *SSH* and *FTP* classes. All new concepts presented in Figure 4 were derived from the dataset used for experimental evaluation of the proposed method. In cases when another dataset with additional types of attacks is used, the ontology may be extended to incorporate new types of attacks. Two additional unions of the existing SSAO concepts (*AttackInfo* and *MitigationInfo*) were added to the ontology to minimize the number of the required object relation types. The concepts represented in red show only a few classes of the SSAO, with the sole purpose of clarifying the added object relations. The full SSAO contains 235 classes, 94 object relations, and 2262 axioms and is not included in the diagram.

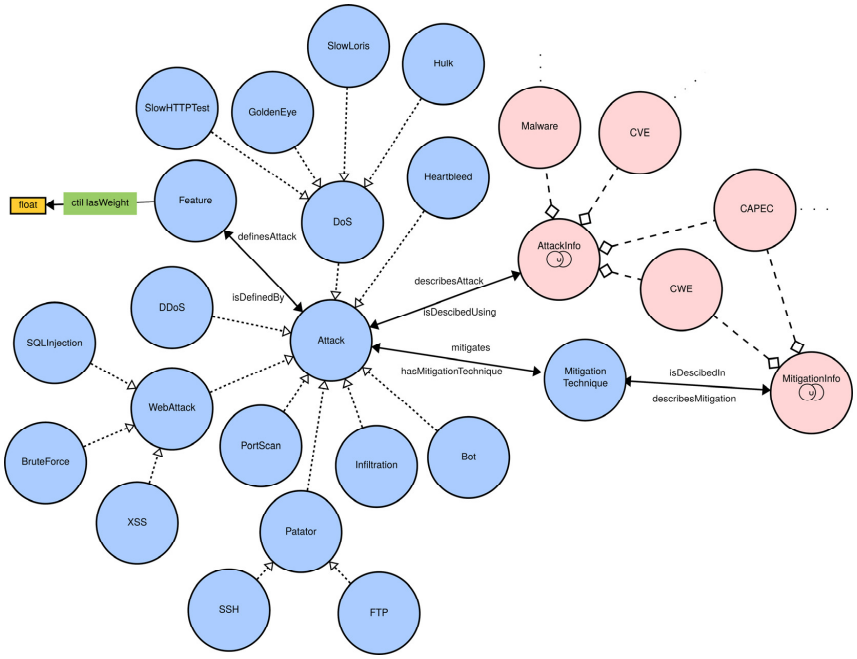

Figure 4. The added part of the extended System Security Assurance Ontology (SSAO).

The most important concept is *Attack*, which represents an event that is recognized as malicious using the Explainable Firewall Penetration Testing Engine and classified into the concrete subclass of the *Attack* using multiclass ML classification. The concept of *Feature* represents the most significant features, determined by using Shapley Importance analysis, which classifies the current malicious event into the concrete class of attacks. Each type of potential attack has specific mitigation techniques, which are described in vulnerabilities databases. The concept of *MitigationTechnique* represents relations between various attack classes and the entities in the vulnerabilities databases, which provide additional information and mitigation techniques.

The list of all object relations among the added concepts is presented in Table 2.

Table 2. List of the object relationships in the extended part of SSAO.

Subject	Forward Relationship	Reverse Relationship	Object
Attack	hasMitigationTechnique	mitigates	MitigationTechnique
Attack	isDescribedUsing	descibesAttack	AttackInfo
Attack	isDefinedBy	definesAttack	Feature
MitigationTechnique	isDescibedIn	describesMitigation	MitigationInfo

The model's development process is depicted in Figure 5.

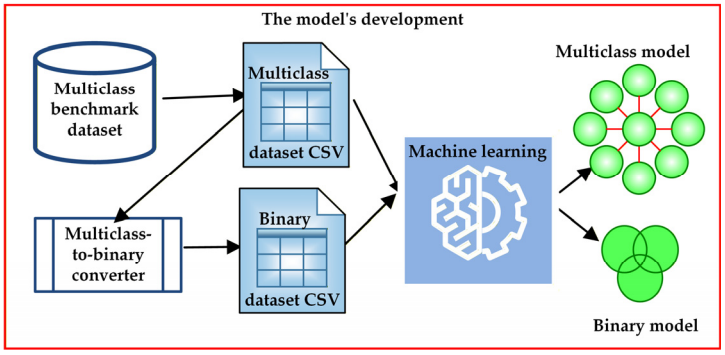

Figure 5. The model's development process.

A multiclass benchmark dataset is used to develop multiclass and binary models. Using the multiclass-to-binary converter, all multiclass benchmark dataset labels are converted to "benign" (0) and "malicious" (1). As a result, we have two datasets: the original multiclass benchmark dataset and the binary dataset, which was created by binarizing the original dataset's labels. Machine learning algorithms are employed to develop models from those datasets, producing multiclass and binary models accordingly.

The simplified pseudo-code (see Algorithm 1) of the EFPT:

Algorithm 1. The Pseudo-Code of the EFPT Engine

Input:
The network packets passed the firewall, the binary model, and the multiclass model.
Workflow:

1. The feature extractor analyzes the network packets passed through the firewall, extracts features, and produces the feature table for testing.
2. The binary model is used to test the feature table.
3. Proceed to Step 1 if an attack is not identified and to Step 4 if an attack is identified.
4. The feature table is tested for attack type prediction using the multiclass model.
5. By combining the binary model, the multiclass model, and the explanatory knowledge base, an explainable AI engine produced in-depth explanations that comprise:
    a. A detailed description of the identified attack type;
    b. Explanation of the rationale behind the execution of that attack;
    c. Advice on how to configure the firewall to effectively defend against that attack;
    d. Encouragement for performing additional testing after the firewall has been appropriately reconfigured.

Output:
In-depth explanation report
End the pseudo-code of the EFPT engine

Both the pretrained binary and multiclass ML models, as well as the network packets that were able to flow through the firewall, are used as inputs by the EFPT engine. The benchmark dataset is utilized in the process of training the models.

The feature extractor analyzes captured raw network packets, generates bidirectional flows, and extracts features from them. The first evaluation produces a binary classifier that determines whether a given packet is an attack or not. If an attack is not detected, the

feature extractor continues to analyze packets that pass the firewall. If an attack is detected, then multiclass classification is performed to more precisely recognize the attack type and provide results to explain vulnerabilities in firewalls and to propose alternative strategies for mitigating those vulnerabilities.

Following the execution of the EFPT engine, a report is generated that provides an explanation of the vulnerabilities that exist inside the firewall, as well as alternate solutions for mitigating those flaws.

The following section presents the experimental setup and the results.

4. Experimental Settings and Results

In this section, we provide the experimental setup and results obtained when evaluating the proposed method of firewall penetration testing using machine learning and explainable artificial intelligence.

4.1. Selecting Appropriate Benchmark Dataset

At the time of selecting the dataset for our experiments, we consider two primary considerations. In the first place, it is necessary to have access to the PCAP files of the network flow. To analyze the suggested method of firewall penetration testing, we need to rebuild the network flow for the firewall that is designed to prevent intrusion. Another requirement is that the dataset must be freely accessible and widely utilized in research projects. Various types of attacks (DoS, DDoS, botnet, infiltration, etc.) and realistic traffic volumes are featured in CIC-IDS2017, which is the most prominent benchmark for modern network intrusion detection systems (NIDS), as demonstrated in [25].

In light of the aforementioned considerations, the intrusion detection evaluation dataset CIC-IDS2017 [26] was chosen for the purpose of evaluating the suggested procedure for a firewall penetration test by utilizing machine learning and explainable artificial intelligence. Using the CICFlowMeter v. 4.0 software, which is freely accessible on the website of the Canadian Institute for Cybersecurity [27,28], the CIC-IDS2017 dataset has been completely labeled, and more than 80 network traffic features have been retrieved and calculated for all benign and intrusive flows. Network flow PCAP and CSV files for deep learning and machine learning are freely available to researchers. The list of attacks and the number of labeled records for each is presented in Table 3.

Table 3. CIC-IDS2017: the list of network attacks and the number of labeled records.

Network Attack Name	The Number of Labeled Records
BENIGN	2,273,097
Bot	1966
DDoS	128,027
DoS GoldenEye	10,293
DoS Hulk	231,073
DoS Slowhttptest	5499
DoS Slowloris	5796
FTP–PATATOR	7938
Heartbleed	11
Infiltration	36
PortScan	158,930
SSH–PATATOR	5897
WebAttack–BruteForce	1507
WebAttack–SQL Injection	21
WebAttack–XSS	652
Total	2,830,743

As a result of looking at Table 3, we may conclude that the CIC-IDS2017 dataset has a significant imbalance. Datasets for ML are typically balanced using a variety of approaches by researchers. Because real networks are highly imbalanced and contain a large number of harmless flows, we avoided explicit balancing efforts.

The full list of features in the CIC-IDS2017 dataset is presented in Table 4.

Table 4. The full list of features in the CIC-IDS2017 dataset.

No	Feature Name	No	Feature Name	No	Feature Name	No	Feature Name
1	FlowID	22	FlowPkts_s	43	FwdPkts_s	64	FwdBlkRateAvg
2	SrcIP	23	FlowIATMean	44	BwdPkts_s	65	BwdByts_bAvg
3	SrcPort	24	FlowIATStd	45	PktLenMin	66	BwdPkts_bAvg
4	DstIP	25	FlowIATMax	46	PktLenMax	67	BwdBlkRateAvg
5	DstPort	26	FlowIATMin	47	PktLenMean	68	SubflowFwdPkts
6	Protocol	27	FwdIATTot	48	PktLenStd	69	SubflowFwdByts
7	Timestamp	28	FwdIATMean	49	PktLenVar	70	SubflowBwdPkts
8	FlowDuration	29	FwdIATStd	50	FINFlagCnt	71	SubflowBwdByts
9	TotFwdPkts	30	FwdIATMax	51	SYNFlagCnt	72	InitFwdWinByts
10	TotBwdPkts	31	FwdIATMin	52	RSTFlagCnt	73	InitBwdWinByts
11	TotLenFwdPkts	32	BwdIATTot	53	PSHFlagCnt	74	FwdActDataPkts
12	TotLenBwdPkts	33	BwdIATMean	54	ACKFlagCnt	75	FwdSegSizeMin
13	FwdPktLenMax	34	BwdIATStd	55	URGFlagCnt	76	ActiveMean
14	FwdPktLenMin	34	BwdIATMax	56	CWEFlagCount	77	ActiveStd
15	FwdPktLenMean	36	BwdIATMin	57	ECEFlagCnt	78	ActiveMax
16	FwdPktLenStd	37	FwdPSHFlags	58	Down_UpRatio	79	ActiveMin
17	BwdPktLenMax	38	BwdPSHFlags	59	PktSizeAvg	80	IdleMean
18	BwdPktLenMin	39	FwdURGFlags	60	FwdSegSizeAvg	81	IdleStd
19	BwdPktLenMean	40	BwdURGFlags	61	BwdSegSizeAvg	82	IdleMax
20	BwdPktLenStd	41	FwdHeaderLen	62	FwdByts_bAvg	83	IdleMin
21	FlowByts_s	42	BwdHeaderLen	63	FwdPkts_bAvg	84	Label

All 2,830,743 records of the CIC-IDS2017 dataset with attack type labels are combined for ML in one CSV file, which was used for model development. We omitted several features in our experiments: FlowID, SrcIP, DstIP, and Timestamp. These features were deleted when developing ML models since, while testing the firewall designed to prevent intrusions, their values in the network flow will differ and should be disregarded. To generate ML models, 80 features were used, 79 as predictors and one (Label) as the response.

4.2. The Model's Development Stage

For the development of artificial intelligence, MATLAB v. 25.2.0 is a strong and integrated environment. It provides specialized toolboxes for deep learning and machine learning for data preparation, model design, and training. Additionally, it permits deployment to an embedded system, which makes it a perfect choice for engineering applications. Various classifiers, such as FT, DT, SVM, ensemble models, and neural network classification models, are utilized in the Classification Learner application to train models to classify data. Such analysis is done to find the most effective model type. Tools for global and local

interpretability, such as LIME and Shapley values, can be utilized to gain an understanding of how the model utilizes predictors to create predictions.

The MATLAB classification leaner application, which provides access to a large number of classifiers, is utilized by us in order to generate models during the development stage of the model, as shown in Figure 5. At this stage, two ML models are developed: a binary model to determine whether an attack has occurred or not and a multiclass model to precisely classify and identify the name of the attack. We conducted experiments using various classification algorithms to determine the best classifier. All 2,830,743 records of the CIC-IDS2017 dataset were used to develop and select the best model. Table 5 displays the models validation accuracy.

Table 5. The model's validation accuracy.

Classifier	Validation Accuracy (%)		Classifier	Validation Accuracy (%)	
	Binary Model	Multiclass Model		Binary Model	Multiclass Model
Fine Tree	99.7	99.6	Narrow Neural Network	98.7	98.7
Medium Tree	98.3	96.5	Medium Neural Network	98.9	99.1
Coarse Tree	93.4	92.3	Wide Neural Network	99.4	99.4
Efficient Linear SVM	89.6	82.0	Bilayered Neural Network	98.8	98.4
Ensemble RUSBoosted Trees	98.5	69.3	Trilayered Neural Network	98.7	98.5

Based on the results presented in Table 5, the best-performing model created using the Fine Tree (FT) classifier with 99.7% binary model validation accuracy and 99.6% multiclass model validation accuracy was selected. The FT binary model validation confusion matrix is depicted in Figure 6.

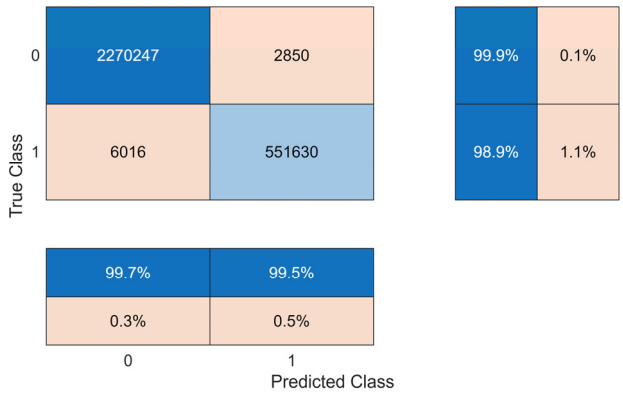

Figure 6. The binary model validation confusion matrix.

The FT model validation results presented in Figure 6 clearly show that the trained model can perform a very accurate classification in both classes: *Attack* and *Benign*. The most relevant is the second row in the matrix, which shows that the model will classify 98.9% of the true attacks' network flows as *Attack*, which is very important for recognizing malicious packets that can penetrate the firewall.

The multiclass FT model validation confusion matrix is depicted in Figure 7.

We employed a standard plot produced by the MATLAB function *confusionchart* to present the confusion matrix. The confusion matrix displays the total number of observations in each cell. The rows of the confusion matrix represent the actual class, while the columns denote the expected class. Diagonal cells represent accurately classified

observations, while off-diagonal cells denote inaccurately classified observations. A row-normalized summary presents the percentages of accurately and inaccurately identified observations for each actual class. For each projected class, the percentages of correctly and incorrectly identified observations are shown in a column-normalized summary.

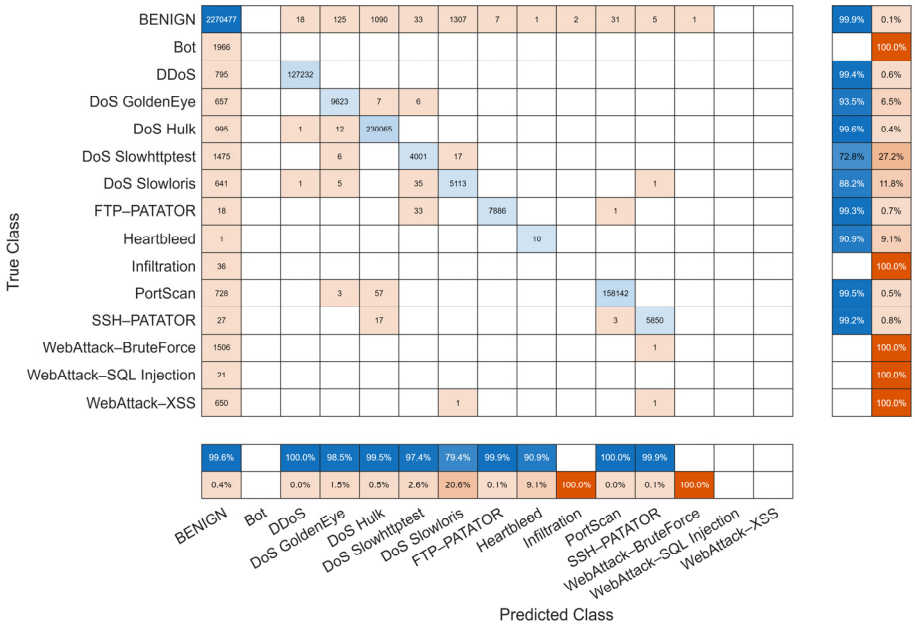

Figure 7. The multiclass validation confusion matrix.

4.3. Experimental Setup

Network flow data extracted from the packets passed through the firewall (see Figure 1) was needed to evaluate the proposed firewall penetration testing method. We rebuild the network architecture, which is identical to the one that was used for the compilation of the CIC-IDS2017 dataset, to evaluate whether the suggested method is able to detect attacks that have passed through the firewall and identify the type of attack. In order to accomplish this goal, the experimental network that is shown in Figure 8 was utilized.

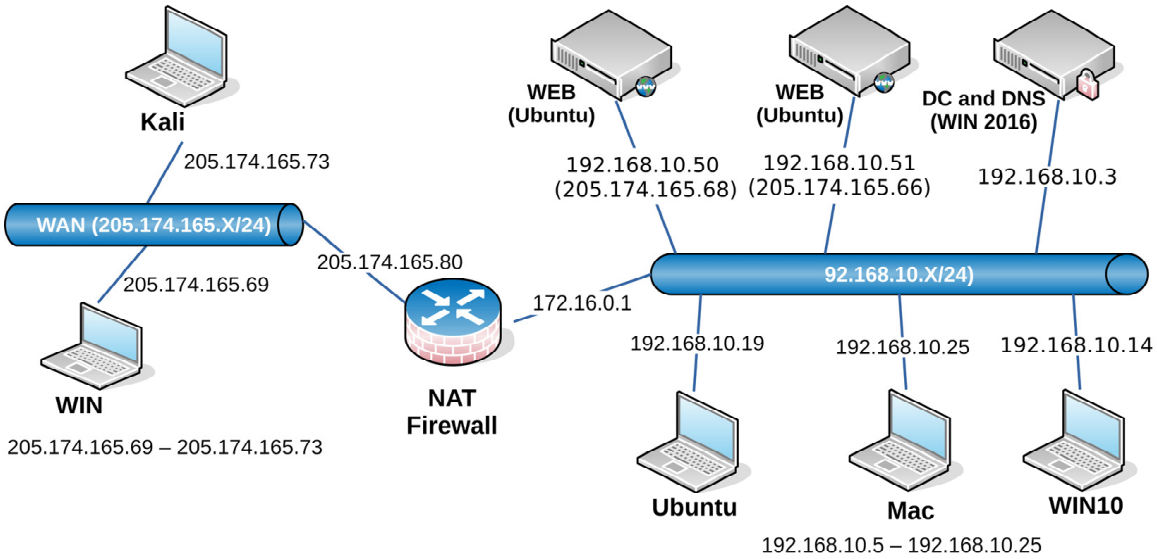

Figure 8. Experimental enterprise network used for firewall penetration testing.

Comprehensive network architecture encompasses router, firewall, and a range of operating systems, including Windows, Ubuntu, and Mac OS. The components of the experimental network are presented in Table 6.

Table 6. The components of the experimental network.

Domain	Network Component	WAN IP Address	LAN IP Address
–	NAT router/firewall	205.174.165.80	172.16.0.1
DC and DNS server	Windows server 2016	–	192.168.10.3
Outsider attackers' network	Kali Linux	205.174.165.73	172.16.0.1
	Windows	205.174.165.69	172.16.0.1
	Windows	205.174.165.70	172.16.0.1
	Windows	205.174.165.71	172.16.0.1
Insider victims' network	Public Web server Ubuntu	205.174.165.68	192.168.10.50
	Public Web server Ubuntu	205.174.165.66	192.168.10.51
	Ubuntu 14.4, 32B	–	192.168.10.19
	Ubuntu 14.4, 64B	–	192.168.10.17
	Ubuntu 16.4, 32B	–	192.168.10.16
	Ubuntu 16.4, 64B	–	192.168.10.12
	Windows 7, 64B	–	192.168.10.9
	Windows 8.1, 64B	–	192.168.10.5
	Windows Vista, 64B	–	192.168.10.8
	Windows 10, 32B	–	192.168.10.14
	Windows 10, 64B	–	192.168.10.15
	Mac OS	–	192.168.10.25

We analyzed all network flows in the original dataset and recreated the network's architecture, as well as hosts on it, for the purposes of the experiment. In this way, we were able to read the original packets from the PCAP files from the CIC-IDS2017 dataset, pass them through the firewall, and capture filtered packets inside the LAN network.

Snort engine [29] was employed to act as an enterprise firewall. Although Snort is an open-source intrusion detection and prevention system (IDS/IPS) that provides network traffic analysis and packet logging capabilities, it could also be used as a simple packet-filtering firewall and packet logger. Snort was used in inline mode and was acting as an IPS, allowing passing or dropping network packets according to the predefined rules. The Default Deny firewall policy with the exceptions specified as Snort rules, which are presented in Listing 1, were used for the firewall.

All firewall rules were deduced from the analysis of the network flows and original PCAP files provided with the CIC-IDS2017 dataset. We believe that these rules are plausible in the real enterprise, with daily network activities visible in the PCAP files, and would allow the users of such an enterprise to perform their daily tasks. The network packets that passed through the firewall (the LAN traffic) were collected and saved into PCAP files. The LAN network flows were extracted from the captured PCAP files using the CICFlowMeter software developed by the authors of the original CIC-IDS2017 dataset. Then we used MATLAB script to classify extracted LAN network flows. This script classifies LAN network flows mainly based on the values of the FlowID fields, which in most cases are the same between original network flows and new LAN network flows. The rest of the data was classified using heuristic analysis based on port numbers, IP addresses, time stamps, etc. The remaining few LAN network flows were classified manually. Results presented in Table 7 were achieved after the classification step.

Listing 1. Snort rules used in network firewall under penetration testing

```
portvar ALL_HTTP [8000:8888,80]

ipvar LAN_NET [192.168.10.0/24]
ipvar WAN_NET !$LAN_NET
ipvar SERVERS [192.168.10.50,192.168.10.51]

#HTTP
pass tcp $LAN_NET any <> $WAN_NET $ALL_HTTP (msg:“Outbound HTTP trfk”; sid:60000010; rev:1; GID:1;)
pass tcp $WAN_NET any <> $SERVERS 80 (msg:“Inbound HTTP traffic”; sid:60000011; rev:1; GID:2;)
#HTTPs
pass tcp $LAN_NET any <> $WAN_NET 443 (msg:“Outbound HTTPs traffic”; sid:60000013; rev:1; GID:3;)
pass tcp $WAN_NET any <> $SERVERS 443 (msg:“Inbound HTTP traffic”; sid:60000014; rev:1; GID:4;)
# SSH
pass tcp $LAN_NET any <> $WAN_NET 22 (msg:“Outbound SSH traffic”; sid:60000004; rev:1; GID:1;)
pass tcp $WAN_NET any <> $SERVERS 22 (msg:“Inbound SSH traffic”; sid:60000005; rev:1; GID:2;)
#FTP
pass tcp $LAN_NET any <> $WAN_NET 21 (msg:“Outbound FTP traffic”; sid:60000006; rev:1; GID:3;)
pass tcp $WAN_NET any <> $SERVERS 21 (msg:“Inbound FTP traffic”; sid:60000007; rev:1; GID:4;)
#DNS
pass tcp $LAN_NET any <> $WAN_NET 53 (msg:“Outbound DNS traffic TCP”; sid:60000008; rev:1; GID:5;)
pass udp $LAN_NET any <> $WAN_NET 53 (msg:“Outbound DNS traffic UDP”; sid:60000009; rev:1; GID:6;)
#NTP
pass udp $LAN_NET any <> $WAN_NET 123 (msg:“Outbound NTO traffic”; sid:60000012; rev:1; GID:1;)
#SMTP
pass tcp $LAN_NET any <> $WAN_NET 465 (msg:“Outbound SMTPs traffic”; sid:60000015; rev:1; GID:1;)
```

Table 7. The list of labeled records extracted from the LAN flows passed through the firewall.

The Number of Labeled Records	Network Attack Name
1,166,760	BENIGN
1966	Bot
96,190	DDoS
10,300	DoS GoldenEye
172,921	DoS Hulk
5498	DoS Slowhttptest
5796	DoS Slowloris
7938	FTP–PATATOR
0	Heartbleed
0	Infiltration
1047	PortScan
5897	SSH–PATATOR
1507	WebAttack–BruteForce
21	WebAttack–SQL Injection
653	WebAttack–XSS
1,476,494	Total

When comparing Tables 3 and 7, one can see that about half of all network flows were filtered by the firewall. Filtered flows mainly include traffic between different hosts inside the LAN, i.e., traffic from Windows workstations to Windows 2016 server, local web traffic, etc. Some of the attacks were completely blocked by the firewall; these include Heartbleed

and Infiltration attacks. On the other hand, our improvised stateless packet-filtering firewall allowed us to pass network packets containing the rest of the attacks.

4.4. Firewall Vulnerability Detection

To perform firewall penetration testing and attack detection, the binary and multiclass models created during the model creation stage (see Section 4.2) were employed. At the firewall vulnerabilities detection stage, the firewall penetration test was evaluated using the testing dataset created during the attack simulation, as shown in Section 4.3. All 1,476,494 records obtained simulating a network attack (see Section 4.3 and Table 7) were used. Table 8 displays the firewall penetration test accuracy.

Table 8. Firewall penetration test accuracy.

Classifier	Test Accuracy (%)		Classifier	Test Accuracy (%)	
	Binary Model	Multiclass Model		Binary Model	Multiclass Model
Fine Tree	82.1	78.7	Narrow Neural Network	79.1	75.7
Medium Tree	78.9	73.1	Medium Neural Network	74.1	66.9
Coarse Tree	71.4	78.2	Wide Neural Network	70.9	77.1
Efficient Linear SVM	81.8	76.8	Bilayered Neural Network	78.3	75.3
Ensemble RUSBoosted Trees	78.7	63.6	Trilayered Neural Network	69.5	77.7

According to the results in Table 7, the best-performing models for the firewall penetration test are the binary FT model, exhibiting an 82.1% test accuracy, and the multiclass FT model, demonstrating a 78.7% test accuracy. Those models were selected for further experiments.

Figure 9 illustrates the confusion matrix for the firewall penetration test employing the FT binary model.

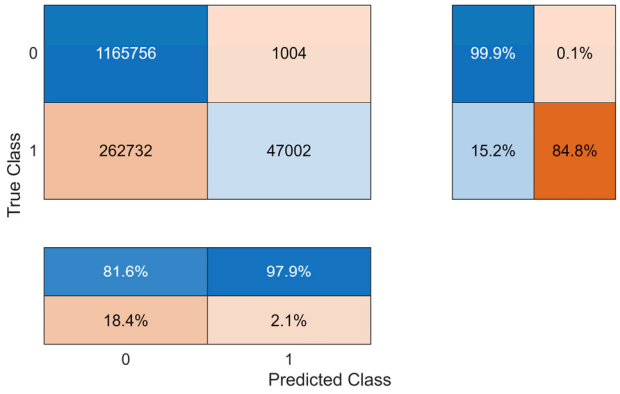

Figure 9. The confusion matrix for the firewall penetration test employing the FT binary model.

A closer analysis of the confusion matrix for the firewall penetration test employing the FT binary model shows that the model trained on original data from the CIC-IDS2017 dataset could be used for classifying network flows that pass the firewall and are captured in the LAN. The second column of the matrix shows that, of all network flows recognized as an attack, 97.9% are true attacks.

Figure 10 illustrates the confusion matrix for the firewall penetration test employing the FT multiclass model.

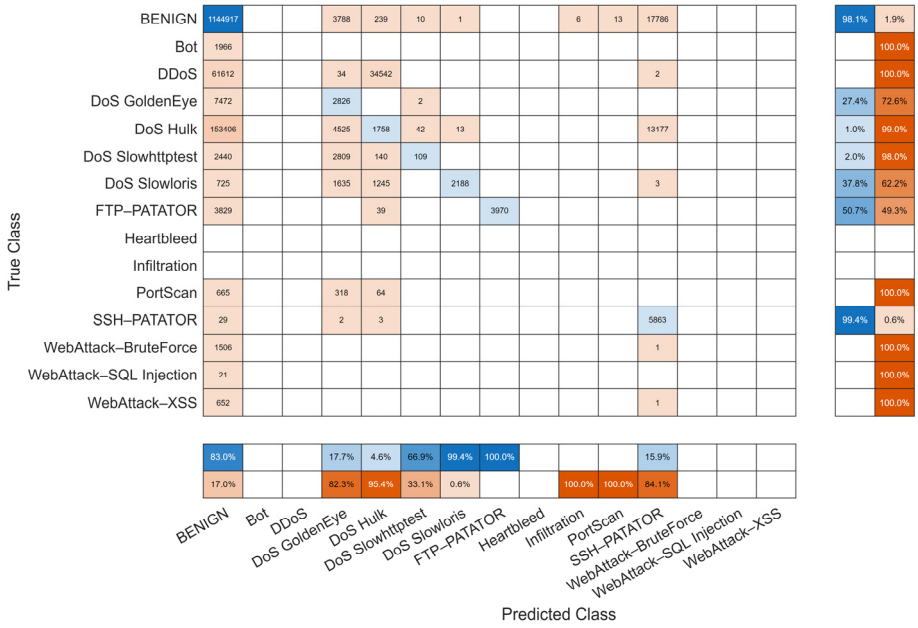

Figure 10. The confusion matrix for the firewall penetration test employing the FT multiclass model.

The rows of the confusion matrix represent the actual class, while the columns denote the expected class. A closer examination of the confusion matrix displayed in Figure 10 reveals that the model trained on the original CIC-IDS2017 dataset could be used for detecting concrete types of malicious network packets captured in the LAN. The overall accuracy of the classification is not as good as in the case of binary classification, but the best accuracy per se is not the main objective of this classification. The main goal is the detection of the types of malicious packets in order to have the situation further analyzed by the firewall security expert and find the real source of vulnerability.

4.5. The Explainability Stage

For the purpose of interpreting predictions, Lundberg and Lee [30] created a unified framework that they called SHAP, which stands for Shapley Additive Explanations. Through the process of computing the contribution of each feature to the prediction, it provides an explanation of the prediction of an instance x. The two approaches, LIME and Shapley Values, are linked together by this perspective. The amount of contribution that each individual feature in the model makes, whether it be positively or adversely, is measured by each SHAP value. Two significant advantages are provided by the SHAP value. To begin, the SHAP value can be computed for any model, contrary to the conventional linear models that are typically used. Secondly, its own unique set of SHAP values is assigned to each record.

Explanations that are locally interpretable and model-agnostic (LIME) were proposed in [31]. An interpretable model that is locally loyal to the classifier is the ultimate objective of the LIME algorithm, which seeks to identify an interpretable model over the interpretable representation.

Shapley's value, which was first presented by Shapley in [32], is a method that is utilized in the field of game theory to ascertain the extent to which every participant in a cooperative game has contributed to the overall success of the game. It is possible to apply this strategy to interpret the predictions made by machine learning [33]. By calculating the average contribution of a feature value to the prediction across all feasible coalitions, the Shapley value can be determined.

As shown in Figure 11, the Shapley importance was utilized in order to determine which predictors have the most (or least) average impact on the scores that were predicted.

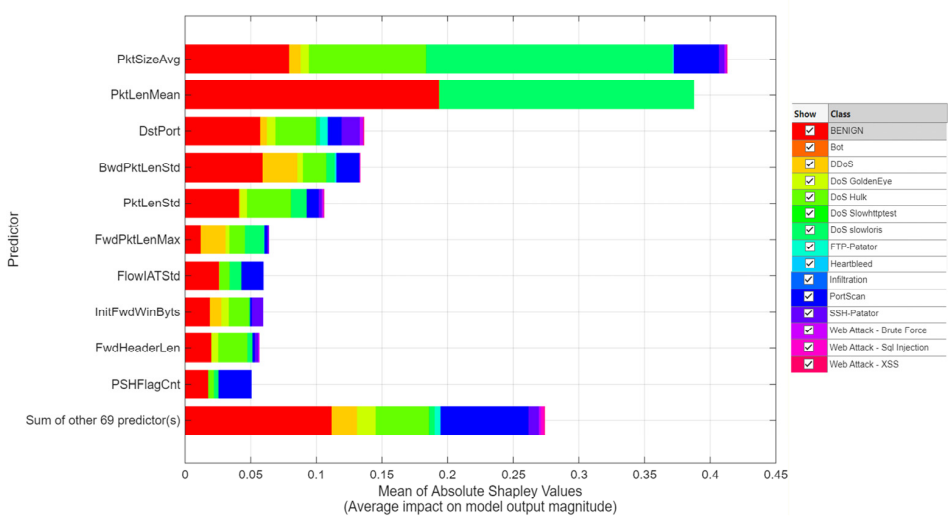

Figure 11. The Shapley importance for FT multiclass model predictors used in the firewall pentest.

As shown in Figure 12, the Shapley importance was utilized in order to determine which predictors have the most average impact on the scores that were predicted for the *WebAttack-XSS* in the firewall penetration test.

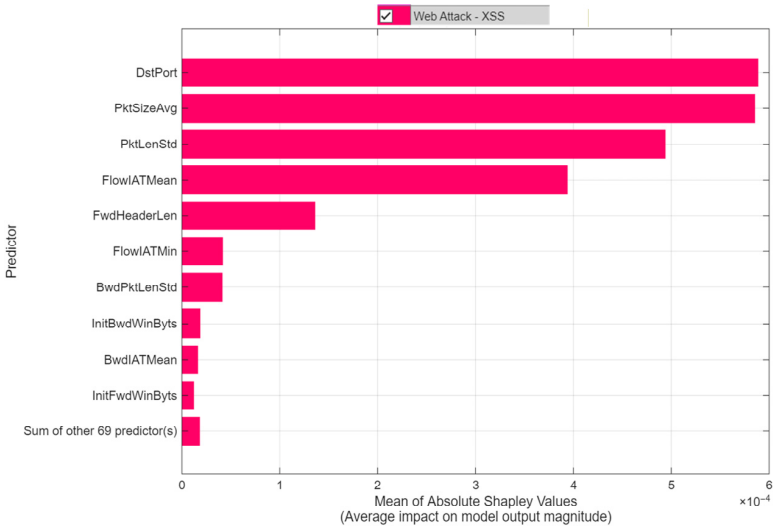

Figure 12. The Shapley importance for the *WebAttack-XSS* model predictors.

When employing SHAP, the only information that is provided is the names of the features, which has a substantial impact on the decision-making process. This is demonstrated in Figure 12, which shows that this is the case. Even though this is essential information for high-quality experts, it does not include any recommendations on how to reduce susceptibility. When only the names of the model's features are used, it is not possible to generate an explainable report that contains extensive information about the vulnerabilities of the firewall and the countermeasures that can be taken to configure the firewall for increased security.

The proposed extended SSAO can be explored for inferences using semantic query language, such as SPARQL. Analytical query activities like joining, sorting, aggregating, and filtering can be performed to extract useful information and relations from the ontologies. The example listing presented in Figure 13 shows a SPARQL query that allows to query

extended SSAO and extract potential mitigation techniques related to the concrete instance of XSS attack *CTI-Att-XSS-1*. The example results of this query are summarized in Table 9.

Snap SPARQL Query:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX cwe: <http://w3id.org/sepses/vocab/ref/cwe#>
PREFIX cti: <http://www.ktu.org/nerijus/ontologies/cti#>
PREFIX ssao: <http://ffrdc.ebiquity.umbc.edu/ns/ontology/#>

SELECT ?Attack ?Description (?CWESummary as ?Mitigation)
WHERE
{
?Attack          cti:hasMitigationTechnique ?MTech .
?MTech           cti:isDescibedIn           ?CWE .
?CWE             ssao:cweID                 ?CWEID .
?Description     ssao:cweID                 ?CWEID ;
                 ssao:cweSummary            ?CWESummary ;
                 rdf:type                   cwe:PotentialMitigation .
FILTER (?Attack=cti:CTI-Att-XSS-1) .
}
```

Figure 13. Example SPARQL query for extracting mitigation techniques related to the *CTI-Att-XSS-1* attack detected by EFPT.

Table 9. Example results of the query presented in Figure 13.

Attack	Description	Mitigation
cti:CTI-Att-XSS-1	cti:CWE-79-XSS-Encoding	Use and specify an output encoding that can be handled by the downstream component that is reading the output. Common . . .
cti:CTI-Att-XSS-1	cti:CWE-79-XSS-Parametrization	If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms . . .
cti:CTI-Att-XSS-1	cti:CWE-79-XSS-Validation	Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a list of acceptable inputs that strictly . . .

The example SPARQL query presented in Figure 14 and Table 10 allows the extraction of more detailed information related to the concrete instance, *CTI-Att-SQLi-1*, of the SQL injection attack.

Snap SPARQL Query:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX cti: <http://www.ktu.org/nerijus/ontologies/cti#>
PREFIX ssao: <http://ffrdc.ebiquity.umbc.edu/ns/ontology/#>

SELECT ?Attack (?Entity as ?Source) ?Summary
where
{
  ?Attack     cti:isDescribedUsing      ?Entity .
  {?Entity    ssao:cweSummary           ?Summary .}
  union
  {?Entity    cti:cveSummary            ?Summary .}
  FILTER (?Attack=cti:CTI-Att-SQLi-1) .
}
```

Figure 14. Example SPARQL query for extracting additional information related to the *CTI-Att-SQLi-1* attack detected by EFPT.

Table 10. Example results of the query presented in Figure 14.

Attack	Source	Summary
cti:CTI-Att-SQLi-1	cti:CWE-89-SQLi	The product constructs all or part of an SQL command using externally influenced input from an upstream component, but it does . . .
cti:CTI-Att-SQLi-1	cti:CVE-2025-7744-SQLi	Improper Neutralization of Special Elements used in a SQL Command ('SQL Injection') vulnerability in Dolusoft Omaspot allows . . .
cti:CTI-Att-SQLi-1	cti:CVE-2025-9926-SQLi	A vulnerability was determined in project worlds Travel Management System 1.0. Impacted is an unknown function . . .

5. Discussion

We propose a method for firewall penetration testing utilizing machine learning and explainable artificial intelligence (XAI). A penetration testing expert generates attack packets that are sent to the firewall. The explainable firewall penetration testing engine processes network packets that pass through the firewall, extracting features for classification. This involves a binary classifier to detect attacks, followed by a multiclass classifier to ascertain the type of attack and suggest mitigation strategies.

Additionally, an ontology-based approach is employed to enhance the explanation of vulnerabilities and mitigation techniques, expanding the System Security Assurance Ontology (SSAO) for this purpose. The extended ontology focuses on concepts such as Attack, Feature, and Mitigation Technique, integrating information from vulnerability databases. Finally, datasets for machine learning models are derived from a multiclass benchmark dataset, facilitating the development of both binary and multiclass classification models.

In selecting a dataset for firewall penetration testing experiments, we prioritized access to PCAP files for network flow analysis and the necessity for a dataset that is freely available and commonly used in research. The CIC-IDS2017 dataset, with its representation of various attack types (e.g., DoS, DDoS, and botnet), was selected as it features over 80 labeled network traffic attributes. Using the CICFlowMeter software, all data was labeled, and the best-performing model utilized the Fine Tree classifier, achieving high validation accuracy.

A recreated enterprise network mimicked the original CIC-IDS2017 structure, allowing the analysis of network flows via a Snort engine as a firewall. Snort's rules were based on the original dataset assessments. While about half of the network flows were filtered, including complete blocks of specific attacks, others were passed, indicating a balance between operational tasks and security. The proposed method also incorporated SHAP and LIME for interpretable predictions in machine learning, alongside semantic querying with SPARQL for extracting relevant mitigation techniques against identified threats.

The experimental results were compared with state-of-the-art research (SOTA), as presented in Table 11.

As we can conclude from Table 11, various SOTA approaches can be used for firewall penetration testing. The proposed method for firewall penetration testing allows testing of installed network firewalls for potential vulnerabilities. This method is not dependent on the type, technology, or implementation of the concrete firewall, as it generates malicious traffic, which is then collected in the network behind the firewall. The proposed approach does not require any special penetration testing tools, nor is it based on the high

qualifications of the tester. It solely relies on network attacks' datasets, which are widely available. The results of the penetration testing could then be explained not only using the industry-standard approaches, but also by providing more detailed, human-readable form of information on potential vulnerabilities for better understanding of the nature of the detected problem. Moreover, potential problem mitigation steps could also be included. This functionality is achieved by incorporating the extended SSAO in the process of the explanation.

Table 11. Experimental results compared with state-of-the-art research.

Research	Method	ML	Knowledge Presentation	XAI	Report for an Expert in the Field
Confido et al., 2022 [34]	RL	Deep Q-Learning	Not used	Markov decision processes	Not generated
Schiller et al., 2023 [35]	Particle swarm optimization, multi-agent penetration testing	Not used	Not used	Not used	Not generated
Fatima et al., 2024 [36]	XAI-based anomaly detection mechanism that classifies normal and anomalous data with the help of ML algorithms	RF, XGBoost, MLP-NN, DNN, and DT	Not used	LIME and SHAP	Not generated
Fathi et al., 2025 [37]	CoverTrack-DQN, RL	Deep Q-Networks	Not used	Only compared with solutions driven by generative AI	Not generated
Proposed	Explainability method that utilizes the list of the most significant features associated with the identified attack type, picked from the multiclass model prediction	10 ML models, binary and multiclass	The extended SSAO	Using SHAP-selected features populated into ontology to provide detailed explanation	Explain vulnerabilities in firewalls and provide alternative strategies for mitigating those vulnerabilities

The SHAP and LIME methods provide the ability to understand which features are dominant in the detected attack and make it possible to make certain decisions about the hardening of the firewall that is being tested. For example, the dominance of destination ports and concrete protocols may indicate the need to block that protocol in the firewall, but further evaluation of the situation by the expert of the field is required. Most of the works summarized in Table 11 stop here. On the other hand, the approach that has been proposed makes it possible to move forward and offers the capability to assist the expert in determining the level of risk and locate potential mitigation techniques for vulnerabilities that have been identified. This step is only possible when using the extended SSAO. No additional computation cost or increased latency is required, as this analysis is performed exclusively on demand from the end user and after the main vulnerabilities' detection phase is finished. The well-established ontology semantic querying language SPARQL offers sufficient performance for basic queries, which this case requires. The only consideration is the automatic collection and the updating of the SSAO with the new information that is constantly added to the CVE, CWE, and CAPEC databases. However, the scope of this paper does not encompass this problem.

6. Conclusions

Cyber adversaries are increasingly sophisticated, leading to complex security challenges as digital services grow. Firewalls are primary defenses, but traditional ones often fail against evolving threats. The rise in XAI addresses transparency issues in ML models,

crucial for informed cybersecurity decisions. Also, organizations need good penetration test reports to help them understand and fix vulnerabilities with customized mitigation options.

The study presents significant contributions in the domain of firewall security through the development of the explainable firewall penetration testing method. It introduces two machine learning classification models: a binary classification model to detect the presence of attacks and a multiclass classification model that identifies the attack type while incorporating an explainability feature.

Multiple ML algorithms were evaluated, namely FT, Medium Tree, Coarse Tree, Efficient Linear SVM, Ensemble RUSBoosted Trees, Narrow Neural Network, Medium Neural Network, Wide Neural Network, Bilayered Neural Network, and Trilayered Neural Network. After evaluating numerous ML algorithms for the full CIC-IDS2017 dataset, the FT model was considered to have the greatest performance. The binary model achieved a validation accuracy of 99.7%, while the multiclass model achieved a validation accuracy of 99.6%. Both models were used to test the firewall for vulnerabilities. Firewall penetration testing using the binary model achieved an accuracy of 82.1%, while the multiclass model achieved an accuracy of 78.7%.

The proposed method uses the extended SSAO to point out the weaknesses in firewalls during penetration testing and to provide different solutions to reduce risks in firewall setup. This contrasts with SHAP, which simply displays the feature names that had a significant impact on predictions.

Author Contributions: Conceptualization, A.V., J.T., and N.M.; methodology, A.V., J.T., and N.M.; software, N.M.; validation, J.T. and N.M.; formal analysis and investigation, A.V., J.T., and N.M.; data curation, N.M.; writing—original draft preparation, A.V., J.T., and N.M.; writing—review and editing, A.V., J.T., and N.M.; visualization, J.T. and N.M.; supervision, A.V. All authors have read and agreed to the published version of the manuscript.

Funding: This work was conducted as part of the execution of the project "Mission-driven Implementation of Science and Innovation Programs" (No. 02-002-P-0001), funded by the Economic Revitalization and Resilience Enhancement Plan "New Generation Lithuania".

Data Availability Statement: No new data were created.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADA	Artificial Discursive Agent
BDI	belief-desire-intention
CAPEC	Common Attack Pattern Enumeration and Classification
CPRF	class probability random forest
CSV	comma-separated values
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
DDoS	Distributed Denial of Service
DNN	deep neural network
DoS	Denial of Service
DT ML	decision tree machine learning
EBM	explainable boosting machines
EFPT	explainable firewall penetration testing
FT ML	Fine Tree Machine Learning
IDS	intrusion detection system
IoT	Internet of Things

KNN	K-Nearest Neighbors
LAN	Local Area Network
LIME	Local Interpretable Model-Agnostic Explanations
LLMs	Large Language Models
MT	Medium Tree
ML	machine learning
MLP	Multilayer Perceptron
NIDS	network intrusion detection systems
PCAP	Packet Capture
RF ML	Random Forest Machine Learning
SHAP	Shapley Additive Explanations
SSAO	System Security Assurance Ontology
SVM	support vector machine
UCO	Unified Cybersecurity Ontology
XAI	explainable artificial intelligence
XSS	Cross-Site Scripting

References

1. Sarker, K.U.; Yunus, F.; Deraman, A. Penetration Taxonomy: A Systematic Review on the Penetration Process, Framework, Standards, Tools, and Scoring Methods. *Sustainability* **2023**, *15*, 10471. [CrossRef]
2. Alhamed, M.; Rahman, M.M.H. A Systematic Literature Review on Penetration Testing in Networks: Future Research Directions. *Appl. Sci.* **2023**, *13*, 6986. [CrossRef]
3. Tiwari, A.; Papini, S.; Hemamalini, V. An enhanced optimization of parallel firewalls filtering rules for scalable high-speed networks. *Mater. Today Proceeedings* **2022**, *62*, 4800–4805. [CrossRef]
4. Chindrus, C.; Caruntu, C.-F. Machine Learning-Based Multilabel Classification for Web Application Firewalls: A Comparative Study. *Electronics* **2025**, *14*, 4172. [CrossRef]
5. Gaspar, D.; Silva, P.; Silva, C. Explainable AI for Intrusion Detection Systems: LIME and SHAP Applicability on Multi-Layer Perceptron. *IEEE Access* **2024**, *12*, 30164–30175. [CrossRef]
6. Galanska, K.; Kruzikova, A.; Murumaa, M.P.; Matyáš, V.; Just, M. From Reports to Actions: Bridging the Customer Usability Gap in Penetration Testing. *IEEE Access* **2025**, *13*, 73975–73986. [CrossRef]
7. Adam, H.M.; Widyawan; Putra, G.D. A Review of Penetration Testing Frameworks, Tools, and Application Areas. In Proceedings of the 7th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE), Purwokerto, Indonesia, 29–30 November 2023; pp. 319–324. [CrossRef]
8. Jayasuryapal, G.; Pranay, P.M.; Kaur, H.; Swati. A Survey on Network Penetration Testing. In Proceedings of the 2nd International Conference on Intelligent Engineering and Management (ICIEM), London, UK, 28–30 April 2021; pp. 373–378. [CrossRef]
9. Hu, Z.; Beuran, R.; Tan, Y. Automated Penetration Testing Using Deep Reinforcement Learning. In Proceedings of the IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Genoa, Italy, 16–18 June 2020; pp. 2–10. [CrossRef]
10. Qian, K.; Zhang, D.; Zhang, P.; Zhou, Z.; Chen, X.; Duan, S. Ontology and Reinforcement Learning Based Intelligent Agent Automatic Penetration Test. In Proceedings of the IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), Dalian, China, 28–30 June 2021; pp. 556–561. [CrossRef]
11. Raza, A.; Munir, K.; Almutairi, M.S.; Sehar, R. Novel Class Probability Features for Optimizing Network Attack Detection with Machine Learning. *IEEE Access* **2023**, *11*, 98685–98694. [CrossRef]
12. Roshan, K.; Zafar, A. Using Kernel SHAP XAI Method to Optimize the Network Anomaly Detection Model. In Proceedings of the 9th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 23–25 March 2022; pp. 74–80. [CrossRef]
13. Patil, S.; Varadarajan, V.; Mazhar, S.M.; Sahibzada, A.; Ahmed, N.; Sinha, O.; Kumar, S.; Shaw, K.; Kotecha, K. Explainable Artificial Intelligence for Intrusion Detection System. *Electronics* **2022**, *11*, 3079. [CrossRef]
14. Chen, M.; Ma, B.; Jiang, H. Research on Intrusion Detection Based on Interpretable Machine Learning. In *Proceedings of the 2025 International Conference on Machine Learning and Neural Networks (MLNN '25)*; Association for Computing Machinery: New York, NY, USA, 2025; pp. 109–114. [CrossRef]
15. Ikram, I.; Huma, Z. An Explainable AI Approach to Intrusion Detection Using Interpretable Machine Learning Models. *EuroVantage J. Artif. Intell.* **2024**, *1*, 57–66.
16. Ahmed, U.; Jiangbin, Z.; Almogren, A.; Khan, S.; Sadiq, M.T.; Altameem, A.; Rehman, A.U. Explainable AI-based innovative hybrid ensemble model for intrusion detection. *J. Cloud Comput.* **2024**, *13*, 150. [CrossRef]

17. Arreche, O.; Guntur, T.; Abdallah, M. XAI-IDS: Toward Proposing an Explainable Artificial Intelligence Framework for Enhancing Network Intrusion Detection Systems. *Appl. Sci.* **2024**, *14*, 4170. [CrossRef]
18. Wang, J.; Ni, T.; Lee, W.B.; Zhao, O. A Contemporary Survey of Large Language Model Assisted Program Analysis. *arXiv* **2025**, arXiv:2502.18474. [CrossRef]
19. Jaffal, N.O.; Alkhanafseh, M.; Mohaisen, D. Large Language Models in Cybersecurity: A Survey of Applications, Vulnerabilities, and Defense Techniques. *AI* **2025**, *6*, 216. [CrossRef]
20. Wang, Y.; Zhao, B.; Li, W.; Zhu, L. An Ontology-Centric Approach for Network Security Situation Awareness. In Proceedings of the 2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC), Torino, Italy, 26–30 June 2023; pp. 777–787.
21. Syed, Z.; Padia, A.; Finin, T.W.; Mathews, M.L.; Joshi, A. UCO: A Unified Cybersecurity Ontology. In Proceedings of the AAAI Workshop: Artificial Intelligence for Cyber Security, Phoenix, AZ, USA, 12–13 February 2016.
22. Lim, J.; Lau, Y.L.; Ming Chan, L.K.; Tristan Paul Goo, J.M.; Zhang, H.; Zhang, Z.; Guo, H. CVE Records of Known Exploited Vulnerabilities. In Proceedings of the 2023 8th International Conference on Computer and Communication Systems (ICCCS), Guangzhou, China, 21–24 April 2023; pp. 738–743.
23. The MITRE Corporation. CWE-Common Weakness Enumeration. Available online: https://cwe.mitre.org (accessed on 6 January 2026).
24. The MITRE Corporation. CAPEC-Common Attack Pattern Enumeration and Classification. Available online: https://capec.mitre.org (accessed on 6 January 2026).
25. Anis, F.M.; Alabdullatif, M.; Aljbli, S.; Hammoudeh, M. A Survey on the Applications of Deep Learning in Network Intrusion Detection Systems to Enhance Network Security. *IEEE Access* **2025**, *13*, 185357–185373. [CrossRef]
26. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. 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, Portugal, 22–24 January 2018.
27. Canadian Institute for Cybersecurity. Applications. Available online: https://www.unb.ca/cic/research/applications.html#CICFlowMeter (accessed on 5 January 2026).
28. CICFlowmeter-V4.0 (Formerly Known as ISCXFlowMeter) Is a Network Traffic Bi-Flow Generator and Analyser for Anomaly Detection. Available online: https://github.com/ISCX/CICFlowMeter (accessed on 5 January 2026).
29. Naldi, L.D.; Siswanto, A. Design and Implement of Intrusion Prevention System Based on Snort and IP Tables. *J. Comput. Res. Innov.* **2025**, *10*, 89–97. [CrossRef]
30. Lundberg, S.M.; Lee, S.-I. A unified approach to interpreting model predictions. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 4765–4774.
31. Ribeiro, M.T.; Singh, S.; Guestrin, C. "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 (KDD '16)*; Association for Computing Machinery: New York, NY, USA, 2016; pp. 1135–1144. [CrossRef]
32. Shapley, L.S. A value for n-person games. *Contrib. Theory Games* **1953**, *2*, 307–317.
33. Wang, M.; Zheng, K.; Yang, Y.; Wang, X. An Explainable Machine Learning Framework for Intrusion Detection Systems. *IEEE Access* **2020**, *8*, 73127–73141. [CrossRef]
34. Confido, A.; Ntagiou, E.V.; Wallum, M. Reinforcing Penetration Testing Using AI. In Proceedings of the IEEE Aerospace Conference (AERO), Big Sky, MT, USA, 5–12 March 2022; pp. 1–15. [CrossRef]
35. Schiller, T.; Caulkins, B.; Wu, A.S.; Mondesire, S. Security Awareness in Smart Homes and Internet of Things Networks through Swarm-Based Cybersecurity Penetration Testing. *Information* **2023**, *14*, 536. [CrossRef]
36. Fatima, Z.; Hussain, R.; Dilshad, A.; Shakir, M.; Laghari, A.A. Explainable AI-Driven Firewall Evaluation: Empowering Cybersecurity Decision-Making for Optimal Network Defense. *Secur. Priv.* **2025**, *8*, e70026. [CrossRef]
37. Fathi, M.K.; Metwally, K.; Sobh, M.; Eldin, A.B. Covertrack-Dqn: Enhancing Penetration Testing Efficiency with Deep Q-Networks for Trace Covering. In Proceedings of the International Telecommunications Conference (ITC-Egypt), Cairo, Egypt, 28–31 July 2025; pp. 681–686. [CrossRef]