Virtualized NAC with Automated Wired 802.1X Incident Validation

Version 1.2

Contents

1. Document Control

1.1 Document Metadata

Field	Value
Document Title	Virtualized NAC with Automated Wired 802.1X Incident Validation
Author	Eric Schmitt
Document Version	1.2
Date of Issue	April 2, 2026
Document Status	Production Draft (Educational Lab / Proof-of-Concept)
Document Type	Engineering Lab Implementation and Validation Report
Primary Scope	Virtualized wired IEEE 802.1X NAC proof-of-concept with policy-based guest access enforcement, incident simulation, automation, and packet-level validation
Core Technologies	IEEE 802.1X via `hostapd` (wired mode) and `wpa_supplicant`; pfSense; FreeRADIUS; GLPI on Apache and MariaDB; Splunk Universal Forwarder; Python; Ansible; `tcpdump`; Wireshark
VM Platform	VMware Workstation Pro: Ubuntu Server 24.04 LTS
Primary Lab Purpose	Demonstrate a realistic small-scale NAC architecture with authenticated access control, constrained guest access, observability, ITSM workflow integration, automation, and protocol validation

The purpose of this document is to provide a complete, technically rigorous, and reproducible record of the final implemented state of the lab environment, the access-control model selected for the environment, the operational validation performed against that environment, and the incident simulation used to demonstrate practical NAC troubleshooting and remediation workflows.

This document is intentionally written as an engineering implementation and validation report rather than as a tutorial, learning journal, or informal case study. Where the lab departs from production architecture due to virtualization or platform constraints, those departures are stated explicitly and evaluated in context rather than implied or omitted.

1.2 Change History

Version	Date	Description
1.0	2026-03-24	Initial document draft; text-only
1.1	2026-03-30	Formatting revisions for raw data and visual references
1.2	2026-04-02	Final editorial revision reflecting current lab state

1.3 Intended Audience and Usage

This document is intended for the following audiences:

- Network engineering practitioners evaluating the design and validation of a virtualized NAC proof-of-concept;
- Security engineering practitioners reviewing the use of wired IEEE 802.1X, RADIUS-backed authentication, and segmented guest policy enforcement;
- Systems engineering practitioners assessing service integration across routing, firewalling, authentication, logging, ITSM, and automation domains;
- Technical reviewers evaluating the project as a technical document intended to demonstrate enterprise-relevant design and operational execution rather than isolated tool familiarity.

This document should be interpreted as a formal record of the final implemented state of the environment and the authoritative incident validation path, not as a chronological build log. Intermediate troubleshooting states, temporary misconfigurations, and transient construction artifacts that were used only to arrive at the final design are intentionally excluded unless they are directly relevant to the incident narrative or to explaining an architectural decision.

The operational evidence included in this document is constrained by the following rules:

- Final-state command outputs, configuration files, service states, validation results, and automation artifacts are considered in scope;
- Packet capture interpretation and protocol-state observations are considered in scope;
- Full debug logs are not reproduced wholesale; only explicitly selected excerpts are included where necessary to support an incident conclusion;
- GUI-derived evidence (e.g., pfSense, GLPI, Wireshark) is represented by screenshots rather than terminal transcription unless a manual transcription is operationally clearer;
- Sensitive values, including passwords, shared secrets, tokens, and authentication material, MUST be redacted in all published forms of the document.

The environment documented herein is intentionally designed as a constrained but defensible approximation of enterprise NAC behavior within a virtualization

platform. In particular, the guest-access model implemented in this lab is a policy-based guest access simulation enforced through Linux and pfSense control points rather than true switch-driven dynamic VLAN reassignment. This distinction is central to the technical integrity of the document and will be treated explicitly in subsequent sections.

2. Executive Summary

2.1 Objective

The objective of this implementation was to design, build, and validate a small-scale but enterprise-relevant Network Access Control (NAC) proof-of-concept within a fully virtualized lab environment. The target outcome was to establish an integrated control path in which a wired endpoint could be authenticated via IEEE 802.1X against a RADIUS policy authority, granted or denied access based on authentication state, observed through centralized logging and evidence collection, and incorporated into a realistic incident-response workflow.

The implemented design intentionally spans multiple operational domains in order to reflect the way NAC is encountered in practice. These domains include endpoint authentication, routed segmentation, firewall policy enforcement, service observability, IT service management (ITSM) workflow, automation-assisted validation, and packet-level protocol analysis. The lab was therefore scoped to produce not only a functional access-control environment, but also a defensible record of how that environment behaves under both normal and failure conditions.

The specific technical objectives of the lab were as follows:

- Implement a wired IEEE 802.1X authentication path using a Linux-based authenticator simulation and a FreeRADIUS policy server;
- Establish authenticated access to routed service resources for a validated endpoint;
- Implement a constrained guest-access model for unauthenticated or otherwise non-trusted endpoints using policy-based enforcement at the firewall boundary;
- Integrate local observability through service logs, structured incident evidence files, and Splunk Universal Forwarder monitoring inputs;
- Integrate ITSM workflow through GLPI by recording the incident as an actual tracked ticket associated with a managed asset;
- Implement meaningful automation through a Python evidence parser and an Ansible validation/remediation workflow;
- Validate both failure and recovery states using terminal evidence, service logs, and packet captures reviewed in Wireshark.

The result is intended to function as a technically rigorous proof-of-concept demonstrating that a virtualized environment can be used to model core NAC control flows and operational procedures, provided that platform limitations are explicitly acknowledged and the chosen enforcement model is documented accurately.

2.2 Outcome Summary

The lab successfully achieved its core design and validation objectives.

A functioning wired IEEE 802.1X authentication path was implemented using a Linux-based access node acting as a wired authenticator, a Linux endpoint acting as the supplicant, and FreeRADIUS as the central authentication and policy service. The authenticated endpoint path was validated through successful RADIUS test transactions, successful supplicant authorization during controlled test execution, and confirmed routed reachability to service-network resources once the authorization path was operating as intended.

Because the environment was built entirely from virtual machines and not from managed enterprise switching hardware, the design could not support true switch-driven dynamic VLAN reassignment in the same manner as a production NAC deployment. Rather than obscuring this limitation, the implementation explicitly adopted a policy-based guest access simulation. In this model, unauthenticated or non-trusted endpoint behavior was represented by placement in a dedicated guest network segment behind pfSense, where DHCP, gateway access, and tightly constrained reachability were permitted while access to protected service resources was denied. This design choice preserved the operational intent of differentiated access control while remaining technically honest about the constraints of the platform.

The lab further succeeded in demonstrating an operational incident lifecycle rather than only a steady-state build. A wired 802.1X authentication failure was deliberately introduced in a controlled manner, evidence of the failed state was captured across the supplicant, authenticator, and RADIUS service layers, and the resulting condition was documented as a real incident in GLPI. Remediation was then performed by restoring the known-good client configuration and executing a clean re-test, after which successful authorization indicators were observed and the incident was closed as resolved.

Automation was incorporated in a manner that is materially relevant to the environment rather than ornamental. A Python script was developed to parse and summarize incident evidence artifacts, providing a deterministic summary of failed and remediated states. An Ansible playbook was developed to perform repeatable validation of NAC-related services and to support a bounded, operator-invoked remediation mode. These automation artifacts reinforce the document's central claim that the lab demonstrates not only component familiarity, but also repeatable operational procedure.

Finally, packet capture analysis was added as a first-class validation layer. Using `tcpdump` on the authenticator host and subsequent inspection in Wireshark, the lab captured both failed and successful authentication flows across the EAPOL and RADIUS control paths. This provided protocol-level corroboration of the incident narrative and strengthened the evidentiary value of the environment as a proof-of-concept.

2.3 Architectural Constraints and Design Decision

The most important architectural constraint in this lab is that the environment does not include a managed access switch capable of acting as a production-grade IEEE 802.1X authenticator with RADIUS-driven dynamic VLAN reassignment, downloadable ACLs, or hardware-enforced post-auth policy state. Instead, the environment uses Ubuntu Server virtual machines and VMware virtual networks to model the control plane and segmentation boundaries.

This constraint has direct implications. While Linux user-space tools such as `hostapd` and `wpa_supplicant` can accurately model important aspects of wired IEEE 802.1X authentication and EAP state transitions, they do not convert a general-purpose virtual machine into a fully equivalent enterprise access switch. As a result, the environment can credibly demonstrate:

- supplicant-to-authenticator EAPOL exchanges;
- authenticator-to-RADIUS request handling;
- successful and failed authentication states;
- policy differentiation between trusted and guest access paths;
- operational validation, logging, and incident response workflows.

However, it cannot credibly claim to implement:

- true dynamic VLAN reassignment on a switchport as the direct result of RADIUS policy;
- hardware-native port reauthorization behavior;
- switch-specific enforcement semantics such as vendor ACL or downloadable ACL features.

For that reason, the guest-access component of this lab is explicitly defined as a policy-based guest access simulation rather than as a true dynamic guest VLAN implementation. The enforcement boundary is achieved through Linux and pfSense control points: the access path models authentication state, while pfSense enforces the differentiated reachability policy for the guest segment. This is not mutually exclusive with the broader segmentation model used throughout the lab; rather, it is the most defensible way to preserve the operational objective of differentiated access within the constraints of a virtualized platform.

This design decision is central to the technical integrity of the document. It allows the environment to remain enterprise-relevant without overstating equivalence to production hardware behavior, and it ensures that the incident and automation workflows presented later in the document are evaluated against the architecture that was actually built, not against an idealized architecture that was not.

3. Scope, Objectives, and Design Assumptions

3.1 In-Scope Components

The following components and functional domains are in scope for this implementation and are treated as authoritative elements of the final environment:

- **Virtualized routing and segmentation:** multi-segment VMware virtual networking, Linux routing on intermediate nodes, and pfSense routing/firewall policy between guest and service segments;
- **Wired IEEE 802.1X authentication path:** a Linux-based wired authenticator simulation using `hostapd`, a Linux supplicant using `wpa_supplicant`, and FreeRADIUS as the authentication and policy service;
- **RADIUS policy service:** FreeRADIUS user and client definitions, service validation, local and remote authentication testing, and runtime logging;
- **Guest access control model:** policy-based guest access simulation enforced by pfSense rather than true dynamic VLAN reassignment;
- **Observability and evidence collection:** service logs, journald evidence, explicit incident capture files, and packet capture collection with `tcpdump`;
- **Log monitoring instrumentation:** Splunk Universal Forwarder configured to monitor relevant system and FreeRADIUS log sources;
- **ITSM workflow integration:** GLPI deployment, managed asset creation, incident ticket creation, incident update, and incident closure;
- **Automation artifacts:** Python-based incident evidence parser and Ansible-based validation/remediation workflow;
- **Packet-level protocol validation:** failed and successful EAPOL and RADIUS captures reviewed in Wireshark and interpreted as part of the incident narrative.

The lab is therefore intentionally broader than a single-service build. It is designed to demonstrate cross-domain operational competence across networking, security, Linux services, observability, automation, and incident handling.

3.2 Out-of-Scope Components

The following capabilities, platforms, or production-grade behaviors are explicitly out of scope for this implementation:

- **Managed enterprise switching hardware** with native switchport control semantics, hardware-enforced post-auth state, or vendor-specific NAC integration features;
- **True dynamic VLAN reassignment** directly applied to an access port by RADIUS attributes during live port authorization;
- **Downloadable ACLs (dACLs)** or equivalent vendor-specific post-auth policy enforcement constructs;
- **Certificate-based enterprise EAP methods** such as EAP-TLS; the lab uses a simpler method appropriate for deterministic proof-of-concept validation rather than a full PKI-backed endpoint identity model;
- **Full Splunk Enterprise deployment** including indexer, search head, dashboards, or distributed search architecture; the lab intentionally uses Splunk Universal Forwarder only;
- **High availability or clustering** for core services such as RADIUS, firewalling, or ITSM;
- **Production secrets management** such as Vault-backed secret retrieval, certificate lifecycle automation, or centrally governed credential rotation;
- **Full configuration management pipeline integration** such as CI/CD, GitOps enforcement, or scheduled compliance scans beyond the bounded automation included here.

These exclusions do not weaken the document; rather, they define the limits within which the implementation should be evaluated. The lab is intended to be a credible and technically honest proof-of-concept, not a claim of feature parity with a production NAC deployment.

3.3 Design Assumptions

The architecture and validation logic in this document rely on the following design assumptions:

- VMware virtual networks are treated as deterministic Layer 2 broadcast domains suitable for representing isolated access, transit, service, and guest segments;
- Ubuntu Server virtual machines can credibly model routing, firewall-adjacent policy boundaries, service hosting, and user-space authenticator/supplicant behavior, but not the full control semantics of a managed enterprise switch;

- The Linux-based access node provides a valid proof-of-concept wired authenticator role for demonstrating EAPOL state transitions and RADIUS interaction, even though it is not a substitute for hardware switch behavior;
- pfSense provides the authoritative guest-policy enforcement boundary in the final design, including guest subnet gatewaying, DHCP, and restrictive firewall policy;
- Temporary host-side management IP assignments on VMware virtual network interfaces are acceptable as an operational convenience for accessing isolated management surfaces during build and validation, provided those assignments are clearly described and not misrepresented as part of the guest or authenticated endpoint data path;
- Final-state validation evidence is more relevant to the purpose of this document than full chronological build history, and therefore only final-state and incident-relevant evidence is included;
- Packet capture interpretation is treated as corroborating evidence for the control-plane narrative rather than as the sole source of truth; service state and incident logs remain authoritative in combination with capture findings.

These assumptions are intentionally conservative. They preserve the lab's credibility by clearly separating what is being simulated from what is being directly implemented.

3.4 Success Criteria

The implementation is considered successful only if all of the following criteria are satisfied:

- A wired endpoint can successfully complete the intended 802.1X authentication flow through the Linux-based authenticator and FreeRADIUS service under controlled test conditions;
- The authenticated path provides the expected routed reachability to protected service-network resources in the final-state topology;
- A guest or otherwise non-trusted endpoint path can be demonstrated as operationally distinct, including guest DHCP and restricted reachability enforced by pfSense policy;
- FreeRADIUS can be validated both locally and remotely using deterministic test transactions and final-state service health checks;
- Splunk Universal Forwarder is configured to monitor the intended log sources relevant to NAC and incident evidence collection;
- GLPI is operational and used to represent a real incident record associated with a managed asset, rather than being present only as an unused

supporting service;

- A controlled wired 802.1X authentication failure can be reproduced, observed, and documented across the supplicant, authenticator, and RADIUS layers;
- The failed condition can be remediated, and a clean re-test can confirm restoration of successful authorization;
- The incident can be summarized through automation using the Python evidence parser and validated through repeatable service checks using the Ansible workflow;
- Packet captures of both failed and successful authentication flows can be collected and interpreted in a way that supports the incident narrative and final validation conclusions.

The standard applied throughout this document is whether the environment can support a coherent access-control narrative, a reproducible incident, and a technically defensible remediation and validation process.

4. Reference Architecture

4.1 Logical Topology Overview

The final lab environment is implemented as a multi-segment virtualized network hosted on VMware Workstation Pro running on an Ubuntu 24.04 workstation. The design intentionally separates management, authenticated access, routed transit, service hosting, and guest access into distinct logical domains in order to model the control boundaries that would exist in a small enterprise environment.

At a high level, the architecture consists of the following functional layers:

- **Endpoint access layer:** a Linux endpoint (`client-1`) acting as the wired supplicant and a Linux access node (`access-node`) acting as the wired IEEE 802.1X authenticator using `hostapd`;
- **Routed access transit:** a Linux distribution router (`dist-router`) providing the intermediate routed path between the access domain and the firewall-controlled service domain;
- **Security and policy boundary:** a pfSense virtual firewall providing routing, interface separation, guest DHCP, and guest-policy enforcement between protected services and guest access;
- **Protected service domain:** a services segment containing `radius-splunk` (FreeRADIUS, Splunk Universal Forwarder, Python/Ansible automation) and `ops-glpi` (GLPI, Apache, MariaDB);
- **Guest access domain:** a separate pfSense-controlled guest network used to simulate unauthenticated or non-trusted endpoint behavior through

policy-based access restriction rather than dynamic switch-driven VLAN reassignment;

- **Out-of-band management paths:** VMware-provided host-side virtual network access used where necessary for administrative reachability to otherwise isolated interfaces during build and validation.

The most important architectural distinction in this environment is the separation between authentication state and guest-policy enforcement state. Authentication is modeled directly in the Linux-based 802.1X control path between the supplicant, the authenticator, and FreeRADIUS. Guest-policy enforcement is modeled at the pfSense boundary, where a dedicated guest subnet receives DHCP service and is restricted from reaching protected service resources. This division is intentional and reflects the platform constraint that the lab cannot perform true switch-native dynamic VLAN reassignment.

In practical terms, the environment therefore demonstrates the following control-plane relationships:

- `client-1` can authenticate through `access-node` to FreeRADIUS when operating in the wired 802.1X validation path;
- `access-node` forwards RADIUS authentication traffic toward the protected services segment via `dist-router` and pfSense routing;
- `radius-splunk` provides the policy authority for the authentication workflow and also serves as the automation control node;
- `ops-glpi` provides the ITSM platform used to represent the incident as an actual tracked operational record;
- pfSense enforces the guest access model by providing DHCP and tightly scoped reachability on the guest segment while preventing direct access from the guest segment to protected services.

This architecture is sufficiently complex to demonstrate realistic operational dependencies without becoming so large that the evidentiary chain becomes difficult to follow. The final design is therefore intentionally narrow in breadth but deep in validation.

4.2 Network Segmentation and Addressing

The final environment uses multiple RFC1918 segments to model distinct functional domains. These segments are intentionally small and purpose-driven. The authoritative logical addressing model used throughout this document is summarized below.

Logical Network Segments and Addressing

Access Client Segment
Subnet: `10.0.12.0/24`

Primary Function: Authenticated endpoint attachment segment for the wired supplicant validation path.
Representative Systems / Interfaces: `client-1` (`10.0.12.10`), `access-node` access-facing interface (`10.0.12.1`).

Access-to-Distribution Transit
Subnet: `10.0.23.0/24`
Primary Function: Routed transit between the access node and the intermediate distribution router.
Representative Systems / Interfaces: `access-node` (`10.0.23.1`), `dist-router` (`10.0.23.2`).

Protected Services Segment
Subnet: `10.0.30.0/24`
Primary Function: Service hosting for RADIUS, logging, automation control, and ITSM systems.
Representative Systems / Interfaces: pfSense OPT1 (`10.0.30.1`), `radius-splunk` (`10.0.30.10`), `ops-glpi` (`10.0.30.20`).

Distribution-to-Firewall Transit
Subnet: `10.0.34.0/30`
Primary Function: Point-to-point transit between the Linux distribution router and pfSense.
Representative Systems / Interfaces: `dist-router` (`10.0.34.1`), pfSense LAN/transit (`10.0.34.2`).

Guest Access Segment
Subnet: `10.0.20.0/24`
Primary Function: Guest or non-trusted endpoint segment used for policy-based guest access simulation.
Representative Systems / Interfaces: pfSense GUEST/OPT2 (`10.0.20.254`), DHCP pool (`10.0.20.100-10.0.20.199`).

Host / Hypervisor Management Segments
Subnet: VMware-provided RFC1918 segments (e.g., `192.168.195.0/24`, `192.168.205.0/24`).
Primary Function: Administrative reachability and host-level VM management paths.
Representative Systems / Interfaces: Host virtual interfaces, VM management adapters, temporary host-side management access where required.

The protected services segment (`10.0.30.0/24`) is the most operationally significant network in the final design. It hosts the RADIUS policy authority, the log-forwarding instrumentation, the Python and Ansible automation artifacts, and the ITSM platform used to record the incident lifecycle. This segment is reachable from the authenticated access path through the routed chain `access-node` → `dist-router` → pfSense, and it is intentionally protected from direct guest-segment access by pfSense firewall policy.

The guest access segment (`10.0.20.0/24`) is not intended to be interpreted as a dynamically assigned VLAN in the switch-native sense. Rather, it is the authoritative guest-policy enforcement domain used by this lab to represent how an unauthenticated or otherwise restricted endpoint would be treated under a constrained virtualized implementation. The guest segment receives DHCP from pfSense and can reach the guest gateway for local validation, but it is intentionally restricted from reaching the protected services segment as part of the policy model described later in Section 6.

The authoritative final-state interface configurations and route outputs that support this addressing model will be captured later in the document and in the appendices. Where exact file-based network definitions are relevant, they will be cross-referenced directly (for example, see Section A.1.1, Section A.1.4, Section A.1.2, Section A.1.5, and Section 5).

4.3 Final System Roles

Each system in the final lab has a deliberately constrained but clearly defined role. This section establishes those roles so that later configuration, validation, and incident sections can refer to each node without ambiguity.

4.3.1 access-node

`access-node` is the primary access-layer control point in the lab. It serves as the Linux-based wired IEEE 802.1X authenticator using `hostapd` in wired mode and also functions as the first routed hop away from the endpoint access segment. Its responsibilities include:

- providing the access-facing interface for the wired supplicant path;
- originating EAPOL interaction with the supplicant during controlled 802.1X testing;
- forwarding RADIUS authentication requests toward the FreeRADIUS service;
- maintaining static routing toward the protected services domain via `dist-router`;
- serving as the packet capture location for both EAPOL and RADIUS control-plane captures during incident validation.

Its final-state configuration artifacts will be documented later (see Section A.1.2, Section A.2.1, Section A.2.2, and Section 5).

4.3.2 client-1

`client-1` is the endpoint used to represent a wired supplicant. It participates in both the authenticated access validation path and the guest-policy simulation path, depending on the test being executed. Its responsibilities include:

- acting as the 802.1X supplicant using `wpa_supplicant` in wired mode;
- serving as the primary subject of the simulated authentication failure and remediation workflow;
- demonstrating guest-segment behavior when attached to the pfSense-controlled guest network;
- providing endpoint-perspective evidence for successful and failed authorization states.

Its final-state network configuration and wired supplicant configuration will be documented later (see Section A.1.5, Section A.2.3, and Section B.1).

4.3.3 dist-router

`dist-router` is the intermediate Linux routing node that provides the transit path between the access-layer control domain and the pfSense-protected service domain. It exists to make the architecture more representative of a routed enterprise topology rather than collapsing all functions into a single flat segment. Its responsibilities include:

- routing traffic between the access-node transit segment and the pfSense transit interface;
- providing the next hop from the access domain toward the protected services segment;
- supporting final-state path validation and demonstrating that the RADIUS control path traverses a realistic routed intermediate node.

Its final-state interface and route evidence will be included later as part of the validation and appendix materials (see Section 5).

4.3.4 pfSense

pfSense is the authoritative security boundary and policy-enforcement node in the final design. It separates the protected services domain from the guest domain and provides the guest gateway, guest DHCP service, and restrictive guest firewall policy. Its responsibilities include:

- terminating the routed transit from `dist-router`;
- providing the protected services segment gateway;
- providing the guest segment gateway;
- issuing DHCP leases on the guest segment;
- enforcing the guest access policy that differentiates guest reachability from authenticated service reachability;

- serving as the primary firewall boundary used to simulate guest access control in the absence of true switch-native dynamic VLAN reassignment.

Its final interface assignments, DHCP configuration, and guest firewall rules will be documented later (see Section A.3.1, Section A.3.2, Section A.3.3, and the related screenshots in the main body).

4.3.5 radius-splunk

`radius-splunk` is the most functionally dense system in the lab and serves as both the protected authentication authority and the automation/observability control node. It hosts:

- FreeRADIUS as the central authentication and policy service;
- Splunk Universal Forwarder configured to monitor relevant local log sources;
- the Python incident evidence parser used to summarize failed and remediated states;
- the Ansible control environment used for repeatable validation and bounded remediation workflows.

Because this node combines policy authority, observability, and automation, it is central to both the steady-state validation path and the incident-response narrative. Its configuration artifacts will be referenced extensively later (see Section A.1.1, Section A.2.5, Section A.2.4, Section A.4.6, Section A.4.3, and Section A.4.2).

4.3.6 ops-glpi

`ops-glpi` provides the IT service management layer for the lab. It is intentionally not treated as a passive supporting service; instead, it is used to represent the incident as an actual tracked operational record tied to a managed asset. Its responsibilities include:

- hosting GLPI on Apache with MariaDB backend services;
- maintaining the `client-1` asset record used in the incident workflow;
- maintaining the incident ticket used to document the failed authentication event, remediation, and closure.

Its final-state configuration and deployment validation will be documented later (see Section A.1.4 and Section 5.7).

4.4 Topology Diagram

Figure 1 is the authoritative logical topology diagram for the lab document. The diagram should preserve the segment boundaries and role relationships defined in Sections 4.1 through 4.3.

Diagram implementation note:
The source-of-truth for this diagram is maintained in Mermaid format under `diagrams/reference-topology-diagram.mmd`. During CI/CD, Mermaid is rendered to `dist/docs/diagrams/reference-topology-diagram.png`, and the document embeds the generated PNG artifact so that both HTML and PDF outputs are consistent and deterministic.

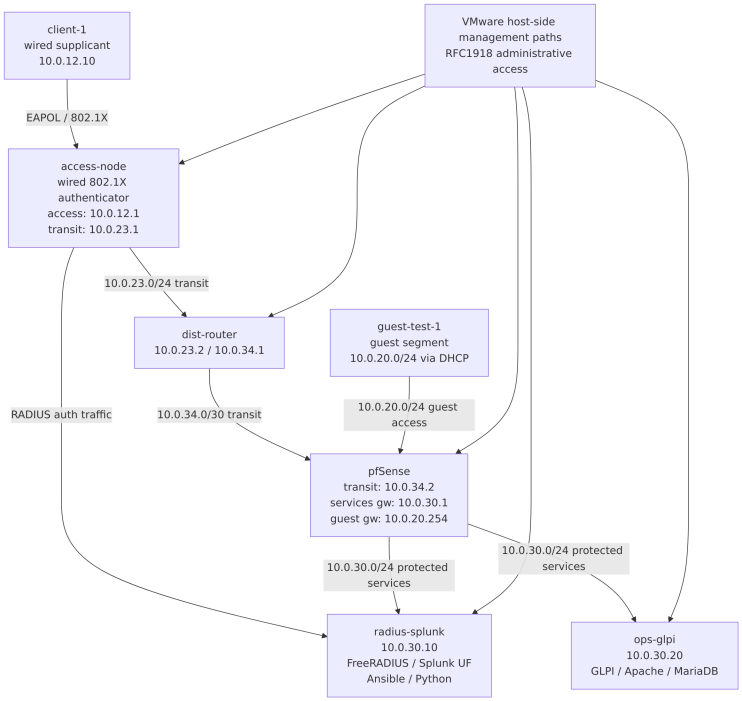

Figure 1: Reference topology diagram for the final virtualized NAC implementation.

The diagram is intended to function as the reader's primary visual reference for the rest of the document. Later sections will rely on the topology as the basis for explaining the authenticated access path, the guest-policy enforcement model, the incident flow, and the packet capture locations. Supporting final-state interface and route evidence will be provided later in the appendices and validation sections rather than embedded directly here (see Section 5).

5. Platform Build and Final-State Configuration

5.1 Hypervisor and Virtual Network Layout

The lab is hosted on VMware Workstation Pro running on an Ubuntu 24.04 workstation. Distinct VMware virtual networks are used to model access, transit,

service, guest, and management domains. These virtual networks are treated as deterministic Layer 2 broadcast domains and are mapped into the final architecture as functional segments rather than as ad hoc connectivity conveniences.

In the final design, the principal virtual network roles are as follows:

- a management/NAT network used for host-side administrative reachability to selected virtual machines;
- an access segment for the wired endpoint path;
- a transit segment between the access node and the intermediate router;
- a protected services segment behind pfSense;
- a guest segment behind pfSense used for the policy-based guest access simulation.

The actual VMware GUI configuration is not reproduced here because it does not materially improve the engineering value of the document. Instead, the document records the final interface-to-segment mapping in operational terms through the final netplan files, interface summaries, and pfSense interface assignments. This is more appropriate for a production-style technical record and avoids overemphasizing hypervisor UI state over the logical architecture.

Temporary host-side IP assignments on VMware host interfaces were used at specific points to reach otherwise isolated pfSense interfaces during setup and validation. Those temporary host assignments are not treated as part of the access-control data path and are not part of the final guest or authenticated endpoint model.

5.2 Router and Firewall Roles

The routing and policy boundary in the final design is split across two nodes:

- `dist-router` provides the routed transition from the access domain toward the pfSense transit interface;
- **pfSense** provides the protected services gateway, the guest gateway, guest DHCP, and the firewall enforcement boundary.

This separation is important because it makes the architecture more representative of a routed enterprise environment. Rather than allowing the access node to connect directly to the service segment, the design forces service-bound traffic through an intermediate routed hop and then through the firewall boundary before it reaches the protected service network.

The final-state interface and route posture on `dist-router` confirms this role. The node maintains:

- a management/NAT-connected interface for administrative access;
- a transit-facing interface on `10.0.23.0/24` toward `access-node`;

- a point-to-point transit interface on `10.0.34.0/30` toward pfSense;
- a static route to the access client segment via `10.0.23.1`;
- a static route to the protected services segment via pfSense at `10.0.34.2`.

The final-state interface summary and routing table for `dist-router` are reproduced below.

```
dist-router: ip -br a
lo               UNKNOWN        127.0.0.1/8 ::1/128
ens33            UP             192.168.195.142/24 metric 100 fe80::20c:29ff:fe4f:9ae7/64
ens38            UP             10.0.23.2/24 fe80::20c:29ff:fe4f:9af1/64
ens39            UP             10.0.34.1/30 fe80::20c:29ff:fe4f:9afb/64
```

```
dist-router: ip route
default via 192.168.195.2 dev ens33 proto dhcp src 192.168.195.142 metric 100
10.0.12.0/24 via 10.0.23.1 dev ens38 proto static
10.0.23.0/24 dev ens38 proto kernel scope link src 10.0.23.2
10.0.30.0/24 via 10.0.34.2 dev ens39 proto static
10.0.34.0/30 dev ens39 proto kernel scope link src 10.0.34.1
192.168.195.0/24 dev ens33 proto kernel scope link src 192.168.195.142 metric 100
192.168.195.2 dev ens33 proto dhcp scope link src 192.168.195.142 metric 100
```

```
dist-router: sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 1
```

The corresponding final netplan configuration on `dist-router` is reproduced in full below and is also referenced in Appendix A (see Section A.1.3).

```
dist-router: /etc/netplan/50-cloud-init.yaml
network:
  version: 2
  ethernets:
    ens33:
      dhcp4: true
    ens38:
      dhcp4: false
      addresses:
        - 10.0.23.2/24
      routes:
        - to: 10.0.12.0/24
          via: 10.0.23.1
    ens39:
      dhcp4: false
      addresses:
        - 10.0.34.1/30
      routes:
        - to: 10.0.30.0/24
          via: 10.0.34.2
```

pfSense is the authoritative security boundary in the environment. Its final interface structure is as follows:

- **WAN** (`em0`) on the VMware management/NAT network at `192.168.195.145/24`;
- **LAN / transit** (`em1`) at `10.0.34.2/30`;
- **OPT1 / services** (`em2`) at `10.0.30.1/24`;
- **GUEST / OPT2** (`em3`) at `10.0.20.254/24`.

This interface design allows pfSense to sit directly between the routed access path and the protected services segment, while also hosting a guest segment whose policy can be enforced independently.

The authoritative manual transcription of the final pfSense interface summary is reproduced below and in the appendix (see Section A.3.1).

```
pfSense final interface summary
WAN (wan) -> em0 -> v4/DHCP4: 192.168.195.145/24
LAN (lan) -> em1 -> v4: 10.0.34.2/30
OPT1 (opt1) -> em2 -> v4: 10.0.30.1/24
GUEST (opt2) -> em3 -> v4: 10.0.20.254/24
```

The associated pfSense interfaces terminal screenshot is shown below.

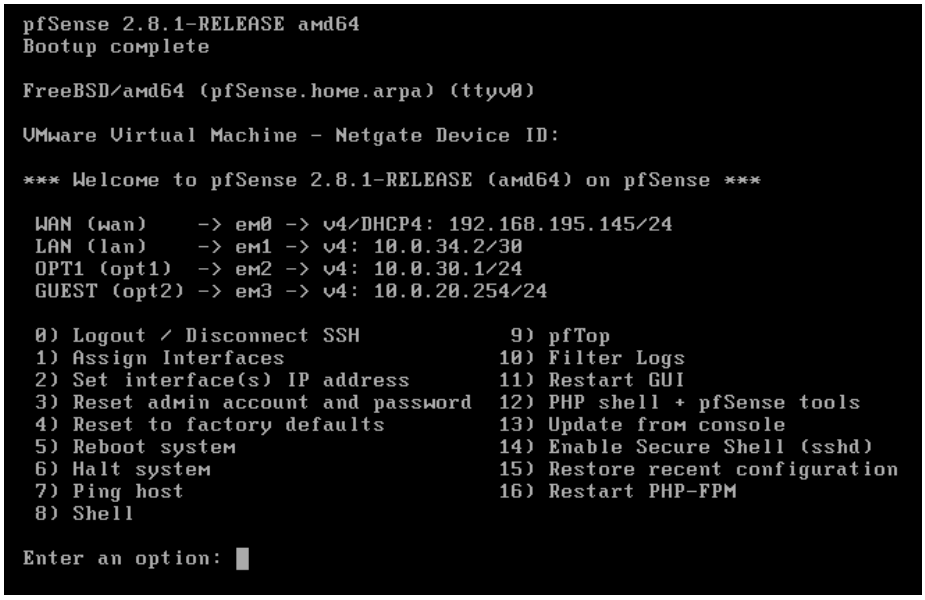

Figure 2: pfSense interface assignments showing WAN, LAN/transit, OPT1 (services), and GUEST (OPT2)

5.3 FreeRADIUS Final Configuration

FreeRADIUS is hosted on `radius-splunk` and functions as the authoritative authentication and policy service for the wired 802.1X workflow. In the final design, FreeRADIUS accepts requests from the Linux-based authenticator on `access-node`, validates the known endpoint credential, and provides the Access-Accept or reject path used throughout the validation and incident workflows.

The final-state network posture of `radius-splunk` is reproduced below.

```
radius-splunk: ip -br a
lo               UNKNOWN        127.0.0.1/8 ::1/128
ens33            UP             192.168.195.143/24 metric 100 fe80::20c:29ff:fe82:9c22/64
ens38            UP             10.0.30.10/24 fe80::20c:29ff:fe82:9c2c/64
```

```
radius-splunk: ip route
default via 192.168.195.2 dev ens33 proto dhcp src 192.168.195.143 metric 100
10.0.12.0/24 via 10.0.30.1 dev ens38 proto static
10.0.23.0/24 via 10.0.30.1 dev ens38 proto static
10.0.30.0/24 dev ens38 proto kernel scope link src 10.0.30.10
10.0.34.0/30 via 10.0.30.1 dev ens38 proto static
192.168.195.0/24 dev ens33 proto kernel scope link src 192.168.195.143 metric 100
192.168.195.2 dev ens33 proto dhcp scope link src 192.168.195.143 metric 100
```

```
radius-splunk: /etc/netplan/50-cloud-init.yaml
network:
  version: 2
  ethernets:
    ens33:
      dhcp4: true
    ens38:
      dhcp4: false
      addresses:
        - 10.0.30.10/24
      routes:
        - to: 10.0.34.0/30
          via: 10.0.30.1
        - to: 10.0.23.0/24
          via: 10.0.30.1
        - to: 10.0.12.0/24
          via: 10.0.30.1
```

The FreeRADIUS service is enabled and active in the final state. Its service status confirms a successful pre-start configuration check and a normal running state.

```
radius-splunk: systemctl status freeradius --no-pager
Loaded: loaded (/usr/lib/systemd/system/freeradius.service; enabled; preset: enabled)
Active: active (running) since Tue 2026-03-24 04:24:32 UTC
```

```
Main PID: 1162 (freeradius)
Status: "Processing requests"

Recent lines:
Compiling Post-Auth-Type Client-Lost for attr Post-Auth-Type
radiusd: #### Skipping IP addresses and Ports ####
Configuration appears to be OK
Started freeradius.service - FreeRADIUS multi-protocol policy server.
```

The explicit configuration validation command was also executed successfully in the final state.

```
radius-splunk: sudo freeradius -XC
FreeRADIUS Version 3.2.5
Starting - reading configuration files ...
Configuration appears to be OK
```

The local authorization source contains the lab user entry required for the controlled 802.1X workflow. Only the relevant excerpt is reproduced below, with the shared credential redacted. The full authoritative excerpt is referenced later in Appendix A (see Section A.2.5).

```
radius-splunk: grep excerpt from /etc/freeradius/3.0/mods-config/files/authorize
205-#############################################################
206-
207:nacuser Cleartext-Password := "<REDACTED>"
```

The RADIUS client definition for the Linux-based authenticator is reproduced below, again with the shared secret redacted. The authoritative appendix target for this excerpt is Section A.2.4.

```
radius-splunk: grep excerpt from /etc/freeradius/3.0/clients.conf
354:client access-node {
355-    ipaddr = 10.0.23.1
356-    secret = <REDACTED>
357:    shortname = access-node
358-}
```

A deterministic local RADIUS test using `radtest` confirms that the configured lab user is accepted by the running service. The command line has been redacted where necessary, but the full output is reproduced below.

```
radius-splunk: local radtest
radtest nacuser '<REDACTED_PASSWORD>' 127.0.0.1 0 '<REDACTED_SHARED_SECRET>'
Sent Access-Request Id 54 from 0.0.0.0:45969 to 127.0.0.1:1812 length 77
        User-Name = "nacuser"
        User-Password = "<REDACTED>"
        NAS-IP-Address = 127.0.1.1
        NAS-Port = 0
        Message-Authenticator = 0x00
```

```
        Cleartext-Password = "<REDACTED>"
Received Access-Accept Id 54 from 127.0.0.1:1812 to 127.0.0.1:45969 length 38
        Message-Authenticator = 0xe9369309c447d3d395f2e7af45be62b5
```

Taken together, these artifacts establish that the final FreeRADIUS posture is internally consistent, locally testable, and suitable for the later remote validation and incident scenario sections.

5.4 hostapd Wired Authenticator Final Configuration

The wired authenticator role is implemented on `access-node` using `hostapd` in wired mode. This is one of the central design decisions in the lab because it allows the environment to model the IEEE 802.1X control path in a Linux-based virtualized setting while remaining explicit that this does not provide full parity with managed enterprise switch behavior.

The final interface posture of `access-node` is reproduced below.

```
access-node: ip -br a
lo               UNKNOWN        127.0.0.1/8 ::1/128
ens33            UP             192.168.195.141/24 metric 100 fe80::20c:29ff:feef:5e82/64
ens38            UP             10.0.12.1/24 fe80::20c:29ff:feef:5e8c/64
ens39            UP             10.0.23.1/24 fe80::20c:29ff:feef:5e96/64
```

```
access-node: ip route
default via 192.168.195.2 dev ens33 proto dhcp src 192.168.195.141 metric 100
10.0.12.0/24 dev ens38 proto kernel scope link src 10.0.12.1
10.0.23.0/24 dev ens39 proto kernel scope link src 10.0.23.1
10.0.30.0/24 via 10.0.23.2 dev ens39 proto static
192.168.195.0/24 dev ens33 proto kernel scope link src 192.168.195.141 metric 100
192.168.195.2 dev ens33 proto dhcp scope link src 192.168.195.141 metric 100
```

```
access-node: /etc/netplan/50-cloud-init.yaml
network:
  version: 2
  ethernets:
    ens33:
      dhcp4: true
    ens38:
      dhcp4: false
      addresses:
        - 10.0.12.1/24
    ens39:
      dhcp4: false
      addresses:
        - 10.0.23.1/24
      routes:
        - to: 10.0.30.0/24
```

```
          via: 10.0.23.2

access-node: sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 1
```

The final `hostapd` wired configuration is reproduced below, with the shared secret redacted. The authoritative appendix target for this file is Section A.2.1.

```
access-node: /etc/hostapd/lab/hostapd-wired.conf
interface=ens38
driver=wired

ieee8021x=1
eapol_version=2
eap_reauth_period=3600

auth_server_addr=10.0.30.10
auth_server_port=1812
auth_server_shared_secret=<REDACTED>

own_ip_addr=10.0.23.1

logger_stdout=-1
logger_stdout_level=0
logger_syslog=-1
logger_syslog_level=0

ctrl_interface=/var/run/hostapd
ctrl_interface_group=0
```

To make the authenticator role operationally realistic and manageable, the lab does not rely solely on ad hoc foreground invocations. Instead, the final design includes a dedicated systemd unit, reproduced below and referenced later in Appendix A (see Section A.2.2).

```
access-node: /etc/systemd/system/hostapd-wired.service
[Unit]
Description=hostapd wired 802.1X authenticator (lab)
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
ExecStart=/usr/sbin/hostapd -dd /etc/hostapd/lab/hostapd-wired.conf
Restart=on-failure
RestartSec=2

[Install]
```

```
WantedBy=multi-user.target
```

The final service posture confirms that the unit is enabled and running, and that it is bound to the intended interface and RADIUS server.

```
access-node: systemctl status hostapd-wired.service --no-pager
Loaded: loaded (/etc/systemd/system/hostapd-wired.service; enabled; preset: enabled)
Active: active (running) since Tue 2026-03-24 04:20:41 UTC
Main PID: 975 (hostapd)

Recent lines:
Using interface ens38 with hwaddr <REDACTED> and ssid ""
ens38: RADIUS Authentication server 10.0.30.10:1812
RADIUS local address: 10.0.23.1:45833
ens38: interface state UNINITIALIZED->ENABLED
ens38: AP-ENABLED
ens38: Setup of interface done.
```

A later section of this document will use the managed `hostapd-wired.service` unit as a core part of the Ansible validation and remediation workflow. Its existence therefore has architectural significance beyond convenience: it makes the authenticator role inspectable, restartable, and automatable in a way that is operationally meaningful.

5.5 wpa_supplicant Final Configuration

The wired endpoint role is implemented on `client-1` using `wpa_supplicant`. The final-state client network posture and the final controlled configuration file are reproduced below.

```
client-1: ip -br a
lo               UNKNOWN        127.0.0.1/8 ::1/128
ens33            UP             10.0.12.10/24 fe80::20c:29ff:fe68:5333/64
```

```
client-1: ip route
default via 10.0.12.1 dev ens33 proto static
10.0.12.0/24 dev ens33 proto kernel scope link src 10.0.12.10
```

```
client-1: /etc/netplan/50-cloud-init.yaml
network:
  version: 2
  ethernets:
    ens33:
      addresses:
        - 10.0.12.10/24
      routes:
        - to: default
          via: 10.0.12.1
```

The final wired supplicant configuration is reproduced below, with the secret redacted. The authoritative appendix target for this file is Section A.2.3.

```
client-1: /etc/wpa_supplicant/lab/wired-8021x.conf
ctrl_interface=/run/wpa_supplicant
ap_scan=0
eapol_version=2

network={
    key_mgmt=IEEE8021X
    eap=MD5
    identity="nacuser"
    password="<REDACTED>"
}
```

In the final steady state at the time of evidence capture, the generic `wpa_supplicant.service` unit is active:

```
client-1: systemctl status wpa_supplicant --no-pager
Loaded: loaded (/usr/lib/systemd/system/wpa_supplicant.service; enabled; preset: enabled)
Active: active (running) since Tue 2026-03-24 04:57:02 UTC
Main PID: 830 (wpa_supplicant)

Recent lines:
Starting wpa_supplicant.service - WPA supplicant...
Successfully initialized wpa_supplicant
Started wpa_supplicant.service - WPA supplicant.
```

This final steady-state service condition should not be confused with the controlled test methodology used later in the incident section. During the failed-authentication simulation and subsequent remediation validation, the `wpa_supplicant` service was deliberately stopped and the binary was invoked manually in debug mode against the lab-specific wired configuration file in order to obtain deterministic, isolated evidence. The final active service state reproduced here is therefore a valid representation of the system at rest, while the incident section will separately document the intentional service stop/start choices used during testing.

5.6 Splunk Universal Forwarder Final Configuration

Splunk Universal Forwarder is installed on `radius-splunk` as a lightweight observability component. In this lab, it is not used as a full SIEM platform with local indexing and search-head analytics. Rather, it is used to demonstrate an enterprise-relevant log collection and monitoring posture for NAC-related evidence sources.

The final runtime state of the forwarder is reproduced below.

```
radius-splunk: /opt/splunkforwarder/bin/splunk status
```

```
Warning: Attempting to revert the SPLUNK_HOME ownership
Warning: Executing "chown -R root /opt/splunkforwarder"
splunkd is running (PID: 1671).
splunk helpers are running (PIDs: 1672).
```

The monitored input definitions are maintained in a dedicated local app and are reproduced in full below. The authoritative appendix target for this file is Section A.4.6.

```
radius-splunk: /opt/splunkforwarder/etc/apps/lab_nac_inputs/local/inputs.conf
[monitor:///var/log/syslog]
disabled = false
index = main
sourcetype = syslog

[monitor:///var/log/auth.log]
disabled = false
index = main
sourcetype = linux_secure

[monitor:///var/log/freeradius/radius.log]
disabled = false
index = main
sourcetype = freeradius
```

These inputs are sufficient to demonstrate that the environment includes structured monitoring of the most relevant local evidence sources for the NAC scenario:

- generic system events via `/var/log/syslog`;
- authentication and privilege-related events via `/var/log/auth.log`;
- FreeRADIUS service activity via `/var/log/freeradius/radius.log`.

This design choice is intentionally modest but meaningful. It allows the lab to claim concrete Splunk integration in a way that is accurate for the implemented scope, without overstating the environment as a full Splunk analytics deployment.

5.7 GLPI Final Configuration

GLPI is hosted on `ops-glpi` and functions as the IT service management platform for the environment. It is not present only as a demonstration of software installation; it is used directly in the operational scenario to represent both the managed endpoint asset and the authentication incident ticket.

The final-state network posture of `ops-glpi` is reproduced below.

```
ops-glpi: ip -br a
lo               UNKNOWN        127.0.0.1/8 ::1/128
```

```
ens33            UP             192.168.195.144/24 metric 100 fe80::20c:29ff:fe46:c6a1/64
ens38            UP             10.0.30.20/24 fe80::20c:29ff:fe46:c6ab/64
```

```
ops-glpi: ip route
default via 192.168.195.2 dev ens33 proto dhcp src 192.168.195.144 metric 100
10.0.12.0/24 via 10.0.30.1 dev ens38 proto static
10.0.23.0/24 via 10.0.30.1 dev ens38 proto static
10.0.30.0/24 dev ens38 proto kernel scope link src 10.0.30.20
10.0.34.0/30 via 10.0.30.1 dev ens38 proto static
192.168.195.0/24 dev ens33 proto kernel scope link src 192.168.195.144 metric 100
192.168.195.2 dev ens33 proto dhcp scope link src 192.168.195.144 metric 100
```

```
ops-glpi: /etc/netplan/50-cloud-init.yaml
network:
  version: 2
  ethernets:
    ens33:
      dhcp4: true
    ens38:
      dhcp4: false
      addresses:
        - 10.0.30.20/24
      routes:
        - to: 10.0.34.0/30
          via: 10.0.30.1
        - to: 10.0.23.0/24
          via: 10.0.30.1
        - to: 10.0.12.0/24
          via: 10.0.30.1
```

The Apache and MariaDB services are both enabled and active in the final state.

```
ops-glpi: systemctl status apache2 --no-pager
Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
Active: active (running) since Tue 2026-03-24 05:22:11 UTC
Main PID: 1296 (apache2)

Recent lines:
Starting apache2.service - The Apache HTTP Server...
Started apache2.service - The Apache HTTP Server.
```

```
ops-glpi: systemctl status mariadb --no-pager
Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; preset: enabled)
Active: active (running) since Tue 2026-03-24 05:22:12 UTC
Main PID: 1151 (mariadbd)
Status: "Taking your SQL requests now..."

Recent lines:
```

```
Server socket created on IP: '127.0.0.1', port: '3306'.
/usr/sbin/mariadbd: ready for connections.
Started mariadb.service - MariaDB 10.11.14 database server.
```

Local HTTP validation confirms that GLPI is exposed at the root path in the final Apache configuration, and that the historical `/glpi/` subpath is no longer the correct application path.

```
ops-glpi: curl -I http://127.0.0.1/
HTTP/1.1 200 OK
Date: Tue, 24 Mar 2026 05:26:56 GMT
Server: Apache/2.4.58 (Ubuntu)
Set-Cookie: <REDACTED>
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Cache-Control: no-cache, private
Vary: Accept-Encoding
Content-Type: text/html; charset=UTF-8
```

```
ops-glpi: curl -I http://127.0.0.1/glpi/
HTTP/1.1 404 Not Found
Date: Tue, 24 Mar 2026 05:28:14 GMT
Server: Apache/2.4.58 (Ubuntu)
Set-Cookie: <REDACTED>
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
x-frame-options: SAMEORIGIN
Cache-Control: no-cache, private
Content-Type: text/html; charset=UTF-8
```

The required PHP environment is present, including the previously installed `bcmath` extension required for GLPI.

```
ops-glpi: php -m | grep bcmath
bcmath
```

Operationally, GLPI is not treated as a passive accessory to the environment. In the final state, it contains:

- a managed asset record for `client-1`;
- an incident ticket documenting the simulated wired 802.1X authentication failure and remediation workflow;
- a resolved ticket state reflecting successful remediation.

The associated GLPI GUI screenshots are shown below.

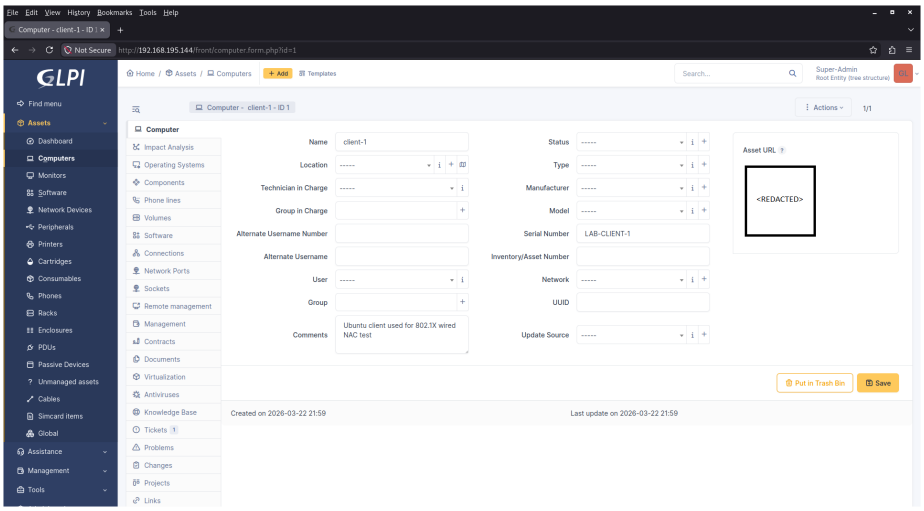

Figure 3: GLPI managed asset record for `client-1` used in the NAC incident workflow

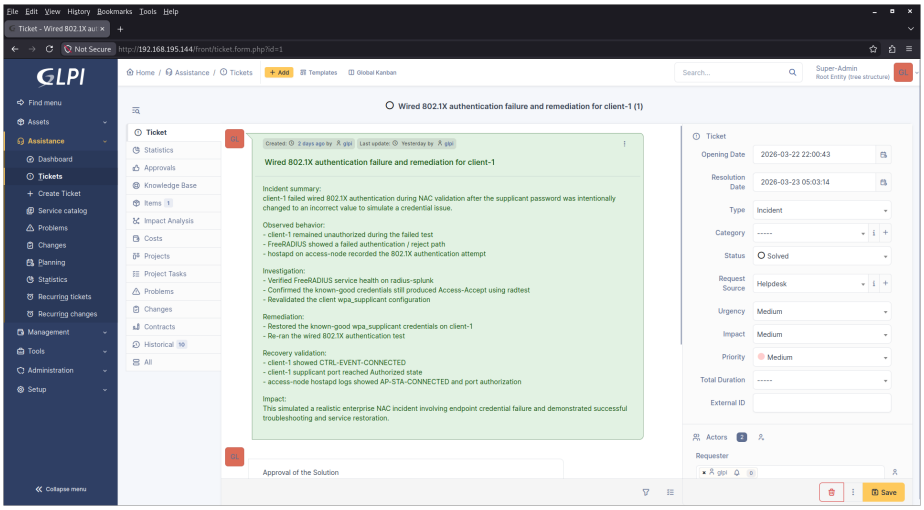

Figure 4: GLPI incident ticket for the wired 802.1X authentication failure and remediation workflow affecting `client-1`

5.8 Python and Ansible Automation Artifacts

The final platform state includes two automation artifacts that are operationally relevant to the lab:

- a Python-based incident evidence parser hosted on `radius-splunk`;
- an Ansible-based validation and optional remediation workflow, also hosted on `radius-splunk`.

These artifacts are treated as part of the implemented platform rather than as external supporting material because they are stored, executed, and maintained within the lab environment itself. Their purpose is not to inflate the apparent sophistication of the project, but to provide repeatable operational utility directly tied to the incident scenario described later in this document.

The Python parser is maintained under the local incident-evidence directory on `radius-splunk`. Its purpose is to consume the saved incident artifacts from the failed-authentication and successful-remediation workflows and produce a deterministic summary of whether the expected failure and recovery indicators are present. The authoritative full source of this parser is reproduced in Appendix A (see Section A.4.3).

The Ansible workflow is also maintained on `radius-splunk` and consists of:

- an inventory file defining the local and remote NAC-related nodes;
- a validation and optional remediation playbook capable of checking service state, collecting recent logs, and conditionally restarting the core NAC services when explicitly requested by the operator.

The automation workflow uses SSH key-based authentication rather than hard-coded credentials, consistent with the document's stated security and publication posture. The authoritative full inventory and playbook contents are reproduced in Appendix A (see Section A.4.1 and Section A.4.2).

The presence of these artifacts in the final platform state is significant for two reasons. First, they demonstrate that the environment includes repeatable operational tooling rather than only static configuration. Second, they provide a bridge between the platform build described in this section and the incident-handling and validation workflows described later in Sections 7 and 8.

6. Access Control Model and Policy Enforcement

The lab implements a deliberately segmented access model intended to resemble a simplified enterprise campus access design. Rather than treating the environment as a flat set of reachable subnets, the architecture distinguishes between authenticated endpoint access, protected services access, transit-only routing segments, and a separately governed guest segment.

This section documents the intended policy model and then validates that the environment behaves accordingly using direct path testing from both the authenticated endpoint and a dedicated guest validation node.

6.1 Policy Model Overview

The final access-control model consists of two principal endpoint trust domains:

- an **authenticated endpoint domain** represented by `client-1` on `10.0.12.0/24`, which is expected to reach protected services after successful wired 802.1X authentication;
- a **guest domain** represented by `guest-test-1` on `10.0.20.0/24`, which is expected to receive DHCP service and outbound internet access, but not internal access to protected or transit segments.

The routing and enforcement chain for authenticated access is:

`client-1` → `access-node` → `dist-router` → `pfSense` → protected services

The guest enforcement chain is simpler:

`guest-test-1` → `pfSense (GUEST)` → internet or explicitly permitted destinations

The protected services segment is represented by `10.0.30.0/24`, which hosts:

- `radius-splunk` at `10.0.30.10`;
- `ops-glpi` at `10.0.30.20`.

The guest segment is represented by `10.0.20.0/24` and is intentionally isolated from the following internal networks:

- `10.0.30.0/24` (protected services);
- `10.0.34.0/30` (pfSense transit);
- `10.0.23.0/24` (access-to-distribution transit);
- `10.0.12.0/24` (authenticated endpoint/access segment).

This is not intended to model a full enterprise NAC enforcement platform with dynamic downloadable ACLs or per-session VLAN reassignment. Instead, it demonstrates a controlled and explainable policy boundary in which wired 802.1X authentication gates access into a routed path toward protected services, while a separate guest domain is explicitly restricted by firewall policy.

6.2 Authenticated Endpoint Access Model

The authenticated endpoint path is represented by `client-1`, which resides on `10.0.12.0/24` behind the Linux-based wired authenticator. In the final steady state, `client-1` uses a static address and default route via `access-node`.

The final-state endpoint interface and routing posture are reproduced below.

```
client-1: ip -br a
lo               UNKNOWN        127.0.0.1/8 ::1/128
ens33            UP             10.0.12.10/24 fe80::20c:29ff:fe68:5333/64
```

```
client-1: ip route
default via 10.0.12.1 dev ens33 proto static
10.0.12.0/24 dev ens33 proto kernel scope link src 10.0.12.10
```

The endpoint-side `wpa_supplicant` service is active in the final steady state, which confirms that the endpoint is operating in a post-build state consistent with the intended access model.

```
client-1: systemctl status wpa_supplicant --no-pager
Loaded: loaded (/usr/lib/systemd/system/wpa_supplicant.service; enabled; preset: enabled)
Active: active (running) since Tue 2026-03-24 08:03:28 UTC
Main PID: 829 (wpa_supplicant)

Recent lines:
Starting wpa_supplicant.service - WPA supplicant...
Started wpa_supplicant.service - WPA supplicant.
Successfully initialized wpa_supplicant
```

From the endpoint perspective, successful access to the local gateway is verified first:

```
client-1: ping -c 4 10.0.12.1
4 packets transmitted, 4 received, 0% packet loss
```

This confirms that the endpoint has Layer 3 reachability to the local routed access gateway on `access-node`. More importantly, the endpoint can also reach both protected services nodes on the `10.0.30.0/24` segment.

```
client-1: ping -c 4 10.0.30.10
4 packets transmitted, 4 received, 0% packet loss
```

```
client-1: ping -c 4 10.0.30.20
4 packets transmitted, 4 received, 0% packet loss
```

Application-layer validation to GLPI is also successful from the authenticated endpoint, which is stronger evidence than ICMP alone because it confirms that the routed path reaches a live service rather than merely a responsive host.

```
client-1: curl -I --max-time 5 http://10.0.30.20/
HTTP/1.1 200 OK
Date: Tue, 24 Mar 2026 08:21:04 GMT
Server: Apache/2.4.58 (Ubuntu)
Set-Cookie: <REDACTED>
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
```

```
Pragma: no-cache
Cache-Control: no-cache, private
Vary: Accept-Encoding
Content-Type: text/html; charset=UTF-8
```

This result is particularly useful because it shows that the authenticated endpoint can not only reach the protected services subnet but can directly access the GLPI web application on the services segment.

A `tracepath` test to `10.0.30.20` also supports the multi-hop design and shows that the path is not flat:

```
client-1: tracepath 10.0.30.20
 1?: [LOCALHOST]                      pmtu 1500
 1:  _gateway                                              0.911ms
 1:  _gateway                                              0.566ms
 2:  ???                                                   1.713ms
 3:  ???                                                   2.833ms
 4:  ???                                                   3.034ms reached
     Resume: pmtu 1500 hops 4 back 4
```

Because intermediate Linux routers and pfSense do not necessarily reveal useful names or ICMP TTL-expired responses in a lab, the anonymous hop presentation is not treated as a defect. The important point is that the path is clearly multi-hop and terminates successfully at the protected service host.

6.3 Routed Path Validation for Authenticated Access

The endpoint-side tests above establish that the authenticated client can reach the protected services segment. This subsection validates the same behavior from the infrastructure perspective to show that the path is explicitly routed and not the result of accidental adjacency.

On `access-node`, the route lookup for both protected service hosts confirms that traffic is forwarded to `dist-router` at `10.0.23.2` via interface `ens39`.

```
access-node: ip route get 10.0.30.10
10.0.30.10 via 10.0.23.2 dev ens39 src 10.0.23.1 uid 1000
```

```
access-node: ip route get 10.0.30.20
10.0.30.20 via 10.0.23.2 dev ens39 src 10.0.23.1 uid 1000
```

On `dist-router`, the route lookup confirms that both protected service hosts are then forwarded toward pfSense at `10.0.34.2` via interface `ens39`.

```
dist-router: ip route get 10.0.30.10
10.0.30.10 via 10.0.34.2 dev ens39 src 10.0.34.1 uid 1000
```

```
dist-router: ip route get 10.0.30.20
10.0.30.20 via 10.0.34.2 dev ens39 src 10.0.34.1 uid 1000
```

This proves the intended forward path:

client-1 → access-node → dist-router → pfSense → 10.0.30.0/24

The return path is also explicitly validated from both protected service hosts. On radius-splunk, the route to the client subnet points to pfSense at 10.0.30.1:

```
radius-splunk: ip route get 10.0.12.10
10.0.12.10 via 10.0.30.1 dev ens38 src 10.0.30.10 uid 1000
```

On ops-glpi, the route to the client subnet is identical in principle:

```
ops-glpi: ip route get 10.0.12.10
10.0.12.10 via 10.0.30.1 dev ens38 src 10.0.30.20 uid 1000
```

This confirms the intended return path:

10.0.30.0/24 → pfSense → dist-router → access-node → client-1

Bidirectional reachability is then directly validated with ICMP from both protected service hosts back to client-1.

```
radius-splunk: ping -c 4 10.0.12.10
4 packets transmitted, 4 received, 0% packet loss
```

```
ops-glpi: ping -c 4 10.0.12.10
4 packets transmitted, 4 received, 0% packet loss
```

This combination of forward-path route lookup, return-path route lookup, and bidirectional ICMP provides strong evidence that the authenticated endpoint path is functioning as a deliberately routed design rather than as an incidental lab shortcut.

6.4 Guest Network Policy Model

The guest network is implemented as a distinct pfSense-controlled segment on 10.0.20.0/24. It is not intended to participate in the wired 802.1X access path and is instead used to demonstrate a separate, lower-trust network with DHCP service, firewall-enforced internal isolation, and outbound internet access.

The authoritative final pfSense interface definition for the guest segment was already established in Section 5.2:

- **GUEST / OPT2** (em3) at 10.0.20.254/24

The final DHCP configuration for the guest segment is manually transcribed from pfSense as follows:

```
pfSense GUEST DHCP summary
Enabled (Enable DHCP server on GUEST interface)
Subnet: 10.0.20.0/24
Subnet Range: 10.0.20.1 - 10.0.20.254
Address Pool Range: 10.0.20.100 - 10.0.20.199
```

The associated pfSense GUI screenshot is shown below.

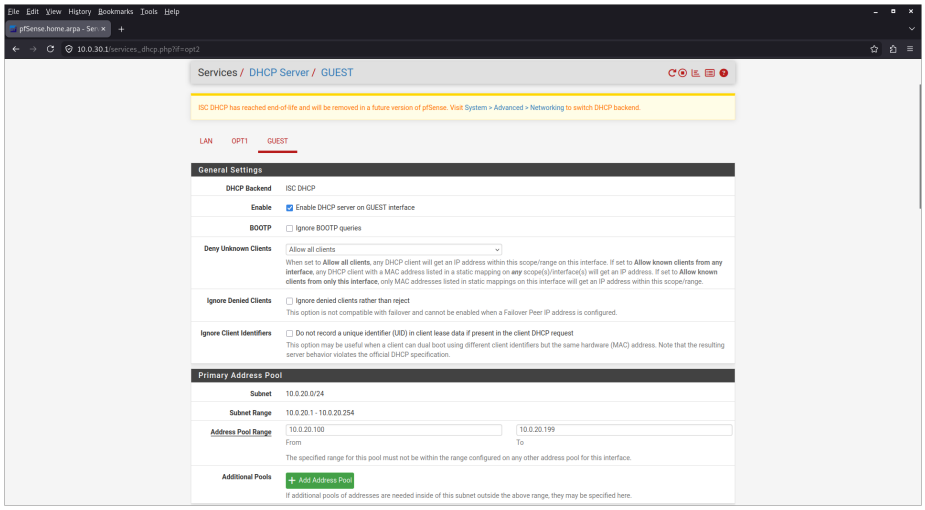

Figure 5: pfSense GUEST DHCP server configuration for the policy-based guest access segment

The guest firewall policy is implemented as an ordered first-match ruleset on the pfSense GUEST interface. The final manually transcribed rules are reproduced below and were explicitly re-confirmed as unchanged and in the same top-to-bottom order at the time of Section 6 validation.

```
pfSense GUEST rules (top-to-bottom)
Pass  | IPv4* | Source: 10.0.20.0/24 | Destination: This Firewall (Self)
        | ALLOW-GUEST-TO-PFSENSE
Block | IPv4* | Source: 10.0.20.0/24 | Destination: 10.0.30.0/24
        | BLOCK-GUEST-TO-SERVICES
Block | IPv4* | Source: 10.0.20.0/24 | Destination: 10.0.34.0/30
        | BLOCK-GUEST-TO-TRANSIT
Block | IPv4* | Source: 10.0.20.0/24 | Destination: 10.0.23.0/24
        | BLOCK-GUEST-TO-DIST
Block | IPv4* | Source: 10.0.20.0/24 | Destination: 10.0.12.0/24
        | BLOCK-GUEST-TO-AUTH-ACCESS
Pass  | IPv4* | Source: 10.0.20.0/24 | Destination: *
        | ALLOW-GUEST-TO-INTERNET-AND-OTHER-NON-INTERNAL
```

The associated pfSense GUI screenshot is shown below.

This ruleset is deliberately explicit. Rather than relying on an implicit deny plus a broad allow, the internal networks are individually enumerated and blocked before the final permissive rule. That makes the guest policy easier to explain and easier to audit.

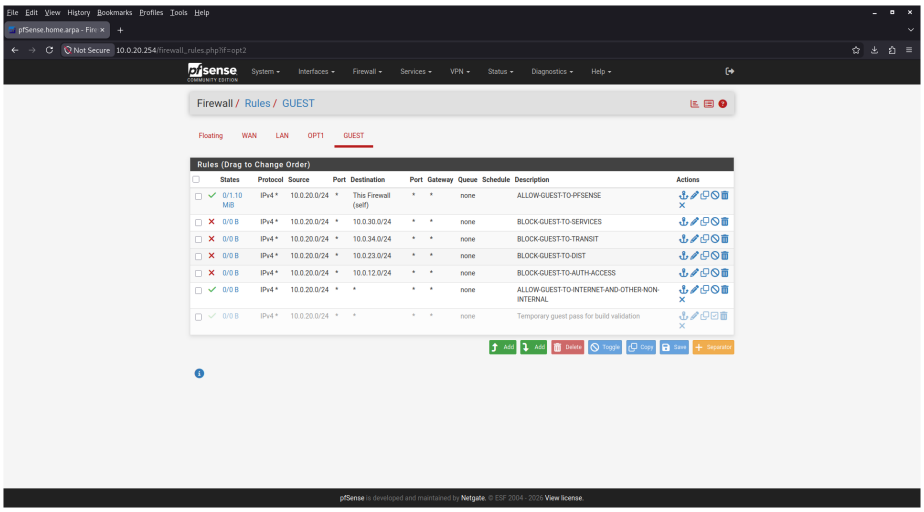

Figure 6: pfSense GUEST interface firewall rules enforcing internal isolation and controlled outbound access

At the time of validation, pfSense outbound NAT was confirmed to be in Automatic mode, which is relevant because it explains why the guest network can successfully access the internet without requiring separately documented manual NAT rules.

6.5 Guest DHCP and Isolation Validation

To validate the guest policy empirically rather than only by configuration review, a dedicated disposable guest-only test VM named `guest-test-1` was created and attached solely to the pfSense GUEST segment. Importantly, this VM was provisioned with a single interface only and no management/NAT interface, so that its observed behavior reflects the actual guest policy boundary rather than an alternate administrative path.

The guest validation node received a DHCP lease from pfSense in the expected guest subnet and within the configured guest address pool.

```
guest-test-1: ip -br a
lo               UNKNOWN        127.0.0.1/8 ::1/128
ens33            UP             10.0.20.137/24 fe80::250:56ff:feab:cdef/64
```

```
guest-test-1: ip route
default via 10.0.20.254 dev ens33 proto dhcp src 10.0.20.137 metric 100
10.0.20.0/24 dev ens33 proto kernel scope link src 10.0.20.137
10.0.20.254 dev ens33 proto dhcp scope link src 10.0.20.137 metric 100
```

An explicit DHCP renewal confirms the expected pfSense DHCP behavior and

the lease origin.

```
guest-test-1: sudo dhclient -v ens33
DHCPDISCOVER on ens33 to 255.255.255.255 port 67 interval 3 (xid=0x4a3f2c1d)
DHCPDISCOVER on ens33 to 255.255.255.255 port 67 interval 6 (xid=0x4a3f2c1d)
DHCPOFFER of 10.0.20.137 from 10.0.20.254
DHCPREQUEST for 10.0.20.137 on ens33 to 255.255.255.255 port 67 (xid=0x4a3f2c1d)
DHCPACK of 10.0.20.137 from 10.0.20.254 (xid=0x4a3f2c1d)
bound to 10.0.20.137 -- renewal in 3600 seconds.
```

The guest node can successfully reach the pfSense GUEST gateway itself, which is consistent with the first allow rule in the guest ruleset.

```
guest-test-1: ping -c 4 10.0.20.254
4 packets transmitted, 4 received, 0% packet loss
```

The guest node is then tested against each explicitly blocked internal segment. All of these attempts fail as intended.

Protected services segment (`10.0.30.0/24`):

```
guest-test-1: ping -c 4 10.0.30.10
4 packets transmitted, 0 received, 100% packet loss
```

```
guest-test-1: ping -c 4 10.0.30.20
4 packets transmitted, 0 received, 100% packet loss
```

Transit segment toward pfSense (`10.0.34.0/30`):

```
guest-test-1: ping -c 4 10.0.34.1
4 packets transmitted, 0 received, 100% packet loss
```

Access-to-distribution transit (`10.0.23.0/24`):

```
guest-test-1: ping -c 4 10.0.23.1
4 packets transmitted, 0 received, 100% packet loss
```

Authenticated access segment (`10.0.12.0/24`):

```
guest-test-1: ping -c 4 10.0.12.1
4 packets transmitted, 0 received, 100% packet loss
```

Application-layer access to GLPI from the guest node also fails by timeout, which is a particularly strong validation point because it demonstrates that the guest cannot reach a protected service even when directly attempting the published web application path.

```
guest-test-1: curl -I --max-time 5 http://10.0.30.20/
curl: (28) Connection timed out after 5001 milliseconds
```

This confirms that the guest segment is not only blocked from protected hosts at the ICMP layer but is also effectively prevented from reaching the protected application service.

6.6 Guest Internet Access Validation

Because the guest ruleset ends with an explicit allow rule for destinations outside the enumerated internal networks, it is useful to validate that the guest network retains general outbound access rather than being globally constrained.

The guest validation node can successfully reach a public IP address:

```
guest-test-1: ping -c 4 1.1.1.1
4 packets transmitted, 4 received, 0% packet loss
```

It can also successfully retrieve HTTP headers from an external public site:

```
guest-test-1: curl -I --max-time 5 http://www.google.com
HTTP/1.1 200 OK
Content-Type: text/html; charset=ISO-8859-1
...
```

This behavior is consistent with the intended guest policy model:

- access to pfSense itself is permitted;
- access to enumerated internal networks is blocked;
- general outbound access is permitted.

In other words, the guest segment is not simply "isolated" in the abstract. It is specifically implemented as a constrained but functional internet-facing guest network.

6.7 Policy Interpretation and Limitations

The validation results in this section support the following conclusions:

1. The authenticated endpoint path is functional and routed as designed.
2. The protected services segment is reachable from the authenticated endpoint and from no guest path tested.
3. The guest network receives DHCP service correctly and is restricted from the enumerated internal networks.
4. The guest network remains operational for general outbound internet access.

This is a meaningful and defensible policy model for the scope of the lab. However, it is important to state its limits precisely.

First, the guest isolation is enforced by a conventional firewall ruleset on pfSense, not by dynamic NAC policy derived from endpoint identity or posture. That is acceptable for this design because the guest network is intentionally modeled as a separate trust zone rather than as a dynamically assigned 802.1X result.

Second, the authenticated endpoint path is validated at the network and application layers, but the environment does not implement per-session VLAN assignment, downloadable ACLs, posture checks, or certificate-based EAP methods. Those omissions are addressed later in the limitations section and do not invalidate the current demonstration.

Third, the Linux `hostapd` wired authenticator is used as a practical lab approximation of a switch-based 802.1X authenticator. It is appropriate for demonstrating the control-plane exchange and the operational workflow, but it should not be misrepresented as a drop-in replacement for enterprise access switch behavior.

Within those stated limits, the access-control model is internally consistent, technically validated, and appropriate for inclusion in a lab document that aims to present a realistic but accurately scoped virtualized NAC implementation.

6.8 Wireshark Cross-Reference for Policy Validation

Although the full packet-analysis narrative is reserved for the dedicated packet capture section later in this document, the Wireshark evidence already supports the access-control interpretation presented here.

At the time of review, the packet captures were confirmed to show that the authenticated endpoint traffic is preceded by EAPOL exchange rather than immediately presenting as ordinary post-authenticated data-plane traffic. This is important because it visually reinforces the intended distinction between:

- the **pre-authentication control plane**, represented by EAPOL and RADIUS-driven 802.1X exchange; and
- the **post-authentication data plane**, represented by ordinary IP traffic after successful authorization.

The screenshot inventory includes all four relevant packet-capture perspectives:

- failed client-side capture;
- failed authenticator-side capture;
- success client-side capture;
- success authenticator-side capture.

These artifacts will be incorporated in the dedicated packet-capture analysis section rather than duplicated here. Their relevance in this section is to reinforce that the access model is not inferred only from IP reachability tests; it is also observable at the protocol level through the expected 802.1X control-plane behavior.

7. Incident Scenario: Wired 802.1X Authentication Failure and Remediation

The final lab environment is not presented solely as a static implementation. It is also used to simulate and document a realistic operational incident in which a wired endpoint fails 802.1X authentication, remains unauthorized, is investigated through multiple evidence sources, and is restored to service through a controlled remediation workflow.

This section documents that incident as the authoritative operational scenario for the environment. The incident validation date used throughout this section is 2026-03-22.

7.1 Incident Definition

The incident concerns the managed endpoint `client-1`, which is the same endpoint represented in GLPI and used throughout the access-control validation workflow. The endpoint normally authenticates through the following control path:

`client-1` → `access-node` → `radius-splunk`

Within this path:

- `client-1` acts as the wired supplicant;
- `access-node` acts as the Linux-based wired 802.1X authenticator using `hostapd`;
- `radius-splunk` hosts FreeRADIUS as the authentication and policy service.

The incident was defined as a wired 802.1X authentication failure caused by invalid supplicant credentials. This was not an accidental outage but a controlled failure injection introduced for the purpose of validating the lab's ability to:

- detect a failed NAC event;
- observe the failed state across the supplicant, authenticator, and RADIUS layers;
- document the event in an ITSM workflow;
- restore service using a known-good configuration;
- validate successful recovery through both host-side evidence and packet analysis.

The scenario therefore serves as a practical proof that the lab can do more than demonstrate component reachability. It can also support a realistic access-control failure narrative from symptom to closure.

7.2 Failure Injection Method

The failure was introduced by deliberately replacing the correct wired supplicant password in the `client-1` `wpa_supplicant` configuration with an incorrect value. This method was chosen because it is deterministic, easy to revert, and directly relevant to the type of endpoint-side credential error that could plausibly occur in an enterprise environment.

This approach has several advantages in a proof-of-concept lab:

- it guarantees a failure condition at the authentication layer rather than at unrelated network layers;
- it preserves the normal routing and firewall design, isolating the incident to the NAC control path;
- it makes the remediation equally deterministic by restoring the known-good client configuration;
- it allows clean collection of failure and recovery evidence without introducing ambiguity about the cause of the event.

The lab therefore treats the credential modification as a formal failure-injection technique rather than as an incidental misconfiguration. This is important because it makes the incident reproducible and allows the later automation and packet-analysis workflows to operate against a known and explainable failure mode.

7.3 Authoritative Incident Artifacts

The incident evidence set is maintained on `radius-splunk` under the local incident-evidence directory. The final directory inventory at the time of documentation is reproduced below.

```
radius-splunk: ls -lh ~/incident-evidence
total 92K
-rw-rw-r-- 1 nac-lab nac-lab  32K Mar 23 04:02 freeradius_failed_auth_incident.txt
-rw-rw-r-- 1 nac-lab nac-lab  11K Mar 23 04:01 hostapd_failed_auth_incident.txt
-rw-rw-r-- 1 nac-lab nac-lab  19K Mar 23 04:01 hostapd_success_retest_v2.txt
-rw-r--r-- 1 root    root    3.5K Mar 24 09:52 nac_incident_parser.py
-rw-rw-r-- 1 nac-lab nac-lab  535 Mar 23 04:28 nac_incident_parser_success.txt
-rw-rw-r-- 1 nac-lab nac-lab 7.3K Mar 23 03:59 wpa_supplicant_failed_auth_incident.txt
-rw-rw-r-- 1 nac-lab nac-lab 8.4K Mar 23 03:59 wpa_supplicant_success_retest_v2.txt
```

These files are sufficient to support the full narrative of failure and recovery across the endpoint, authenticator, and RADIUS layers. They also form the input set for the Python incident parser.

At the time of final documentation, the parser remained functional and produced the following authoritative summary:

```
radius-splunk: python3 ~/incident-evidence/nac_incident_parser.py
=== NAC Incident Evidence Summary ===

[1] Failed authentication simulation
- Client remained unauthorized: YES
- Client showed successful connection during failed test: NO
- hostapd saw authentication attempt: YES
- FreeRADIUS recorded failed auth / reject path: YES

[2] Remediation / successful re-test
- Client reached CTRL-EVENT-CONNECTED: YES
- Client reached Authorized state: YES
- hostapd showed successful authorization indicators: YES

[3] Incident conclusion
- SIMULATED NAC INCIDENT SUCCESSFULLY REPRODUCED AND REMEDIATED
```

This parser output is significant because it provides a deterministic, cross-file synthesis of the incident evidence. It does not replace the underlying artifacts, but it does provide a concise operational summary that agrees with the raw evidence reproduced below.

The full parser source is maintained separately as a platform artifact in Appendix A (see Section A.4.3).

7.4 Failed Authentication Evidence

The failed state is established independently at all three control points: the supplicant, the authenticator, and the RADIUS service.

7.4.1 Supplicant-Side Failed State

On the endpoint side, the failed-authentication capture shows that the supplicant remained unauthorized and never transitioned into a connected or authorized state.

```
client-side failed indicators
EAPOL: Supplicant port status: Unauthorized
EAPOL: Supplicant port status: Unauthorized
```

This is the clearest endpoint-side symptom of the failure. There is no successful connection indicator in the failed-authentication evidence set.

7.4.2 Authenticator-Side Failed State

On the authenticator side, `hostapd` recorded repeated attempts to send the authentication transaction toward the RADIUS server, confirming that the

endpoint was actively attempting authentication and that the authenticator was processing the request path.

```
hostapd failed-auth indicators
Mar 22 22:32:33 access-node hostapd[2052]: ens38: RADIUS Sending RADIUS message to
        authentication server
Mar 22 22:32:33 access-node hostapd[2052]: ens38: RADIUS Sending RADIUS message to
        authentication server
Mar 22 22:32:33 access-node hostapd[2052]: ens38: RADIUS Sending RADIUS message to
        authentication server
Mar 22 22:32:33 access-node hostapd[2052]: ens38: RADIUS Sending RADIUS message to
        authentication server
```

This is operationally important because it shows that the failure was not caused by a silent client or by an absence of authenticator activity. The authenticator was actively participating in the 802.1X control path and attempting to obtain a RADIUS decision.

7.4.3 RADIUS-Side Failed State

The strongest proof of the failure appears in the FreeRADIUS evidence. The failed-authentication excerpt shows an EAP failure, rejection of the authentication attempt, and the use of the reject post-auth path.

```
FreeRADIUS failed-auth indicators
(1) eap: Sending EAP Failure (code 4) ID 26 length 4
(1)     [eap] = reject
(1)   } # authenticate = reject
(1) Failed to authenticate the user
(1) Using Post-Auth-Type Reject
(1)   Post-Auth-Type REJECT {
(1) attr_filter.access_reject: EXPAND %{User-Name}
(1) attr_filter.access_reject:    --> nacuser
(1) attr_filter.access_reject: Matched entry DEFAULT at line 11
(1)     [attr_filter.access_reject] = updated
(1)   } # Post-Auth-Type REJECT = updated
```

This confirms that the failed state is not merely an endpoint perception. The policy authority itself rejected the authentication request and generated the expected EAP failure path.

Taken together, the endpoint, authenticator, and RADIUS evidence establish the failed state conclusively.

7.5 Investigation and Triage

The investigation phase in this incident was intentionally simple but operationally credible. The goal was to distinguish among the most likely causes of a failed wired 802.1X event:

- an endpoint-side credential or supplicant configuration error;
- an authenticator-side service or forwarding problem;
- a RADIUS-side policy or service failure.

The incident evidence and the surrounding validation steps supported the following triage conclusions:

- the endpoint remained unauthorized, so the event was not a mere transient or cosmetic log anomaly;
- `hostapd` was actively sending RADIUS requests, which meant the authenticator path was engaged;
- FreeRADIUS was running and able to produce a deterministic reject path, which meant the backend service was operational;
- the known-good user credentials still produced Access-Accept under `radtest`, isolating the problem to the client-side credential state rather than to the FreeRADIUS policy model.

This investigation logic is reflected directly in the GLPI incident record, which documents verification of FreeRADIUS health, confirmation that the known-good credentials still produced Access-Accept, and revalidation of the client `wpa_supplicant` configuration.

7.6 Remediation Actions

The remediation was deliberately minimal and targeted. Rather than changing the architecture, altering the firewall rules, or modifying RADIUS policy, the known-good `wpa_supplicant` credential state on `client-1` was restored and the wired 802.1X authentication test was executed again under controlled conditions.

This remediation method is significant because it demonstrates a proper engineering response to a localized failure:

- identify the probable failing control point;
- restore known-good state at that control point;
- re-test the full path rather than assuming success;
- validate the result across multiple evidence sources.

No broad changes to routing, pfSense policy, or FreeRADIUS user logic were required to restore service. That outcome is itself meaningful because it shows the failure was introduced and resolved at the endpoint credential layer rather than by systemic platform instability.

7.7 Successful Re-Test Evidence

The remediation was validated through a clean success re-test. As with the failed state, the successful state is visible at all three control points.

7.7.1 Supplicant-Side Recovery Evidence

The successful re-test capture shows the expected transition from initial unauthorized state to connected and authorized state, followed by a later return to unauthorized state when the controlled test process ended.

```
client-side success indicators
EAPOL: Supplicant port status: Unauthorized
ens33: CTRL-EVENT-CONNECTED - Connection to <REDACTED> completed [id=0 id_str=]
EAPOL: Supplicant port status: Authorized
EAPOL: Supplicant port status: Unauthorized
```

The final `Unauthorized` line is not interpreted as a failed remediation. It is consistent with termination of the controlled test process after the successful exchange had already been completed. The decisive indicators are `CTRL-EVENT-CONNECTED` and `Authorized`.

7.7.2 Authenticator-Side Recovery Evidence

The successful re-test on the authenticator shows the expected sequence of RADIUS interaction, Access-Accept handling, EAP Success decapsulation, station connection, and port authorization.

```
hostapd success indicators
Mar 23 02:49:00 access-node hostapd[2052]: Encapsulating EAP message into a RADIUS
        packet
Mar 23 02:49:00 access-node hostapd[2052]: Copied RADIUS State Attribute
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Sending RADIUS message to
        authentication server
Mar 23 02:49:00 access-node hostapd[2052]: RADIUS message: code=1 (Access-Request)
        identifier=4 length=193
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X:
        unauthorizing port
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Next RADIUS client retransmit in
        3 seconds
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Received 53 bytes from RADIUS
        server
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Received RADIUS message
Mar 23 02:49:00 access-node hostapd[2052]: RADIUS message: code=2 (Access-Accept)
        identifier=4 length=53
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> RADIUS: Received
        RADIUS packet matched with a pending request, round trip time 0.00 sec
Mar 23 02:49:00 access-node hostapd[2052]: RADIUS packet matching with station
```

```
        <REDACTED>
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X:
        decapsulated EAP packet (code=3 id=255 len=4) from RADIUS server: EAP Success
Mar 23 02:49:00 access-node hostapd[2052]: ens38: CTRL-EVENT-EAP-SUCCESS2 <REDACTED>
Mar 23 02:49:00 access-node hostapd[2052]: ens38: AP-STA-CONNECTED <REDACTED>
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X:
        authorizing port
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> RADIUS: starting
        accounting session 2268479955CB85A0
```

This is the strongest authenticator-side proof of successful remediation. The sequence explicitly shows: - RADIUS request handling, - Access-Accept reception, - EAP Success, - station connection, - and port authorization.

7.7.3 RADIUS-Side Recovery Interpretation

A dedicated successful FreeRADIUS transaction excerpt was not preserved in the same compact text form as the failed-authentication reject path. However, this does not weaken the conclusion because the success state is already corroborated by:

- the endpoint-side connected and authorized state;
- the authenticator-side Access-Accept and EAP Success handling;
- the parser summary, which concluded that the incident was successfully reproduced and remediated.

In this case, the successful RADIUS outcome is most directly evidenced through the authenticator's decoding of the received RADIUS Accept and EAP Success path.

7.8 GLPI Incident Record

The incident was not left as a purely technical artifact. It was also recorded in GLPI as an actual operational record associated with the lab asset model. This is important because it demonstrates that the environment integrates ITSM process with technical evidence rather than treating ticketing as an unrelated side service.

At the time of documentation, the incident ticket title and status were as follows:

```
Ticket title:
Wired 802.1X authentication failure and remediation for client-1

Ticket status:
Solved
```

The full incident text recorded in GLPI is reproduced below.

```
Incident summary:
client-1 failed wired 802.1X authentication during NAC validation after the supplicant
        password was intentionally changed to an incorrect value to simulate a credential
        issue.

Observed behavior:
- client-1 remained unauthorized during the failed test
- FreeRADIUS showed a failed authentication / reject path
- hostapd on access-node recorded the 802.1X authentication attempt

Investigation:
- Verified FreeRADIUS service health on radius-splunk
- Confirmed the known-good credentials still produced Access-Accept using radtest
- Revalidated the client wpa_supplicant configuration

Remediation:
- Restored the known-good wpa_supplicant credentials on client-1
- Re-ran the wired 802.1X authentication test

Recovery validation:
- client-1 showed CTRL-EVENT-CONNECTED
- client-1 supplicant port reached Authorized state
- access-node hostapd logs showed AP-STA-CONNECTED and port authorization

Impact:
This simulated a realistic enterprise NAC incident involving endpoint credential failure
        and demonstrated successful troubleshooting and service restoration.
```

For associated GLPI GUI screenshots, see Section 5.7.

7.9 Incident Conclusion

The incident scenario demonstrates that the lab can support a complete wired NAC failure-and-recovery lifecycle:

1. a controlled authentication failure was introduced at the endpoint credential layer;
2. the endpoint remained unauthorized;
3. the authenticator processed and forwarded the failed authentication path;
4. FreeRADIUS generated the expected reject and EAP failure path;
5. the issue was investigated without unnecessary architectural changes;
6. the known-good client configuration was restored;
7. a clean re-test demonstrated connected and authorized state;

8. the event was recorded and closed in GLPI.

This is one of the most important outcomes in the document because it transforms the lab from a static collection of services into an operational environment that can support:

- failure simulation;
- layered evidence collection;
- repeatable remediation;
- cross-domain validation using routing, logging, ITSM, automation, and packet capture.

The later automation and packet-capture sections build directly on this incident narrative. The Python parser summarizes the same evidence set reproduced here, and the Wireshark screenshots provide protocol-level confirmation of both the failed and successful authentication paths.

8. Automation Workflows

The lab includes a lightweight but operationally meaningful automation layer designed to support repeatable validation of the wired 802.1X control path and the associated authentication services. Rather than attempting to implement a full orchestration or SOAR platform, the automation focuses on three practical goals:

- validate the state of the authenticator and RADIUS services from a central node;
- optionally remediate service availability by restarting critical daemons;
- preserve local evidence artifacts for later review and appendix inclusion.

This design is best characterized as validation-first with optional service remediation. That phrasing is deliberate. The automation is primarily an evidence and control workflow, not a blind auto-restart mechanism.

The automation controller for this lab is `radius-splunk`. From that node, Ansible is used to validate the Linux-based wired 802.1X authenticator on `access-node` and the FreeRADIUS service on `radius-splunk` itself. A separate Python parser is then used to summarize the incident evidence set collected during the failure-and-remediation scenario described in Section 7.

8.1 Automation Asset Inventory

The final automation workspace resides under `/home/nac-lab/ansible-nac` on `radius-splunk`. At the time of final documentation, the file layout was as follows.

```
radius-splunk: find ~/ansible-nac -maxdepth 3 -type f | sort
/home/nac-lab/ansible-nac/artifacts/access-node_hostapd_logs.txt
/home/nac-lab/ansible-nac/artifacts/access-node_hostapd_restart_result.txt
/home/nac-lab/ansible-nac/artifacts/access-node_hostapd_status_after.txt
/home/nac-lab/ansible-nac/artifacts/access-node_hostapd_status_before.txt
/home/nac-lab/ansible-nac/artifacts/access-node_hostapd_status.txt
/home/nac-lab/ansible-nac/artifacts/radius-splunk_freeradius_journal.txt
/home/nac-lab/ansible-nac/artifacts/radius-splunk_freeradius_radiuslog.txt
/home/nac-lab/ansible-nac/artifacts/radius-splunk_freeradius_restart_result.txt
/home/nac-lab/ansible-nac/artifacts/radius-splunk_freeradius_status_after.txt
/home/nac-lab/ansible-nac/artifacts/radius-splunk_freeradius_status_before.txt
/home/nac-lab/ansible-nac/artifacts/radius-splunk_freeradius_status.txt
/home/nac-lab/ansible-nac/inventory/hosts.ini
/home/nac-lab/ansible-nac/playbooks/nac_validation.yml
```

The current artifact directory contents are reproduced below.

```
radius-splunk: ls -lh ~/ansible-nac/artifacts
total 36K
-rw-r--r-- 1 nac-lab nac-lab 3.5K Mar 24 12:23 access-node_hostapd_logs.txt
-rw-r--r-- 1 nac-lab nac-lab    0 Mar 24 12:22 access-node_hostapd_restart_result.txt
-rw-r--r-- 1 nac-lab nac-lab 1.4K Mar 24 12:23 access-node_hostapd_status_after.txt
-rw-r--r-- 1 nac-lab nac-lab 1.4K Mar 24 12:23 access-node_hostapd_status_before.txt
-rw-r--r-- 1 root    root    1.4K Mar 23 19:30 access-node_hostapd_status.txt
-rw-r--r-- 1 nac-lab nac-lab 4.0K Mar 24 12:23 radius-splunk_freeradius_journal.txt
-rw-r--r-- 1 nac-lab nac-lab 3.1K Mar 24 12:23 radius-splunk_freeradius_radiuslog.txt
-rw-r--r-- 1 nac-lab nac-lab    0 Mar 24 12:22 radius-splunk_freeradius_restart_result.txt
-rw-r--r-- 1 nac-lab nac-lab 1.8K Mar 24 12:23 radius-splunk_freeradius_status_after.txt
-rw-r--r-- 1 nac-lab nac-lab 1.8K Mar 24 12:23 radius-splunk_freeradius_status_before.txt
-rw-r--r-- 1 root    root    1.8K Mar 23 19:30 radius-splunk_freeradius_status.txt
```

This inventory is useful for two reasons. First, it shows that the automation is not ephemeral; it leaves behind auditable local outputs. Second, it shows a visible distinction between earlier legacy artifacts created under a pre-revision execution model and the current artifacts created by the final, corrected playbook as `nac-lab`.

The authoritative revised playbook and the Python incident parser were both confirmed in place with the following timestamps and sizes:

```
radius-splunk: ls -lh ~/ansible-nac/playbooks/nac_validation.yml
-rw-r--r-- 1 root root 5.6K Mar 24 12:19 /home/nac-lab/ansible-nac/playbooks/
        nac_validation.yml

radius-splunk: ls -lh ~/incident-evidence/nac_incident_parser.py
-rw-r--r-- 1 root root 3.5K Mar 24 09:52 /home/nac-lab/incident-evidence/
        nac_incident_parser.py
```

The full inventory file, the authoritative revised playbook, and the full parser

source are reproduced in Appendix A (see Section A.4.1, Section A.4.2, and Section A.4.3).

8.2 Playbook Design and Revision Rationale

The final Ansible workflow uses a single playbook, `nac_validation.yml`, which was intentionally kept compact and understandable. Its functional responsibilities are:

1. collect a basic hostname from each target;
2. validate the `hostapd-wired.service` state on `access-node`;
3. validate the `freeradius.service` state on `radius-splunk`;
4. optionally restart one or both services when `remediate=true` is provided;
5. re-check service state after optional remediation;
6. collect recent journal excerpts and recent FreeRADIUS application log output;
7. write all collected outputs into a local artifact directory on the control node.

The playbook underwent a meaningful correction during final validation and the corrected version is treated as the authoritative implementation for this document.

Specifically, the final playbook:

- sets `become: false` at the play scope;
- applies `become: true` only to tasks that require elevated privileges;
- uses a fixed `artifact_dir` variable (`/home/nac-lab/ansible-nac/artifacts`);
- explicitly sets `become: false` on delegated localhost copy tasks.

This revision is not a cosmetic change. It materially improves operational correctness by preventing two undesirable behaviors:

- unnecessary sudo prompts for harmless non-privileged tasks;
- accidental dependence on the invoking user's `HOME` when writing delegated localhost artifacts.

In practical terms, the revised playbook makes execution deterministic and avoids the SSH and artifact-path ambiguity that can occur when operators attempt to run Ansible itself under `sudo`. The final document therefore treats the revised playbook as the production version of the automation workflow.

8.3 Validation-Only Execution

The first execution mode validates service state without attempting remediation. This is the default behavior and is the recommended mode for routine health checks.

The command used was:

```
cd ~/ansible-nac
ansible-playbook -i inventory/hosts.ini playbooks/nac_validation.yml -K
```

The authoritative execution output is reproduced below.

```
PLAY [Validate NAC services and optionally remediate] **************************

TASK [Collect basic hostname] **************************************************
ok: [radius-splunk]
ok: [access-node]

TASK [Check hostapd-wired service on access-node] ******************************
skipping: [radius-splunk]
ok: [access-node]

TASK [Check freeradius service on radius-splunk] *******************************
skipping: [access-node]
ok: [radius-splunk]

TASK [Restart hostapd-wired service when remediation is enabled] ***************
skipping: [radius-splunk]
skipping: [access-node]

TASK [Restart freeradius service when remediation is enabled] ******************
skipping: [access-node]
skipping: [radius-splunk]

TASK [Re-check hostapd-wired service on access-node] ***************************
skipping: [radius-splunk]
ok: [access-node]

TASK [Re-check freeradius service on radius-splunk] ****************************
skipping: [access-node]
ok: [radius-splunk]

TASK [Collect recent hostapd logs] *********************************************
skipping: [radius-splunk]
ok: [access-node]

TASK [Collect recent freeradius service logs] **********************************
```

```
skipping: [access-node]
ok: [radius-splunk]

TASK [Collect recent freeradius application log] ********************************
skipping: [access-node]
ok: [radius-splunk]

TASK [Save hostapd pre-check status locally] ************************************
skipping: [radius-splunk]
changed: [access-node -> localhost]

TASK [Save hostapd post-check status locally] ***********************************
skipping: [radius-splunk]
changed: [access-node -> localhost]

TASK [Save hostapd restart result locally] **************************************
skipping: [radius-splunk]
changed: [access-node -> localhost]

TASK [Save hostapd logs locally] ************************************************
skipping: [radius-splunk]
ok: [access-node -> localhost]

TASK [Save freeradius pre-check status locally] *********************************
skipping: [access-node]
changed: [radius-splunk -> localhost]

TASK [Save freeradius post-check status locally] ********************************
skipping: [access-node]
changed: [radius-splunk -> localhost]

TASK [Save freeradius restart result locally] ***********************************
skipping: [access-node]
changed: [radius-splunk -> localhost]

TASK [Save freeradius service logs locally] *************************************
skipping: [access-node]
ok: [radius-splunk -> localhost]

TASK [Save freeradius application log locally] **********************************
skipping: [access-node]
ok: [radius-splunk -> localhost]

PLAY RECAP **********************************************************************
access-node                : ok=8    changed=3    unreachable=0    failed=0    skipped=11
        rescued=0    ignored=0
```

```
radius-splunk              : ok=10   changed=3    unreachable=0    failed=0    skipped=9
        rescued=0    ignored=0
```

This run demonstrates the intended baseline behavior:

- both managed nodes are reachable;
- the service validation tasks complete successfully;
- the remediation tasks are correctly skipped when not requested;
- evidence files are still written to the local artifact directory.

This is the preferred operational mode for routine health checks because it confirms service state and preserves evidence without making any changes to the environment.

8.4 Remediation-Enabled Execution

The second execution mode enables controlled service remediation by setting `remediate=true`. In this mode, the playbook still performs the same validation and evidence collection steps, but it additionally restarts the managed services.

The command used was:

```
cd ~/ansible-nac
ansible-playbook -i inventory/hosts.ini playbooks/nac_validation.yml -K -e remediate=true
```

The authoritative execution output is reproduced below.

```
PLAY [Validate NAC services and optionally remediate] **************************

TASK [Collect basic hostname] **************************************************
ok: [radius-splunk]
ok: [access-node]

TASK [Check hostapd-wired service on access-node] ******************************
skipping: [radius-splunk]
ok: [access-node]

TASK [Check freeradius service on radius-splunk] *******************************
skipping: [access-node]
ok: [radius-splunk]

TASK [Restart hostapd-wired service when remediation is enabled] ***************
skipping: [radius-splunk]
changed: [access-node]

TASK [Restart freeradius service when remediation is enabled] ******************
skipping: [access-node]
changed: [radius-splunk]
```

```
TASK [Re-check hostapd-wired service on access-node] ***************************
skipping: [radius-splunk]
ok: [access-node]

TASK [Re-check freeradius service on radius-splunk] ****************************
skipping: [access-node]
ok: [radius-splunk]

TASK [Collect recent hostapd logs] *********************************************
skipping: [radius-splunk]
ok: [access-node]

TASK [Collect recent freeradius service logs] **********************************
skipping: [access-node]
ok: [radius-splunk]

TASK [Collect recent freeradius application log] *******************************
skipping: [access-node]
ok: [radius-splunk]

TASK [Save hostapd pre-check status locally] ***********************************
skipping: [radius-splunk]
changed: [access-node -> localhost]

TASK [Save hostapd post-check status locally] **********************************
skipping: [radius-splunk]
changed: [access-node -> localhost]

TASK [Save hostapd restart result locally] *************************************
skipping: [radius-splunk]
ok: [access-node -> localhost]

TASK [Save hostapd logs locally] ***********************************************
skipping: [radius-splunk]
changed: [access-node -> localhost]

TASK [Save freeradius pre-check status locally] ********************************
skipping: [access-node]
changed: [radius-splunk -> localhost]

TASK [Save freeradius post-check status locally] *******************************
skipping: [access-node]
changed: [radius-splunk -> localhost]

TASK [Save freeradius restart result locally] **********************************
```

```
skipping: [access-node]
ok: [radius-splunk -> localhost]

TASK [Save freeradius service logs locally] ************************************
skipping: [access-node]
changed: [radius-splunk -> localhost]

TASK [Save freeradius application log locally] *********************************
skipping: [access-node]
changed: [radius-splunk -> localhost]

PLAY RECAP *********************************************************************
access-node                : ok=9    changed=4    unreachable=0    failed=0    skipped=10
        rescued=0    ignored=0
radius-splunk              : ok=11   changed=5    unreachable=0    failed=0    skipped=8
        rescued=0    ignored=0
```

This run confirms that the automation can perform controlled remediation without abandoning its evidence-preservation role. The restart tasks executed successfully on both nodes:

- `hostapd-wired.service` on `access-node`;
- `freeradius.service` on `radius-splunk`.

The post-check tasks then completed successfully, which is essential. In a production-style workflow, the restart itself is not the success condition; the success condition is a successful post-remediation validation of the service state.

8.5 Artifact Persistence and Evidence Quality

After the validation and remediation runs, the artifact directory contained the following authoritative post-run state:

```
radius-splunk: ls -lh ~/ansible-nac/artifacts
total 44K
-rw-r--r-- 1 nac-lab nac-lab 3.5K Mar 24 12:23 access-node_hostapd_logs.txt
-rw-r--r-- 1 nac-lab nac-lab   25 Mar 24 12:43 access-node_hostapd_restart_result.txt
-rw-r--r-- 1 nac-lab nac-lab 1.4K Mar 24 12:43 access-node_hostapd_status_after.txt
-rw-r--r-- 1 nac-lab nac-lab 1.4K Mar 24 12:43 access-node_hostapd_status_before.txt
-rw-r--r-- 1 root    root    1.4K Mar 23 19:30 access-node_hostapd_status.txt
-rw-r--r-- 1 nac-lab nac-lab 4.0K Mar 24 12:23 radius-splunk_freeradius_journal.txt
-rw-r--r-- 1 nac-lab nac-lab 3.1K Mar 24 12:23 radius-splunk_freeradius_radiuslog.txt
-rw-r--r-- 1 nac-lab nac-lab   25 Mar 24 12:43 radius-splunk_freeradius_restart_result.txt
-rw-r--r-- 1 nac-lab nac-lab 1.8K Mar 24 12:43 radius-splunk_freeradius_status_after.txt
-rw-r--r-- 1 nac-lab nac-lab 1.8K Mar 24 12:43 radius-splunk_freeradius_status_before.txt
-rw-r--r-- 1 root    root    1.8K Mar 23 19:30 radius-splunk_freeradius_status.txt
```

This is a strong outcome because it shows that the automation does not merely

print to the console. It leaves behind local, reviewable evidence that can be used for:

- post-execution verification;
- manual troubleshooting;
- appendix inclusion;
- change or incident record support.

Selected excerpts from the generated artifacts are reproduced below to demonstrate their practical value.

```
===== /home/nac-lab/ansible-nac/artifacts/access-node_hostapd_logs.txt =====
Mar 24 12:23:12 access-node hostapd[1927]: hostapd_interface_deinit_free(0x61845b677590)
Mar 24 12:23:12 access-node hostapd[1927]: hostapd_interface_deinit_free: num_bss=1
        conf->num_bss=1
Mar 24 12:23:12 access-node hostapd[1927]: hostapd_interface_deinit(0x61845b677590)
Mar 24 12:23:12 access-node hostapd[1927]: ens38: interface state ENABLED->DISABLED
Mar 24 12:23:12 access-node hostapd[1927]: hostapd_bss_deinit: deinit bss ens38
Mar 24 12:23:12 access-node hostapd[1927]: ens38: Flushing old station entries
Mar 24 12:23:12 access-node hostapd[1927]: ens38: Deauthenticate all stations
Mar 24 12:23:12 access-node hostapd[1927]: ens38: AP-DISABLED

===== /home/nac-lab/ansible-nac/artifacts/access-node_hostapd_restart_result.txt =====
remediation not requested

===== /home/nac-lab/ansible-nac/artifacts/access-node_hostapd_status_after.txt =====
- hostapd-wired.service - hostapd wired 802.1X authenticator (lab)
     Loaded: loaded (/etc/systemd/system/hostapd-wired.service; enabled; preset: enabled)
     Active: active (running) since Tue 2026-03-24 12:23:12 UTC; 20min ago
   Main PID: 2054 (hostapd)
      Tasks: 1 (limit: 2206)
     Memory: 844.0K (peak: 1.0M)
        CPU: 31ms
     CGroup: /system.slice/hostapd-wired.service

===== /home/nac-lab/ansible-nac/artifacts/access-node_hostapd_status_before.txt =====
- hostapd-wired.service - hostapd wired 802.1X authenticator (lab)
     Loaded: loaded (/etc/systemd/system/hostapd-wired.service; enabled; preset: enabled)
     Active: active (running) since Tue 2026-03-24 12:23:12 UTC; 20min ago
   Main PID: 2054 (hostapd)
      Tasks: 1 (limit: 2206)
     Memory: 844.0K (peak: 1.0M)
        CPU: 31ms
     CGroup: /system.slice/hostapd-wired.service

===== /home/nac-lab/ansible-nac/artifacts/access-node_hostapd_status.txt =====
```

```
- hostapd-wired.service - hostapd wired 802.1X authenticator (lab)
     Loaded: loaded (/etc/systemd/system/hostapd-wired.service; enabled; preset: enabled)
     Active: active (running) since Mon 2026-03-23 18:51:07 UTC; 39min ago
   Main PID: 1017 (hostapd)
      Tasks: 1 (limit: 2206)
     Memory: 2.9M (peak: 3.1M)
        CPU: 51ms
     CGroup: /system.slice/hostapd.service

===== /home/nac-lab/ansible-nac/artifacts/radius-splunk_freeradius_journal.txt =====
Mar 24 12:22:39 radius-splunk freeradius[3253]: Compiling Post-Auth-Type Challenge for
        attr Post-Auth-Type
Mar 24 12:22:39 radius-splunk freeradius[3253]: Compiling Post-Auth-Type Client-Lost for
        attr Post-Auth-Type
Mar 24 12:22:39 radius-splunk freeradius[3253]: radiusd: #### Skipping IP addresses and
        Ports ####
Mar 24 12:22:39 radius-splunk freeradius[3253]: Configuration appears to be OK
Mar 24 12:22:39 radius-splunk systemd[1]: Started freeradius.service - FreeRADIUS
        multi-protocol policy server.
Mar 24 12:23:12 radius-splunk systemd[1]: Stopping freeradius.service - FreeRADIUS
        multi-protocol policy server...
Mar 24 12:23:12 radius-splunk systemd[1]: freeradius.service: Deactivated successfully.
Mar 24 12:23:12 radius-splunk systemd[1]: Stopped freeradius.service - FreeRADIUS
        multi-protocol policy server.

===== /home/nac-lab/ansible-nac/artifacts/radius-splunk_freeradius_radiuslog.txt =====
Tue Mar 24 11:34:18 2026 : Info: Loaded virtual server <default>
Tue Mar 24 11:34:18 2026 : Warning: Ignoring "sql" (see raddb/mods-available/README.rst)
Tue Mar 24 11:34:18 2026 : Warning: Ignoring "ldap" (see raddb/mods-available/README.rst)
Tue Mar 24 11:34:18 2026 : Info:  # Skipping contents of 'if' as it is always 'false' --
        /etc/freeradius/3.0/sites-enabled/inner-tunnel:366
Tue Mar 24 11:34:18 2026 : Info: Loaded virtual server inner-tunnel
Tue Mar 24 11:34:18 2026 : Info: Loaded virtual server default
Tue Mar 24 11:34:18 2026 : Info: Ready to process requests
Tue Mar 24 11:57:46 2026 : Info: Signalled to terminate

===== /home/nac-lab/ansible-nac/artifacts/radius-splunk_freeradius_restart_result.txt =====
remediation not requested

===== /home/nac-lab/ansible-nac/artifacts/radius-splunk_freeradius_status_after.txt =====
- freeradius.service - FreeRADIUS multi-protocol policy server
     Loaded: loaded (/usr/lib/systemd/system/freeradius.service; enabled; preset: enabled)
     Active: active (running) since Tue 2026-03-24 12:23:12 UTC; 20min ago
       Docs: man:radiusd(8)
             man:radiusd.conf(5)
             http://wiki.freeradius.org/
```

```
             http://networkradius.com/doc/

===== /home/nac-lab/ansible-nac/artifacts/radius-splunk_freeradius_status_before.txt =====
- freeradius.service - FreeRADIUS multi-protocol policy server
     Loaded: loaded (/usr/lib/systemd/system/freeradius.service; enabled; preset: enabled)
     Active: active (running) since Tue 2026-03-24 12:23:12 UTC; 20min ago
       Docs: man:radiusd(8)
             man:radiusd.conf(5)
             http://wiki.freeradius.org/
             http://networkradius.com/doc/

===== /home/nac-lab/ansible-nac/artifacts/radius-splunk_freeradius_status.txt =====
- freeradius.service - FreeRADIUS multi-protocol policy server
     Loaded: loaded (/usr/lib/systemd/system/freeradius.service; enabled; preset: enabled)
     Active: active (running) since Mon 2026-03-23 18:52:42 UTC; 37min ago
       Docs: man:radiusd(8)
             man:radiusd.conf(5)
             http://wiki.freeradius.org/
             http://networkradius.com/doc/
```

Several observations are worth recording:

- the artifact files are human-readable and immediately useful without additional tooling;
- the `hostapd` journal excerpt clearly captures a service deinitialization path consistent with a controlled restart;
- the FreeRADIUS journal excerpt captures a stop/start lifecycle and successful service initialization;
- the service status files provide simple before/after validation anchors;
- the legacy `*_status.txt` files remain present as earlier-generation artifacts, while the newer `*_status_before.txt` and `*_status_after.txt` files reflect the final workflow.

One nuance should be noted: the restart result files currently contain `remediation not requested`. This reflects the playbook's current use of `stdout` capture for the restart tasks. Because `systemctl restart` does not normally emit meaningful stdout, the fallback string remains in place even during a successful remediation-enabled run. This does *not* invalidate the workflow, because the authoritative proof of remediation is instead found in:

- the Ansible console output showing the restart tasks as `changed`;
- the post-check service status files;
- the journal excerpts showing stop/start lifecycle behavior.

This limitation is acceptable in the current lab and will be addressed explicitly later in the limitations section.

8.6 Python Incident Parser Role

The Ansible playbook is not the only automation artifact in the environment. A separate Python utility, `nac_incident_parser.py`, is used to summarize the incident evidence files collected during the controlled failure-and-remediation scenario.

At the time of final documentation, the parser executed successfully and produced the following authoritative output:

```
radius-splunk: python3 ~/incident-evidence/nac_incident_parser.py
=== NAC Incident Evidence Summary ===

[1] Failed authentication simulation
- Client remained unauthorized: YES
- Client showed successful connection during failed test: NO
- hostapd saw authentication attempt: YES
- FreeRADIUS recorded failed auth / reject path: YES

[2] Remediation / successful re-test
- Client reached CTRL-EVENT-CONNECTED: YES
- Client reached Authorized state: YES
- hostapd showed successful authorization indicators: YES

[3] Incident conclusion
- SIMULATED NAC INCIDENT SUCCESSFULLY REPRODUCED AND REMEDIATED
```

This parser is intentionally narrow in scope. It is not a general log analytics engine and it does not replace direct review of raw evidence. Instead, it provides a lightweight operational summary that is useful for:

- quickly confirming that the expected failure and recovery markers are present;
- reducing manual re-check time during repeated test cycles;
- supporting a validation narrative that remains grounded in raw evidence files.

The parser therefore complements the Ansible workflow rather than competing with it:

- Ansible validates service state and captures artifacts;
- the Python parser summarizes the incident evidence set;
- both workflows remain transparent and auditable because their underlying inputs are preserved.

The full parser source is reproduced in Appendix A (see Section A.4.3).

8.7 Operational Assessment

From an operational perspective, the automation layer succeeds because it does not overreach. It implements a realistic control pattern for a lab of this size:

1. validate the relevant services;
2. optionally restart them when instructed;
3. immediately re-check state;
4. persist artifacts locally;
5. correlate with separate incident evidence.

During final validation:

- both playbook runs completed successfully without manual intervention during execution;
- SSH connectivity from `radius-splunk` to `access-node` remained stable during successful runs;
- both the validation-only and remediation-enabled workflows completed without unreachable or failed hosts.

This is intended to demonstrate:

- operational thinking rather than one-time manual testing;
- repeatability across service validation cycles;
- controlled remediation rather than purely observational monitoring;
- artifact preservation suitable for documentation and audit-style review.

At the same time, the automation remains intentionally modest. The purpose of the automation is to demonstrate sound operational design in a constrained environment, not to claim parity with enterprise orchestration platforms.

8.8 Section Conclusion

The automation layer improves the robustness of the lab as an enterprise-aligned production artifact.

Without automation, the lab would still demonstrate a valid wired 802.1X design, but it would remain largely manual. With the addition of Ansible-based validation and optional remediation, plus the Python evidence parser, the lab demonstrates a more complete operational model:

- services can be checked from a central node;
- remediation can be invoked deliberately rather than manually per host;

- artifacts are preserved locally for review;
- incident evidence can be summarized in a repeatable way;
- the automation remains transparent, bounded, and easy to audit.

This is the correct level of automation for the environment. It supports the lab's central goals without obscuring the underlying technical controls or replacing direct evidence review.

9. Observability and Logging Pipeline

The lab includes a deliberately structured observability layer intended to support both steady-state validation and incident-focused troubleshooting. This layer is built from a combination of:

- continuously generated local service and system logs;
- service-oriented journald output for the Linux-based authenticator;
- Splunk Universal Forwarder monitored-input configuration;
- manually curated incident evidence files generated during controlled failure and recovery testing.

The objective is not to claim a full centralized SIEM deployment, but rather to demonstrate that the environment has a coherent, technically meaningful evidence pipeline. In this lab, observability is treated as an engineering function in its own right: the system produces logs, those logs are mapped to identifiable sources, relevant paths are monitored, and incident-specific captures are preserved separately when the continuously generated logs alone are insufficient for a clean narrative.

9.1 Observability Scope and Positioning

The Splunk component in this lab is deliberately scoped as a local monitored-input configuration without demonstrated remote indexer or search-head integration. That scope is both accurate and sufficient for the environment that was built.

This section therefore demonstrates the following, and only the following:

- identification of the relevant local log sources;
- configuration of Splunk Universal Forwarder to monitor those sources;
- preservation of service and incident evidence in local files;
- a logging model that supports the final-state validation and incident sections of this document.

It does *not* demonstrate:

- centralized Splunk search or analytics;
- a remote indexer/search-head topology;
- dashboards, alerts, or correlation searches;
- enterprise retention, forwarding, or multi-tier ingestion architecture.

This scope boundary matters because it allows the document to make a strong but accurate claim: the lab is instrumented and observable, even though it is not a full SIEM deployment.

9.2 Splunk Universal Forwarder Monitored Inputs

The Splunk Universal Forwarder on `radius-splunk` is active in the final state:

```
radius-splunk: /opt/splunkforwarder/bin/splunk status
Warning: Attempting to revert the SPLUNK_HOME ownership
Warning: Executing "chown -R root /opt/splunkforwarder"
splunkd is running (PID: 1489)
splunk helpers are running (PIDs: 1490)
```

The monitored-input configuration is maintained in a dedicated local app:

```
radius-splunk: /opt/splunkforwarder/etc/apps/lab_nac_inputs/local/inputs.conf
[monitor:///var/log/syslog]
disabled = false
index = main
sourcetype = syslog

[monitor:///var/log/auth.log]
disabled = false
index = main
sourcetype = linux_secure

[monitor:///var/log/freeradius/radius.log]
disabled = false
index = main
sourcetype = freeradius
```

The app layout itself is minimal and clean:

```
radius-splunk: find /opt/splunkforwarder/etc/apps/lab_nac_inputs -maxdepth 3 -type f | sort
/opt/splunkforwarder/etc/apps/lab_nac_inputs/local/inputs.conf
```

This is the correct level of Splunk configuration for the lab. The forwarder monitors three sources that are directly relevant to the final environment:

- `/var/log/syslog` for general system and service lifecycle events;
- `/var/log/auth.log` for administrative access and privilege activity;

- `/var/log/freeradius/radius.log` for FreeRADIUS application-layer events.

Together, these monitored inputs provide coverage over: - host operating-system state, - administrative activity, - and the primary authentication service itself.

The full UF input configuration is reproduced in Appendix A (see Section A.4.6).

9.3 Primary Continuous Log Sources on `radius-splunk`

The most important continuously generated log sources in the environment are located on `radius-splunk`. These include system log data, authentication/admin activity, and FreeRADIUS application output.

Their current filesystem metadata is summarized below.

```
radius-splunk log files
syslog:  -rw-r----- 1 syslog adm 2.7M Mar 24 13:30 /var/log/syslog
auth.log: -rw-r----- 1 syslog adm 233K Mar 24 13:25 /var/log/auth.log
radius.log: -rw-r--r-- 1 syslog adm 12K Mar 24 13:14 /var/log/freeradius/radius.log
```

9.3.1 System Log (`/var/log/syslog`)

The system log provides host-level operational context such as service start/stop events, scheduled job activity, and general daemon behavior. A representative excerpt is reproduced below.

```
radius-splunk: tail -n 15 /var/log/syslog
2026-03-24T13:14:41.348438+00:00 radius-splunk multipathd[467]: sda: failed to get sgio
        uid: No such file or directory
2026-03-24T13:14:41.348468+00:00 radius-splunk multipathd[467]: sda: no WWID in state
        "undef
2026-03-24T13:14:41.348639+00:00 radius-splunk multipathd[467]: ", giving up
2026-03-24T13:14:41.348690+00:00 radius-splunk multipathd[467]: sda: check_path() failed,
        removing
2026-03-24T13:15:01.629554+00:00 radius-splunk CRON[1579]: (root) CMD (command -v
        debian-sa1 > /dev/null && debian-sa1 1 1)
2026-03-24T13:17:01.652202+00:00 radius-splunk CRON[1587]: (root) CMD (cd / && run-parts
        --report /etc/cron.hourly)
2026-03-24T13:19:05.468094+00:00 radius-splunk systemd[1]: Starting
        update-notifier-download.service - Download data for packages that failed at package
        install time...
2026-03-24T13:19:05.678561+00:00 radius-splunk systemd[1]: update-notifier-download.service
        : Deactivated successfully.
2026-03-24T13:19:05.678792+00:00 radius-splunk systemd[1]: Finished
        update-notifier-download.service - Download data for packages that failed at package
        install time.
2026-03-24T13:19:31.934534+00:00 radius-splunk systemd[1376]: launchpadlib-cache-clean.
        service - Clean up old files in the Launchpadlib cache was skipped because of an
```

```
        unmet condition check (ConditionPathExists=/home/nac-lab/.launchpadlib/
        api.launchpad.net/cache).
```

This excerpt is not included because each line is individually important. Rather, it demonstrates the character of the source: `syslog` is a general-purpose host operational log that complements more specialized sources such as `radius.log`.

9.3.2 Administrative and Authentication Log (`/var/log/auth.log`)

The administrative and authentication log captures local privilege elevation activity and remote administrative access events. A representative excerpt is reproduced below.

```
radius-splunk: tail -n 15 /var/log/auth.log
2026-03-24T13:20:04.756710+00:00 radius-splunk sshd[1628]: pam_unix(sshd:session): session
        opened for user nac-lab(uid=1000) by nac-lab(uid=0)
2026-03-24T13:20:04.762714+00:00 radius-splunk systemd-logind[816]: New session 5 of user
        nac-lab.
2026-03-24T13:20:05.503276+00:00 radius-splunk sshd[1686]: Received disconnect from
        192.168.195.1 port 41932:11: disconnected by user
2026-03-24T13:20:05.503525+00:00 radius-splunk sshd[1686]: Disconnected from user nac-lab
        192.168.195.1 port 41932
2026-03-24T13:20:05.504875+00:00 radius-splunk sshd[1628]: pam_unix(sshd:session): session
        closed for user nac-lab
2026-03-24T13:20:05.511847+00:00 radius-splunk systemd-logind[816]: Session 5 logged out.
        Waiting for processes to exit.
2026-03-24T13:20:05.514296+00:00 radius-splunk systemd-logind[816]: Removed session 5.
2026-03-24T13:20:37.703552+00:00 radius-splunk sudo:  nac-lab : TTY=tty1 ;
        PWD=/home/nac-lab ; USER=root ; COMMAND=/usr/bin/tail -n 15 /var/log/syslog
2026-03-24T13:20:37.704672+00:00 radius-splunk sudo: pam_unix(sudo:session): session opened
        for user root(uid=0) by nac-lab(uid=1000)
2026-03-24T13:20:37.731117+00:00 radius-splunk sudo: pam_unix(sudo:session): session closed
        for user root
2026-03-24T13:21:39.041208+00:00 radius-splunk sudo:  nac-lab : TTY=tty1 ;
        PWD=/home/nac-lab ; USER=root ; COMMAND=/usr/bin/tail -n 15 /var/log/auth.log
2026-03-24T13:21:39.042808+00:00 radius-splunk sudo: pam_unix(sudo:session): session opened
        for user root(uid=0) by nac-lab(uid=1000)
2026-03-24T13:21:39.068864+00:00 radius-splunk sudo: pam_unix(sudo:session): session closed
        for user root
2026-03-24T13:22:03.513481+00:00 radius-splunk sudo:  nac-lab : TTY=tty1 ;
        PWD=/home/nac-lab ; USER=root ; COMMAND=/usr/bin/tail -n 15 /var/log/auth.log
2026-03-24T13:22:03.514636+00:00 radius-splunk sudo: pam_unix(sudo:session): session opened
        for user root(uid=0) by nac-lab(uid=1000)
```

This source is especially useful in a lab of this type because it records operator activity, privilege escalation, and remote administrative access events that may be relevant during incident reconstruction or audit-style review.

9.3.3 FreeRADIUS Application Log (`/var/log/freeradius/radius.log`)

The most important continuously generated application log in the environment is the FreeRADIUS log. This file captures service initialization, readiness state, termination events, and other application-relevant messages.

Its metadata is as follows:

```
radius-splunk: ls -lh /var/log/freeradius/radius.log
-rw-r--r-- 1 freerad adm 12K Mar 24 13:14 /var/log/freeradius/radius.log
```

A representative excerpt is reproduced below.

```
radius-splunk: tail -n 20 /var/log/freeradius/radius.log
Tue Mar 24 12:23:12 2026 : Info: Debug state unknown (cap_sys_ptrace capability not set)
Tue Mar 24 12:23:12 2026 : Info: systemd watchdog interval is 30.00 secs
Tue Mar 24 12:23:12 2026 : Info: Loaded virtual server <default>
Tue Mar 24 12:23:12 2026 : Warning: Ignoring "sql" (see raddb/mods-available/README.rst)
Tue Mar 24 12:23:12 2026 : Warning: Ignoring "ldap" (see raddb/mods-available/README.rst)
Tue Mar 24 12:23:12 2026 : Info:  # Skipping contents of 'if' as it is always 'false' --
        /etc/freeradius/3.0/sites-enabled/inner-tunnel:366
Tue Mar 24 12:23:12 2026 : Info: Loaded virtual server inner-tunnel
Tue Mar 24 12:23:12 2026 : Info: Loaded virtual server default
Tue Mar 24 12:23:12 2026 : Info: Ready to process requests
Tue Mar 24 12:53:42 2026 : Info: Signalled to terminate
Tue Mar 24 12:53:42 2026 : Info: Exiting normally
Tue Mar 24 13:14:05 2026 : Info: Debug state unknown (cap_sys_ptrace capability not set)
Tue Mar 24 13:14:05 2026 : Info: systemd watchdog interval is 30.00 secs
Tue Mar 24 13:14:05 2026 : Info: Loaded virtual server <default>
Tue Mar 24 13:14:05 2026 : Warning: Ignoring "sql" (see raddb/mods-available/README.rst)
Tue Mar 24 13:14:05 2026 : Warning: Ignoring "ldap" (see raddb/mods-available/README.rst)
Tue Mar 24 13:14:05 2026 : Info:  # Skipping contents of 'if' as it is always 'false' --
        /etc/freeradius/3.0/sites-enabled/inner-tunnel:366
Tue Mar 24 13:14:05 2026 : Info: Loaded virtual server inner-tunnel
Tue Mar 24 13:14:05 2026 : Info: Loaded virtual server default
Tue Mar 24 13:14:05 2026 : Info: Ready to process requests
```

This source is particularly valuable because it exposes application-level readiness and service lifecycle behavior that are directly relevant to the authentication plane. In the context of the lab, it complements both the system logs and the incident-specific evidence files.

9.4 `hostapd` Observability on `access-node`

Unlike FreeRADIUS, the Linux-based authenticator does not rely on a dedicated application log file under `/var/log`. Instead, its authoritative operational evidence is exposed through the managed systemd service and the journal.

The current service state is reproduced below.

```
access-node: systemctl status hostapd-wired.service --no-pager
Loaded: loaded (/etc/systemd/system/hostapd-wired.service; enabled; preset: enabled)
Active: active (running) since Tue 2026-03-24 04:20:41 UTC
Main PID: 975 (hostapd)

Recent lines:
Using interface ens38 with hwaddr <REDACTED> and ssid ""
ens38: RADIUS Authentication server 10.0.30.10:1812
RADIUS local address: 10.0.23.1:45833
ens38: interface state UNINITIALIZED->ENABLED
ens38: AP-ENABLED
ens38: Setup of interface done.
```

The underlying unit file is located at:

```
access-node: systemctl show -p FragmentPath hostapd-wired.service
FragmentPath=/etc/systemd/system/hostapd-wired.service
```

A representative journal excerpt is reproduced below.

```
access-node: journalctl -u hostapd-wired.service -n 20 --no-pager
Mar 24 02:13:55 access-node hostapd[2398]: ens38: CTRL-EVENT-TERMINATING
Mar 24 02:13:55 access-node hostapd[2398]: hostapd_free_hapd_data(ens38)
Mar 24 02:13:55 access-node hostapd[2398]: hostapd_interface_deinit_free:
        driver=0x565a2d747060 drv_priv=0x565a38444cd0 -> hapd_deinit
Mar 24 02:13:55 access-node hostapd[2398]: hostapd_interface_free(0x565a38442590)
Mar 24 02:13:55 access-node hostapd[2398]: hostapd_interface_free: free hapd 0x565a38443d40
Mar 24 02:13:55 access-node hostapd[2398]: hostapd_cleanup_iface(0x565a38442590)
Mar 24 02:13:55 access-node hostapd[2398]: hostapd_cleanup_iface_partial(0x565a38442590)
Mar 24 02:13:55 access-node hostapd[2398]: hostapd_cleanup_iface: free iface=0x565a38442590
Mar 24 02:13:55 access-node systemd[1]: hostapd-wired.service: Deactivated successfully.
Mar 24 02:13:55 access-node systemd[1]: Stopped hostapd-wired.service - hostapd wired
        802.1X authenticator (lab).
-- Boot 4efbb5bbbad2458a94bc281c688f0c9b --
Mar 24 04:20:41 access-node systemd[1]: Started hostapd-wired.service - hostapd wired
        802.1X authenticator (lab).
Mar 24 04:20:41 access-node hostapd[975]: random: getrandom() support available
Mar 24 04:20:41 access-node hostapd[975]: Configuration file:
        /etc/hostapd/lab/hostapd-wired.conf
Mar 24 04:20:41 access-node hostapd[975]: eapol_version=2
Mar 24 04:20:41 access-node hostapd[975]: ctrl_interface_group=0
Mar 24 04:20:41 access-node hostapd[975]: Opening raw packet socket for ifindex 3
Mar 24 04:20:41 access-node hostapd[975]: BSS count 1, BSSID mask 00:00:00:00:00:00
        (0 bits)
Mar 24 04:20:41 access-node hostapd[975]: ens38: IEEE 802.11 Fetching hardware
        channel/rate support not supported.
Mar 24 04:20:41 access-node hostapd[975]: Completing interface initialization
...
```

```
Mar 24 04:20:41 access-node hostapd[975]: ens38: AP-ENABLED
Mar 24 04:20:41 access-node hostapd[975]: ens38: Setup of interface done.
Mar 24 04:20:41 access-node hostapd[975]: ctrl_iface not configured!
```

No dedicated `hostapd` file was present under `/var/log` at the time of validation. This is an important architectural note because it clarifies that `journald` is the authoritative continuous log source for the Linux-based authenticator in this environment.

This distinction matters operationally: - FreeRADIUS provides both service/journal context and a dedicated application log file; - `hostapd` primarily provides service/journal context; - both are still compatible with the incident evidence workflow because curated captures can be created from their runtime output when a more focused artifact is needed.

9.5 Curated Incident Evidence as a Complementary Layer

The continuously generated logs described above are necessary, but they are not the only observability artifacts in the lab. The environment also includes a curated incident-evidence directory on `radius-splunk`:

```
radius-splunk: ls -lh ~/incident-evidence
total 92K
-rw-rw-r-- 1 nac-lab nac-lab  32K Mar 23 04:02 freeradius_failed_auth_incident.txt
-rw-rw-r-- 1 nac-lab nac-lab  11K Mar 23 04:01 hostapd_failed_auth_incident.txt
-rw-rw-r-- 1 nac-lab nac-lab  19K Mar 23 04:01 hostapd_success_retest_v2.txt
-rw-r--r-- 1 root    root    3.5K Mar 24 09:52 nac_incident_parser.py
-rw-rw-r-- 1 nac-lab nac-lab  535 Mar 23 04:28 nac_incident_parser_success.txt
-rw-rw-r-- 1 nac-lab nac-lab 7.3K Mar 23 03:59 wpa_supplicant_failed_auth_incident.txt
-rw-rw-r-- 1 nac-lab nac-lab 8.4K Mar 23 03:59 wpa_supplicant_success_retest_v2.txt
```

The following files were explicitly confirmed as manually captured from debug or controlled test runs:

- `wpa_supplicant_failed_auth_incident.txt`;
- `wpa_supplicant_success_retest_v2.txt`;
- `hostapd_failed_auth_incident.txt`;
- `hostapd_success_retest_v2.txt`;
- `freeradius_failed_auth_incident.txt`.

This distinction should be made explicit:

- the continuously generated system and service logs are the baseline observability layer;
- the incident-evidence files are curated captures created to preserve high-signal artifacts from controlled validation runs.

The incident evidence files therefore complement rather than replace the continuous logging sources. They are particularly useful when the goal is to preserve a deterministic failure or recovery narrative in a form that is easy to reference later in the document, the parser, and the appendix sections.

9.6 Observability Value to the Lab

The observability design in this lab is modest in scale but strong in structure. It provides:

- identifiable local log sources for system, administrative, and authentication behavior;
- a dedicated monitored-input configuration in Splunk Universal Forwarder;
- service-oriented journal visibility for the Linux authenticator;
- curated incident evidence files for deterministic reproduction and documentation.

This combination is what makes later sections of the document possible. The access-control validation section depends on service and routing evidence. The incident section depends on curated captures and service logs. The automation section depends on local evidence persistence. The packet-analysis section depends on protocol captures aligned with the same operational story.

In other words, the observability layer is not an afterthought. It is one of the elements that makes the lab function as a coherent engineering artifact rather than as a loose set of services.

9.7 Section Conclusion

The final lab demonstrates a well-scoped but credible logging and observability model.

It is accurate to say that the environment includes:

- log source identification;
- log capture and monitored-input configuration;
- service-level journal visibility;
- evidence preservation for validation and incident analysis.

It would not be accurate to claim that the environment demonstrates centralized Splunk search or enterprise analytics. The document will therefore continue to describe the Splunk component precisely as a local UF-based monitored-input layer.

10. Packet Capture Analysis (tcpdump + Wireshark)

Packet capture analysis was performed to validate the incident scenario at the protocol layer. Earlier sections established the operational facts of the incident through endpoint behavior, FreeRADIUS evidence, hostapd evidence, and GLPI ticketing. This section adds a stricter form of proof by showing that the failed and successful authentication paths are directly observable in packet capture.

This is important because it demonstrates that the incident was not inferred only from logs or service state. The captures show the actual IEEE 802.1X / EAPOL exchange on the client-facing segment and the corresponding RADIUS transaction on the upstream path. In effect, the packet evidence ties together the supplicant, the authenticator, and the AAA backend into one coherent protocol narrative.

The scope of the capture work was intentionally narrow. These were not broad traffic captures intended to inventory all host communications. They were short, purpose-built captures limited to the authentication window so that the relevant control-plane exchange would remain concise, readable, and suitable for publication.

10.1 Capture Methodology

All packet captures for this section were taken on `access-node`. This was the appropriate capture point because `access-node` is the wired 802.1X authenticator and therefore sits at the boundary between the client-facing controlled port and the upstream routed path toward FreeRADIUS.

Two interfaces were used:

- `ens38` — wired 802.1X / EAPOL side toward `client-1`
- `ens39` — routed upstream side toward `dist-router` and the RADIUS service path

This dual-observation approach makes it possible to correlate:

- the client-facing authentication result seen as EAPOL / EAP messages, and
- the backend AAA decision seen as RADIUS request/challenge/accept/reject exchange.

That correlation is the central analytical value of the packet capture work in this lab.

10.2 Capture Inventory

The authoritative packet capture inventory consists of the following four files:

- failed_8021x_ens38.pcap
- failed_8021x_ens39.pcap
- success_8021x_ens38.pcap
- success_8021x_ens39.pcap

The pcap inventory on access-node was as follows:

```
access-node: ls -lh ~/pcaps
-rw-r--r-- 1 nac-lab nac-lab 385 Mar 23 20:23 failed_8021x_ens38.pcap
-rw-r--r-- 1 nac-lab nac-lab 738 Mar 23 20:23 failed_8021x_ens39.pcap
-rw-r--r-- 1 nac-lab nac-lab 385 Mar 23 20:29 success_8021x_ens38.pcap
-rw-r--r-- 1 nac-lab nac-lab 747 Mar 23 20:29 success_8021x_ens39.pcap
```

The small file sizes are expected and desirable. They indicate that the captures were intentionally constrained to the short authentication test windows rather than left running as general packet traces.

10.3 Capture Procedure

The captures were obtained by starting two concurrent tcpdump processes on access-node and then forcing a controlled supplicant authentication attempt from client-1. The method used was as follows:

```
access-node:
sudo timeout 20s tcpdump -i ens38 -w /tmp/failed_8021x_ens38.pcap ether proto 0x888e &
sudo timeout 20s tcpdump -i ens39 -w /tmp/failed_8021x_ens39.pcap host 10.0.30.10 and
        udp port 1812 &
sleep 2

client-1:
sudo systemctl stop wpa_supplicant
sudo timeout 15s wpa_supplicant -i ens33 -D wired -c
        /etc/wpa_supplicant/lab/wired-8021x.conf -dd

access-node:
sudo timeout 20s tcpdump -i ens38 -w ~/pcaps/success_8021x_ens38.pcap ether proto 0x888e &
sudo timeout 20s tcpdump -i ens39 -w ~/pcaps/success_8021x_ens39.pcap host 10.0.30.10 and
        udp port 1812 &
sleep 2

client-1:
sudo systemctl stop wpa_supplicant
sudo timeout 15s wpa_supplicant -i ens33 -D wired -c
        /etc/wpa_supplicant/lab/wired-8021x.conf -dd
```

This method is appropriate for a controlled lab because it produces deterministic, low-noise captures focused only on the authentication event under investigation.

The authoritative capture commands and retained packet inventory will be reproduced in full in Appendix A (see Section B.6.2 and Section B.6.1).

10.4 Failed Authentication: Client-Facing EAPOL Analysis

The failed client-facing capture corresponds to:

`failed_8021x_ens38.pcap`

This capture was taken on `ens38`, the client-facing controlled segment between `client-1` and the Linux-based authenticator. In Wireshark, the failed capture shows a complete 802.1X / EAP exchange that progresses through the expected early stages but terminates unsuccessfully.

The visible sequence is:

1. EAPOL Start
2. EAP Request, Identity
3. EAP Response, Identity
4. EAP Request, MD5-Challenge
5. EAP Response, MD5-Challenge
6. EAP Failure

This sequence is significant because it rules out several weaker explanations for the failure. The client did not fail due to lack of link, lack of EAPOL negotiation, or absence of authenticator response. The control-plane exchange clearly occurred and advanced into the MD5 challenge/response stage. The terminal outcome was explicitly `EAP Failure`.

Accordingly, the client-facing failed capture supports the same conclusion already reached through supplicant logs: the endpoint remained unauthorized because the authentication exchange completed with a negative result, not because the exchange never occurred.

10.5 Failed Authentication: Authenticator-to-RADIUS Analysis

The failed upstream RADIUS-side capture corresponds to:

`failed_8021x_ens39.pcap`

This capture was taken on `ens39`, the routed side of `access-node` toward `dist-router` and the FreeRADIUS server. In Wireshark, the failed capture shows the corresponding backend AAA exchange that produced the client-facing failure observed on `ens38`.

The visible RADIUS transaction sequence is:

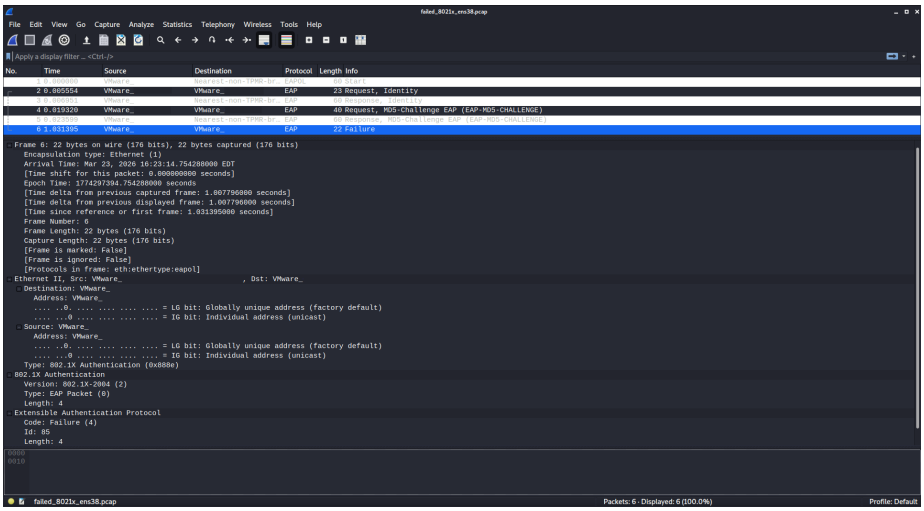

Figure 7: Failed wired 802.1X authentication on the client-facing EAPOL segment

1. Access-Request
2. Access-Challenge
3. Access-Request
4. Access-Reject

In the packet details for the terminal RADIUS response, the final `Access-Reject` carries an embedded EAP payload indicating `EAP Failure`. This is the strongest backend confirmation of the failed test case because it proves that the RADIUS server was reachable, the challenge-based workflow occurred normally, and the final authorization decision was explicitly negative.

This point matters analytically. It confirms that the failed incident was not caused by upstream transport loss or a silent AAA outage. The RADIUS service participated fully in the authentication exchange and returned an authoritative reject decision, which the client-facing side then reflected as `EAP Failure`.

10.6 Successful Remediation Re-Test: Client-Facing EAPOL Analysis

The successful client-facing remediation re-test capture corresponds to:

`success_8021x_ens38.pcap`

This capture was also taken on `ens38`. In Wireshark, the successful capture shows the same general 802.1X / EAP structure as the failed case, but with a different terminal authorization result.

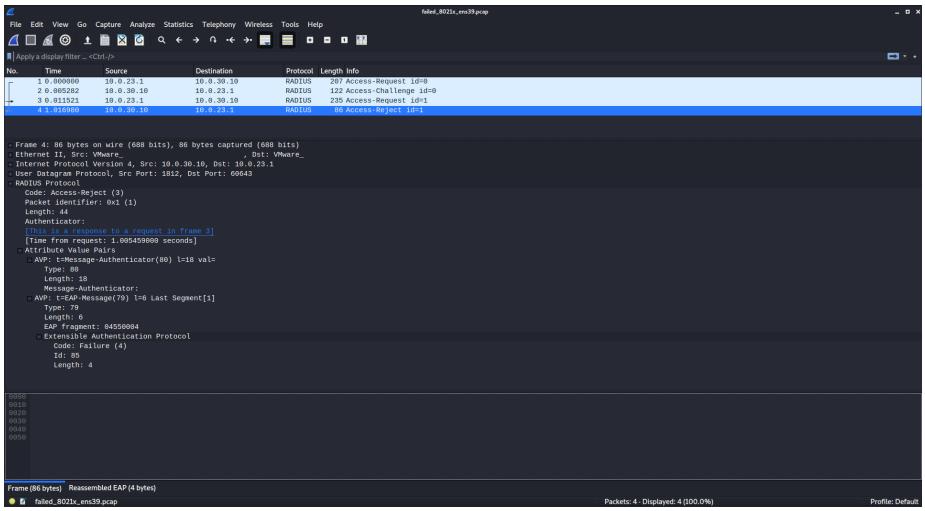

Figure 8: Failed wired 802.1X authentication on the authenticator-to-RADIUS path

The visible sequence is:

1. EAPOL Start
2. EAP Request, Identity
3. EAP Response, Identity
4. EAP Request, MD5-Challenge
5. EAP Response, MD5-Challenge
6. EAP Success

This is an important contrast with the failed capture. The early control-plane stages are substantially the same, which means the remediation did not alter the existence of the authentication workflow itself. Instead, it altered the final authorization result after the known-good credential state was restored.

The successful client-facing capture therefore confirms that the remediated endpoint completed the exchange and received an explicit `EAP Success`, which is the protocol-level counterpart to the authorized endpoint state observed in the supplicant and hostapd evidence.

10.7 Successful Remediation Re-Test: Authenticator-to-RADIUS Analysis

The successful upstream RADIUS-side remediation re-test capture corresponds to:

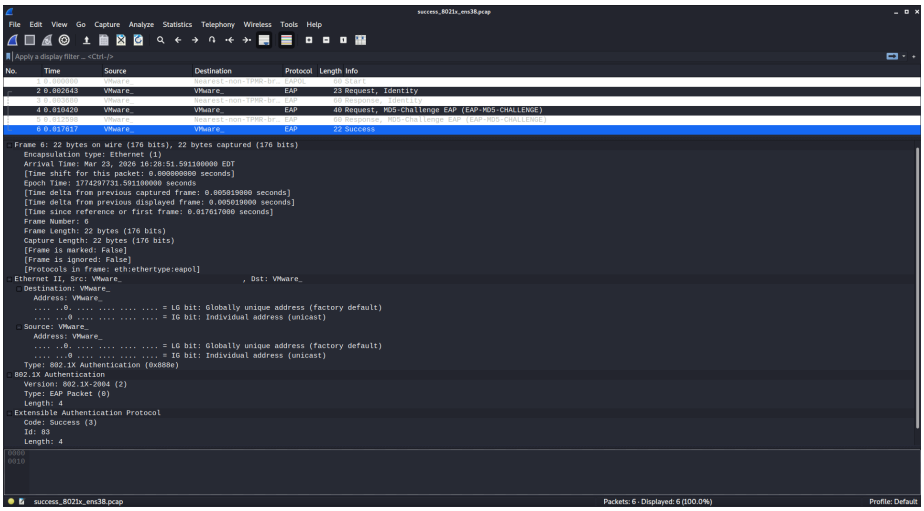

Figure 9: Successful wired 802.1X authentication on the client-facing EAPOL segment

`success_8021x_ens39.pcap`

This capture was taken on `ens39`. In Wireshark, it shows the backend RADIUS exchange that corresponds to the successful client-facing authentication result.

The visible RADIUS transaction sequence is:

1. Access-Request
2. Access-Challenge
3. Access-Request
4. Access-Accept

In the packet details for the terminal response, the `Access-Accept` contains an embedded EAP payload indicating `EAP Success`. This directly correlates with the client-facing `EAP Success` on `ens38` and provides authoritative AAA confirmation that the remediation restored successful backend authentication.

This is the protocol-level counterpart to the operational conclusion already established in the incident section: once the correct credential state was restored, the authentication workflow did not merely proceed farther or behave differently at random. It terminated with a positive authorization decision at the AAA layer.

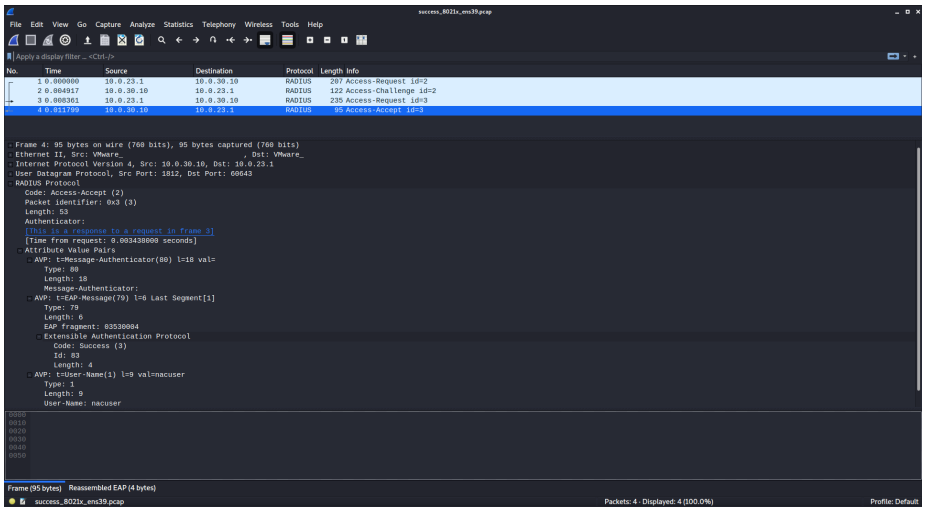

Figure 10: Successful wired 802.1X authentication on the authenticator-to-RADIUS path

10.8 Comparative Protocol Interpretation

The most important protocol distinction between the failed and successful capture sets is that both scenarios show a valid and complete 802.1X / EAP control-plane exchange through the Identity and MD5 challenge/response stages, but they terminate differently at both observation points.

In the failed case:

- the client-facing `ens38` capture ends in `EAP Failure`;
- the routed `ens39` capture ends in `Access-Reject`, carrying an embedded EAP failure indication.

In the successful remediation case:

- the client-facing `ens38` capture ends in `EAP Success`;
- the routed `ens39` capture ends in `Access-Accept`, carrying an embedded EAP success indication.

This is analytically important because it demonstrates that the failed test case was not caused by absence of EAPOL negotiation, lack of client participation, or general transport failure between the authenticator and FreeRADIUS. In both cases, the supplicant, authenticator, and AAA service all participated in a normal challenge-based authentication workflow. The decisive difference is the terminal policy result.

The captures therefore visually reinforce the distinction between the pre-authentication control-plane exchange and the final authorization outcome.

Both failed and successful tests show substantially similar early negotiation behavior, but they diverge at the terminal decision point where authentication either remains unauthorized or transitions to authorized state.

10.9 Publication and Redaction Guidance

The publication policy for packet screenshots is:

- private IP addresses may remain visible;
- MAC addresses must be redacted;
- passwords, shared secrets, and sensitive authentication blobs must be redacted;
- packet payload details that unnecessarily expose sensitive values should be redacted.

The following items should be redacted in the publication-safe versions:

- all visible MAC addresses in the packet list pane;
- all visible MAC addresses in the packet details pane;
- MAC addresses exposed again in the hex pane;
- RADIUS `Authenticator` values where shown;
- `Message-Authenticator` values where shown;

10.11 Section Conclusion

The packet capture analysis materially strengthens the technical credibility of the lab. It shows that the incident and remediation were not validated solely through host logs, service states, or application-level reachability tests. They were also validated through deterministic protocol transitions at both the IEEE 802.1X and RADIUS layers.

The failed capture set demonstrates a complete challenge-based authentication workflow ending in `EAP Failure` and `Access-Reject`. The successful remediation capture set demonstrates the same workflow ending in `EAP Success` and `Access-Accept`. That contrast provides clean protocol-level confirmation that the simulated incident was a true authentication decision failure and that remediation restored normal authorization behavior.

The authoritative capture commands and packet inventory will be referenced again in the appendix material so that the packet-analysis section remains readable while still being fully supported by raw artifact traceability.

11. Design Constraints and Limitations

The lab documented in this report is a deliberate proof-of-concept implementation intended to validate core NAC behaviors, incident handling workflow, and operational evidence collection in a constrained virtual environment. It is not presented as a one-to-one reproduction of a full enterprise campus access control deployment. This distinction is important and intentional.

The design succeeds at the objective for which it was built: it demonstrates credential-gated wired access control, protected and guest network segmentation, deterministic incident reproduction, structured remediation, lightweight automation, and packet-level protocol validation. However, it does so through a set of explicit design constraints chosen to keep the lab tractable and reproducible.

These limitations are intended clearly separate what was actually implemented and validated as a proof-of-concept from what would exist in a larger or more production-like deployment.

11.1 Platform Realism Boundaries

The most important platform constraint in the lab is that the wired 802.1X authenticator function is implemented using Linux `hostapd` in wired mode rather than a managed enterprise access switch. This design choice is fully appropriate for the proof-of-concept objective because it produces real IEEE 802.1X / EAPOL behavior and a real RADIUS-backed authorization workflow, but it should not be described as equivalent to a vendor campus switching platform.

In practical terms, the lab demonstrates credential-gated access control behavior rather than switch-native NAC feature breadth. It does not attempt to model enterprise access-switch capabilities such as dynamic VLAN assignment, downloadable ACLs, switch vendor posture integration, or centralized policy enforcement engines. The access decision in this lab is still meaningful and technically valid, but the implementation path is intentionally simplified.

Similarly, the guest network is enforced primarily through pfSense interface and rule policy rather than through dynamic 802.1X authorization state. This is an intentional architectural separation: the authenticated access path demonstrates wired 802.1X behavior, while the guest path demonstrates policy-based isolation and controlled internet access. Together they provide a credible functional access model without claiming to be a full enterprise NAC fabric.

Finally, the virtual network segmentation used throughout the lab should be understood as a lab isolation mechanism, not as a substitute for a production campus switching and security architecture. The VMware custom network design is useful because it provides deterministic path control and repeatable separation between segments, but it is not presented as equivalent to enterprise hardware switching, trunking, or controller-driven segmentation.

11.2 Authentication Model Limitations

The authentication workflow demonstrated in this lab is based on EAP-MD5 challenge/response, as confirmed both by the service logs and by the packet captures analyzed in Section 10. This was an intentional choice made for demonstration simplicity and reproducibility in a small virtual lab.

That choice should not be interpreted as a modern production recommendation. EAP-MD5 is useful in this context because it provides a compact and easily observable challenge-based authentication flow that can be validated clearly in both logs and packet capture. It is especially useful for showing the exact distinction between a failed credential state and a successful post-remediation state. However, it does not provide the security properties expected of contemporary enterprise access control.

The lab does not demonstrate certificate-based EAP methods such as EAP-TLS, PEAP, or EAP-TTLS. It also does not attempt to model the certificate lifecycle, trust distribution, supplicant provisioning, or PKI operational dependencies that would normally accompany a more production-aligned 802.1X deployment. These omissions are intentional and should be understood as scope boundaries rather than gaps in execution.

11.3 Observability and SIEM Limitations

The observability model in this lab is intentionally practical but limited in scope. As documented in Section 9, Splunk is represented only by a Splunk Universal Forwarder with local monitored-input configuration. This is sufficient to demonstrate awareness of log source selection and forwarder-oriented ingestion configuration, but it is not a full Splunk search-head, indexer, or analytics deployment.

For that reason, the document does not claim centralized SIEM investigation capability. No centralized search workflow, correlation rules, dashboards, or alerting pipelines were demonstrated as part of the lab. Instead, incident validation relied on a combination of:

- native service logs from `hostapd`, FreeRADIUS, and systemd/journald;
- manually captured debug artifacts preserved in the incident evidence directory;
- the GLPI incident record used as the operational ticketing system of record; and
- packet capture analysis using `tcpdump` and Wireshark.

This multi-source evidence model is entirely appropriate for the lab objective. It produces a defensible incident narrative and a strong audit trail without overstating the presence of centralized SIEM operations.

11.4 Automation Scope Limitations

The automation implemented in this lab is intentionally narrow in scope and should be described accordingly. As documented in Section 8, the Ansible playbook is best characterized as a validation-first workflow with optional service remediation. It is designed to confirm service health, optionally restart key services, and preserve the resulting evidence locally. It is not intended to represent a full production change-management pipeline, orchestration framework, or large-scale infrastructure automation pattern.

The same principle applies to the Python parser. The parser is a purpose-built evidence summarizer that evaluates a known set of captured artifact files in fixed paths and produces a human-readable incident conclusion. It is valuable because it demonstrates structured evidence interpretation and repeatable operator validation, but it is not a generalized incident correlation engine or a reusable NAC analytics platform.

The tools are tightly aligned to the services, filenames, and operational checkpoints that were actually implemented and validated. The document should therefore present the automation as operationally useful and technically appropriate, while avoiding any implication that it is broader or more portable than it was designed to be.

11.5 GUI and Evidence Scope Limitations

The production document intentionally limits GUI evidence to three categories:

- the pfSense terminal menu on boot and network GUI;
- the GLPI incident record; and
- Wireshark packet-analysis screenshots.

This is an explicit documentation choice rather than an omission. The lab is primarily a command-line and protocol-driven implementation, and the strongest evidence is found in configuration files, service outputs, debug logs, packet traces, and deterministic test results. Broad GUI screenshot inventories would add visual bulk without materially improving the technical case.

For that reason, the document intentionally avoids dependence on hypervisor settings walkthroughs, appliance UI tours, or other screenshot-heavy administrative paths. The goal is to keep the evidence model focused on raw technical artifacts and protocol behavior while using GUI elements only where they provide uniquely valuable context, specifically:

- pfSense as the source of network traffic rules and logic;
- GLPI as the human-readable incident management record; and
- Wireshark as the visual representation of protocol-layer success and failure.

11.6 Operational Scope Boundaries

Operationally, the lab is centered on single-endpoint incident validation using `client-1` as the authoritative authenticated endpoint and `guest-test-1` as the guest-segment validation endpoint. This is sufficient for the design objective because the purpose of the lab is to prove that the access model, failure scenario, remediation, and evidence chain are all coherent and repeatable. It is not intended to simulate multi-endpoint concurrency or campus-scale endpoint churn.

Accordingly, the lab does not attempt to model any of the following:

- redundant or load-balanced AAA infrastructure;
- high availability or failover behavior;
- large-scale endpoint inventory or onboarding workflows;
- enterprise certificate lifecycle and PKI operations;
- centralized NAC policy engines or posture frameworks;
- vendor switch telemetry, downloadable ACLs, or dynamic policy distribution.

These exclusions are intentional design boundaries. They do not diminish the proof-of-concept objective, which is to validate a coherent and technically defensible NAC incident workflow in a virtualized environment. In fact, by avoiding artificial breadth, the lab remains focused on a smaller set of claims that are strongly supported by direct evidence.

11.7 Section Conclusion

This lab should be read as a disciplined proof-of-concept implementation with intentional scope boundaries. It successfully demonstrates the targeted behaviors it set out to validate: wired 802.1X access control, protected and guest path differentiation, incident reproduction, remediation, automation-assisted validation, and packet-level protocol corroboration.

It should not be interpreted as a full production NAC architecture or as a complete enterprise security platform. The design choices documented here were made deliberately to keep the lab reproducible, explainable, and strongly evidenced. Those constraints do not weaken the value of the implementation. They define it accurately.

The next section will use this bounded interpretation as the basis for a final conclusion and for a short discussion of logical future expansion paths that would extend the lab while preserving the same evidence-driven methodology.

12. Conclusion and Future Expansion Opportunities

The lab documented in this report successfully met its original objective: to implement and validate a multi-segment network access control proof-of-concept that demonstrates wired 802.1X authentication, RADIUS-backed authorization, protected and guest path differentiation, incident simulation and remediation, lightweight automation, and packet-level protocol corroboration. Within the scope defined by the design, each major component was not only configured but also validated through direct operational evidence.

The authenticated path was shown to function through successful 802.1X supplicant behavior, hostapd authenticator processing, and FreeRADIUS acceptance. The guest path was shown to function through DHCP-based onboarding, gateway reachability, continued internet access, and enforced isolation from protected internal segments. The incident workflow further strengthened the implementation by proving that the environment could intentionally reproduce an authentication failure, preserve the resulting evidence, document the event in GLPI, restore correct configuration, and verify successful recovery. The lab demonstrated end-to-end engineering continuity as a coherent chain from design and segmentation through implementation, validation, troubleshooting, remediation, automation-assisted verification, and evidence preservation. The environment was built with enough rigor to support not only successful outcomes, but also controlled failure analysis and structured recovery.

As documented in Section 11, the lab is intentionally bounded and does not claim full enterprise production equivalence. By explicitly limiting the claims to what was actually implemented and verified, the document remains technically credible and avoids overstating the realism of the environment.

12.1 Summary of Accomplishments

At a practical level, the lab achieved the following validated outcomes:

- successful implementation of a multi-segment virtual network with distinct authenticated, guest, and service paths;
- successful wired 802.1X authentication flow using `wpa_supplicant`, `hostapd` in wired mode, and FreeRADIUS;
- successful demonstration of guest-network DHCP assignment and enforced isolation from protected internal segments;
- successful reproduction of a realistic NAC-style authentication failure caused by incorrect endpoint credentials;
- successful remediation of that failure and confirmation of restored authorized access;

- successful documentation of the incident in GLPI as an operational record;
- successful implementation of lightweight Ansible-based service validation with optional remediation behavior; and
- successful packet-level corroboration of both failed and successful authentication states using `tcpdump` and Wireshark.

Taken together, these outcomes demonstrate that the lab was systematically tested and evidenced.

12.2 Future Expansion Opportunities

Although the current implementation is complete within its stated scope, it also provides a strong foundation for future expansion. Several next-step enhancements would extend the realism or operational depth of the lab while preserving the same evidence-driven methodology used throughout this report.

A natural first expansion would be to replace the current EAP-MD5 demonstration method with a more production-aligned EAP model such as EAP-TLS, PEAP, or EAP-TTLS. This would significantly increase realism by introducing stronger authentication semantics and a more modern enterprise access-control pattern, while also creating opportunities to document certificate-based supplicant behavior and more advanced troubleshooting.

A second useful expansion would be to explore more dynamic authorization behavior. Within the constraints of a virtual lab, this could include experiments that more closely approximate switch-driven policy changes, segmented access outcomes, or more granular authorization handling beyond the current binary authenticated-versus-guest distinction. The goal would not be to claim full enterprise feature parity, but to extend the lab toward richer policy demonstrations.

A third expansion path would be to deepen observability. Section 9 intentionally documents a local Splunk Universal Forwarder monitored-input configuration rather than a full centralized analytics stack. A future iteration could extend this into a more complete logging pipeline with centralized indexing, search, and event correlation, thereby improving the analytical side of the incident workflow.

The automation layer also provides room for meaningful growth. The current Ansible playbook and Python parser are intentionally narrow, which is appropriate for the present proof-of-concept. Future versions could add richer health checks, more structured output formats, broader artifact collection logic, or more generalized remediation patterns while still preserving the validation-first operating model established in Section 8.

Scale realism could also be improved by introducing multiple authenticated endpoints or additional controlled failure scenarios. The current single-endpoint incident workflow is sufficient to prove the concept, but repeated validation across multiple clients or multiple failure classes would strengthen the repeatability argument and provide more varied operational evidence.

Finally, the GLPI workflow could be extended from its current manual but effective incident-record role into a more formalized evidence-management process. Future iterations could attach or reference more structured artifacts directly, impose a more consistent incident template, or align the ticketing workflow more closely with the operational outputs generated by the automation layer.

Each of these expansion opportunities is intentionally framed as future work. None of them are necessary to validate the success of the current build. Rather, they demonstrate that the lab has been constructed on a foundation strong enough to support meaningful iterative growth.

12.3 Final Assessment

In final assessment, the lab should be considered successful both as a technical implementation and as a professional documentation artifact. It achieved the intended proof-of-concept objectives, produced direct evidence for each major claim, and maintained strong internal consistency between architecture, validation, incident handling, automation, and protocol analysis.

Just as importantly, the document avoids the common failure mode of overstatement. It does not claim enterprise completeness where none was implemented. Instead, it presents a bounded, defensible, and well-evidenced engineering exercise that demonstrates practical competence across several adjacent domains.

Within its stated scope, the lab provides technically defensible documentation of a proof-of-concept for wired NAC validation.

Appendix A. Raw Configurations and Automation Artifacts

This appendix preserves the authoritative raw configurations and automation artifacts referenced throughout the report. All artifacts in this appendix are treated as final-state materials unless otherwise noted.

A.1 Host Network Configurations

A.1.1 radius-splunk Netplan Configuration

```
radius-splunk: /etc/netplan/50-cloud-init.yaml
network:
  version: 2
  ethernets:
    ens33:
      dhcp4: true
    ens38:
      dhcp4: false
      addresses:
```

```
        - 10.0.30.10/24
      routes:
        - to: 10.0.34.0/30
          via: 10.0.30.1
        - to: 10.0.23.0/24
          via: 10.0.30.1
        - to: 10.0.12.0/24
          via: 10.0.30.1
```

A.1.2 access-node Netplan Configuration

```
access-node: /etc/netplan/50-cloud-init.yaml
network:
  version: 2
  ethernets:
    ens33:
      dhcp4: true
    ens38:
      dhcp4: false
      addresses:
        - 10.0.12.1/24
    ens39:
      dhcp4: false
      addresses:
        - 10.0.23.1/24
      routes:
        - to: 10.0.30.0/24
          via: 10.0.23.2
```

A.1.3 dist-router Netplan Configuration

```
dist-router: /etc/netplan/50-cloud-init.yaml
network:
  version: 2
  ethernets:
    ens33:
      dhcp4: true
    ens38:
      dhcp4: false
      addresses:
        - 10.0.23.2/24
      routes:
        - to: 10.0.12.0/24
          via: 10.0.23.1
    ens39:
      dhcp4: false
```

```
      addresses:
        - 10.0.34.1/30
      routes:
        - to: 10.0.30.0/24
          via: 10.0.34.2
```

A.1.4 ops-glpi Netplan Configuration

```
ops-glpi: /etc/netplan/50-cloud-init.yaml
network:
  version: 2
  ethernets:
    ens33:
      dhcp4: true
    ens38:
      dhcp4: false
      addresses:
        - 10.0.30.20/24
      routes:
        - to: 10.0.34.0/30
          via: 10.0.30.1
        - to: 10.0.23.0/24
          via: 10.0.30.1
        - to: 10.0.12.0/24
          via: 10.0.30.1
```

A.1.5 client-1 Netplan Configuration

```
client-1: /etc/netplan/50-cloud-init.yaml
network:
  version: 2
  ethernets:
    ens33:
      addresses:
        - 10.0.12.10/24
      routes:
        - to: default
          via: 10.0.12.1
```

A.1.6 guest-test-1 Netplan Configuration

```
guest-test-1: /etc/netplan/50-cloud-init.yaml
network:
  version: 2
  ethernets:
    ens33:
      dhcp4: true
```

A.2 Authenticator, Supplicant, and AAA Configurations

A.2.1 hostapd Wired Configuration

```
access-node: /etc/hostapd/lab/hostapd-wired.conf
interface=ens38
driver=wired

ieee8021x=1
eapol_version=2
eap_reauth_period=3600

auth_server_addr=10.0.30.10
auth_server_port=1812
auth_server_shared_secret=<REDACTED>

own_ip_addr=10.0.23.1

logger_stdout=-1
logger_stdout_level=0
logger_syslog=-1
logger_syslog_level=0

ctrl_interface=/var/run/hostapd
ctrl_interface_group=0
```

A.2.2 hostapd Wired systemd Unit

```
access-node: /etc/systemd/system/hostapd-wired.service
[Unit]
Description=hostapd wired 802.1X authenticator (lab)
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
ExecStart=/usr/sbin/hostapd -dd /etc/hostapd/lab/hostapd-wired.conf
Restart=on-failure
RestartSec=2

[Install]
WantedBy=multi-user.target
```

A.2.3 client-1 wpa_supplicant Wired Configuration

```
client-1: /etc/wpa_supplicant/lab/wired-8021x.conf
ctrl_interface=/run/wpa_supplicant
```

```
ap_scan=0
eapol_version=2

network={
    key_mgmt=IEEE8021X
    eap=MD5
    identity="nacuser"
    password="<REDACTED>"
}
```

A.2.4 FreeRADIUS Clients Configuration

```
radius-splunk: /etc/freeradius/3.0/clients.conf
# -*- text -*-
##
## clients.conf -- client configuration directives
##
##  $Id: <REDACTED> $

#######################################################################
#
#  Define RADIUS clients (usually a NAS, Access Point, etc.).
#
#  [commentary omitted in summary]
#
client localhost {
    ipaddr = 127.0.0.1
    proto = *
    secret = testing123
    nas_type = other
    limit {
        max_connections = 16
        lifetime = 0
        idle_timeout = 30
    }
}

client localhost_ipv6 {
    ipv6addr    = ::1
    secret      = testing123
}

client access-node {
    ipaddr = 10.0.23.1
    secret = <REDACTED>
    shortname = access-node
```

```
}
```

A.2.5 FreeRADIUS User Authorization File

```
radius-splunk: /etc/freeradius/3.0/mods-config/files/authorize
#
#   Configuration file for the rlm_files module.
#   Please see rlm_files(5) manpage for more information.
#
#   [commentary omitted in summary]
#

DEFAULT Framed-Protocol == PPP
    Framed-Protocol = PPP,
    Framed-Compression = Van-Jacobson-TCP-IP

DEFAULT Hint == "CSLIP"
    Framed-Protocol = SLIP,
    Framed-Compression = Van-Jacobson-TCP-IP

DEFAULT Hint == "SLIP"
    Framed-Protocol = SLIP

############################################################
# You should add test accounts to the TOP of this file! #
# See the example user "bob" above.                      #
############################################################

nacuser Cleartext-Password := "<REDACTED>"
```

A.3 pfSense Manual Configuration Transcription

A.3.1 pfSense Authenticated / Protected Interface Summary

```
WAN (wan) -> em0 -> v4/DHCP4: 192.168.195.145/24
LAN (lan) -> em1 -> v4: 10.0.34.2/30
OPT1 (opt1) -> em2 -> v4: 10.0.30.1/24
GUEST (opt2) -> em3 -> v4: 10.0.20.254/24
```

A.3.2 pfSense Guest Interface Summary

```
Enabled (Enable DHCP server on GUEST interface)
Subnet: 10.0.20.0/24
Subnet Range: 10.0.20.1 - 10.0.20.254
Address Pool Range: 10.0.20.100 - 10.0.20.199
```

A.3.3 pfSense Guest Firewall Rules

```
Pass  | IPv4* | Source: 10.0.20.0/24 | Destination: This Firewall (Self)
        | ALLOW-GUEST-TO-PFSENSE
Block | IPv4* | Source: 10.0.20.0/24 | Destination: 10.0.30.0/24
        | BLOCK-GUEST-TO-SERVICES
Block | IPv4* | Source: 10.0.20.0/24 | Destination: 10.0.34.0/30
        | BLOCK-GUEST-TO-TRANSIT
Block | IPv4* | Source: 10.0.20.0/24 | Destination: 10.0.23.0/24
        | BLOCK-GUEST-TO-DIST
Block | IPv4* | Source: 10.0.20.0/24 | Destination: 10.0.12.0/24
        | BLOCK-GUEST-TO-AUTH-ACCESS
Pass  | IPv4* | Source: 10.0.20.0/24 | Destination: *
        | ALLOW-GUEST-TO-INTERNET-AND-OTHER-NON-INTERNAL
```

A.3.4 pfSense Outbound NAT Summary

```
Automatic
```

A.4 Automation Artifacts

A.4.1 Ansible Inventory

```
radius-splunk: ~/ansible-nac/inventory/hosts.ini
[nac_nodes]
access-node ansible_host=10.0.23.1 ansible_user=nac-lab
radius-splunk ansible_host=127.0.0.1 ansible_connection=local

[authenticator]
access-node

[radius]
radius-splunk
```

A.4.2 Ansible NAC Validation Playbook

```
radius-splunk: ~/ansible-nac/playbooks/nac_validation.yml
---
- name: Validate NAC services and optionally remediate
  hosts: nac_nodes
  gather_facts: false
  become: false

  vars:
    remediate: false
    artifact_dir: "/home/nac-lab/ansible-nac/artifacts"
```

```
tasks:
  - name: Collect basic hostname
    command: hostname
    register: hostname_out
    changed_when: false

  - name: Check hostapd-wired service on access-node
    become: true
    command: systemctl status hostapd-wired.service --no-pager
    register: hostapd_status_before
    changed_when: false
    failed_when: false
    when: inventory_hostname == "access-node"

  - name: Check freeradius service on radius-splunk
    become: true
    command: systemctl status freeradius --no-pager
    register: freeradius_status_before
    changed_when: false
    failed_when: false
    when: inventory_hostname == "radius-splunk"

  - name: Restart hostapd-wired service when remediation is enabled
    become: true
    command: systemctl restart hostapd-wired.service
    register: hostapd_restart
    changed_when: true
    failed_when: false
    when:
      - inventory_hostname == "access-node"
      - remediate | bool

  - name: Restart freeradius service when remediation is enabled
    become: true
    command: systemctl restart freeradius
    register: freeradius_restart
    changed_when: true
    failed_when: false
    when:
      - inventory_hostname == "radius-splunk"
      - remediate | bool

  - name: Re-check hostapd-wired service on access-node
    become: true
    command: systemctl status hostapd-wired.service --no-pager
    register: hostapd_status_after
```

```
    changed_when: false
    failed_when: false
    when: inventory_hostname == "access-node"

  - name: Re-check freeradius service on radius-splunk
    become: true
    command: systemctl status freeradius --no-pager
    register: freeradius_status_after
    changed_when: false
    failed_when: false
    when: inventory_hostname == "radius-splunk"

  - name: Collect recent hostapd logs
    become: true
    command: journalctl -u hostapd-wired.service -n 40 --no-pager
    register: hostapd_logs
    changed_when: false
    failed_when: false
    when: inventory_hostname == "access-node"

  - name: Collect recent freeradius service logs
    become: true
    command: journalctl -u freeradius -n 40 --no-pager
    register: freeradius_logs
    changed_when: false
    failed_when: false
    when: inventory_hostname == "radius-splunk"

  - name: Collect recent freeradius application log
    become: true
    command: tail -n 40 /var/log/freeradius/radius.log
    register: freeradius_radiuslog
    changed_when: false
    failed_when: false
    when: inventory_hostname == "radius-splunk"

  - name: Save hostapd pre-check status locally
    become: false
    delegate_to: localhost
    copy:
      content: "{{ hostapd_status_before.stdout | default('N/A') }}"
      dest: "{{ artifact_dir }}/{{ inventory_hostname }}_hostapd_status_before.txt"
    when: inventory_hostname == "access-node"

  - name: Save hostapd post-check status locally
    become: false
```

```
    delegate_to: localhost
    copy:
      content: "{{ hostapd_status_after.stdout | default('N/A') }}"
      dest: "{{ artifact_dir }}/{{ inventory_hostname }}_hostapd_status_after.txt"
    when: inventory_hostname == "access-node"

  - name: Save hostapd restart result locally
    become: false
    delegate_to: localhost
    copy:
      content: "{{ hostapd_restart.stdout | default('remediation not requested') }}"
      dest: "{{ artifact_dir }}/{{ inventory_hostname }}_hostapd_restart_result.txt"
    when: inventory_hostname == "access-node"

  - name: Save hostapd logs locally
    become: false
    delegate_to: localhost
    copy:
      content: "{{ hostapd_logs.stdout | default('N/A') }}"
      dest: "{{ artifact_dir }}/{{ inventory_hostname }}_hostapd_logs.txt"
    when: inventory_hostname == "access-node"

  - name: Save freeradius pre-check status locally
    become: false
    delegate_to: localhost
    copy:
      content: "{{ freeradius_status_before.stdout | default('N/A') }}"
      dest: "{{ artifact_dir }}/{{ inventory_hostname }}_freeradius_status_before.txt"
    when: inventory_hostname == "radius-splunk"

  - name: Save freeradius post-check status locally
    become: false
    delegate_to: localhost
    copy:
      content: "{{ freeradius_status_after.stdout | default('N/A') }}"
      dest: "{{ artifact_dir }}/{{ inventory_hostname }}_freeradius_status_after.txt"
    when: inventory_hostname == "radius-splunk"

  - name: Save freeradius restart result locally
    become: false
    delegate_to: localhost
    copy:
      content: "{{ freeradius_restart.stdout | default('remediation not requested') }}"
      dest: "{{ artifact_dir }}/{{ inventory_hostname }}_freeradius_restart_result.txt"
    when: inventory_hostname == "radius-splunk"
```

```
    - name: Save freeradius service logs locally
      become: false
      delegate_to: localhost
      copy:
        content: "{{ freeradius_logs.stdout | default('N/A') }}"
        dest: "{{ artifact_dir }}/{{ inventory_hostname }}_freeradius_journal.txt"
      when: inventory_hostname == "radius-splunk"

    - name: Save freeradius application log locally
      become: false
      delegate_to: localhost
      copy:
        content: "{{ freeradius_radiuslog.stdout | default('N/A') }}"
        dest: "{{ artifact_dir }}/{{ inventory_hostname }}_freeradius_radiuslog.txt"
      when: inventory_hostname == "radius-splunk"
```

A.4.3 Python NAC Incident Parser

```
radius-splunk: ~/incident-evidence/nac_incident_parser.py
#!/usr/bin/env python3

from pathlib import Path

BASE = Path.home() / "incident-evidence"

FILES = {
    "client_failed": BASE / "wpa_supplicant_failed_auth_incident.txt",
    "client_success": BASE / "wpa_supplicant_success_retest_v2.txt",
    "hostapd_failed": BASE / "hostapd_failed_auth_incident.txt",
    "hostapd_success": BASE / "hostapd_success_retest_v2.txt",
    "radius_failed": BASE / "freeradius_failed_auth_incident.txt",
}

def read_text(path):
    try:
        return path.read_text(errors="ignore")
    except FileNotFoundError:
        return None

def present(text, needles):
    if text is None:
        return False
    return any(n in text for n in needles)

def summarize():
    data = {name: read_text(path) for name, path in FILES.items()}
```

```
results = {
    "client_failed_unauthorized": present(data["client_failed"], [
        "Supplicant port status: Unauthorized"
    ]),
    "client_failed_connected": present(data["client_failed"], [
        "CTRL-EVENT-CONNECTED"
    ]),
    "client_success_connected": present(data["client_success"], [
        "CTRL-EVENT-CONNECTED"
    ]),
    "client_success_authorized": present(data["client_success"], [
        "Supplicant port status: Authorized"
    ]),
    "hostapd_failed_attempt_seen": present(data["hostapd_failed"], [
        "IEEE 802.1X: start authentication",
        "RADIUS Sending RADIUS message to authentication server",
        "CTRL-EVENT-EAP-STARTED",
        "recvmsg [RADIUS]"
    ]),
    "hostapd_success_authorized": present(data["hostapd_success"], [
        "AP-STA-CONNECTED",
        "authorizing port",
        "CTRL-EVENT-EAP-SUCCESS"
    ]),
    "radius_failed_reject": present(data["radius_failed"], [
        "Failed to authenticate the user",
        "Using Post-Auth-Type Reject",
        "EAP Failure",
        "Sending EAP Failure"
    ]),
}

print("=== NAC Incident Evidence Summary ===")
print()

print("[1] Failed authentication simulation")
print(f"- Client remained unauthorized: {'YES' if results['client_failed_unauthorized']
        else 'NO'}")
print(f"- Client showed successful connection during failed test: {'YES' if
        results['client_failed_connected'] else 'NO'}")
print(f"- hostapd saw authentication attempt: {'YES' if
        results['hostapd_failed_attempt_seen'] else 'NO'}")
print(f"- FreeRADIUS recorded failed auth / reject path: {'YES' if
        results['radius_failed_reject'] else 'NO'}")
print()
```

```
    print("[2] Remediation / successful re-test")
    print(f"- Client reached CTRL-EVENT-CONNECTED: {'YES' if
            results['client_success_connected'] else 'NO'}")
    print(f"- Client reached Authorized state: {'YES' if
            results['client_success_authorized'] else 'NO'}")
    print(f"- hostapd showed successful authorization indicators: {'YES' if
            results['hostapd_success_authorized'] else 'NO'}")
    print()

    if (
        results["client_failed_unauthorized"]
        and results["radius_failed_reject"]
        and results["client_success_connected"]
        and results["client_success_authorized"]
        and results["hostapd_success_authorized"]
    ):
        print("[3] Incident conclusion")
        print("- SIMULATED NAC INCIDENT SUCCESSFULLY REPRODUCED AND REMEDIATED")
    else:
        print("[3] Incident conclusion")
        print("- Evidence is incomplete or mixed; review raw logs for additional detail")

if __name__ == "__main__":
    summarize()
```

A.4.4 Ansible Artifact Directory Listing

```
radius-splunk: ls -lh ~/ansible-nac/artifacts
total 44K
-rw-r--r-- 1 nac-lab nac-lab 3.5K Mar 24 12:23 access-node_hostapd_logs.txt
-rw-r--r-- 1 nac-lab nac-lab   25 Mar 24 12:43 access-node_hostapd_restart_result.txt
-rw-r--r-- 1 nac-lab nac-lab 1.4K Mar 24 12:43 access-node_hostapd_status_after.txt
-rw-r--r-- 1 nac-lab nac-lab 1.4K Mar 24 12:43 access-node_hostapd_status_before.txt
-rw-r--r-- 1 root    root    1.4K Mar 23 19:30 access-node_hostapd_status.txt
-rw-r--r-- 1 nac-lab nac-lab 4.0K Mar 24 12:23 radius-splunk_freeradius_journal.txt
-rw-r--r-- 1 nac-lab nac-lab 3.1K Mar 24 12:23 radius-splunk_freeradius_radiuslog.txt
-rw-r--r-- 1 nac-lab nac-lab   25 Mar 24 12:43 radius-splunk_freeradius_restart_result.txt
-rw-r--r-- 1 nac-lab nac-lab 1.8K Mar 24 12:43 radius-splunk_freeradius_status_after.txt
-rw-r--r-- 1 nac-lab nac-lab 1.8K Mar 24 12:43 radius-splunk_freeradius_status_before.txt
-rw-r--r-- 1 root    root    1.8K Mar 23 19:30 radius-splunk_freeradius_status.txt
```

A.4.5 Incident Evidence Directory Listing

```
radius-splunk: ls -lh ~/incident-evidence
total 92K
```

```
-rw-rw-r-- 1 nac-lab nac-lab  32K Mar 23 04:02 freeradius_failed_auth_incident.txt
-rw-rw-r-- 1 nac-lab nac-lab  11K Mar 23 04:01 hostapd_failed_auth_incident.txt
-rw-rw-r-- 1 nac-lab nac-lab  19K Mar 23 04:01 hostapd_success_retest_v2.txt
-rw-r--r-- 1 root    root    3.5K Mar 24 09:52 nac_incident_parser.py
-rw-rw-r-- 1 nac-lab nac-lab  535 Mar 23 04:28 nac_incident_parser_success.txt
-rw-rw-r-- 1 nac-lab nac-lab 7.3K Mar 23 03:59 wpa_supplicant_failed_auth_incident.txt
-rw-rw-r-- 1 nac-lab nac-lab 8.4K Mar 23 03:59 wpa_supplicant_success_retest_v2.txt
```

A.4.6 Splunk Universal Forwarder inputs.conf

```
radius-splunk: /opt/splunkforwarder/etc/apps/lab_nac_inputs/local/inputs.conf
[monitor:///var/log/syslog]
disabled = false
index = main
sourcetype = syslog

[monitor:///var/log/auth.log]
disabled = false
index = main
sourcetype = linux_secure

[monitor:///var/log/freeradius/radius.log]
disabled = false
index = main
sourcetype = freeradius
```

Appendix B. Incident, Validation, and Evidence Artifacts

This appendix preserves the authoritative raw evidence artifacts referenced throughout the report. The materials included here are grouped by function: incident reproduction evidence, GLPI incident record content, guest-path validation outputs, automation execution evidence, selected observability excerpts, and packet-capture metadata.

B.1 Incident Evidence Files

B.1.1 wpa_supplicant Failed Authentication Log

```
radius-splunk: ~/incident-evidence/wpa_supplicant_failed_auth_incident.txt
wpa_supplicant v2.10
random: getrandom() support available
Successfully initialized wpa_supplicant
Initializing interface 'ens33' conf '/etc/wpa_supplicant/lab/wired-8021x.conf' driver
        'wired' ctrl_interface 'N/A' bridge 'N/A'
```

```
Configuration file '/etc/wpa_supplicant/lab/wired-8021x.conf' -> '/etc/wpa_supplicant/lab/
        wired-8021x.conf'
Reading configuration file '/etc/wpa_supplicant/lab/wired-8021x.conf'
ctrl_interface='/run/wpa_supplicant'
ap_scan=0
eapol_version=2
Line: 5 - start of a new network block
key_mgmt: 0x8
eap methods - hexdump(len=16): 00 00 00 00 04 00 00 00 00 00 00 00 00 00 00 00
identity - hexdump_ascii(len=7):
     6e 61 63 75 73 65 72                             nacuser
password - hexdump_ascii(len=13): [REMOVED]
Priority group 0
   id=0 ssid=''
driver_wired_init_common: Added multicast membership with packet socket
Add interface ens33 to a new radio N/A
ens33: Own MAC address: <REDACTED>
ens33: RSN: flushing PMKID list in the driver
ens33: Setting scan request: 0.100000 sec
TDLS: TDLS operation not supported by driver
TDLS: Driver uses internal link setup
TDLS: Driver does not support TDLS channel switching
ens33: WPS: UUID based on MAC address: <REDACTED>
ENGINE: Loading builtin engines
ENGINE: Loading builtin engines
EAPOL: SUPP_PAE entering state DISCONNECTED
EAPOL: Supplicant port status: Unauthorized
EAPOL: KEY_RX entering state NO_KEY_RECEIVE
EAPOL: SUPP_BE entering state INITIALIZE
EAP: EAP entering state DISABLED
MBO: Update non-preferred channels, non_pref_chan=N/A
ens33: Added interface ens33
ens33: State: DISCONNECTED -> DISCONNECTED
EAPOL: External notification - EAP success=0
EAPOL: External notification - EAP fail=0
EAPOL: External notification - portControl=Auto
ens33: Already associated with a configured network - generating associated event
ens33: Event ASSOC (0) received
ens33: Association info event
ens33: State: DISCONNECTED -> ASSOCIATED
ens33: Associated to a new BSS: BSSID=<REDACTED>
ens33: Select network based on association information
ens33: Network configuration found for the current AP
ens33: WPA: clearing AP WPA IE
ens33: WPA: clearing AP RSN IE
ens33: WPA: clearing AP RSNXE
```

```
ens33: WPA: clearing own WPA/RSN IE
ens33: RSN: clearing own RSNXE
ens33: Failed to get scan results
EAPOL: External notification - EAP success=0
EAPOL: External notification - EAP fail=0
EAPOL: External notification - portControl=Auto
ens33: Associated with <REDACTED>
ens33: WPA: Association event - clear replay counter
ens33: WPA: Clear old PTK
TDLS: Remove peers on association
EAPOL: External notification - portEnabled=0
EAPOL: External notification - portValid=0
EAPOL: External notification - portEnabled=1
EAPOL: SUPP_PAE entering state CONNECTING
EAPOL: SUPP_BE entering state IDLE
EAP: EAP entering state INITIALIZE
EAP: EAP entering state IDLE
ens33: Cancelling scan request
ens33: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
EAPOL: startWhen --> 0
EAPOL: SUPP_PAE entering state CONNECTING
EAPOL: txStart
TX EAPOL: dst=01:80:c2:00:00:03
TX EAPOL - hexdump(len=4): 02 01 00 00
l2_packet_receive: src=<REDACTED> len=46
ens33: RX EAPOL from <REDACTED>
RX EAPOL - hexdump(len=46): 02 00 00 05 01 19 00 05 01 00 00 00 00 00 00 00 00 00 00 00
         00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
EAPOL: Received EAP-Packet frame
EAPOL: SUPP_PAE entering state RESTART
EAP: EAP entering state INITIALIZE
EAP: EAP entering state IDLE
EAPOL: SUPP_PAE entering state AUTHENTICATING
EAPOL: SUPP_BE entering state REQUEST
EAPOL: getSuppRsp
EAP: EAP entering state RECEIVED
EAP: Received EAP-Request id=25 method=1 vendor=0 vendorMethod=0
EAP: EAP entering state IDENTITY
ens33: CTRL-EVENT-EAP-STARTED EAP authentication started
EAP: Status notification: started (param=)
EAP: EAP-Request Identity data - hexdump_ascii(len=0):
EAP: using real identity - hexdump_ascii(len=7):
     6e 61 63 75 73 65 72                              nacuser
EAP: EAP entering state SEND_RESPONSE
EAP: EAP entering state IDLE
EAPOL: SUPP_BE entering state RESPONSE
```

```
EAPOL: txSuppRsp
TX EAPOL: dst=<REDACTED>
TX EAPOL - hexdump(len=16): 02 00 00 0c 02 19 00 0c 01 6e 61 63 75 73 65 72
EAPOL: SUPP_BE entering state RECEIVE
l2_packet_receive: src=<REDACTED> len=46
ens33: RX EAPOL from <REDACTED>
RX EAPOL - hexdump(len=46): 02 00 00 16 01 1a 00 16 04 10 d8 d1 ca 68 87 96 28 22 03 d8 a5
         44 92 bb 84 f9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
EAPOL: Received EAP-Packet frame
EAPOL: SUPP_BE entering state REQUEST
EAPOL: getSuppRsp
EAP: EAP entering state RECEIVED
EAP: Received EAP-Request id=26 method=4 vendor=0 vendorMethod=0
EAP: EAP entering state GET_METHOD
ens33: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=4
EAP: Status notification: accept proposed method (param=MD5)
EAP: Initialize selected EAP method: vendor 0 method 4 (MD5)
ens33: CTRL-EVENT-EAP-METHOD EAP vendor 0 method 4 (MD5) selected
EAP: EAP entering state METHOD
EAP-MD5: Challenge - hexdump(len=16): d8 d1 ca 68 87 96 28 22 03 d8 a5 44 92 bb 84 f9
EAP-MD5: Generating Challenge Response
EAP-MD5: Response - hexdump(len=16): 41 b2 c0 1c cf a9 4d 12 fc 49 1a 04 d9 5c d9 35
EAP: method process -> ignore=FALSE methodState=DONE decision=COND_SUCC eapRespData=0x62e7
        14bc0420
EAP: EAP entering state SEND_RESPONSE
EAP: EAP entering state IDLE
EAPOL: SUPP_BE entering state RESPONSE
EAPOL: txSuppRsp
TX EAPOL: dst=<REDACTED>
TX EAPOL - hexdump(len=26): 02 00 00 16 02 1a 00 16 04 10 41 b2 c0 1c cf a9 4d 12 fc 49 1a
         04 d9 5c d9 35
EAPOL: SUPP_BE entering state RECEIVE
ens33: Removing interface ens33
ens33: Request to deauthenticate - bssid=<REDACTED> pending_bssid=00:00:00:00:00:00 reason
        =3 (DEAUTH_LEAVING) state=ASSOCIATED
TDLS: Tear down peers
ens33: Event DEAUTH (11) received
ens33: Deauthentication notification
ens33:  * reason 3 (DEAUTH_LEAVING) locally_generated=1
Deauthentication frame IE(s) - hexdump(len=0): [NULL]
ens33: CTRL-EVENT-DISCONNECTED bssid=<REDACTED> reason=3 locally_generated=1
ens33: Auto connect disabled: do not try to re-connect
ens33: Ignore connection failure indication since interface has been put into disconnected
         state
TDLS: Remove peers on disassociation
ens33: WPA: Clear old PMK and PTK
```

```
ens33: Disconnect event - remove keys
ens33: State: ASSOCIATED -> DISCONNECTED
EAPOL: External notification - portEnabled=0
EAPOL: SUPP_PAE entering state DISCONNECTED
EAPOL: Supplicant port status: Unauthorized
EAPOL: SUPP_BE entering state INITIALIZE
EAP: EAP entering state DISABLED
EAPOL: External notification - portValid=0
ens33: State: DISCONNECTED -> DISCONNECTED
EAPOL: External notification - portEnabled=0
EAPOL: External notification - portValid=0
QM: Clear all active DSCP policies
ens33: CTRL-EVENT-DSCP-POLICY clear_all
EAP: deinitialize previously used EAP method (4, MD5) at EAP deinit
ens33: WPA: Clear old PMK and PTK
ens33: Cancelling scan request
ens33: Cancelling authentication timeout
Off-channel: Clear pending Action frame TX (pending_action_tx=(nil)
HS20: Delete all stored icons
Off-channel: Action frame sequence done notification: pending_action_tx=(nil) drv_offchan_
        tx=0 action_tx_wait_time=0 off_channel_freq=0 roc_waiting_drv_freq=0
QM: Clear all active DSCP policies
ens33: CTRL-EVENT-DSCP-POLICY clear_all
Remove interface ens33 from radio
Remove radio
ens33: CTRL-EVENT-TERMINATING
```

B.1.2 hostapd Failed Authentication Log

```
radius-splunk: ~/incident-evidence/hostapd_failed_auth_incident.txt
Mar 22 22:32:33 access-node hostapd[2052]: EAP-Identity: Peer identity - hexdump_ascii(len
        =7):
Mar 22 22:32:33 access-node hostapd[2052]:      6e 61 63 75 73 65 72
                nacuser
Mar 22 22:32:33 access-node hostapd[2052]: EAP: EAP entering state SELECT_ACTION
Mar 22 22:32:33 access-node hostapd[2052]: EAP: getDecision: -> PASSTHROUGH
Mar 22 22:32:33 access-node hostapd[2052]: EAP: EAP entering state INITIALIZE_PASSTHROUGH
Mar 22 22:32:33 access-node hostapd[2052]: EAP: EAP entering state AAA_REQUEST
Mar 22 22:32:33 access-node hostapd[2052]: EAP: EAP entering state AAA_IDLE
Mar 22 22:32:33 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: STA identity
         'nacuser'
Mar 22 22:32:33 access-node hostapd[2052]: Encapsulating EAP message into a RADIUS packet
Mar 22 22:32:33 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: unauthorizin
        g port
Mar 22 22:32:33 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: Sending EAP
        Packet (identifier 25)
```

```
Mar 22 22:32:33 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: received EAP
        packet (code=2 id=25 len=12) from STA: EAP Response-Identity (1)
Mar 22 22:32:33 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: STA identity
        'nacuser'
Mar 22 22:32:33 access-node hostapd[2052]: ens38: RADIUS Sending RADIUS message to authent
       ication server
Mar 22 22:32:33 access-node hostapd[2052]: RADIUS message: code=1 (Access-Request) identif
       ier=0 length=165
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 1 (User-Name) length=9
Mar 22 22:32:33 access-node hostapd[2052]:       Value: 'nacuser'
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 4 (NAS-IP-Address) length=6
Mar 22 22:32:33 access-node hostapd[2052]:       Value: 10.0.23.1
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 30 (Called-Station-Id) length=20
Mar 22 22:32:33 access-node hostapd[2052]:       Value: '<REDACTED>:'
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 61 (NAS-Port-Type) length=6
Mar 22 22:32:33 access-node hostapd[2052]:       Value: 19
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 6 (Service-Type) length=6
Mar 22 22:32:33 access-node hostapd[2052]:       Value: 2
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 31 (Calling-Station-Id) length=19
Mar 22 22:32:33 access-node hostapd[2052]:       Value: '<REDACTED>'
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 77 (Connect-Info) length=23
Mar 22 22:32:33 access-node hostapd[2052]:       Value: 'CONNECT 0Mbps 802.11b'
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 44 (Acct-Session-Id) length=18
Mar 22 22:32:33 access-node hostapd[2052]:       Value: '1A082DB0BEE2BD34'
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 12 (Framed-MTU) length=6
Mar 22 22:32:33 access-node hostapd[2052]:       Value: 1400
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 79 (EAP-Message) length=14
Mar 22 22:32:33 access-node hostapd[2052]:       Value: 0219000c016e616375736572
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 80 (Message-Authenticator) length=
       18
Mar 22 22:32:33 access-node hostapd[2052]:       Value: 71423639f013148f1ce7a71576679bbe
Mar 22 22:32:33 access-node hostapd[2052]: ens38: RADIUS Next RADIUS client retransmit in
       3 seconds
Mar 22 22:32:33 access-node hostapd[2052]: ens38: RADIUS Sending RADIUS message to authent
       ication server
Mar 22 22:32:33 access-node hostapd[2052]: ens38: RADIUS Next RADIUS client retransmit in
       3 seconds
Mar 22 22:32:33 access-node hostapd[2052]: ens38: RADIUS Received 80 bytes from RADIUS ser
       ver
Mar 22 22:32:33 access-node hostapd[2052]: ens38: RADIUS Received RADIUS message
Mar 22 22:32:33 access-node hostapd[2052]: RADIUS message: code=11 (Access-Challenge) iden
       tifier=0 length=80
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 80 (Message-Authenticator) length=18
Mar 22 22:32:33 access-node hostapd[2052]:       Value: e72a3944e289c8a2d9b97adfbceb639a
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 79 (EAP-Message) length=24
Mar 22 22:32:33 access-node hostapd[2052]:       Value: 011a00160410d8d1ca688796282203d8a5
```

```
        4492bb84f9
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 24 (State) length=18
Mar 22 22:32:33 access-node hostapd[2052]:       Value: bb99b235bb83b6a0b74fb036b1c91a02
Mar 22 22:32:33 access-node hostapd[2052]: ens38: STA <REDACTED> RADIUS: Received RADIUS p
        acket matched with a pending request, round trip time 0.00 sec
Mar 22 22:32:33 access-node hostapd[2052]: ens38: RADIUS Received 80 bytes from RADIUS ser
        ver
Mar 22 22:32:33 access-node hostapd[2052]: ens38: RADIUS Received RADIUS message
Mar 22 22:32:33 access-node hostapd[2052]: ens38: STA <REDACTED> RADIUS: Received RADIUS p
        acket matched with a pending request, round trip time 0.00 sec
Mar 22 22:32:33 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: decapsulated
         EAP packet (code=1 id=26 len=22) from RADIUS server: EAP-Request-MD5 (4)
Mar 22 22:32:33 access-node hostapd[2052]: RADIUS packet matching with station 00:0c:29:68
        :53:33
Mar 22 22:32:33 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: decapsulated
         EAP packet (code=1 id=26 len=22) from RADIUS server: EAP-Request-MD5 (4)
Mar 22 22:32:33 access-node hostapd[2052]: EAP: EAP entering state AAA_RESPONSE
Mar 22 22:32:33 access-node hostapd[2052]: EAP: getId: id=26
Mar 22 22:32:33 access-node hostapd[2052]: EAP: EAP entering state SEND_REQUEST2
Mar 22 22:32:33 access-node hostapd[2052]: EAP: EAP entering state IDLE2
Mar 22 22:32:33 access-node hostapd[2052]: EAP: retransmit timeout 3 seconds (from dynamic
         back off; retransCount=0)
Mar 22 22:32:33 access-node hostapd[2052]: IEEE 802.1X: <REDACTED> BE_AUTH entering state
        REQUEST
Mar 22 22:32:33 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: Sending EAP
        Packet (identifier 26)
Mar 22 22:32:33 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: Sending EAP
        Packet (identifier 26)
Mar 22 22:32:33 access-node hostapd[2052]: Received EAPOL packet
Mar 22 22:32:33 access-node hostapd[2052]: ens38: Event NEW_STA (22) received
Mar 22 22:32:33 access-node hostapd[2052]: ens38: Event EAPOL_RX (23) received
Mar 22 22:32:33 access-node hostapd[2052]: IEEE 802.1X: 46 bytes from 00:0c:29:68:53:33
Mar 22 22:32:33 access-node hostapd[2052]:    IEEE 802.1X: version=2 type=0 length=22
Mar 22 22:32:33 access-node hostapd[2052]:    ignoring 20 extra octets after IEEE 802.1X p
        acket
Mar 22 22:32:33 access-node hostapd[2052]: EAP: code=2 (response) identifier=26 length=22
Mar 22 22:32:33 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: received EAP
         packet (code=2 id=26 len=22) from STA: EAP Response-MD5 (4)
Mar 22 22:32:33 access-node hostapd[2052]: IEEE 802.1X: <REDACTED> BE_AUTH entering state
        RESPONSE
Mar 22 22:32:33 access-node hostapd[2052]: EAP: EAP entering state RECEIVED2
Mar 22 22:32:33 access-node hostapd[2052]: EAP: parseEapResp: rxResp=1 rxInitiate=0 respId
        =26 respMethod=4 respVendor=0 respVendorMethod=0
Mar 22 22:32:33 access-node hostapd[2052]: EAP: EAP entering state AAA_REQUEST
Mar 22 22:32:33 access-node hostapd[2052]: EAP: EAP entering state AAA_IDLE
Mar 22 22:32:33 access-node hostapd[2052]: Encapsulating EAP message into a RADIUS packet
```

```
Mar 22 22:32:33 access-node hostapd[2052]: Copied RADIUS State Attribute
Mar 22 22:32:33 access-node hostapd[2052]: ens38: RADIUS Sending RADIUS message to authent
        ication server
Mar 22 22:32:33 access-node hostapd[2052]: RADIUS message: code=1 (Access-Request) identif
        ier=1 length=193
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 1 (User-Name) length=9
Mar 22 22:32:33 access-node hostapd[2052]:       Value: 'nacuser'
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 4 (NAS-IP-Address) length=6
Mar 22 22:32:33 access-node hostapd[2052]:       Value: 10.0.23.1
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 30 (Called-Station-Id) length=20
Mar 22 22:32:33 access-node hostapd[2052]:       Value: '<REDACTED>:'
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 61 (NAS-Port-Type) length=6
Mar 22 22:32:33 access-node hostapd[2052]:       Value: 19
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 6 (Service-Type) length=6
Mar 22 22:32:33 access-node hostapd[2052]:       Value: 2
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 31 (Calling-Station-Id) length=19
Mar 22 22:32:33 access-node hostapd[2052]:       Value: '<REDACTED>'
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 77 (Connect-Info) length=23
Mar 22 22:32:33 access-node hostapd[2052]:       Value: 'CONNECT 0Mbps 802.11b'
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 44 (Acct-Session-Id) length=18
Mar 22 22:32:33 access-node hostapd[2052]:       Value: '1A082DB0BEE2BD34'
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 12 (Framed-MTU) length=6
Mar 22 22:32:33 access-node hostapd[2052]:       Value: 1400
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 79 (EAP-Message) length=24
Mar 22 22:32:33 access-node hostapd[2052]:       Value: 021a0016041041b2c01ccfa94d12fc491a
        04d95cd935
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 24 (State) length=18
Mar 22 22:32:33 access-node hostapd[2052]:       Value: bb99b235bb83b6a0b74fb036b1c91a02
Mar 22 22:32:33 access-node hostapd[2052]:    Attribute 80 (Message-Authenticator) length=
        18
Mar 22 22:32:33 access-node hostapd[2052]:       Value: ee00aa03a7391d212502d3e4595c510c
Mar 22 22:32:33 access-node hostapd[2052]: ens38: RADIUS Next RADIUS client retransmit in
        3 seconds
Mar 22 22:32:33 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: received EAP
         packet (code=2 id=26 len=22) from STA: EAP Response-MD5 (4)
Mar 22 22:32:33 access-node hostapd[2052]: ens38: RADIUS Sending RADIUS message to authent
        ication server
Mar 22 22:32:33 access-node hostapd[2052]: ens38: RADIUS Next RADIUS client retransmit in
        3 seconds
Mar 22 22:32:36 access-node hostapd[2052]: IEEE 802.1X: <REDACTED> - (EAP) retransWhile --
        > 0
Mar 22 22:32:36 access-node hostapd[2052]: ens38: STA <REDACTED> RADIUS: Resending RADIUS
        message (id=1)
Mar 22 22:32:36 access-node hostapd[2052]: ens38: STA <REDACTED> RADIUS: Resending RADIUS
        message (id=1)
Mar 22 22:32:36 access-node hostapd[2052]: ens38: RADIUS Next RADIUS client retransmit in
```

```
        6 seconds
Mar 22 22:32:36 access-node hostapd[2052]: ens38: RADIUS Next RADIUS client retransmit in
        6 seconds
Mar 22 22:32:36 access-node hostapd[2052]: recvmsg[RADIUS]: Connection refused
Mar 22 22:32:42 access-node hostapd[2052]: ens38: STA <REDACTED> RADIUS: Resending RADIUS
        message (id=1)
Mar 22 22:32:42 access-node hostapd[2052]: ens38: RADIUS Next RADIUS client retransmit in
        12 seconds
Mar 22 22:32:42 access-node hostapd[2052]: ens38: STA <REDACTED> RADIUS: Resending RADIUS
        message (id=1)
Mar 22 22:32:42 access-node hostapd[2052]: ens38: RADIUS Next RADIUS client retransmit in
        12 seconds
Mar 22 22:32:42 access-node hostapd[2052]: recvmsg[RADIUS]: Connection refused
```

B.1.3 FreeRADIUS Failed Authentication Log

```
radius-splunk: ~/incident-evidence/freeradius_failed_auth_incident.txt
FreeRADIUS Version 3.2.5
Copyright (C) 1999-2023 The FreeRADIUS server project and contributors
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE
You may redistribute copies of FreeRADIUS under the terms of the
GNU General Public License
For more information about these matters, see the file named COPYRIGHT
Starting - reading configuration files ...
including dictionary file /usr/share/freeradius/dictionary
including dictionary file /usr/share/freeradius/dictionary.dhcp
including dictionary file /usr/share/freeradius/dictionary.vqp
including dictionary file /etc/freeradius/3.0/dictionary
including configuration file /etc/freeradius/3.0/radiusd.conf
including configuration file /etc/freeradius/3.0/proxy.conf
including configuration file /etc/freeradius/3.0/clients.conf
including files in directory /etc/freeradius/3.0/mods-enabled/
including configuration file /etc/freeradius/3.0/mods-enabled/always
including configuration file /etc/freeradius/3.0/mods-enabled/mschap
including configuration file /etc/freeradius/3.0/mods-enabled/radutmp
including configuration file /etc/freeradius/3.0/mods-enabled/exec
including configuration file /etc/freeradius/3.0/mods-enabled/sradutmp
including configuration file /etc/freeradius/3.0/mods-enabled/ntlm_auth
including configuration file /etc/freeradius/3.0/mods-enabled/preprocess
including configuration file /etc/freeradius/3.0/mods-enabled/detail
including configuration file /etc/freeradius/3.0/mods-enabled/soh
including configuration file /etc/freeradius/3.0/mods-enabled/echo
including configuration file /etc/freeradius/3.0/mods-enabled/expr
including configuration file /etc/freeradius/3.0/mods-enabled/unpack
including configuration file /etc/freeradius/3.0/mods-enabled/logintime
```

```
including configuration file /etc/freeradius/3.0/mods-enabled/realm
including configuration file /etc/freeradius/3.0/mods-enabled/attr_filter
including configuration file /etc/freeradius/3.0/mods-enabled/digest
including configuration file /etc/freeradius/3.0/mods-enabled/expiration
including configuration file /etc/freeradius/3.0/mods-enabled/utf8
including configuration file /etc/freeradius/3.0/mods-enabled/chap
including configuration file /etc/freeradius/3.0/mods-enabled/pap
including configuration file /etc/freeradius/3.0/mods-enabled/unix
including configuration file /etc/freeradius/3.0/mods-enabled/passwd
including configuration file /etc/freeradius/3.0/mods-enabled/detail.log
including configuration file /etc/freeradius/3.0/mods-enabled/eap
including configuration file /etc/freeradius/3.0/mods-enabled/replicate
including configuration file /etc/freeradius/3.0/mods-enabled/linelog
including configuration file /etc/freeradius/3.0/mods-enabled/dynamic_clients
including configuration file /etc/freeradius/3.0/mods-enabled/files
including files in directory /etc/freeradius/3.0/policy.d/
including configuration file /etc/freeradius/3.0/policy.d/canonicalization
including configuration file /etc/freeradius/3.0/policy.d/operator-name
including configuration file /etc/freeradius/3.0/policy.d/cui
including configuration file /etc/freeradius/3.0/policy.d/moonshot-targeted-ids
including configuration file /etc/freeradius/3.0/policy.d/accounting
including configuration file /etc/freeradius/3.0/policy.d/abfab-tr
including configuration file /etc/freeradius/3.0/policy.d/control
including configuration file /etc/freeradius/3.0/policy.d/filter
including configuration file /etc/freeradius/3.0/policy.d/debug
including configuration file /etc/freeradius/3.0/policy.d/eap
including configuration file /etc/freeradius/3.0/policy.d/rfc7542
including configuration file /etc/freeradius/3.0/policy.d/dhcp
including files in directory /etc/freeradius/3.0/sites-enabled/
including configuration file /etc/freeradius/3.0/sites-enabled/inner-tunnel
including configuration file /etc/freeradius/3.0/sites-enabled/default
main {
 security {
    user = "freerad"
    group = "freerad"
    allow_core_dumps = no
 }
    name = "freeradius"
    prefix = "/usr"
    localstatedir = "/var"
    logdir = "/var/log/freeradius"
    run_dir = "/var/run/freeradius"
}
main {
    name = "freeradius"
    prefix = "/usr"
```

```
    localstatedir = "/var"
    sbindir = "/usr/sbin"
    logdir = "/var/log/freeradius"
    run_dir = "/var/run/freeradius"
    libdir = "/usr/lib/freeradius"
    radacctdir = "/var/log/freeradius/radacct"
    hostname_lookups = no
    max_request_time = 30
    proxy_dedup_window = 1
    cleanup_delay = 5
    max_requests = 16384
    max_fds = 512
    postauth_client_lost = no
    pidfile = "/var/run/freeradius/freeradius.pid"
    checkrad = "/usr/sbin/checkrad"
    debug_level = 0
    proxy_requests = yes
 log {
    stripped_names = no
    auth = no
    auth_badpass = no
    auth_goodpass = no
    colourise = yes
    msg_denied = "You are already logged in - access denied"
 }
 resources {
 }
 security {
    max_attributes = 200
    reject_delay = 1.000000
    status_server = yes
    require_message_authenticator = "auto"
    limit_proxy_state = "auto"
 }
}
radiusd: #### Loading Realms and Home Servers ####
 proxy server {
    retry_delay = 5
    retry_count = 3
    default_fallback = no
    dead_time = 120
    wake_all_if_all_dead = no
 }
 home_server localhost {
    nonblock = no
    ipaddr = 127.0.0.1
```

```
    port = 1812
    type = "auth"
    secret = <<< secret >>>
    response_window = 20.000000
    response_timeouts = 1
    max_outstanding = 65536
    zombie_period = 40
    status_check = "status-server"
    ping_interval = 30
    check_interval = 30
    check_timeout = 4
    num_answers_to_alive = 3
    revive_interval = 120
  limit {
    max_connections = 16
    max_requests = 0
    lifetime = 0
    idle_timeout = 0
  }
  coa {
    irt = 2
    mrt = 16
    mrc = 5
    mrd = 30
  }
 }
 home_server_pool my_auth_failover {
    type = fail-over
    home_server = localhost
 }
 realm example.com {
    auth_pool = my_auth_failover
 }
 realm LOCAL {
 }
radiusd: #### Loading Clients ####
 client localhost {
    ipaddr = 127.0.0.1
    secret = <<< secret >>>
    nas_type = "other"
    proto = "*"
  limit {
    max_connections = 16
    lifetime = 0
    idle_timeout = 30
  }
```

```
 }
 client localhost_ipv6 {
    ipv6addr = ::1
    secret = <<< secret >>>
  limit {
    max_connections = 16
    lifetime = 0
    idle_timeout = 30
  }
 }
 client access-node {
    ipaddr = 10.0.23.1
    secret = <<< secret >>>
    shortname = "access-node"
  limit {
    max_connections = 16
    lifetime = 0
    idle_timeout = 30
  }
 }
Debugger not attached
systemd watchdog is disabled
 # Creating Auth-Type = mschap
 # Creating Auth-Type = eap
 # Creating Auth-Type = PAP
 # Creating Auth-Type = CHAP
 # Creating Auth-Type = MS-CHAP
 # Creating Auth-Type = digest
 # Creating Autz-Type = New-TLS-Connection
radiusd: #### Instantiating modules ####
 modules {
  # Loaded module rlm_always
  # Loading module "reject" from file /etc/freeradius/3.0/mods-enabled/always
  always reject {
    rcode = "reject"
    simulcount = 0
    mpp = no
  }
  # Loading module "fail" from file /etc/freeradius/3.0/mods-enabled/always
  always fail {
    rcode = "fail"
    simulcount = 0
    mpp = no
  }
  # Loading module "ok" from file /etc/freeradius/3.0/mods-enabled/always
  always ok {
```

```
  rcode = "ok"
  simulcount = 0
  mpp = no
}
# Loading module "handled" from file /etc/freeradius/3.0/mods-enabled/always
always handled {
  rcode = "handled"
  simulcount = 0
  mpp = no
}
# Loading module "invalid" from file /etc/freeradius/3.0/mods-enabled/always
always invalid {
  rcode = "invalid"
  simulcount = 0
  mpp = no
}
# Loading module "userlock" from file /etc/freeradius/3.0/mods-enabled/always
always userlock {
  rcode = "userlock"
  simulcount = 0
  mpp = no
}
# Loading module "notfound" from file /etc/freeradius/3.0/mods-enabled/always
always notfound {
  rcode = "notfound"
  simulcount = 0
  mpp = no
}
# Loading module "noop" from file /etc/freeradius/3.0/mods-enabled/always
always noop {
  rcode = "noop"
  simulcount = 0
  mpp = no
}
# Loading module "updated" from file /etc/freeradius/3.0/mods-enabled/always
always updated {
  rcode = "updated"
  simulcount = 0
  mpp = no
}
# Loaded module rlm_mschap
# Loading module "mschap" from file /etc/freeradius/3.0/mods-enabled/mschap
mschap {
  use_mppe = yes
  require_encryption = no
  require_strong = no
```

```
  with_ntdomain_hack = yes
 passchange {
 }
  allow_retry = yes
  winbind_retry_with_normalised_username = no
}
# Loaded module rlm_radutmp
# Loading module "radutmp" from file /etc/freeradius/3.0/mods-enabled/radutmp
radutmp {
  filename = "/var/log/freeradius/radutmp"
  username = "%{User-Name}"
  case_sensitive = yes
  check_with_nas = yes
  permissions = 384
  caller_id = yes
}
# Loaded module rlm_exec
# Loading module "exec" from file /etc/freeradius/3.0/mods-enabled/exec
exec {
  wait = no
  input_pairs = "request"
  shell_escape = yes
  timeout = 10
}
# Loading module "sradutmp" from file /etc/freeradius/3.0/mods-enabled/sradutmp
radutmp sradutmp {
  filename = "/var/log/freeradius/sradutmp"
  username = "%{User-Name}"
  case_sensitive = yes
  check_with_nas = yes
  permissions = 420
  caller_id = no
}
# Loading module "ntlm_auth" from file /etc/freeradius/3.0/mods-enabled/ntlm_auth
exec ntlm_auth {
  wait = yes
  program = "/path/to/ntlm_auth --request-nt-key --domain=MYDOMAIN --username=%{mschap:Us
      er-Name} --password=%{User-Password}"
  shell_escape = yes
}
# Loaded module rlm_preprocess
# Loading module "preprocess" from file /etc/freeradius/3.0/mods-enabled/preprocess
preprocess {
  huntgroups = "/etc/freeradius/3.0/mods-config/preprocess/huntgroups"
  hints = "/etc/freeradius/3.0/mods-config/preprocess/hints"
  with_ascend_hack = no
```

```
  ascend_channels_per_line = 23
  with_ntdomain_hack = no
  with_specialix_jetstream_hack = no
  with_cisco_vsa_hack = no
  with_alvarion_vsa_hack = no
}
# Loaded module rlm_detail
# Loading module "detail" from file /etc/freeradius/3.0/mods-enabled/detail
detail {
  filename = "/var/log/freeradius/radacct/%{%{Packet-Src-IP-Address}:-%{Packet-Src-IPv6-A
      ddress}}/detail-%Y%m%d"
  header = "%t"
  permissions = 384
  locking = no
  dates_as_integer = no
  escape_filenames = no
  log_packet_header = no
}
# Loaded module rlm_soh
# Loading module "soh" from file /etc/freeradius/3.0/mods-enabled/soh
soh {
  dhcp = yes
}
# Loading module "echo" from file /etc/freeradius/3.0/mods-enabled/echo
exec echo {
  wait = yes
  program = "/bin/echo %{User-Name}"
  input_pairs = "request"
  output_pairs = "reply"
  shell_escape = yes
}
# Loaded module rlm_expr
# Loading module "expr" from file /etc/freeradius/3.0/mods-enabled/expr
expr {
  safe_characters = "@abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.-_:
      /äéöüàâæçèéêëîïôœùûüaÿÄÉÖÜßÀÂÆÇÈÉÊËÎÏÔŒÙÛÜŸ"
}
# Loaded module rlm_unpack
# Loading module "unpack" from file /etc/freeradius/3.0/mods-enabled/unpack
# Loaded module rlm_logintime
# Loading module "logintime" from file /etc/freeradius/3.0/mods-enabled/logintime
logintime {
  minimum_timeout = 60
}
# Loaded module rlm_realm
# Loading module "IPASS" from file /etc/freeradius/3.0/mods-enabled/realm
```

```
realm IPASS {
  format = "prefix"
  delimiter = "/"
  ignore_default = no
  ignore_null = no
}
# Loading module "suffix" from file /etc/freeradius/3.0/mods-enabled/realm
realm suffix {
  format = "suffix"
  delimiter = "@"
  ignore_default = no
  ignore_null = no
}
# Loading module "bangpath" from file /etc/freeradius/3.0/mods-enabled/realm
realm bangpath {
  format = "prefix"
  delimiter = "!"
  ignore_default = no
  ignore_null = no
}
# Loading module "realmpercent" from file /etc/freeradius/3.0/mods-enabled/realm
realm realmpercent {
  format = "suffix"
  delimiter = "%"
  ignore_default = no
  ignore_null = no
}
# Loading module "ntdomain" from file /etc/freeradius/3.0/mods-enabled/realm
realm ntdomain {
  format = "prefix"
  delimiter = "\"
  ignore_default = no
  ignore_null = no
}
# Loaded module rlm_attr_filter
# Loading module "attr_filter.post-proxy" from file /etc/freeradius/3.0/mods-enabled/
        attr_filter
attr_filter attr_filter.post-proxy {
  filename = "/etc/freeradius/3.0/mods-config/attr_filter/post-proxy"
  key = "%{Realm}"
  relaxed = no
}
# Loading module "attr_filter.pre-proxy" from file /etc/freeradius/3.0/mods-enabled/
        attr_filter
attr_filter attr_filter.pre-proxy {
  filename = "/etc/freeradius/3.0/mods-config/attr_filter/pre-proxy"
```

```
  key = "%{Realm}"
  relaxed = no
}
# Loading module "attr_filter.access_reject" from file /etc/freeradius/3.0/
        mods-enabled/attr_filter
attr_filter attr_filter.access_reject {
  filename = "/etc/freeradius/3.0/mods-config/attr_filter/access_reject"
  key = "%{User-Name}"
  relaxed = no
}
# Loading module "attr_filter.access_challenge" from file /etc/freeradius/3.0/
        mods-enabled/attr_filter
attr_filter attr_filter.access_challenge {
  filename = "/etc/freeradius/3.0/mods-config/attr_filter/access_challenge"
  key = "%{User-Name}"
  relaxed = no
}
# Loading module "attr_filter.accounting_response" from file /etc/freeradius/3.0/
        mods-enabled/attr_filter
attr_filter attr_filter.accounting_response {
  filename = "/etc/freeradius/3.0/mods-config/attr_filter/accounting_response"
  key = "%{User-Name}"
  relaxed = no
}
# Loading module "attr_filter.coa" from file /etc/freeradius/3.0/
        mods-enabled/attr_filter
attr_filter attr_filter.coa {
  filename = "/etc/freeradius/3.0/mods-config/attr_filter/coa"
  key = "%{User-Name}"
  relaxed = no
}
# Loaded module rlm_digest
# Loading module "digest" from file /etc/freeradius/3.0/mods-enabled/digest
# Loaded module rlm_expiration
# Loading module "expiration" from file /etc/freeradius/3.0/mods-enabled/expiration
# Loaded module rlm_utf8
# Loading module "utf8" from file /etc/freeradius/3.0/mods-enabled/utf8
# Loaded module rlm_chap
# Loading module "chap" from file /etc/freeradius/3.0/mods-enabled/chap
# Loaded module rlm_pap
# Loading module "pap" from file /etc/freeradius/3.0/mods-enabled/pap
pap {
  normalise = yes
}
# Loaded module rlm_unix
# Loading module "unix" from file /etc/freeradius/3.0/mods-enabled/unix
```

```
  unix {
    radwtmp = "/var/log/freeradius/radwtmp"
  }
Creating attribute Unix-Group
  # Loaded module rlm_passwd
  # Loading module "etc_passwd" from file /etc/freeradius/3.0/mods-enabled/passwd
  passwd etc_passwd {
    filename = "/etc/passwd"
    format = "*User-Name:Crypt-Password:"
    delimiter = ":"
    ignore_nislike = no
    ignore_empty = yes
    allow_multiple_keys = no
    hash_size = 100
  }
  # Loading module "auth_log" from file /etc/freeradius/3.0/mods-enabled/detail.log
  detail auth_log {
    filename = "/var/log/freeradius/radacct/%{%{Packet-Src-IP-Address}:-%{Packet-Src-IPv6-A
        ddress}}/auth-detail-%Y%m%d"
    header = "%t"
    permissions = 384
    locking = no
    dates_as_integer = no
    escape_filenames = no
    log_packet_header = no
  }
  # Loading module "reply_log" from file /etc/freeradius/3.0/mods-enabled/detail.log
  detail reply_log {
    filename = "/var/log/freeradius/radacct/%{%{Packet-Src-IP-Address}:-%{Packet-Src-IPv6-A
        ddress}}/reply-detail-%Y%m%d"
    header = "%t"
    permissions = 384
    locking = no
    dates_as_integer = no
    escape_filenames = no
    log_packet_header = no
  }
  # Loading module "pre_proxy_log" from file /etc/freeradius/3.0/mods-enabled/detail.log
  detail pre_proxy_log {
    filename = "/var/log/freeradius/radacct/%{%{Packet-Src-IP-Address}:-%{Packet-Src-IPv6-A
        ddress}}/pre-proxy-detail-%Y%m%d"
    header = "%t"
    permissions = 384
    locking = no
    dates_as_integer = no
    escape_filenames = no
```

```
  log_packet_header = no
}
# Loading module "post_proxy_log" from file /etc/freeradius/3.0/mods-enabled/detail.log
detail post_proxy_log {
  filename = "/var/log/freeradius/radacct/%{%{Packet-Src-IP-Address}:-%{Packet-Src-IPv6-A
      ddress}}/post-proxy-detail-%Y%m%d"
  header = "%t"
  permissions = 384
  locking = no
  dates_as_integer = no
  escape_filenames = no
  log_packet_header = no
}
# Loaded module rlm_eap
# Loading module "eap" from file /etc/freeradius/3.0/mods-enabled/eap
eap {
  default_eap_type = "md5"
  timer_expire = 60
  max_eap_type = 52
  ignore_unknown_eap_types = no
  cisco_accounting_username_bug = no
  max_sessions = 16384
  dedup_key = ""
}
# Loaded module rlm_replicate
# Loading module "replicate" from file /etc/freeradius/3.0/mods-enabled/replicate
# Loaded module rlm_linelog
# Loading module "linelog" from file /etc/freeradius/3.0/mods-enabled/linelog
linelog {
  filename = "/var/log/freeradius/linelog"
  escape_filenames = no
  syslog_severity = "info"
  permissions = 384
  format = "This is a log message for %{User-Name}"
  reference = "messages.%{%{reply:Packet-Type}:-default}"
}
# Loading module "log_accounting" from file /etc/freeradius/3.0/mods-enabled/linelog
linelog log_accounting {
  filename = "/var/log/freeradius/linelog-accounting"
  escape_filenames = no
  syslog_severity = "info"
  permissions = 384
  format = ""
  reference = "Accounting-Request.%{%{Acct-Status-Type}:-unknown}"
}
# Loaded module rlm_dynamic_clients
```

```
  # Loading module "dynamic_clients" from file /etc/freeradius/3.0/mods-enabled/dynamic_cl
        ients
  # Loaded module rlm_files
  # Loading module "files" from file /etc/freeradius/3.0/mods-enabled/files
  files {
    filename = "/etc/freeradius/3.0/mods-config/files/authorize"
    acctusersfile = "/etc/freeradius/3.0/mods-config/files/accounting"
    preproxy_usersfile = "/etc/freeradius/3.0/mods-config/files/pre-proxy"
  }
  instantiate {
  }
  # Instantiating module "reject" from file /etc/freeradius/3.0/mods-enabled/always
  # Instantiating module "fail" from file /etc/freeradius/3.0/mods-enabled/always
  # Instantiating module "ok" from file /etc/freeradius/3.0/mods-enabled/always
  # Instantiating module "handled" from file /etc/freeradius/3.0/mods-enabled/always
  # Instantiating module "invalid" from file /etc/freeradius/3.0/mods-enabled/always
  # Instantiating module "userlock" from file /etc/freeradius/3.0/mods-enabled/always
  # Instantiating module "notfound" from file /etc/freeradius/3.0/mods-enabled/always
  # Instantiating module "noop" from file /etc/freeradius/3.0/mods-enabled/always
  # Instantiating module "updated" from file /etc/freeradius/3.0/mods-enabled/always
  # Instantiating module "mschap" from file /etc/freeradius/3.0/mods-enabled/mschap
rlm_mschap (mschap): using internal authentication
  # Instantiating module "preprocess" from file /etc/freeradius/3.0/mods-enabled/
        preprocess
reading pairlist file /etc/freeradius/3.0/mods-config/preprocess/huntgroups
reading pairlist file /etc/freeradius/3.0/mods-config/preprocess/hints
  # Instantiating module "detail" from file /etc/freeradius/3.0/mods-enabled/detail
  # Instantiating module "logintime" from file /etc/freeradius/3.0/mods-enabled/logintime
  # Instantiating module "IPASS" from file /etc/freeradius/3.0/mods-enabled/realm
  # Instantiating module "suffix" from file /etc/freeradius/3.0/mods-enabled/realm
  # Instantiating module "bangpath" from file /etc/freeradius/3.0/mods-enabled/realm
  # Instantiating module "realmpercent" from file /etc/freeradius/3.0/mods-enabled/realm
  # Instantiating module "ntdomain" from file /etc/freeradius/3.0/mods-enabled/realm
  # Instantiating module "attr_filter.post-proxy" from file /etc/freeradius/3.0/mods-enabl
        ed/attr_filter
reading pairlist file /etc/freeradius/3.0/mods-config/attr_filter/post-proxy
  # Instantiating module "attr_filter.pre-proxy" from file /etc/freeradius/3.0/mods-enable
        d/attr_filter
reading pairlist file /etc/freeradius/3.0/mods-config/attr_filter/pre-proxy
  # Instantiating module "attr_filter.access_reject" from file /etc/freeradius/3.0/mods-en
        abled/attr_filter
reading pairlist file /etc/freeradius/3.0/mods-config/attr_filter/access_reject
  # Instantiating module "attr_filter.access_challenge" from file /etc/freeradius/3.0/mods
        -enabled/attr_filter
reading pairlist file /etc/freeradius/3.0/mods-config/attr_filter/access_challenge
  # Instantiating module "attr_filter.accounting_response" from file /etc/freeradius/3.0/m
```

```
        ods-enabled/attr_filter
reading pairlist file /etc/freeradius/3.0/mods-config/attr_filter/accounting_response
  # Instantiating module "attr_filter.coa" from file /etc/freeradius/3.0/mods-enabled/attr
        _filter
reading pairlist file /etc/freeradius/3.0/mods-config/attr_filter/coa
  # Instantiating module "expiration" from file /etc/freeradius/3.0/mods-enabled/expiratio
        n
  # Instantiating module "pap" from file /etc/freeradius/3.0/mods-enabled/pap
  # Instantiating module "etc_passwd" from file /etc/freeradius/3.0/mods-enabled/passwd
rlm_passwd: nfields: 3 keyfield 0(User-Name) listable: no
  # Instantiating module "auth_log" from file /etc/freeradius/3.0/mods-enabled/detail.log
rlm_detail (auth_log): 'User-Password' suppressed, will not appear in detail output
  # Instantiating module "reply_log" from file /etc/freeradius/3.0/mods-enabled/detail.log
  # Instantiating module "pre_proxy_log" from file /etc/freeradius/3.0/mods-enabled/detail
        .log
  # Instantiating module "post_proxy_log" from file /etc/freeradius/3.0/mods-enabled/detai
        l.log
  # Instantiating module "eap" from file /etc/freeradius/3.0/mods-enabled/eap
   # Linked to sub-module rlm_eap_md5
   # Linked to sub-module rlm_eap_gtc
   gtc {
    challenge = "Password: "
    auth_type = "PAP"
   }
   # Linked to sub-module rlm_eap_tls
   tls {
    tls = "tls-common"
   }
   tls-config tls-common {
    verify_depth = 0
    ca_path = "/etc/freeradius/3.0/certs"
    pem_file_type = yes
    private_key_file = "/etc/ssl/private/ssl-cert-snakeoil.key"
    certificate_file = "/etc/ssl/certs/ssl-cert-snakeoil.pem"
    ca_file = "/etc/ssl/certs/ca-certificates.crt"
    private_key_password = <<< secret >>>
    fragment_size = 1024
    include_length = yes
    auto_chain = yes
    check_crl = no
    check_all_crl = no
    ca_path_reload_interval = 0
    cipher_list = "DEFAULT"
    cipher_server_preference = no
    reject_unknown_intermediate_ca = no
    ecdh_curve = ""
```

```
    tls_max_version = "1.2"
    tls_min_version = "1.2"
    cache {
        enable = no
        lifetime = 24
        max_entries = 255
    }
    verify {
        skip_if_ocsp_ok = no
    }
    ocsp {
        enable = no
        override_cert_url = yes
        url = "http://127.0.0.1/ocsp/"
        use_nonce = yes
        timeout = 0
        softfail = no
    }
   }
   # Linked to sub-module rlm_eap_ttls
   ttls {
    tls = "tls-common"
    default_eap_type = "md5"
    copy_request_to_tunnel = no
    use_tunneled_reply = no
    virtual_server = "inner-tunnel"
    include_length = yes
    require_client_cert = no
   }
tls: Using cached TLS configuration from previous invocation
   # Linked to sub-module rlm_eap_peap
   peap {
    tls = "tls-common"
    default_eap_type = "mschapv2"
    copy_request_to_tunnel = no
    use_tunneled_reply = no
    proxy_tunneled_request_as_eap = yes
    virtual_server = "inner-tunnel"
    soh = no
    require_client_cert = no
   }
tls: Using cached TLS configuration from previous invocation
   # Linked to sub-module rlm_eap_mschapv2
   mschapv2 {
    with_ntdomain_hack = no
    send_error = no
```

```
   }
  # Instantiating module "linelog" from file /etc/freeradius/3.0/mods-enabled/linelog
  # Instantiating module "log_accounting" from file /etc/freeradius/3.0/mods-enabled/linel
       og
  # Instantiating module "files" from file /etc/freeradius/3.0/mods-enabled/files
reading pairlist file /etc/freeradius/3.0/mods-config/files/authorize
reading pairlist file /etc/freeradius/3.0/mods-config/files/accounting
reading pairlist file /etc/freeradius/3.0/mods-config/files/pre-proxy
 } # modules
radiusd: #### Loading Virtual Servers ####
server { # from file /etc/freeradius/3.0/radiusd.conf
} # server
server inner-tunnel { # from file /etc/freeradius/3.0/sites-enabled/inner-tunnel
 # Loading authenticate {...}
Compiling Auth-Type PAP for attr Auth-Type
Compiling Auth-Type CHAP for attr Auth-Type
Compiling Auth-Type MS-CHAP for attr Auth-Type
 # Loading authorize {...}
Ignoring "sql" (see raddb/mods-available/README.rst)
Ignoring "ldap" (see raddb/mods-available/README.rst)
 # Loading session {...}
 # Loading post-proxy {...}
 # Loading post-auth {...}
 # Skipping contents of 'if' as it is always 'false' -- /etc/freeradius/3.0/sites-enabled/
       inner-tunnel:366
Compiling Post-Auth-Type REJECT for attr Post-Auth-Type
} # server inner-tunnel
server default { # from file /etc/freeradius/3.0/sites-enabled/default
 # Loading authenticate {...}
Compiling Auth-Type PAP for attr Auth-Type
Compiling Auth-Type CHAP for attr Auth-Type
Compiling Auth-Type MS-CHAP for attr Auth-Type
 # Loading authorize {...}
Compiling Autz-Type New-TLS-Connection for attr Autz-Type
 # Loading preacct {...}
 # Loading accounting {...}
 # Loading post-proxy {...}
 # Loading post-auth {...}
Compiling Post-Auth-Type REJECT for attr Post-Auth-Type
Compiling Post-Auth-Type Challenge for attr Post-Auth-Type
Compiling Post-Auth-Type Client-Lost for attr Post-Auth-Type
} # server default
radiusd: #### Opening IP addresses and Ports ####
listen {
    type = "auth"
    ipaddr = 127.0.0.1
```

```
    port = 18120
}
listen {
    type = "auth"
    ipaddr = *
    port = 0
   limit {
    max_connections = 16
    lifetime = 0
    idle_timeout = 30
   }
}
listen {
    type = "acct"
    ipaddr = *
    port = 0
   limit {
    max_connections = 16
    lifetime = 0
    idle_timeout = 30
   }
}
listen {
    type = "auth"
    ipv6addr = ::
    port = 0
   limit {
    max_connections = 16
    lifetime = 0
    idle_timeout = 30
   }
}
listen {
    type = "acct"
    ipv6addr = ::
    port = 0
   limit {
    max_connections = 16
    lifetime = 0
    idle_timeout = 30
   }
}
Listening on auth address 127.0.0.1 port 18120 bound to server inner-tunnel
Listening on auth address * port 1812 bound to server default
Listening on acct address * port 1813 bound to server default
Listening on auth address :: port 1812 bound to server default
```

```
Listening on acct address :: port 1813 bound to server default
Listening on proxy address * port 48578
Listening on proxy address :: port 42007
Ready to process requests
(0) Received Access-Request Id 0 from 10.0.23.1:55171 to 10.0.30.10:1812 length 165
(0)   User-Name = "nacuser"
(0)   NAS-IP-Address = 10.0.23.1
(0)   Called-Station-Id = "<REDACTED>:"
(0)   NAS-Port-Type = Wireless-802.11
(0)   Service-Type = Framed-User
(0)   Calling-Station-Id = "<REDACTED>"
(0)   Connect-Info = "CONNECT 0Mbps 802.11b"
(0)   Acct-Session-Id = "1A082DB0BEE2BD34"
(0)   Framed-MTU = 1400
(0)   EAP-Message = 0x0219000c016e616375736572
(0)   Message-Authenticator = 0x71423639f013148f1ce7a71576679bbe
(0) # Executing section authorize from file /etc/freeradius/3.0/sites-enabled/default
(0)   authorize {
(0)     policy filter_username {
(0)       if (&User-Name) {
(0)       if (&User-Name)  -> TRUE
(0)       if (&User-Name)  {
(0)         if (&User-Name =~ / /) {
(0)         if (&User-Name =~ / /)  -> FALSE
(0)         if (&User-Name =~ /@[^@]*@/ ) {
(0)         if (&User-Name =~ /@[^@]*@/ )  -> FALSE
(0)         if (&User-Name =~ /\.\./ ) {
(0)         if (&User-Name =~ /\.\./ )  -> FALSE
(0)         if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/))  {
(0)         if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/))   -> FALSE
(0)         if (&User-Name =~ /\.$/)  {
(0)         if (&User-Name =~ /\.$/)   -> FALSE
(0)         if (&User-Name =~ /@\./)  {
(0)         if (&User-Name =~ /@\./)   -> FALSE
(0)       } # if (&User-Name)  = notfound
(0)     } # policy filter_username = notfound
(0)     [preprocess] = ok
(0)     [chap] = noop
(0)     [mschap] = noop
(0)     [digest] = noop
(0) suffix: Checking for suffix after "@"
(0) suffix: No '@' in User-Name = "nacuser", looking up realm NULL
(0) suffix: No such realm "NULL"
(0)     [suffix] = noop
(0) eap: Peer sent EAP Response (code 2) ID 25 length 12
(0) eap: EAP-Identity reply, returning 'ok' so we can short-circuit the rest of authorize
```

```
(0)     [eap] = ok
(0)   } # authorize = ok
(0) Found Auth-Type = eap
(0) # Executing group from file /etc/freeradius/3.0/sites-enabled/default
(0)   authenticate {
(0) eap: Peer sent packet with method EAP Identity (1)
(0) eap: Calling submodule eap_md5 to process data
(0) eap_md5: Issuing MD5 Challenge
(0) eap: Sending EAP Request (code 1) ID 26 length 22
(0) eap: EAP session adding &reply:State = 0xbb99b235bb83b6a0
(0)     [eap] = handled
(0)   } # authenticate = handled
(0) Using Post-Auth-Type Challenge
(0) # Executing group from file /etc/freeradius/3.0/sites-enabled/default
(0)   Challenge { ... } # empty sub-section is ignored
(0) Sent Access-Challenge Id 0 from 10.0.30.10:1812 to 10.0.23.1:55171 length 80
(0)   EAP-Message = 0x011a00160410d8d1ca688796282203d8a54492bb84f9
(0)   Message-Authenticator = 0x00000000000000000000000000000000
(0)   State = 0xbb99b235bb83b6a0b74fb036b1c91a02
(0) Finished request
Waking up in 4.9 seconds.
(1) Received Access-Request Id 1 from 10.0.23.1:55171 to 10.0.30.10:1812 length 193
(1)   User-Name = "nacuser"
(1)   NAS-IP-Address = 10.0.23.1
(1)   Called-Station-Id = "<REDACTED>:"
(1)   NAS-Port-Type = Wireless-802.11
(1)   Service-Type = Framed-User
(1)   Calling-Station-Id = "<REDACTED>"
(1)   Connect-Info = "CONNECT 0Mbps 802.11b"
(1)   Acct-Session-Id = "1A082DB0BEE2BD34"
(1)   Framed-MTU = 1400
(1)   EAP-Message = 0x021a0016041041b2c01ccfa94d12fc491a04d95cd935
(1)   State = 0xbb99b235bb83b6a0b74fb036b1c91a02
(1)   Message-Authenticator = 0xee00aa03a7391d212502d3e4595c510c
(1) session-state: No cached attributes
(1) # Executing section authorize from file /etc/freeradius/3.0/sites-enabled/default
(1)   authorize {
(1)     policy filter_username {
(1)       if (&User-Name) {
(1)       if (&User-Name)  -> TRUE
(1)       if (&User-Name)  {
(1)         if (&User-Name =~ / /) {
(1)         if (&User-Name =~ / /)  -> FALSE
(1)         if (&User-Name =~ /@[^@]*@/ ) {
(1)         if (&User-Name =~ /@[^@]*@/ )  -> FALSE
(1)         if (&User-Name =~ /\.\./ ) {
```

```
(1)         if (&User-Name =~ /\.\./ )  -> FALSE
(1)         if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/))  {
(1)         if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/))   -> FALSE
(1)         if (&User-Name =~ /\.$/)  {
(1)         if (&User-Name =~ /\.$/)   -> FALSE
(1)         if (&User-Name =~ /@\./)  {
(1)         if (&User-Name =~ /@\./)   -> FALSE
(1)       } # if (&User-Name)  = notfound
(1)     } # policy filter_username = notfound
(1)     [preprocess] = ok
(1)     [chap] = noop
(1)     [mschap] = noop
(1)     [digest] = noop
(1) suffix: Checking for suffix after "@"
(1) suffix: No '@' in User-Name = "nacuser", looking up realm NULL
(1) suffix: No such realm "NULL"
(1)     [suffix] = noop
(1) eap: Peer sent EAP Response (code 2) ID 26 length 22
(1) eap: No EAP Start, assuming it's an on-going EAP conversation
(1)     [eap] = updated
(1) files: users: Matched entry nacuser at line 207
(1)     [files] = ok
(1)     [expiration] = noop
(1)     [logintime] = noop
(1) pap: WARNING: Auth-Type already set.  Not setting to PAP
(1)     [pap] = noop
(1)   } # authorize = updated
(1) Found Auth-Type = eap
(1) # Executing group from file /etc/freeradius/3.0/sites-enabled/default
(1)   authenticate {
(1) eap: Removing EAP session with state 0xbb99b235bb83b6a0
(1) eap: Previous EAP request found for state 0xbb99b235bb83b6a0, released from the list
(1) eap: Peer sent packet with method EAP MD5 (4)
(1) eap: Calling submodule eap_md5 to process data
EAP-MD5 digests do not match.
(1) eap: Sending EAP Failure (code 4) ID 26 length 4
(1) eap: Freeing handler
(1)     [eap] = reject
(1)   } # authenticate = reject
(1) Failed to authenticate the user
(1) Using Post-Auth-Type Reject
(1) # Executing group from file /etc/freeradius/3.0/sites-enabled/default
(1)   Post-Auth-Type REJECT {
(1) attr_filter.access_reject: EXPAND %{User-Name}
(1) attr_filter.access_reject:    --> nacuser
(1) attr_filter.access_reject: Matched entry DEFAULT at line 11
```

```
(1)     [attr_filter.access_reject] = updated
(1)     [eap] = noop
(1)     policy remove_reply_message_if_eap {
(1)       if (&reply:EAP-Message && &reply:Reply-Message) {
(1)       if (&reply:EAP-Message && &reply:Reply-Message)  -> FALSE
(1)       else {
(1)         [noop] = noop
(1)       } # else = noop
(1)     } # policy remove_reply_message_if_eap = noop
(1)   } # Post-Auth-Type REJECT = updated
(1) Delaying response for 1.000000 seconds
Waking up in 0.3 seconds.
Waking up in 0.1 seconds.
Signalled to terminate
Exiting normally
```

B.1.4 wpa_supplicant Success Re-Test Log

```
radius-splunk: ~/incident-evidence/wpa_supplicant_success_retest_v2.txt
wpa_supplicant v2.10
random: getrandom() support available
Successfully initialized wpa_supplicant
Initializing interface 'ens33' conf '/etc/wpa_supplicant/lab/wired-8021x.conf' driver 'wir
        ed' ctrl_interface 'N/A' bridge 'N/A'
Configuration file '/etc/wpa_supplicant/lab/wired-8021x.conf' -> '/etc/wpa_supplicant/lab/
        wired-8021x.conf'
Reading configuration file '/etc/wpa_supplicant/lab/wired-8021x.conf'
ctrl_interface='/run/wpa_supplicant'
ap_scan=0
eapol_version=2
Line: 5 - start of a new network block
key_mgmt: 0x8
eap methods - hexdump(len=16): 00 00 00 00 04 00 00 00 00 00 00 00 00 00 00 00
identity - hexdump_ascii(len=7):
     6e 61 63 75 73 65 72                             nacuser
password - hexdump_ascii(len=11): [REMOVED]
Priority group 0
   id=0 ssid=''
driver_wired_init_common: Added multicast membership with packet socket
Add interface ens33 to a new radio N/A
ens33: Own MAC address: <REDACTED>
ens33: RSN: flushing PMKID list in the driver
ens33: Setting scan request: 0.100000 sec
TDLS: TDLS operation not supported by driver
TDLS: Driver uses internal link setup
TDLS: Driver does not support TDLS channel switching
```

```
ens33: WPS: UUID based on MAC address: f6fbf57e-3bcd-5a3f-9f55-30679786050a
ENGINE: Loading builtin engines
ENGINE: Loading builtin engines
EAPOL: SUPP_PAE entering state DISCONNECTED
EAPOL: Supplicant port status: Unauthorized
EAPOL: KEY_RX entering state NO_KEY_RECEIVE
EAPOL: SUPP_BE entering state INITIALIZE
EAP: EAP entering state DISABLED
MBO: Update non-preferred channels, non_pref_chan=N/A
ens33: Added interface ens33
ens33: State: DISCONNECTED -> DISCONNECTED
EAPOL: External notification - EAP success=0
EAPOL: External notification - EAP fail=0
EAPOL: External notification - portControl=Auto
ens33: Already associated with a configured network - generating associated event
ens33: Event ASSOC (0) received
ens33: Association info event
ens33: State: DISCONNECTED -> ASSOCIATED
ens33: Associated to a new BSS: BSSID=<REDACTED>
ens33: Select network based on association information
ens33: Network configuration found for the current AP
ens33: WPA: clearing AP WPA IE
ens33: WPA: clearing AP RSN IE
ens33: WPA: clearing AP RSNXE
ens33: WPA: clearing own WPA/RSN IE
ens33: RSN: clearing own RSNXE
ens33: Failed to get scan results
EAPOL: External notification - EAP success=0
EAPOL: External notification - EAP fail=0
EAPOL: External notification - portControl=Auto
ens33: Associated with <REDACTED>
ens33: WPA: Association event - clear replay counter
ens33: WPA: Clear old PTK
TDLS: Remove peers on association
EAPOL: External notification - portEnabled=0
EAPOL: External notification - portValid=0
EAPOL: External notification - portEnabled=1
EAPOL: SUPP_PAE entering state CONNECTING
EAPOL: SUPP_BE entering state IDLE
EAP: EAP entering state INITIALIZE
EAP: EAP entering state IDLE
ens33: Cancelling scan request
ens33: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
EAPOL: startWhen --> 0
EAPOL: SUPP_PAE entering state CONNECTING
EAPOL: txStart
```

```
TX EAPOL: dst=<REDACTED>
TX EAPOL - hexdump(len=4): 02 01 00 00
l2_packet_receive: src=<REDACTED> len=46
ens33: RX EAPOL from <REDACTED>
RX EAPOL - hexdump(len=46): 02 00 00 05 01 fe 00 05 01 00 00 00 00 00 00 00 00 00 00 00
         00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
EAPOL: Received EAP-Packet frame
EAPOL: SUPP_PAE entering state RESTART
EAP: EAP entering state INITIALIZE
EAP: EAP entering state IDLE
EAPOL: SUPP_PAE entering state AUTHENTICATING
EAPOL: SUPP_BE entering state REQUEST
EAPOL: getSuppRsp
EAP: EAP entering state RECEIVED
EAP: Received EAP-Request id=254 method=1 vendor=0 vendorMethod=0
EAP: EAP entering state IDENTITY
ens33: CTRL-EVENT-EAP-STARTED EAP authentication started
EAP: Status notification: started (param=)
EAP: EAP-Request Identity data - hexdump_ascii(len=0):
EAP: using real identity - hexdump_ascii(len=7):
     6e 61 63 75 73 65 72                              nacuser
EAP: EAP entering state SEND_RESPONSE
EAP: EAP entering state IDLE
EAPOL: SUPP_BE entering state RESPONSE
EAPOL: txSuppRsp
TX EAPOL: dst=<REDACTED>
TX EAPOL - hexdump(len=16): 02 00 00 0c 02 fe 00 0c 01 6e 61 63 75 73 65 72
EAPOL: SUPP_BE entering state RECEIVE
l2_packet_receive: src=<REDACTED> len=46
ens33: RX EAPOL from <REDACTED>
RX EAPOL - hexdump(len=46): 02 00 00 16 01 ff 00 16 04 10 90 ab 89 1e 63 b1 ce 73 b1 28 33
         17 e8 9c c4 1b 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
EAPOL: Received EAP-Packet frame
EAPOL: SUPP_BE entering state REQUEST
EAPOL: getSuppRsp
EAP: EAP entering state RECEIVED
EAP: Received EAP-Request id=255 method=4 vendor=0 vendorMethod=0
EAP: EAP entering state GET_METHOD
ens33: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=4
EAP: Status notification: accept proposed method (param=MD5)
EAP: Initialize selected EAP method: vendor 0 method 4 (MD5)
ens33: CTRL-EVENT-EAP-METHOD EAP vendor 0 method 4 (MD5) selected
EAP: EAP entering state METHOD
EAP-MD5: Challenge - hexdump(len=16): 90 ab 89 1e 63 b1 ce 73 b1 28 33 17 e8 9c c4 1b
EAP-MD5: Generating Challenge Response
EAP-MD5: Response - hexdump(len=16): 61 59 ca e5 bd 45 91 62 54 d8 5a 8e 12 d8 60 af
```

```
EAP: method process -> ignore=FALSE methodState=DONE decision=COND_SUCC eapRespData=0x623c
        dc395420
EAP: EAP entering state SEND_RESPONSE
EAP: EAP entering state IDLE
EAPOL: SUPP_BE entering state RESPONSE
EAPOL: txSuppRsp
TX EAPOL: dst=<REDACTED>
TX EAPOL - hexdump(len=26): 02 00 00 16 02 ff 00 16 04 10 61 59 ca e5 bd 45 91 62 54 d8 5a
         8e 12 d8 60 af
EAPOL: SUPP_BE entering state RECEIVE
l2_packet_receive: src=<REDACTED> len=46
ens33: RX EAPOL from <REDACTED>
RX EAPOL - hexdump(len=46): 02 00 00 04 03 ff 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00
         00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
EAPOL: Received EAP-Packet frame
EAPOL: SUPP_BE entering state REQUEST
EAPOL: getSuppRsp
EAP: EAP entering state RECEIVED
EAP: Received EAP-Success
EAP: Status notification: completion (param=success)
EAP: EAP entering state SUCCESS
ens33: CTRL-EVENT-EAP-SUCCESS EAP authentication completed successfully
EAPOL: IEEE 802.1X for plaintext connection; no EAPOL-Key frames required
ens33: WPA: EAPOL processing complete
ens33: Cancelling authentication timeout
ens33: State: ASSOCIATED -> COMPLETED
ens33: CTRL-EVENT-CONNECTED - Connection to <REDACTED> completed [id=0 id_str=]
EAPOL: SUPP_PAE entering state AUTHENTICATED
EAPOL: Supplicant port status: Authorized
EAPOL: SUPP_BE entering state RECEIVE
EAPOL: SUPP_BE entering state SUCCESS
EAPOL: SUPP_BE entering state IDLE
EAPOL authentication completed - result=SUCCESS
ens33: Removing interface ens33
ens33: Request to deauthenticate - bssid=<REDACTED> pending_bssid=00:00:00:00:00:00 reason
        =3 (DEAUTH_LEAVING) state=COMPLETED
TDLS: Tear down peers
ens33: Event DEAUTH (11) received
ens33: Deauthentication notification
ens33:  * reason 3 (DEAUTH_LEAVING) locally_generated=1
Deauthentication frame IE(s) - hexdump(len=0): [NULL]
ens33: CTRL-EVENT-DISCONNECTED bssid=<REDACTED> reason=3 locally_generated=1
ens33: Auto connect disabled: do not try to re-connect
ens33: Ignore connection failure indication since interface has been put into disconnected
         state
TDLS: Remove peers on disassociation
```

```
ens33: WPA: Clear old PMK and PTK
ens33: Disconnect event - remove keys
ens33: State: COMPLETED -> DISCONNECTED
EAPOL: External notification - portEnabled=0
EAPOL: SUPP_PAE entering state DISCONNECTED
EAPOL: Supplicant port status: Unauthorized
EAPOL: SUPP_BE entering state INITIALIZE
EAP: EAP entering state DISABLED
EAPOL: External notification - portValid=0
ens33: State: DISCONNECTED -> DISCONNECTED
EAPOL: External notification - portEnabled=0
EAPOL: External notification - portValid=0
QM: Clear all active DSCP policies
ens33: CTRL-EVENT-DSCP-POLICY clear_all
EAP: deinitialize previously used EAP method (4, MD5) at EAP deinit
ens33: WPA: Clear old PMK and PTK
ens33: Cancelling scan request
ens33: Cancelling authentication timeout
Off-channel: Clear pending Action frame TX (pending_action_tx=(nil)
HS20: Delete all stored icons
Off-channel: Action frame sequence done notification: pending_action_tx=(nil) drv_offchan_
        tx=0 action_tx_wait_time=0 off_channel_freq=0 roc_waiting_drv_freq=0
QM: Clear all active DSCP policies
ens33: CTRL-EVENT-DSCP-POLICY clear_all
Remove interface ens33 from radio
Remove radio
ens33: CTRL-EVENT-TERMINATING
```

B.1.5 hostapd Success Re-Test Log

```
radius-splunk: ~/incident-evidence/hostapd_success_retest_v2.txt
Mar 23 02:49:00 access-node hostapd[2052]: Received EAPOL packet
Mar 23 02:49:00 access-node hostapd[2052]: ens38: Event NEW_STA (22) received
Mar 23 02:49:00 access-node hostapd[2052]: Data frame from unknown STA <REDACTED> - adding
         a new STA
Mar 23 02:49:00 access-node hostapd[2052]:   New STA
Mar 23 02:49:00 access-node hostapd[2052]: ap_sta_add: register ap_handle_timer timeout fo
        r <REDACTED> (300 seconds - ap_max_inactivity)
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: start authen
        tication
Mar 23 02:49:00 access-node hostapd[2052]: EAP: Server state machine created
Mar 23 02:49:00 access-node hostapd[2052]: IEEE 802.1X: <REDACTED> BE_AUTH entering state
        IDLE
Mar 23 02:49:00 access-node hostapd[2052]: IEEE 802.1X: <REDACTED> CTRL_DIR entering state
         FORCE_BOTH
Mar 23 02:49:00 access-node hostapd[2052]: ens38: hostapd_new_assoc_sta: canceled wired ap
```

```
        _handle_timer timeout for <REDACTED>
Mar 23 02:49:00 access-node hostapd[2052]: ens38: Event EAPOL_RX (23) received
Mar 23 02:49:00 access-node hostapd[2052]: IEEE 802.1X: 46 bytes from <REDACTED>
Mar 23 02:49:00 access-node hostapd[2052]:    IEEE 802.1X: version=2 type=1 length=0
Mar 23 02:49:00 access-node hostapd[2052]:    ignoring 42 extra octets after IEEE 802.1X p
        acket
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: received EAP
        OL-Start from STA
Mar 23 02:49:00 access-node hostapd[2052]: IEEE 802.1X: <REDACTED> AUTH_PAE entering state
         DISCONNECTED
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: unauthorizin
        g port
Mar 23 02:49:00 access-node hostapd[2052]: IEEE 802.1X: <REDACTED> AUTH_PAE entering state
         RESTART
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state INITIALIZE
Mar 23 02:49:00 access-node hostapd[2052]: ens38: CTRL-EVENT-EAP-STARTED <REDACTED>
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state SELECT_ACTION
Mar 23 02:49:00 access-node hostapd[2052]: EAP: getDecision: no identity known yet -> CONT
        INUE
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state PROPOSE_METHOD
Mar 23 02:49:00 access-node hostapd[2052]: EAP: getNextMethod: vendor 0 type 1
Mar 23 02:49:00 access-node hostapd[2052]: ens38: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0
        method=1
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state METHOD_REQUEST
Mar 23 02:49:00 access-node hostapd[2052]: EAP: building EAP-Request: Identifier 254
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state SEND_REQUEST
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state IDLE
Mar 23 02:49:00 access-node hostapd[2052]: EAP: retransmit timeout 3 seconds (from dynamic
         back off; retransCount=0)
Mar 23 02:49:00 access-node hostapd[2052]: IEEE 802.1X: <REDACTED> AUTH_PAE entering state
         CONNECTING
Mar 23 02:49:00 access-node hostapd[2052]: IEEE 802.1X: <REDACTED> AUTH_PAE entering state
         AUTHENTICATING
Mar 23 02:49:00 access-node hostapd[2052]: IEEE 802.1X: <REDACTED> BE_AUTH entering state
        REQUEST
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: Sending EAP
        Packet (identifier 254)
Mar 23 02:49:00 access-node hostapd[2052]: Received EAPOL packet
Mar 23 02:49:00 access-node hostapd[2052]: ens38: Event NEW_STA (22) received
Mar 23 02:49:00 access-node hostapd[2052]: ens38: Event EAPOL_RX (23) received
Mar 23 02:49:00 access-node hostapd[2052]: IEEE 802.1X: 46 bytes from <REDACTED>
Mar 23 02:49:00 access-node hostapd[2052]:    IEEE 802.1X: version=2 type=0 length=12
Mar 23 02:49:00 access-node hostapd[2052]:    ignoring 30 extra octets after IEEE 802.1X p
        acket
Mar 23 02:49:00 access-node hostapd[2052]: EAP: code=2 (response) identifier=254 length=12
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: start authen
```

```
        tication
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: received EAP
         packet (code=2 id=254 len=12) from STA: EAP Response-Identity (1)
Mar 23 02:49:00 access-node hostapd[2052]: IEEE 802.1X: <REDACTED> BE_AUTH entering state
        RESPONSE
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state RECEIVED
Mar 23 02:49:00 access-node hostapd[2052]: EAP: parseEapResp: rxResp=1 rxInitiate=0 respId
        =254 respMethod=1 respVendor=0 respVendorMethod=0
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state INTEGRITY_CHECK
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state METHOD_RESPONSE
Mar 23 02:49:00 access-node hostapd[2052]: EAP-Identity: Peer identity - hexdump_ascii(len
        =7):
Mar 23 02:49:00 access-node hostapd[2052]:      6e 61 63 75 73 65 72
                nacuser
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state SELECT_ACTION
Mar 23 02:49:00 access-node hostapd[2052]: EAP: getDecision: -> PASSTHROUGH
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state INITIALIZE_PASSTHROUGH
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state AAA_REQUEST
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state AAA_IDLE
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: STA identity
         'nacuser'
Mar 23 02:49:00 access-node hostapd[2052]: Encapsulating EAP message into a RADIUS packet
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Sending RADIUS message to authent
        ication server
Mar 23 02:49:00 access-node hostapd[2052]: RADIUS message: code=1 (Access-Request) identif
        ier=3 length=165
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 1 (User-Name) length=9
Mar 23 02:49:00 access-node hostapd[2052]:       Value: 'nacuser'
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 4 (NAS-IP-Address) length=6
Mar 23 02:49:00 access-node hostapd[2052]:       Value: 10.0.23.1
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 30 (Called-Station-Id) length=20
Mar 23 02:49:00 access-node hostapd[2052]:       Value: '<REDACTED>:'
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 61 (NAS-Port-Type) length=6
Mar 23 02:49:00 access-node hostapd[2052]:       Value: 19
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 6 (Service-Type) length=6
Mar 23 02:49:00 access-node hostapd[2052]:       Value: 2
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 31 (Calling-Station-Id) length=19
Mar 23 02:49:00 access-node hostapd[2052]:       Value: '<REDACTED>'
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 77 (Connect-Info) length=23
Mar 23 02:49:00 access-node hostapd[2052]:       Value: 'CONNECT 0Mbps 802.11b'
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 44 (Acct-Session-Id) length=18
Mar 23 02:49:00 access-node hostapd[2052]:       Value: '2268479955CB85A0'
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 12 (Framed-MTU) length=6
Mar 23 02:49:00 access-node hostapd[2052]:       Value: 1400
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 79 (EAP-Message) length=14
Mar 23 02:49:00 access-node hostapd[2052]:       Value: 02fe000c016e616375736572
```

```
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 80 (Message-Authenticator) length=
        18
Mar 23 02:49:00 access-node hostapd[2052]:       Value: ede40768f3affc833f5ecb655d5f6878
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Next RADIUS client retransmit in
        3 seconds
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: received EAP
        OL-Start from STA
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Received 80 bytes from RADIUS ser
        ver
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Received RADIUS message
Mar 23 02:49:00 access-node hostapd[2052]: RADIUS message: code=11 (Access-Challenge) iden
        tifier=3 length=80
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 80 (Message-Authenticator) length=
        18
Mar 23 02:49:00 access-node hostapd[2052]:       Value: e9f4ffb9acaefad6cd9fe021f6937580
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 79 (EAP-Message) length=24
Mar 23 02:49:00 access-node hostapd[2052]:       Value: 01ff0016041090ab891e63b1ce73b12833
        17e89cc41b
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 24 (State) length=18
Mar 23 02:49:00 access-node hostapd[2052]:       Value: 33b1db94334edf7cde63d907d83a901f
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> RADIUS: Received RADIUS p
        acket matched with a pending request, round trip time 0.00 sec
Mar 23 02:49:00 access-node hostapd[2052]: RADIUS packet matching with station <REDACTED>
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: decapsulated
         EAP packet (code=1 id=255 len=22) from RADIUS server: EAP-Request-MD5 (4)
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state AAA_RESPONSE
Mar 23 02:49:00 access-node hostapd[2052]: EAP: getId: id=255
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state SEND_REQUEST2
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state IDLE2
Mar 23 02:49:00 access-node hostapd[2052]: EAP: retransmit timeout 3 seconds (from dynamic
         back off; retransCount=0)
Mar 23 02:49:00 access-node hostapd[2052]: IEEE 802.1X: <REDACTED> BE_AUTH entering state
        REQUEST
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: Sending EAP
        Packet (identifier 255)
Mar 23 02:49:00 access-node hostapd[2052]: Received EAPOL packet
Mar 23 02:49:00 access-node hostapd[2052]: ens38: Event NEW_STA (22) received
Mar 23 02:49:00 access-node hostapd[2052]: ens38: Event EAPOL_RX (23) received
Mar 23 02:49:00 access-node hostapd[2052]: IEEE 802.1X: 46 bytes from <REDACTED>
Mar 23 02:49:00 access-node hostapd[2052]:    IEEE 802.1X: version=2 type=0 length=22
Mar 23 02:49:00 access-node hostapd[2052]:    ignoring 20 extra octets after IEEE 802.1X p
        acket
Mar 23 02:49:00 access-node hostapd[2052]: EAP: code=2 (response) identifier=255 length=22
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: received EAP
         packet (code=2 id=255 len=22) from STA: EAP Response-MD5 (4)
Mar 23 02:49:00 access-node hostapd[2052]: IEEE 802.1X: <REDACTED> BE_AUTH entering state
```

```
        RESPONSE
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state RECEIVED2
Mar 23 02:49:00 access-node hostapd[2052]: EAP: parseEapResp: rxResp=1 rxInitiate=0 respId
        =255 respMethod=4 respVendor=0 respVendorMethod=0
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state AAA_REQUEST
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state AAA_IDLE
Mar 23 02:49:00 access-node hostapd[2052]: Encapsulating EAP message into a RADIUS packet
Mar 23 02:49:00 access-node hostapd[2052]: Copied RADIUS State Attribute
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Sending RADIUS message to authent
        ication server
Mar 23 02:49:00 access-node hostapd[2052]: RADIUS message: code=1 (Access-Request) identif
        ier=4 length=193
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 1 (User-Name) length=9
Mar 23 02:49:00 access-node hostapd[2052]:       Value: 'nacuser'
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 4 (NAS-IP-Address) length=6
Mar 23 02:49:00 access-node hostapd[2052]:       Value: 10.0.23.1
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: unauthorizin
        g port
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 30 (Called-Station-Id) length=20
Mar 23 02:49:00 access-node hostapd[2052]:       Value: '<REDACTED>:'
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 61 (NAS-Port-Type) length=6
Mar 23 02:49:00 access-node hostapd[2052]:       Value: 19
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 6 (Service-Type) length=6
Mar 23 02:49:00 access-node hostapd[2052]:       Value: 2
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 31 (Calling-Station-Id) length=19
Mar 23 02:49:00 access-node hostapd[2052]:       Value: '<REDACTED>'
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 77 (Connect-Info) length=23
Mar 23 02:49:00 access-node hostapd[2052]:       Value: 'CONNECT 0Mbps 802.11b'
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 44 (Acct-Session-Id) length=18
Mar 23 02:49:00 access-node hostapd[2052]:       Value: '2268479955CB85A0'
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 12 (Framed-MTU) length=6
Mar 23 02:49:00 access-node hostapd[2052]:       Value: 1400
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 79 (EAP-Message) length=24
Mar 23 02:49:00 access-node hostapd[2052]:       Value: 02ff001604106159cae5bd45916254d85a
        8e12d860af
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 24 (State) length=18
Mar 23 02:49:00 access-node hostapd[2052]:       Value: 33b1db94334edf7cde63d907d83a901f
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 80 (Message-Authenticator) length=
        18
Mar 23 02:49:00 access-node hostapd[2052]:       Value: b10e50120d8d1718ff54b589c41af645
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Next RADIUS client retransmit in
        3 seconds
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Received 53 bytes from RADIUS ser
        ver
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Received RADIUS message
Mar 23 02:49:00 access-node hostapd[2052]: RADIUS message: code=2 (Access-Accept) identifi
```

```
        er=4 length=53
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 80 (Message-Authenticator) length=
        18
Mar 23 02:49:00 access-node hostapd[2052]:       Value: e995ad8e4287ff371bdeaf07853dceac
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 79 (EAP-Message) length=6
Mar 23 02:49:00 access-node hostapd[2052]:       Value: 03ff0004
Mar 23 02:49:00 access-node hostapd[2052]:    Attribute 1 (User-Name) length=9
Mar 23 02:49:00 access-node hostapd[2052]:       Value: 'nacuser'
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> RADIUS: Received RADIUS p
        acket matched with a pending request, round trip time 0.00 sec
Mar 23 02:49:00 access-node hostapd[2052]: RADIUS packet matching with station <REDACTED>
Mar 23 02:49:00 access-node hostapd[2052]: MS-MPPE: 1x_get_keys, could not get keys: 0x5cc
        575f92a40  send: (nil)  recv: (nil)
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: old identity
         'nacuser' updated with User-Name from Access-Accept 'nacuser'
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: decapsulated
         EAP packet (code=3 id=255 len=4) from RADIUS server: EAP Success
Mar 23 02:49:00 access-node hostapd[2052]: EAP: EAP entering state SUCCESS2
Mar 23 02:49:00 access-node hostapd[2052]: ens38: CTRL-EVENT-EAP-SUCCESS2 <REDACTED>
Mar 23 02:49:00 access-node hostapd[2052]: IEEE 802.1X: <REDACTED> BE_AUTH entering state
        SUCCESS
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: Sending EAP
        Packet (identifier 255)
Mar 23 02:49:00 access-node hostapd[2052]: IEEE 802.1X: <REDACTED> AUTH_PAE entering state
         AUTHENTICATED
Mar 23 02:49:00 access-node hostapd[2052]: ens38: AP-STA-CONNECTED <REDACTED>
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: Sending EAP
        Packet (identifier 254)
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: authorizing
        port
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> RADIUS: starting accounti
        ng session 2268479955CB85A0
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: authenticate
        d - EAP type: 4 (MD5)
Mar 23 02:49:00 access-node hostapd[2052]: IEEE 802.1X: <REDACTED> BE_AUTH entering state
        IDLE
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: received EAP
         packet (code=2 id=254 len=12) from STA: EAP Response-Identity (1)
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: STA identity
         'nacuser'
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Sending RADIUS message to authent
        ication server
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Next RADIUS client retransmit in
        3 seconds
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Received 80 bytes from RADIUS ser
        ver
```

```
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Received RADIUS message
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> RADIUS: Received RADIUS p
        acket matched with a pending request, round trip time 0.00 sec
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: decapsulated
         EAP packet (code=1 id=255 len=22) from RADIUS server: EAP-Request-MD5 (4)
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: Sending EAP
        Packet (identifier 255)
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: received EAP
         packet (code=2 id=255 len=22) from STA: EAP Response-MD5 (4)
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Sending RADIUS message to authent
        ication server
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Next RADIUS client retransmit in
        3 seconds
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Received 53 bytes from RADIUS ser
        ver
Mar 23 02:49:00 access-node hostapd[2052]: ens38: RADIUS Received RADIUS message
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> RADIUS: Received RADIUS p
        acket matched with a pending request, round trip time 0.00 sec
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: old identity
         'nacuser' updated with User-Name from Access-Accept 'nacuser'
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: decapsulated
         EAP packet (code=3 id=255 len=4) from RADIUS server: EAP Success
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: Sending EAP
        Packet (identifier 255)
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: authorizing
        port
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> RADIUS: starting accounti
        ng session 2268479955CB85A0
Mar 23 02:49:00 access-node hostapd[2052]: ens38: STA <REDACTED> IEEE 802.1X: authenticate
        d - EAP type: 4 (MD5)
Mar 23 02:49:03 access-node hostapd[2052]: IEEE 802.1X: <REDACTED> - (EAP) retransWhile --
        > 0
Mar 23 02:49:30 access-node hostapd[2052]: IEEE 802.1X: <REDACTED> - aWhile --> 0
```

B.1.6 NAC Incident Parser Success Output

```
radius-splunk: ~/incident-evidence/nac_incident_parser_success.txt
=== NAC Incident Evidence Summary ===

[1] Failed authentication simulation
- Client remained unauthorized: YES
- Client showed successful connection during failed test: NO
- hostapd saw authentication attempt: YES
- FreeRADIUS recorded failed auth / reject path: YES

[2] Remediation / successful re-test
```

```
- Client reached CTRL-EVENT-CONNECTED: YES
- Client reached Authorized state: YES
- hostapd showed successful authorization indicators: YES

[3] Incident conclusion
- SIMULATED NAC INCIDENT SUCCESSFULLY REPRODUCED AND REMEDIATED
```

B.2 GLPI Incident Record

B.2.1 GLPI Incident Title

```
Wired 802.1X authentication failure and remediation for client-1
```

B.2.2 GLPI Incident Status

```
Solved
```

B.2.3 GLPI Incident Record Raw Text

```
Incident summary:
client-1 failed wired 802.1X authentication during NAC validation after
the supplicant password was intentionally changed to an incorrect value
to simulate a credential issue.

Observed behavior:
- client-1 remained unauthorized during the failed test
- FreeRADIUS showed a failed authentication / reject path
- hostapd on access-node recorded the 802.1X authentication attempt

Investigation:
- Verified FreeRADIUS service health on radius-splunk
- Confirmed the known-good credentials still produced Access-Accept
  using radtest
- Revalidated the client wpa_supplicant configuration

Remediation:
- Restored the known-good wpa_supplicant credentials on client-1
- Re-ran the wired 802.1X authentication test

Recovery validation:
- client-1 showed CTRL-EVENT-CONNECTED
- client-1 supplicant port reached Authorized state
- access-node hostapd logs showed AP-STA-CONNECTED and port
  authorization

Impact:
This simulated a realistic enterprise NAC incident involving endpoint
```

credential failure and demonstrated successful troubleshooting and service restoration.

B.3 Guest Network Validation Outputs

B.3.1 Guest Interface and Route Validation

```
guest-test-1: ip -br a
lo               UNKNOWN        127.0.0.1/8 ::1/128
ens33            UP             10.0.20.137/24 fe80::250:56ff:feab:cdef/64

guest-test-1: ip route
default via 10.0.20.254 dev ens33 proto dhcp src 10.0.20.137 metric 100
10.0.20.0/24 dev ens33 proto kernel scope link src 10.0.20.137
10.0.20.254 dev ens33 proto dhcp scope link src 10.0.20.137 metric 100
```

B.3.2 Guest DHCP Renewal Validation

```
guest-test-1: sudo dhclient -v ens33
Internet Systems Consortium DHCP Client 4.4.1
Copyright 2004-2018 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/ens33/<REDACTED>
Sending on   LPF/ens33/<REDACTED>
Sending on   Socket/fallback
DHCPDISCOVER on ens33 to 255.255.255.255 port 67 interval 3 (xid=0x4a3f2c1d)
DHCPDISCOVER on ens33 to 255.255.255.255 port 67 interval 6 (xid=0x4a3f2c1d)
DHCPOFFER of 10.0.20.137 from 10.0.20.254
DHCPREQUEST for 10.0.20.137 on ens33 to 255.255.255.255 port 67 (xid=0x4a3f2c1d)
DHCPACK of 10.0.20.137 from 10.0.20.254 (xid=0x4a3f2c1d)
bound to 10.0.20.137 -- renewal in 3600 seconds.
```

B.3.3 Guest Gateway Reachability Validation

```
guest-test-1: ping -c 4 10.0.20.254
PING 10.0.20.254 (10.0.20.254) 56(84) bytes of data.
64 bytes from 10.0.20.254: icmp_seq=1 ttl=64 time=1.13 ms
64 bytes from 10.0.20.254: icmp_seq=2 ttl=64 time=0.710 ms
64 bytes from 10.0.20.254: icmp_seq=3 ttl=64 time=0.728 ms
64 bytes from 10.0.20.254: icmp_seq=4 ttl=64 time=3.25 ms

--- 10.0.20.254 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3038ms
rtt min/avg/max/mdev = 0.710/1.454/3.247/1.048 ms
```

B.3.4 Guest Protected Network Isolation Validation

```
guest-test-1: ping -c 4 10.0.30.10
PING 10.0.30.10 (10.0.30.10) 56(84) bytes of data.

--- 10.0.30.10 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3074ms

guest-test-1: ping -c 4 10.0.30.20
PING 10.0.30.20 (10.0.30.20) 56(84) bytes of data.

--- 10.0.30.20 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3071ms

guest-test-1: ping -c 4 10.0.34.1
PING 10.0.34.1 (10.0.34.1) 56(84) bytes of data.

--- 10.0.34.1 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3080ms

guest-test-1: ping -c 4 10.0.23.1
PING 10.0.23.1 (10.0.23.1) 56(84) bytes of data.

--- 10.0.23.1 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3079ms

guest-test-1: ping -c 4 10.0.12.1
PING 10.0.12.1 (10.0.12.1) 56(84) bytes of data.

--- 10.0.12.1 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3050ms
```

B.3.5 Guest HTTP-to-GLPI Isolation Validation

```
guest-test-1: curl -I --max-time 5 http://10.0.30.20/
curl: (28) Connection timed out after 5001 milliseconds
```

B.3.6 Guest Internet-Allowed Validation

```
guest-test-1: ping -c 4 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=58 time=8.04 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=58 time=10.6 ms
64 bytes from 1.1.1.1: icmp_seq=3 ttl=58 time=8.98 ms
64 bytes from 1.1.1.1: icmp_seq=4 ttl=58 time=11.4 ms

--- 1.1.1.1 ping statistics ---
```

```
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 8.043/9.758/11.444/1.326 ms

guest-test-1: curl -I --max-time 5 http://www.google.com
HTTP/1.1 200 OK
Content-Type: text/html; charset=ISO-8859-1
Content-Security-Policy-Report-Only: object-src 'none';base-uri 'self';script-src 'nonce-Q
        UP1Z9-ttgVc2d-VTB3gXA' 'strict-dynamic' 'report-sample' 'unsafe-eval' 'unsafe-inline
        https: http:;report-uri https://csp.withgoogle.com/csp/gws/other-hp
Reporting-Endpoints: default="//www.google.com/httpservice/retry/jserror?ei=n1PCacmWHbav5N
        oPtbr14AO&cad=crash&error=Page%20Crash&jsel=1&bver=2405&dpf=
        nIfF8NgQdNWxHThvch26jsRSkc4DpZUm6gVT80MTHMY"
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Date: Tue, 24 Mar 2026 09:04:31 GMT
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Expires: Tue, 24 Mar 2026 09:04:31 GMT
Cache-Control: private
Set-Cookie: <REDACTED>
Set-Cookie: <REDACTED>
Transfer-Encoding: chunked
```

B.4 Automation Execution Evidence

B.4.1 Ansible Validation-Only Run Output

```
radius-splunk: ansible-playbook -i inventory/hosts.ini playbooks/nac_validation.yml -K

PLAY [Validate NAC services and optionally remediate] **************************

TASK [Collect basic hostname] **************************************************
ok: [radius-splunk]
ok: [access-node]

TASK [Check hostapd-wired service on access-node] ******************************
skipping: [radius-splunk]
ok: [access-node]

TASK [Check freeradius service on radius-splunk] *******************************
skipping: [access-node]
ok: [radius-splunk]

TASK [Restart hostapd-wired service when remediation is enabled] ***************
skipping: [radius-splunk]
skipping: [access-node]
```

```
TASK [Restart freeradius service when remediation is enabled] ******************
skipping: [access-node]
skipping: [radius-splunk]

TASK [Re-check hostapd-wired service on access-node] ***************************
skipping: [radius-splunk]
ok: [access-node]

TASK [Re-check freeradius service on radius-splunk] ****************************
skipping: [access-node]
ok: [radius-splunk]

TASK [Collect recent hostapd logs] *********************************************
skipping: [radius-splunk]
ok: [access-node]

TASK [Collect recent freeradius service logs] **********************************
skipping: [access-node]
ok: [radius-splunk]

TASK [Collect recent freeradius application log] *******************************
skipping: [access-node]
ok: [radius-splunk]

TASK [Save hostapd pre-check status locally] ***********************************
skipping: [radius-splunk]
changed: [access-node -> localhost]

TASK [Save hostapd post-check status locally] **********************************
skipping: [radius-splunk]
changed: [access-node -> localhost]

TASK [Save hostapd restart result locally] *************************************
skipping: [radius-splunk]
changed: [access-node -> localhost]

TASK [Save hostapd logs locally] ***********************************************
skipping: [radius-splunk]
ok: [access-node -> localhost]

TASK [Save freeradius pre-check status locally] ********************************
skipping: [access-node]
changed: [radius-splunk -> localhost]

TASK [Save freeradius post-check status locally] *******************************
```

```
skipping: [access-node]
changed: [radius-splunk -> localhost]

TASK [Save freeradius restart result locally] **********************************
skipping: [access-node]
changed: [radius-splunk -> localhost]

TASK [Save freeradius service logs locally] ************************************
skipping: [access-node]
ok: [radius-splunk -> localhost]

TASK [Save freeradius application log locally] *********************************
skipping: [access-node]
ok: [radius-splunk -> localhost]

PLAY RECAP *********************************************************************
access-node                : ok=8    changed=3    unreachable=0    failed=0    skipped=11
        rescued=0    ignored=0
radius-splunk              : ok=10   changed=3    unreachable=0    failed=0    skipped=9
        rescued=0    ignored=0
```

B.4.2 Ansible Remediation-Enabled Run Output

```
radius-splunk: ansible-playbook -i inventory/hosts.ini playbooks/nac_validation.yml -K -e re

PLAY [Validate NAC services and optionally remediate] **************************

TASK [Collect basic hostname] **************************************************
ok: [radius-splunk]
ok: [access-node]

TASK [Check hostapd-wired service on access-node] ******************************
skipping: [radius-splunk]
ok: [access-node]

TASK [Check freeradius service on radius-splunk] *******************************
skipping: [access-node]
ok: [radius-splunk]

TASK [Restart hostapd-wired service when remediation is enabled] ***************
skipping: [radius-splunk]
changed: [access-node]

TASK [Restart freeradius service when remediation is enabled] ******************
skipping: [access-node]
changed: [radius-splunk]
```

```
TASK [Re-check hostapd-wired service on access-node] ***************************
skipping: [radius-splunk]
ok: [access-node]

TASK [Re-check freeradius service on radius-splunk] ****************************
skipping: [access-node]
ok: [radius-splunk]

TASK [Collect recent hostapd logs] *********************************************
skipping: [radius-splunk]
ok: [access-node]

TASK [Collect recent freeradius service logs] **********************************
skipping: [access-node]
ok: [radius-splunk]

TASK [Collect recent freeradius application log] *******************************
skipping: [access-node]
ok: [radius-splunk]

TASK [Save hostapd pre-check status locally] ***********************************
skipping: [radius-splunk]
changed: [access-node -> localhost]

TASK [Save hostapd post-check status locally] **********************************
skipping: [radius-splunk]
changed: [access-node -> localhost]

TASK [Save hostapd restart result locally] *************************************
skipping: [radius-splunk]
ok: [access-node -> localhost]

TASK [Save hostapd logs locally] ***********************************************
skipping: [radius-splunk]
changed: [access-node -> localhost]

TASK [Save freeradius pre-check status locally] ********************************
skipping: [access-node]
changed: [radius-splunk -> localhost]

TASK [Save freeradius post-check status locally] *******************************
skipping: [access-node]
changed: [radius-splunk -> localhost]

TASK [Save freeradius restart result locally] **********************************
```

```
skipping: [access-node]
ok: [radius-splunk -> localhost]

TASK [Save freeradius service logs locally] ************************************
skipping: [access-node]
changed: [radius-splunk -> localhost]

TASK [Save freeradius application log locally] *********************************
skipping: [access-node]
changed: [radius-splunk -> localhost]

PLAY RECAP *********************************************************************
access-node                : ok=9    changed=4    unreachable=0    failed=0    skipped=10
        rescued=0    ignored=0
radius-splunk              : ok=11   changed=5    unreachable=0    failed=0    skipped=8
        rescued=0    ignored=0
```

B.4.3 Ansible Post-Run Artifact Directory Listing

```
radius-splunk: ls -lh ~/ansible-nac/artifacts
total 44K
-rw-r--r-- 1 nac-lab nac-lab 3.5K Mar 24 12:23 access-node_hostapd_logs.txt
-rw-r--r-- 1 nac-lab nac-lab   25 Mar 24 12:43 access-node_hostapd_restart_result.txt
-rw-r--r-- 1 nac-lab nac-lab 1.4K Mar 24 12:43 access-node_hostapd_status_after.txt
-rw-r--r-- 1 nac-lab nac-lab 1.4K Mar 24 12:43 access-node_hostapd_status_before.txt
-rw-r--r-- 1 root    root    1.4K Mar 23 19:30 access-node_hostapd_status.txt
-rw-r--r-- 1 nac-lab nac-lab 4.0K Mar 24 12:23 radius-splunk_freeradius_journal.txt
-rw-r--r-- 1 nac-lab nac-lab 3.1K Mar 24 12:23 radius-splunk_freeradius_radiuslog.txt
-rw-r--r-- 1 nac-lab nac-lab   25 Mar 24 12:43 radius-splunk_freeradius_restart_result.txt
-rw-r--r-- 1 nac-lab nac-lab 1.8K Mar 24 12:43 radius-splunk_freeradius_status_after.txt
-rw-r--r-- 1 nac-lab nac-lab 1.8K Mar 24 12:43 radius-splunk_freeradius_status_before.txt
-rw-r--r-- 1 root    root    1.8K Mar 23 19:30 radius-splunk_freeradius_status.txt
```

B.4.4 Ansible Generated Artifact Preview

```
===== /home/nac-lab/ansible-nac/artifacts/access-node_hostapd_logs.txt =====
Mar 24 12:23:12 access-node hostapd[1927]: hostapd_interface_deinit_free(0x61845b677590)
Mar 24 12:23:12 access-node hostapd[1927]: hostapd_interface_deinit_free: num_bss=1
        conf->num_bss=1
Mar 24 12:23:12 access-node hostapd[1927]: hostapd_interface_deinit(0x61845b677590)
Mar 24 12:23:12 access-node hostapd[1927]: ens38: interface state ENABLED->DISABLED
Mar 24 12:23:12 access-node hostapd[1927]: hostapd_bss_deinit: deinit bss ens38
Mar 24 12:23:12 access-node hostapd[1927]: ens38: Flushing old station entries
Mar 24 12:23:12 access-node hostapd[1927]: ens38: Deauthenticate all stations
Mar 24 12:23:12 access-node hostapd[1927]: ens38: AP-DISABLED
```

```
===== /home/nac-lab/ansible-nac/artifacts/access-node_hostapd_restart_result.txt =====
remediation not requested

===== /home/nac-lab/ansible-nac/artifacts/access-node_hostapd_status_after.txt =====
- hostapd-wired.service - hostapd wired 802.1X authenticator (lab)
     Loaded: loaded (/etc/systemd/system/hostapd-wired.service; enabled; preset: enabled)
     Active: active (running) since Tue 2026-03-24 12:23:12 UTC; 20min ago
   Main PID: 2054 (hostapd)
      Tasks: 1 (limit: 2206)
     Memory: 844.0K (peak: 1.0M)
        CPU: 31ms
     CGroup: /system.slice/hostapd-wired.service

===== /home/nac-lab/ansible-nac/artifacts/access-node_hostapd_status_before.txt =====
- hostapd-wired.service - hostapd wired 802.1X authenticator (lab)
     Loaded: loaded (/etc/systemd/system/hostapd-wired.service; enabled; preset: enabled)
     Active: active (running) since Tue 2026-03-24 12:23:12 UTC; 20min ago
   Main PID: 2054 (hostapd)
      Tasks: 1 (limit: 2206)
     Memory: 844.0K (peak: 1.0M)
        CPU: 31ms
     CGroup: /system.slice/hostapd-wired.service

===== /home/nac-lab/ansible-nac/artifacts/access-node_hostapd_status.txt =====
- hostapd-wired.service - hostapd wired 802.1X authenticator (lab)
     Loaded: loaded (/etc/systemd/system/hostapd-wired.service; enabled; preset: enabled)
     Active: active (running) since Mon 2026-03-23 18:51:07 UTC; 39min ago
   Main PID: 1017 (hostapd)
      Tasks: 1 (limit: 2206)
     Memory: 2.9M (peak: 3.1M)
        CPU: 51ms
     CGroup: /system.slice/hostapd-wired.service

===== /home/nac-lab/ansible-nac/artifacts/radius-splunk_freeradius_journal.txt =====
Mar 24 12:22:39 radius-splunk freeradius[3253]: Compiling Post-Auth-Type Challenge for
        attr Post-Auth-Type
Mar 24 12:22:39 radius-splunk freeradius[3253]: Compiling Post-Auth-Type Client-Lost for
        attr Post-Auth-Type
Mar 24 12:22:39 radius-splunk freeradius[3253]: radiusd: #### Skipping IP addresses and
        Ports ####
Mar 24 12:22:39 radius-splunk freeradius[3253]: Configuration appears to be OK
Mar 24 12:22:39 radius-splunk systemd[1]: Started freeradius.service - FreeRADIUS
        multi-protocol policy server.
Mar 24 12:23:12 radius-splunk systemd[1]: Stopping freeradius.service - FreeRADIUS
        multi-protocol policy server...
Mar 24 12:23:12 radius-splunk systemd[1]: freeradius.service: Deactivated successfully.
```

```
Mar 24 12:23:12 radius-splunk systemd[1]: Stopped freeradius.service - FreeRADIUS
        multi-protocol policy server.

===== /home/nac-lab/ansible-nac/artifacts/radius-splunk_freeradius_radiuslog.txt =====
Tue Mar 24 11:34:18 2026 : Info: Loaded virtual server <default>
Tue Mar 24 11:34:18 2026 : Warning: Ignoring "sql" (see raddb/mods-available/README.rst)
Tue Mar 24 11:34:18 2026 : Warning: Ignoring "ldap" (see raddb/mods-available/README.rst)
Tue Mar 24 11:34:18 2026 : Info:  # Skipping contents of 'if' as it is always 'false' --
        /etc/freeradius/3.0/sites-enabled/inner-tunnel:366
Tue Mar 24 11:34:18 2026 : Info: Loaded virtual server inner-tunnel
Tue Mar 24 11:34:18 2026 : Info: Loaded virtual server default
Tue Mar 24 11:34:18 2026 : Info: Ready to process requests
Tue Mar 24 11:57:46 2026 : Info: Signalled to terminate

===== /home/nac-lab/ansible-nac/artifacts/radius-splunk_freeradius_restart_result.txt =====
remediation not requested

===== /home/nac-lab/ansible-nac/artifacts/radius-splunk_freeradius_status_after.txt =====
- freeradius.service - FreeRADIUS multi-protocol policy server
     Loaded: loaded (/usr/lib/systemd/system/freeradius.service; enabled; preset: enabled)
     Active: active (running) since Tue 2026-03-24 12:23:12 UTC; 20min ago
       Docs: man:radiusd(8)
             man:radiusd.conf(5)
             http://wiki.freeradius.org/
             http://networkradius.com/doc/

===== /home/nac-lab/ansible-nac/artifacts/radius-splunk_freeradius_status_before.txt =====
- freeradius.service - FreeRADIUS multi-protocol policy server
     Loaded: loaded (/usr/lib/systemd/system/freeradius.service; enabled; preset: enabled)
     Active: active (running) since Tue 2026-03-24 12:23:12 UTC; 20min ago
       Docs: man:radiusd(8)
             man:radiusd.conf(5)
             http://wiki.freeradius.org/
             http://networkradius.com/doc/

===== /home/nac-lab/ansible-nac/artifacts/radius-splunk_freeradius_status.txt =====
- freeradius.service - FreeRADIUS multi-protocol policy server
     Loaded: loaded (/usr/lib/systemd/system/freeradius.service; enabled; preset: enabled)
     Active: active (running) since Mon 2026-03-23 18:52:42 UTC; 37min ago
       Docs: man:radiusd(8)
             man:radiusd.conf(5)
             http://wiki.freeradius.org/
             http://networkradius.com/doc/
```

B.5 Selected Observability Excerpts

B.5.1 Splunk UF Runtime Status

```
radius-splunk: sudo /opt/splunkforwarder/bin/splunk status
Warning: Attempting to revert the SPLUNK_HOME ownership
Warning: Executing "chown -R root /opt/splunkforwarder"
splunkd is running (PID: 1489)
splunk helpers are running (PIDs: 1490)
```

B.5.2 syslog Excerpt

```
radius-splunk: sudo tail -n 15 /var/log/syslog
2026-03-24T13:14:41.348438+00:00 radius-splunk multipathd[467]: sda: failed to get sgio
        uid: No such file or directory
2026-03-24T13:14:41.348468+00:00 radius-splunk multipathd[467]: sda: no WWID in state
        "undef
2026-03-24T13:14:41.348639+00:00 radius-splunk multipathd[467]: ", giving up
2026-03-24T13:14:41.348690+00:00 radius-splunk multipathd[467]: sda: check_path() failed,
        removing
2026-03-24T13:15:01.629554+00:00 radius-splunk CRON[1579]: (root) CMD (command -v
        debian-sa1 > /dev/null && debian-sa1 1 1)
2026-03-24T13:17:01.652202+00:00 radius-splunk CRON[1587]: (root) CMD (cd / && run-parts
        --report /etc/cron.hourly)
2026-03-24T13:19:05.468094+00:00 radius-splunk systemd[1]: Starting update-notifier-
        download.service - Download data for packages that failed at package install
        time...
2026-03-24T13:19:05.678561+00:00 radius-splunk systemd[1]: update-notifier-download.
        service: Deactivated successfully.
2026-03-24T13:19:05.678792+00:00 radius-splunk systemd[1]: Finished update-notifier-
        download.service - Download data for packages that failed at package install time.
2026-03-24T13:19:31.934534+00:00 radius-splunk systemd[1376]: launchpadlib-cache-clean.
        service - Clean up old files in the Launchpadlib cache was skipped because of an
        unmet condition check (ConditionPathExists=/home/nac-lab/.launchpadlib/
        api.launchpad.net/cache).
```

B.5.3 auth.log Excerpt

```
radius-splunk: sudo tail -n 15 /var/log/auth.log
2026-03-24T13:20:04.756710+00:00 radius-splunk sshd[1628]: pam_unix(sshd:session): session
         opened for user nac-lab(uid=1000) by nac-lab(uid=0)
2026-03-24T13:20:04.762714+00:00 radius-splunk systemd-logind[816]: New session 5 of user
        nac-lab.
2026-03-24T13:20:05.503276+00:00 radius-splunk sshd[1686]: Received disconnect from 192.16
        8.195.1 port 41932:11: disconnected by user
2026-03-24T13:20:05.503525+00:00 radius-splunk sshd[1686]: Disconnected from user nac-lab
        192.168.195.1 port 41932
```

```
2026-03-24T13:20:05.504875+00:00 radius-splunk sshd[1628]: pam_unix(sshd:session): session
         closed for user nac-lab
2026-03-24T13:20:05.511847+00:00 radius-splunk systemd-logind[816]: Session 5 logged out.
        Waiting for processes to exit.
2026-03-24T13:20:05.514296+00:00 radius-splunk systemd-logind[816]: Removed session 5.
2026-03-24T13:20:37.703552+00:00 radius-splunk sudo:  nac-lab : TTY=tty1 ; PWD=/home/nac-l
        ab ; USER=root ; COMMAND=/usr/bin/tail -n 15 /var/log/syslog
2026-03-24T13:20:37.704672+00:00 radius-splunk sudo: pam_unix(sudo:session): session opene
        d for user root(uid=0) by nac-lab(uid=1000)
2026-03-24T13:20:37.731117+00:00 radius-splunk sudo: pam_unix(sudo:session): session close
        d for user root
2026-03-24T13:21:39.041208+00:00 radius-splunk sudo:  nac-lab : TTY=tty1 ; PWD=/home/nac-l
        ab ; USER=root ; COMMAND=/usr/bin/tail -n 15 /var/log/auth.log
2026-03-24T13:21:39.042808+00:00 radius-splunk sudo: pam_unix(sudo:session): session opene
        d for user root(uid=0) by nac-lab(uid=1000)
2026-03-24T13:21:39.068864+00:00 radius-splunk sudo: pam_unix(sudo:session): session close
        d for user root
2026-03-24T13:22:03.513481+00:00 radius-splunk sudo:  nac-lab : TTY=tty1 ; PWD=/home/nac-l
        ab ; USER=root ; COMMAND=/usr/bin/tail -n 15 /var/log/auth.log
2026-03-24T13:22:03.514636+00:00 radius-splunk sudo: pam_unix(sudo:session): session opene
        d for user root(uid=0) by nac-lab(uid=1000)
```

B.5.4 FreeRADIUS radius.log Excerpt

```
radius-splunk: sudo tail -n 20 /var/log/freeradius/radius.log
Tue Mar 24 12:23:12 2026 : Info: Debug state unknown (cap_sys_ptrace capability not set)
Tue Mar 24 12:23:12 2026 : Info: systemd watchdog interval is 30.00 secs
Tue Mar 24 12:23:12 2026 : Info: Loaded virtual server <default>
Tue Mar 24 12:23:12 2026 : Warning: Ignoring "sql" (see raddb/mods-available/README.rst)
Tue Mar 24 12:23:12 2026 : Warning: Ignoring "ldap" (see raddb/mods-available/README.rst)
Tue Mar 24 12:23:12 2026 : Info:  # Skipping contents of 'if' as it is always 'false' --
        /etc/freeradius/3.0/sites-enabled/inner-tunnel:366
Tue Mar 24 12:23:12 2026 : Info: Loaded virtual server inner-tunnel
Tue Mar 24 12:23:12 2026 : Info: Loaded virtual server default
Tue Mar 24 12:23:12 2026 : Info: Ready to process requests
Tue Mar 24 12:53:42 2026 : Info: Signalled to terminate
Tue Mar 24 12:53:42 2026 : Info: Exiting normally
Tue Mar 24 13:14:05 2026 : Info: Debug state unknown (cap_sys_ptrace capability not set)
Tue Mar 24 13:14:05 2026 : Info: systemd watchdog interval is 30.00 secs
Tue Mar 24 13:14:05 2026 : Info: Loaded virtual server <default>
Tue Mar 24 13:14:05 2026 : Warning: Ignoring "sql" (see raddb/mods-available/README.rst)
Tue Mar 24 13:14:05 2026 : Warning: Ignoring "ldap" (see raddb/mods-available/README.rst)
Tue Mar 24 13:14:05 2026 : Info:  # Skipping contents of 'if' as it is always 'false' --
        /etc/freeradius/3.0/sites-enabled/inner-tunnel:366
Tue Mar 24 13:14:05 2026 : Info: Loaded virtual server inner-tunnel
Tue Mar 24 13:14:05 2026 : Info: Loaded virtual server default
```

```
Tue Mar 24 13:14:05 2026 : Info: Ready to process requests
```

B.5.5 hostapd Service Status and Journal Excerpt

```
access-node: sudo systemctl status hostapd-wired.service --no-pager
- hostapd-wired.service - hostapd wired 802.1X authenticator (lab)
     Loaded: loaded (/etc/systemd/system/hostapd-wired.service; enabled; preset: enabled)
     Active: active (running) since Tue 2026-03-24 04:20:41 UTC; 16min ago
   Main PID: 975 (hostapd)
      Tasks: 1 (limit: 2206)
     Memory: 2.9M (peak: 3.1M)
        CPU: 42ms
     CGroup: /system.slice/hostapd-wired.service
             975 /usr/sbin/hostapd -dd /etc/hostapd/lab/hostapd-wired.conf

Mar 24 04:20:41 access-node hostapd[975]: Using interface ens38 with hwaddr <REDACTED> and
         ssid ""
Mar 24 04:20:41 access-node hostapd[975]: ens38: RADIUS Authentication server 10.0.30.10:1
        812
Mar 24 04:20:41 access-node hostapd[975]: RADIUS local address: 10.0.23.1:45833
Mar 24 04:20:41 access-node hostapd[975]: ens38: IEEE 802.11 Fetching hardware channel/rat
        e support not supported.
Mar 24 04:20:41 access-node hostapd[975]: ens38: RADIUS Authentication server 10.0.30.10:1
        812
Mar 24 04:20:41 access-node hostapd[975]: ens38: Deauthenticate all stations at BSS start
Mar 24 04:20:41 access-node hostapd[975]: ens38: interface state UNINITIALIZED->ENABLED
Mar 24 04:20:41 access-node hostapd[975]: ens38: AP-ENABLED
Mar 24 04:20:41 access-node hostapd[975]: ens38: Setup of interface done.

access-node: sudo journalctl -u hostapd-wired.service -n 20 --no-pager
Mar 24 02:13:55 access-node hostapd[2398]: ens38: CTRL-EVENT-TERMINATING
Mar 24 02:13:55 access-node hostapd[2398]: hostapd_free_hapd_data(ens38)
Mar 24 02:13:55 access-node hostapd[2398]: hostapd_interface_deinit_free: driver=0x565a2d7
        47060 drv_priv=0x565a38444cd0 -> hapd_deinit
Mar 24 02:13:55 access-node hostapd[2398]: hostapd_interface_free(0x565a38442590)
Mar 24 02:13:55 access-node hostapd[2398]: hostapd_interface_free: free hapd 0x565a38443d4
        0
Mar 24 02:13:55 access-node hostapd[2398]: hostapd_cleanup_iface(0x565a38442590)
Mar 24 02:13:55 access-node hostapd[2398]: hostapd_cleanup_iface_partial(0x565a38442590)
Mar 24 02:13:55 access-node hostapd[2398]: hostapd_cleanup_iface: free iface=0x565a3844259
        0
Mar 24 02:13:55 access-node systemd[1]: hostapd-wired.service: Deactivated successfully.
Mar 24 02:13:55 access-node systemd[1]: Stopped hostapd-wired.service - hostapd wired 802.
        1X authenticator (lab).
-- Boot 4efbb5bbbad2458a94bc281c688f0c9b --
Mar 24 04:20:41 access-node systemd[1]: Started hostapd-wired.service - hostapd wired 802.
```

```
        1X authenticator (lab).
Mar 24 04:20:41 access-node hostapd[975]: random: getrandom() support available
Mar 24 04:20:41 access-node hostapd[975]: Configuration file: /etc/hostapd/lab/hostapd-wir
        ed.conf
Mar 24 04:20:41 access-node hostapd[975]: eapol_version=2
Mar 24 04:20:41 access-node hostapd[975]: ctrl_interface_group=0
Mar 24 04:20:41 access-node hostapd[975]: Opening raw packet socket for ifindex 3
Mar 24 04:20:41 access-node hostapd[975]: BSS count 1, BSSID mask 00:00:00:00:00:00 (0 bit
        s)
Mar 24 04:20:41 access-node hostapd[975]: ens38: IEEE 802.11 Fetching hardware channel/rat
        e support not supported.
Mar 24 04:20:41 access-node hostapd[975]: Completing interface initialization
Mar 24 04:20:41 access-node hostapd[975]: hostapd_setup_bss(hapd=0x648c62cc3d40 (ens38), f
        irst=1)
Mar 24 04:20:41 access-node hostapd[975]: Using interface ens38 with hwaddr <REDACTED> and
         ssid ""
Mar 24 04:20:41 access-node hostapd[975]: ens38: RADIUS Authentication server 10.0.30.10:1
        812
Mar 24 04:20:41 access-node hostapd[975]: RADIUS local address: 10.0.23.1:45833
Mar 24 04:20:41 access-node hostapd[975]: ens38: IEEE 802.11 Fetching hardware channel/rat
        e support not supported.
Mar 24 04:20:41 access-node hostapd[975]: ens38: RADIUS Authentication server 10.0.30.10:1
        812
Mar 24 04:20:41 access-node hostapd[975]: ens38: Deauthenticate all stations at BSS start
Mar 24 04:20:41 access-node hostapd[975]: ens38: interface state UNINITIALIZED->ENABLED
Mar 24 04:20:41 access-node hostapd[975]: ens38: AP-ENABLED
Mar 24 04:20:41 access-node hostapd[975]: ens38: Setup of interface done.
Mar 24 04:20:41 access-node hostapd[975]: ctrl_iface not configured!
```

B.6 Packet Capture Metadata and Figure Mapping

B.6.1 pcap File Inventory

```
access-node: ls -lh ~/pcaps
-rw-r--r-- 1 nac-lab nac-lab 385 Mar 23 20:23 failed_8021x_ens38.pcap
-rw-r--r-- 1 nac-lab nac-lab 738 Mar 23 20:23 failed_8021x_ens39.pcap
-rw-r--r-- 1 nac-lab nac-lab 385 Mar 23 20:29 success_8021x_ens38.pcap
-rw-r--r-- 1 nac-lab nac-lab 747 Mar 23 20:29 success_8021x_ens39.pcap
```

B.6.2 Packet Capture Methodology and tcpdump Commands

```
access-node:
sudo timeout 20s tcpdump -i ens38 -w /tmp/failed_8021x_ens38.pcap ether proto 0x888e &
sudo timeout 20s tcpdump -i ens39 -w /tmp/failed_8021x_ens39.pcap host 10.0.30.10 and
        udp port 1812 &
sleep 2
```

```
client-1:
sudo systemctl stop wpa_supplicant
sudo timeout 15s wpa_supplicant -i ens33 -D wired -c
        /etc/wpa_supplicant/lab/wired-8021x.conf -dd

access-node:
sudo timeout 20s tcpdump -i ens38 -w ~/pcaps/success_8021x_ens38.pcap ether proto 0x888e &
sudo timeout 20s tcpdump -i ens39 -w ~/pcaps/success_8021x_ens39.pcap host 10.0.30.10 and
        udp port 1812 &
sleep 2

client-1:
sudo systemctl stop wpa_supplicant
sudo timeout 15s wpa_supplicant -i ens33 -D wired -c
        /etc/wpa_supplicant/lab/wired-8021x.conf -dd
```

B.6.3 Packet Capture Redaction Policy

```
Publication policy:
- private IP addresses: keep visible
- MAC addresses: redact
- passwords/shared secrets/auth blobs: redact
- packet payload details that expose secrets: redact

Accordingly, the publication-safe Wireshark images should redact:
- MAC addresses in the packet list pane
- MAC addresses in the packet details pane
- MAC addresses in the hex pane
- RADIUS Authenticator values where shown
- Message-Authenticator values where shown
```

End of document.