

Research on APT Attack Behavior Tracing Method Based on Deep Neural Network

Mengyuan Tao, Shuo Yang, Fangru Duan, Dong Zhu, Guliasiman Abudukadier, Aobo Zhang

Xinjiang College of Science & Technology, Korla, Xinjiang, 841000, China

Abstract: Advanced Persistent Threat (APT) attack tracing aims to identify attack behavior patterns and trace the attack source, which is a key step in security operations and incident response. To address the difficulties of multi-type attack behavior identification and severe class imbalance, this paper proposes an APT attack behavior tracing method based on a Deep Neural Network (DNN). Based on the public UNSW-NB15 network traffic dataset, network behaviors are divided into 10 categories, namely normal behavior and 9 types of attack behaviors. Through data cleaning, categorical feature encoding, and numerical standardization, a 70-dimensional feature sample set is constructed, and a multilayer perceptron network is employed to perform fine-grained classification of attack behaviors; the identification results are then mapped to APT kill-chain stages to assist attacker profiling and attribution. The proposed method is compared with XGBoost, linear support vector machine, AdaBoost, and random forest. Experimental results show that XGBoost achieves the best macro-averaged F1 of 0.5160, while the proposed DNN ranks second with a macro-averaged F1 of 0.4600; both significantly outperform the linear model and AdaBoost. Per-class analysis reveals that majority attack behaviors are identified with high accuracy, whereas minority classes such as Analysis and Backdoor are difficult to recognize, indicating that class imbalance is the main factor limiting tracing performance. A further oversampling ablation experiment shows that simple resampling cannot effectively improve minority-class identification, and more sophisticated cost-sensitive or semantic enhancement strategies are needed. This study provides a reproducible experimental reference for APT attack behavior tracing and attribution assistance.

Keywords: Advanced Persistent Threat; Attack Tracing; Deep Neural Network; Multi-class Classification; Class Imbalance; Network Traffic.

1. Introduction

Advanced Persistent Threats (APTs) are carried out by organized attackers and are characterized by well-defined objectives, long attack cycles, and strong stealth, posing a broad threat to governments, national defense, and critical infrastructure. The Cyber Kill Chain model [1] describes the phased progression of such attacks from reconnaissance and weaponization to exploitation, installation, and exfiltration. Compared with detecting whether an attack has occurred, attack tracing aims to answer where the attack comes from, what behavior patterns the attacker adopts, and how the attack chain advances, which is a prerequisite for incident response, forensics, and defense decision-making. According to the granularity of tracing, APT tracing can be divided into attacker attribution and attack behavior tracing, and a comprehensive review of APT tracing and reasoning methods is given in Ref. [2]. Since attackers often hide their true identities through proxies, relay nodes, and forged information, attacker attribution is highly uncertain, whereas attack behavior tracing, which relies on observable behavioral evidence, is more feasible and can support attribution[2].

Network traffic is one of the most direct observable carriers of APT attack behaviors. During the reconnaissance, exploitation, persistence, and data exfiltration stages, attackers leave traces at the traffic level, such as abnormal scanning, exploit payloads, C2 communication, and covert channels. Marchetti et al.[3] proposed an APT detection method for high-volume network traffic. Machine learning methods have been widely applied to APT detection and tracing by learning discriminative patterns from traffic or log data[4].

However, attack behavior tracing faces two prominent

problems. First, attack behaviors are diverse and intertwined; the same attacker exhibits different behavior patterns at different stages, so fine-grained multi-class identification is significantly more difficult than binary detection. Second, the sample distribution across attack behavior categories in real attack scenarios is extremely imbalanced; key behaviors such as backdoor persistence and worm propagation are scarce, and conventional models tend to misclassify minority samples into majority classes, making the critical stages difficult to trace. Deep Neural Networks (DNNs) possess strong nonlinear feature representation ability and have been widely used in intrusion detection, e.g., multi-stage autoencoder-based APT detection[5]. Provenance-graph-based methods[6] provide another path for attack chain reconstruction. In this paper, the public UNSW-NB15 dataset[7,8], which contains fine-grained attack category labels and is widely used in intrusion detection research, is adopted as the experimental benchmark.

To address the above problems, this paper proposes an APT attack behavior tracing method based on a DNN. The main contributions are as follows: (1) attack tracing is modeled as a fine-grained multi-class identification task of attack behaviors, and a mapping from attack behavior types to APT kill-chain stages is established to provide stage-level semantics for tracing results; (2) a multilayer perceptron-based tracing model is designed and systematically compared with four representative methods, namely XGBoost, linear support vector machine, AdaBoost, and random forest; (3) per-class performance analysis and an oversampling ablation experiment are conducted to reveal the difficulty of minority-class tracing and to point out directions for further improvement.

2. Data Source and Task Modeling

2.1. Dataset and Tracing Task Modeling

This paper adopts the public UNSW-NB15 dataset[7,8], created by the Cyber Range Lab of the University of New South Wales Canberra, which simulates modern normal network behaviors and various synthetic attack behaviors and contains 45-dimensional network flow features and fine-grained attack category labels. Following the official partition, the training set contains 175,341 records and the test set contains 82,332 records.

The attack behavior tracing task is modeled as a 10-class fine-grained classification: normal behavior (Normal) and 9 types of attack behaviors. According to the phase division of the APT Cyber Kill Chain[1], each attack behavior type is mapped to the corresponding kill-chain stage, as shown in Table 1, so that the classification results can be converted into interpretable tracing conclusions.

Table 1. Data category distribution and APT kill-chain stage mapping

Category	Training samples	Proportion/%	Kill-chain stage
Normal	56,000	31.94	Normal behavior
Generic	40,000	22.81	Attack behavior (generic)
Exploits	33,393	19.04	Exploitation
Fuzzers	18,184	10.37	Probing / fuzzing
DoS	12,264	6.99	Denial of service
Reconnaissance	10,491	5.98	Reconnaissance
Analysis	2,000	1.14	Analysis probing
Backdoor	1,746	1.00	Persistence
Shellcode	1,133	0.65	Delivery / exploitation
Worms	130	0.07	Propagation / spread
Total	175,341	100.00	—

Table 1 shows that the sample sizes of the categories differ dramatically: Normal and Generic together account for more than 54%, whereas Worms accounts for only 0.07%, and Backdoor and Shellcode are both below 1.5%. The persistence (Backdoor) and propagation (Worms) stages, which are the most threatening and most in need of tracing in APT attacks, have the fewest samples, so the class imbalance problem has a prominent impact on the tracing task.

2.2. Data Preprocessing

The preprocessing procedure is as follows: (1) delete the id and label columns irrelevant to modeling; (2) unify the placeholder “-” in the categorical feature service into “none” and check for missing values; (3) for the high-cardinality categorical feature proto (originally 133 categories), retain the top 8 protocol types by frequency and merge the rest into other, then apply one-hot encoding to the three categorical features proto, service, and state, generating 31 binary features; (4) apply Z-score standardization to the 39 numerical features. After preprocessing, each sample is represented by a 70-dimensional feature vector, and the training and test sets share the same encoding and standardization parameters.

3. Proposed Tracing Method Based on Deep Neural Network

3.1. Overall Framework

The overall framework of the proposed method is shown in Fig. 1: after preprocessing and feature encoding, the network traffic data are divided into training and test sets; a 10-class deep neural network classifier is trained on the training set; in the testing stage, the attack behavior types output by the model are mapped to APT kill-chain stages to form tracing conclusions.

Framework of the Proposed APT Behavior Tracing Method

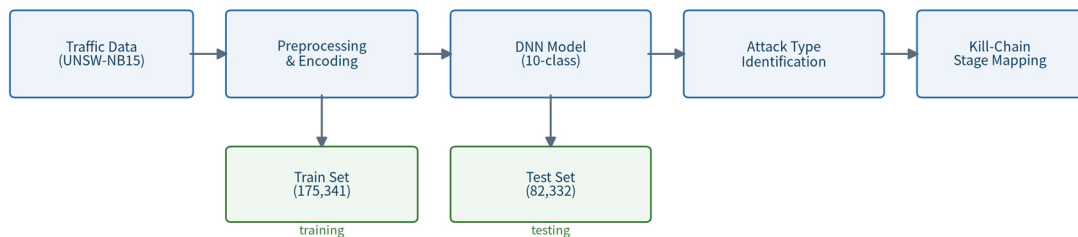


Fig 1. Overall framework of the proposed APT attack behavior tracing method based on deep neural network

3.2. Deep Neural Network Model

A multilayer perceptron (MLP) is adopted as the deep neural network implementation. The network structure is input layer (70 dimensions)—hidden layer 256—hidden layer 128—hidden layer 64—output layer (10-class Softmax). The hidden layers use the ReLU activation function, and L2 regularization (regularization coefficient 1×10^{-4}) is applied to suppress overfitting. Compared with shallow models, the multi-layer nonlinear structure can capture complex interactions among features and is suitable for attack behavior type discrimination[5].

3.3. Training Strategy

The model is trained with the Adam optimizer, with an initial learning rate of 1×10^{-3} , a batch size of 512, and a maximum of 60 iterations; 10% of the training set is held out as a validation set, and early stopping is adopted (training terminates when the validation loss does not improve for 5 consecutive iterations). The random seed is set to 42 to ensure reproducibility.

3.4. Imbalance Handling Scheme

To evaluate the impact of class imbalance, two training schemes are designed: (1) the baseline scheme, which directly

uses the training set with the original distribution; (2) the oversampling scheme, in which minority classes with fewer than 20,000 samples are expanded to 20,000 samples via repeated sampling, so that the proportions of majority and minority classes become relatively balanced. The two schemes are trained with the same network structure and hyperparameters, and the comparison reveals the actual effect of resampling strategies on tracing performance.

4. Experiments and Results Analysis

4.1. Experimental Setup

The experiments are implemented with Python 3.12 and scikit-learn 1.4, and the DNN model is implemented with the

scikit-learn multilayer perceptron. The compared methods include XGBoost (200 trees, maximum depth 8), linear support vector machine (solved by SGD), AdaBoost (60 weak learners), and random forest (200 trees, maximum depth 24). Evaluation metrics include accuracy, macro-averaged precision, macro-averaged recall, macro-averaged F1, weighted F1, and training time; among them, the macro-averaged metrics weight each class equally and better reflect the overall performance under imbalance.

4.2. Overall Performance Comparison

Table 2 presents the overall performance of each model on the test set.

Table 2. Overall performance comparison of different models

Model	Accuracy/%	Macro-Precision/%	Macro-Recall/%	Macro-F1/%	Weighted-F1/%	Training time/s
DNN	76.27	51.33	48.38	46.00	76.80	34.1
XGBoost	76.77	54.70	55.48	51.60	78.15	20.8
Linear SVM	69.91	34.58	38.32	35.11	71.61	1.9
AdaBoost	59.93	28.05	28.38	25.73	60.52	21.4
Random Forest	75.86	53.53	49.60	46.95	77.78	13.4

It can be seen from Table 2 that: (1) XGBoost achieves the best macro-averaged F1 (51.60%) and macro-averaged recall (55.48%), and the proposed DNN ranks second with a macro-averaged F1 of 46.00%; both significantly outperform the linear support vector machine and AdaBoost; (2) random forest performs close to DNN (macro-F1 46.95%), indicating that tree ensembles and deep networks have comparable expressive ability in this task; (3) the linear support vector machine is limited by its linear decision boundary, with a macro-averaged F1 of only 35.11%; (4) AdaBoost is sensitive to imbalance and noise, and most minority classes are ignored,

giving the lowest macro-averaged F1 (25.73%). Overall, fine-grained multi-class identification of attack behaviors is considerably more difficult than binary detection: the macro-averaged F1 varies between 0.26 and 0.52, and class imbalance is the key limiting factor.

4.3. Per-Class Performance Analysis

To further analyze the tracing ability of each category, Fig. 2 compares the per-class F1 of the five models, and Table 3 lists the per-class F1 values of representative models.

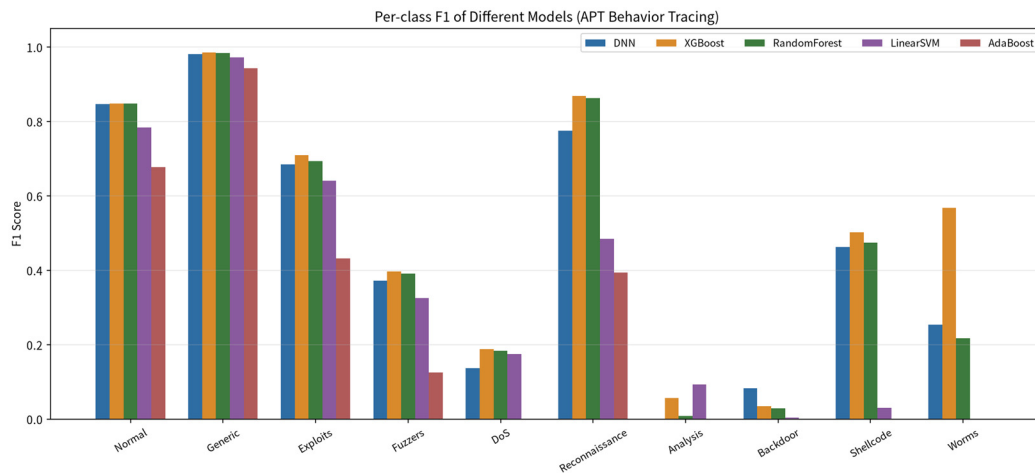


Fig 2. Per-class F1 comparison of different models

Table 3. Per-class F1 of representative models

Category	DNN	XGBoost	Random Forest
Normal	0.8463	0.8480	0.8487
Generic	0.9812	0.9850	0.9838
Exploits	0.6856	0.7091	0.6935
Fuzzers	0.3728	0.3968	0.3911
DoS	0.1379	0.1886	0.1833
Reconnaissance	0.7757	0.8688	0.8636
Analysis	0.0000	0.0568	0.0089
Backdoor	0.0833	0.0357	0.0288
Shellcode	0.4622	0.5029	0.4750
Worms	0.2545	0.5679	0.2182

It can be seen from Table 3 and Fig. 2 that: (1) categories with larger sample sizes, such as Normal, Generic, Exploits, and Reconnaissance, are identified with high accuracy (F1 above 0.69), with Generic reaching 0.98, indicating that most attack behaviors can be effectively traced; (2) the Analysis and Backdoor categories can hardly be recognized (F1 below 0.1 for most models), and minority classes such as Worms, DoS, and Fuzzers also show notably lower identification performance; (3) XGBoost has obvious advantages on Worms (0.5679) and Reconnaissance (0.8688), reflecting the ability of boosted trees to learn minority classes; (4) DNN is slightly better than XGBoost on Backdoor (0.0833 vs. 0.0357) but remains at a low level overall. The per-class results confirm a strong correlation between sample size and identification performance, and minority-class tracing is the main

shortcoming of the current methods.

4.4. Confusion Matrix Analysis

Fig. 3 presents the confusion matrices of DNN and XGBoost. It can be observed that the errors of both models mainly concentrate on minority classes being misclassified into majority classes such as Generic and Exploits: for example, a large number of DoS samples are identified as Generic, and Analysis and Backdoor samples are absorbed into Generic and Exploits. This indicates that the models can learn effective features when samples are sufficient, but tend to conservatively output majority-class labels on minority classes, which is typical behavior of classifiers under class imbalance.

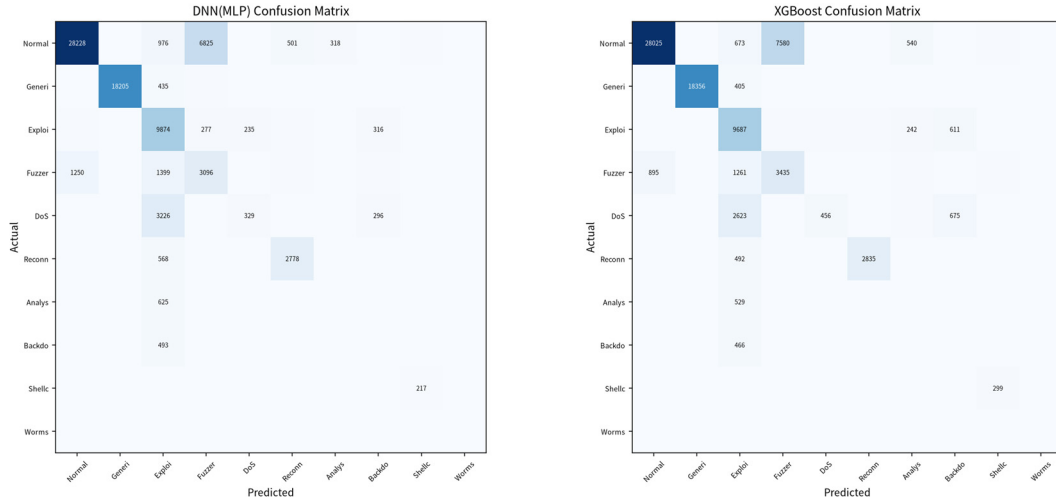


Fig 3. Confusion matrix comparison of DNN and XGBoost

4.5. Training Convergence Analysis

Fig. 4 shows the training loss curve of the DNN model. The loss decreases rapidly within about 20 iterations and then converges; the early stopping mechanism terminates training when the validation loss stops improving. The training process is stable without divergence, verifying the rationality of the network structure and hyperparameter settings.

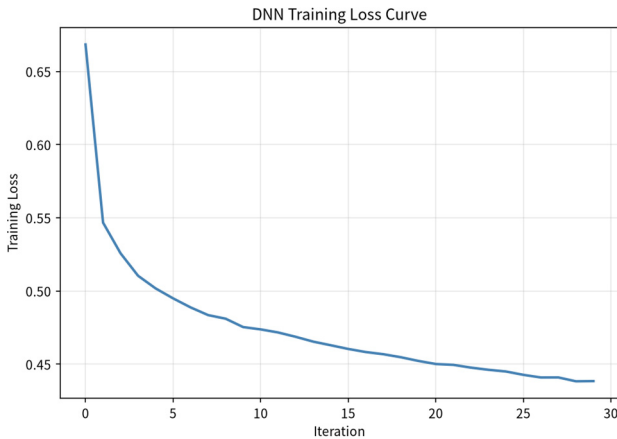


Fig 4. DNN training loss curve

4.6. Oversampling Ablation Experiment

To evaluate the effect of the class imbalance handling strategy, the DNN is retrained with the oversampling scheme,

and the results are shown in Table 4.

Table 4. Results of the oversampling ablation experiment

Training scheme	Accuracy/%	Macro-F1/%	Weighted-F1/%
Baseline	76.27	46.00	76.80
Oversampling	70.28	43.90	75.00

Table 4 shows that the oversampling scheme does not improve performance: the macro-averaged F1 even drops from 46.00% to 43.90%, and accuracy and weighted F1 also decrease. The per-class results show that oversampling only slightly improves a few minority classes such as DoS (0.1379→0.2221), whereas the F1 of Shellcode and Worms decreases. The reasons are as follows: simply duplicating minority samples causes redundancy and overfitting in the training set, making the decision boundaries on minority classes over-adapted to the duplicated samples and disturbing the discrimination of majority classes. This indicates that in the imbalanced multi-class tracing task, more sophisticated strategies such as cost-sensitive loss, generative data augmentation, or attack-semantic-based feature enhancement are needed, rather than simple sample resampling.

4.7. Tracing Results at Kill-Chain Stages

Mapping the per-class identification results to APT kill-chain stages yields the following: the reconnaissance stage (Reconnaissance) and the exploitation stage (Exploits) are traced effectively, with F1 up to 0.87 and 0.71, respectively;

delivery/exploitation-related behaviors (Shellcode) are traced at a medium level (F1 about 0.50); while the persistence stage (Backdoor) and the propagation stage (Worms, DoS) show unstable and low tracing performance. This means that traffic-feature-based tracing methods can effectively support behavior identification in the early stage of an attack, whereas for the critical stages after the attacker has successfully persisted, multi-source evidence such as host-side audit logs, provenance graphs, and threat intelligence should be integrated for comprehensive analysis[6].

4.8. Discussion

The limitations of this study include: (1) UNSW-NB15 is a public synthetic benchmark and differs from real APT traffic, so further validation on real or semi-synthetic APT datasets is needed; (2) the difficulty of minority-class tracing caused by class imbalance has not been fundamentally solved; cost-sensitive learning, focal loss, and generative data augmentation are promising directions; (3) only network traffic features are utilized without fusing host-side behaviors and threat intelligence; multi-source evidence fusion and interpretability enhancement are potential paths to improve tracing explainability.

5. Conclusion

This paper proposes and evaluates an attack behavior tracing method based on a deep neural network for APT tracing requirements. The method models tracing as a fine-grained multi-class identification task of normal behavior and 9 types of attack behaviors and maps the identification results to APT kill-chain stages. Experiments on the UNSW-NB15 dataset show that the proposed DNN achieves a macro-averaged F1 of 46.00%, ranking second among the compared methods, while XGBoost achieves the best macro-averaged F1 of 51.60%. Majority behaviors such as normal, generic, exploitation, and reconnaissance can be effectively traced, whereas minority behaviors such as persistence and propagation are difficult to identify, and class imbalance is the core factor limiting tracing performance. The oversampling ablation experiment further shows that simple resampling cannot solve the problem and more sophisticated imbalance handling strategies are needed. This study provides reproducible experimental evidence for APT attack behavior tracing and attribution assistance and clarifies the direction for future research on minority-class tracing. Future work will focus on cost-sensitive and generative imbalance handling, multi-source evidence fusion, and validation on real APT data.

Acknowledgments

the Youth Fund of Southern Xinjiang Joint Fund of the Natural Science Foundation of Xinjiang Uygur Autonomous Region (Grant No. 2026D01C566).

Open Project of the Key Laboratory of Industrial Internet Security Technology Testing and Evaluation, Ministry of Industry and Information Technology (Grant No. ZCQP2026000613).

Xinjiang College of Science & Technology College Students' Innovation and Entrepreneurship Training Program (Project No. S202613561029).

References

- [1] Hutchins, E. M., Cloppert, M. J., & Amin, R. M. (2011). Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1(1), 80–106.
- [2] Yang, X. Z., Peng, G. J., Liu, S. D., et al. (2025). Survey on tracing and reasoning for APT attacks. *Journal of Software*, 36(1), 203–252.
- [3] Marchetti, M., Pierazzi, F., Colajanni, M., et al. (2016). Analysis of high volumes of network traffic for advanced persistent threat detection. *Computer Networks*, 109, 127–141. <https://doi.org/10.1016/j.comnet.2016.04.020>.
- [4] AL-Aamri, A. S., Abdulghafor, R., Turaev, S., et al. (2023). Machine learning for APT detection. *Sustainability*, 15(18), 13820. <https://doi.org/10.3390/su151813820>.
- [5] Neuschmied, H., Winter, M., Stojanović, B., et al. (2022). APT-attack detection based on multi-stage autoencoders. *Applied Sciences*, 12(13), 6816. <https://doi.org/10.3390/app12136816>.
- [6] Zipperle, M., Gottwalt, F., Chang, E., et al. (2022). Provenance-based intrusion detection systems: A survey. *ACM Computing Surveys*, 55(7), 1–36. <https://doi.org/10.1145/3527849>.
- [7] Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)* (pp. 1–6). IEEE. <https://doi.org/10.1109/MilCIS.2015.7348942>.
- [8] Moustafa, N., & Slay, J. (2016). The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective*, 25(1–3), 18–31. <https://doi.org/10.1080/19393555.2015.1125454>.