Beacon Frame Analysis for Passive Rogue AP Detection in Enterprise WLANs

Akintoye Oyedola
School of Computing, University of North Florida
CNT 6519 Wireless Network Security
Project Type: Experimentation & Evaluation

Abstract

Rogue access points (APs), including evil-twin deployments that clone legitimate network identifiers, are a persistent threat in enterprise wireless LAN (WLAN) environments. This project presents a passive, client-agnostic detection pipeline that analyzes IEEE 802.11 Layer 2 management frames—specifically beacon and probe response frames—to distinguish legitimate APs from rogue devices without active probing or client-side instrumentation. Eight labeled capture sessions were collected using a controlled testbed: a home router as the legitimate AP and an iPhone 15 Personal Hotspot as an evil-twin. Features including inter-frame timing, sequence number deltas, RSSI statistics, and SSID/BSSID consistency were evaluated using Random Forest and SVM classifiers. The Random Forest achieved 83% recall on the rogue class and 78% overall accuracy. A rule-based CLI detection script was also developed. Limitations including MAC address randomization and SSID cloaking are analyzed in the context of RFC 9414 and IEEE 802.11aq.

1 Introduction

Rogue access points pose a significant threat to enterprise wireless networks. Da et al. note that an attacker can deploy one to intercept credentials, perform man-in-the-middle attacks, or silently redirect traffic [1]. Evil-twin APs clone a legitimate SSID and attempt to match the BSSID closely enough to mislead users and clients. Traditional detection relies on active probing, client-side agents, or infrastructure-dependent Wireless Intrusion Prevention Systems (WIPS), all of which introduce overhead, privacy concerns, or deployment complexity [2].

This project investigates whether passive Layer 2 management frame analysis alone can reliably distinguish legitimate APs from rogue devices—without touching client traffic or deploying sensors. The primary research question is: *Can passive management-frame features alone support reliable rogue AP detection using supervised machine learning?*

2 Research Objective

The objective is to detect evil-twin rogue APs by analyzing only the structural and signal properties of IEEE 802.11 beacon and probe response frames captured passively in monitor mode, without active probing, client-side data, or spatial positioning infrastructure. Features evaluated include inter-frame timing, sequence number patterns, RSSI statistics, and SSID/BSSID consistency, benchmarked against published thresholds from the literature.

3 Methodology

Figure 1 summarizes the full experimental pipeline from hardware setup through detection and evaluation.

3.1 Experimental Setup

Experiments ran on Kali Linux 2025.4 (Live USB) on a MacBook with a Qualcomm Atheros AR9271 802.11n USB adapter in monitor mode via `airmon-ng`. Two APs were used: a residential Wi-Fi router (`HOME_WIFI`, identifiers pseudonymized) as the legitimate AP, and an iPhone 15 Personal Hotspot with the same SSID as the evil-twin, forced to 2.4 GHz for adapter compatibility. Captures used `airodump-ng` with PCAP output. All PCAP files, Wireshark screenshots, and Python scripts are included in the supplementary archive (`WirNetSec_D4_supplementary.zip`).

3.2 Data Collection

Seven labeled sessions were collected across two sittings on different days (Table 1). Sessions S1, S2, S4, and S5 formed the training set; S7 and S8 were captured two days later as a genuine held-out test set. Session S6 (SSID-cloaked router) was used only for qualitative analysis. The S3 same-channel evil-twin condition

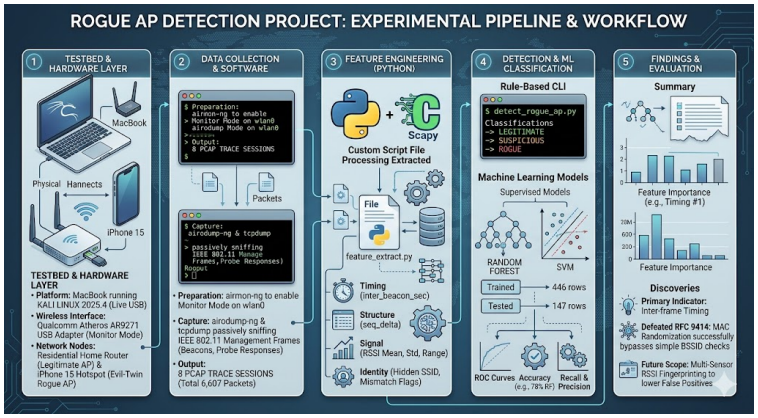

Figure 1: Experimental pipeline and workflow for rogue AP detection, covering the testbed, data collection, feature engineering, ML classification, and evaluation stages.

was not captured due to iOS channel selection behavior—a notable limitation, as same-channel attacks are common in enterprise scenarios.

Table 1: Data Collection Sessions

Session	Condition	Frames	Label	Role
S1	Legitimate-only baseline	207	0	Training
S2	Evil-twin, different channel	62	1	Training
S4	Legitimate-only, different time	192	0	Training
S5	Evil-twin, distance variation	287	1	Training
S6	SSID-cloaked legitimate AP	182	−1	Qualitative
S7	Legitimate-only repeat	218	0	Test
S8	Evil-twin repeat	121	1	Test

3.3 Feature Extraction

A Python 3 script using Scapy parsed all PCAP files. Due to a layer-matching regression in Python 3.13, frame type and subtype were extracted directly from raw 802.11 frame control bytes rather than Scapy's high-level `.haslayer()` API. Management frames of subtype 5 (probe response) and subtype 8 (beacon) were retained. Extracted features included: BSSID, SSID, `is_hidden_ssid`, `seq_num`, `seq_delta` [3], `inter_beacon_sec`, RSSI statistics (`mean`, `std`, `range`) [4, 5], and `ssid_bssid_mismatch` [1].

3.4 Classification and Rule-Based Detection

The filtered dataset (593 rows) was split by session: training on S1, S2, S4, S5 (446 rows) and test on S7, S8 (147 rows). This session-stratified split prevents data leakage across captures [4]. Missing values were imputed as zero. Two classifiers were evaluated: Random Forest (100 estimators, `class_weight='balanced'`) and SVM (RBF kernel, `class_weight='balanced'`). Metrics included precision, recall, F1-score, accuracy, and ROC/AUC.

A complementary CLI script (`detect_rogue_ap.py`) applies four interpretable rules to any input PCAP: (1) SSID/BSSID mismatch, (2) RSSI range anomaly >20 dBm [5], (3) hidden SSID flag, and (4) inter-frame timing gap anomaly. Each AP receives a LEGITIMATE, SUSPICIOUS, or ROGUE verdict written to a CSV report.

4 Implementation and Analysis

The pipeline ran entirely on the Kali Linux live environment. The AR9271 adapter would occasionally drop out of monitor mode when NetworkManager was not fully terminated; this was resolved by adding `systemctl stop NetworkManager` and `killall wpa_supplicant` as explicit pre-capture steps. The Scapy

parser was revised to use direct raw byte inspection of the frame control field after the high-level API silently dropped a substantial number of frames under Python 3.13.

The resulting dataset had a notable 5:1 class imbalance in the test set (121 rogue vs. 26 legitimate frames), which heavily influenced false positive rate results. This was partially mitigated with `class_weight='balanced'` in both classifiers but not fully corrected.

5 Results and Discussion

5.1 Classifier Performance

Table 2 presents held-out test set results. Figure 2 shows the full classification report and feature importances; Figure 3 shows the ROC curves.

Table 2: Classification Results on Held-Out Test Set (S7, S8)

Classifier	Class	Precision	Recall	F1
Random Forest	Legitimate	0.39	0.50	0.44
	Rogue	0.89	0.83	0.86
	Accuracy		0.78	
SVM	Legitimate	0.26	1.00	0.42
	Rogue	1.00	0.48	0.57
	Accuracy		0.50	

The Random Forest achieved 83% recall on the rogue class (78% overall accuracy), with a confusion matrix of 13 true legitimate, 13 false positives, 20 false negatives, and 101 true rogue detections. The SVM collapsed into predicting the majority class, producing 100% legitimate recall but only 48% rogue recall—consistent with class imbalance destabilizing an RBF kernel on this feature set.

5.2 Feature Importance and Key Findings

Random Forest feature importance showed `inter_beacon_sec` dominating at 0.83, followed by `seq_num` at 0.16; all other features scored near zero. The timing signal reflects a fundamental behavioral difference: the router broadcasts beacons at a regular 100 ms interval, while the iPhone generates probe responses reactively, producing irregular gaps. This finding should not be over-generalized—in enterprise environments where rogue devices also broadcast regular beacons, timing alone would not be a reliable discriminator.

The near-zero score of `ssid_bssid_mismatch` is a direct consequence of MAC address randomization [6]. The iPhone hotspot used a randomized BSSID that changed between sessions, so the mismatch rule never fired—a live demonstration that RFC 9414 defeats simple BSSID-based detection. Applied to the S2 evil-twin capture, the rule-based script classified 1 AP as LEGITIMATE, 0 as ROGUE, and 51 as SUSPICIOUS; the high SUSPICIOUS volume in a dense residential environment underscores the need for a known-good whitelist in any deployment. SSID cloaking (S6) similarly neutralized the mismatch rule, confirming predictions by Liu et al. [7] that beacon-only detectors cannot distinguish cloaked legitimate APs from cloaked rogues without external context.

5.3 Comparison to Literature

Zhang et al. [4] report a false positive rate below 5% for PRAPD; the Random Forest here reached ≈50% on the legitimate class, driven by the 5:1 class imbalance and the single-device legitimate AP dataset. Lin et al. [3] report high accuracy using sequence numbers and inter-arrival times; the inter-frame timing result here partially replicates that finding, but sequence deltas contributed almost nothing, likely because probe responses—which have less regular sequence progression than beacons—dominated the captures.

6 Conclusion and Future Work

This project built and evaluated a passive Layer 2 rogue AP detection pipeline using 802.11 management frame features. The Random Forest achieved 83% rogue recall and 78% overall accuracy, with inter-frame timing as the dominant feature. The SVM failed due to class imbalance. Two significant limitations were

```
┌──(kali㉿kali)-[~/rogue_ap_project]
└─$ nano ~/rogue_ap_project/train_classifier.py

┌──(kali㉿kali)-[~/rogue_ap_project]
└─$ python3 train_classifier.py
Training rows: 440
Test rows: 147
Training labels: {1: np.int64(349), 0: np.int64(91)}
Test labels: {1: np.int64(121), 0: np.int64(26)}
/home/kali/rogue_ap_project/train_classifier.py:43: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  train_df[col] = pd.to_numeric(train_df[col], errors='coerce')
/home/kali/rogue_ap_project/train_classifier.py:44: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_df[col] = pd.to_numeric(test_df[col], errors='coerce')

== Random Forest ==
              precision    recall  f1-score   support

  Legitimate       0.39      0.50      0.44        26
       Rogue       0.89      0.83      0.86       121

    accuracy                           0.78       147
   macro avg       0.64      0.67      0.65       147
weighted avg       0.80      0.78      0.79       147

Confusion Matrix:
[[ 13  13]
 [ 20 101]]

== SVM ==
              precision    recall  f1-score   support

  Legitimate       0.26      1.00      0.42        26
       Rogue       1.00      0.40      0.57       121

    accuracy                           0.50       147
   macro avg       0.63      0.70      0.49       147
weighted avg       0.87      0.50      0.54       147

Confusion Matrix:
[[26  0]
 [73 48]]

ROC curve saved to roc_curves.png

== Feature Importance (Random Forest) ==
    inter_beacon_sec: 0.8346
    seq_num: 0.1649
    is_hidden_ssid: 0.0004
    ssid_bssid_mismatch: 0.0001
    seq_delta: 0.0000
    rssi: 0.0000
    rssi_mean: 0.0000
    rssi_std: 0.0000
    rssi_range: 0.0000
```

Figure 2: Terminal output of `train_classifier.py` showing classification report, confusion matrix, and Random Forest feature importances on the held-out test set (S7, S8).

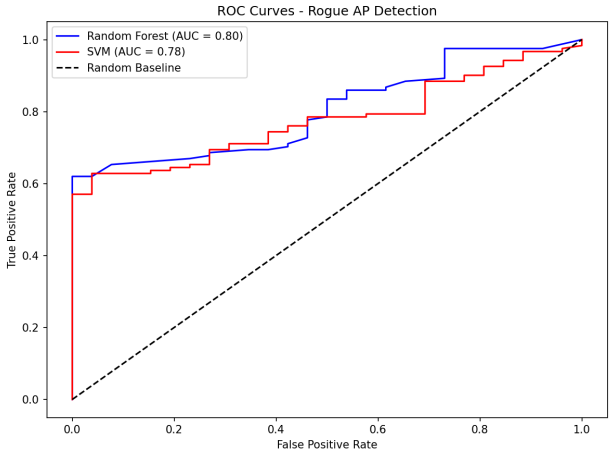

Figure 3: ROC curves for Random Forest and SVM on the held-out test set. Random Forest (AUC = 0.80) outperforms SVM (AUC = 0.78).

demonstrated experimentally: MAC address randomization (RFC 9414 [6]) defeated the primary evil-twin detection rule, and SSID cloaking made beacon-only detection unreliable [7]. The ≈50% false positive rate on the legitimate class far exceeds the 5% target of Zhang et al. [4], driven by class imbalance and a single-device legitimate AP dataset.

Future work should (1) collect larger, more diverse legitimate AP datasets; (2) prioritize beacon over probe

response captures to exploit sequence number fingerprinting [3]; (3) incorporate multi-sensor RSSI aggregation as in PRAPD [4]; (4) use dedicated rogue hardware (e.g., Raspberry Pi) rather than an iPhone hotspot; and (5) explore MAC-randomization-resistant approaches such as timing jitter profiles and physical-layer features.

References

[1] B. Da, Z. Zhang, and X. Wang, "Rogue access point detection: A survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 245–268, 2021.

[2] Cisco Systems, Inc., "Rogue access point detection and mitigation in the enterprise," white paper, Cisco Systems, Inc., 2023.

[3] J. Lin, J. Wang, and T. Xu, "Detecting rogue access points via beacon frame analysis and machine learning," *IEEE Transactions on Network and Service Management*, vol. 19, pp. 2105–2118, Sept. 2022.

[4] L. Zhang, Y. Yang, and C. Wang, "PRAPD: Passive rogue access point detection using RSSI fingerprinting," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, 2018.

[5] W. Li, M. Li, and R. D. Pietro, "Exploiting wireless received signal strength indicators to detect evil-twin access points," *Security and Communication Networks*, vol. 2017, pp. 1–13, 2017.

[6] J. C. Zúñiga, C. J. Bernardos, and A. Andersdotter, "Privacy considerations for IEEE 802.11 operating procedures," Request for Comments 9414, IETF, Jan. 2023.

[7] H. Liu, S. Chen, and M. Zhou, "Position-based rogue AP detection using spatial signal properties," *Journal of Network and Computer Applications*, vol. 221, p. 103784, Jan. 2024.