NIST 800-63-4 Auditor — Operational Guide

Version 1.0 · Generated:

> **Overview.** This guide explains how to install, configure, and operate the NIST 800-63-4 Auditor Python tool so others in your organization can run consistent assessments. The tool evaluates IAL, AAL, FAL and checks password/ authenticator requirements, MFA, session management, credential storage, transport security, privacy, and account recovery.

Key Capabilities

- Automated data collection from Microsoft Entra ID (policies, MFA, conditional access)
- Local environment/system probing for crypto, TLS, and endpoint hygiene
- Manual configuration intake for systems without API access
- Standards-based checks aligned to NIST SP 800-63-4
- Report output in JSON, text, and HTML

Supported Data Sources

- Manual input: prompts or YAML config file
- Entra ID collector: Microsoft Graph read-only policy collection
- Auto collector: local host info (crypto/TLS probes, libraries)
- HTTPS probe: target URL TLS and header checks

System Requirements

- Python 3.9+ (3.11 recommended)
- Internet access for Entra ID/Graph and HTTPS probing
- OS: Linux, macOS, or Windows
- Permissions: Azure app registration with read-only Graph scopes for Entra ID

Installation

1) Clone and set up environment

```
git clone <your-repo-or-path>
cd <auditor-directory>
python -m venv .venv
source .venv/bin/activate  # Windows: .venv\Scripts\activate
pip install -U pip wheel
pip install -r requirements.txt
```

If you do not have a requirements.txt, install typical dependencies:

```
pip install requests msal cryptography pyopenssl pyyaml rich
```

2) Optional: Docker

```
# Dockerfile
FROM python:3.11-slim
WORKDIR /app
COPY . /app
RUN pip install -U pip && pip install -r requirements.txt
ENV PYTHONUNBUFFERED=1
CMD ["python", "auditor.py", "--help"]
```

```
docker build -t nist-auditor:latest .
docker run --rm -it nist-auditor:latest --help
```

Configuration

A) Entra ID (Microsoft Graph) Setup

Create an Azure App Registration and capture Tenant ID, Client ID, and Client Secret (store securely). Grant read-only Microsoft Graph application permissions using least privilege. Examples:

- Policy.Read.All
- Directory.Read.All
- AuditLog.Read.All
- IdentityUserFlow.Read.All
- AuthenticationMethodPolicy.Read.All
- ConditionalAccess.Read.All

Set environment variables (preferred) or pass via CLI:

```
export ENTRA_TENANT_ID="xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
export ENTRA_CLIENT_ID="yyyyyyyy-yyyy-yyyy-yyyy-yyyyyyyyyyyy"
export ENTRA_CLIENT_SECRET="your-super-secret"
```

B) Manual Config (YAML)

```
system:
  name: "Payment-Portal"
  owner: "IAM Team"
  environment: "production"

authentication:
  password_policy:
    min_length: 12
    requires_uppercase: true
    requires_lowercase: true
    requires_number: true
    requires_symbol: true
    reuse_prevention: 24
    max_age_days: 365
    lockout_threshold: 10
    lockout_duration_minutes: 15
  mfa:
    enabled: true
    allowed_factors: ["TOTP", "FIDO2", "Push"]
    required_for_admins: true
    required_for_all_users: true
  recovery:
    email_reset_allowed: false
    sms_reset_allowed: false
    kbq_allowed: false

session:
  idle_timeout_minutes: 15
  absolute_timeout_hours: 8
  reauth_for_sensitive_actions: true

transport:
  https_required: true
  hsts_enabled: true
  min_tls_version: "1.2"
  allowed_ciphersuites: ["TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384", "TLS_AES_256_GCM_SHA384"]

privacy:
  pii_minimization: true
  pii_retention_days: 365
  pii_encrypted_at_rest: true
```

Running the Auditor

```
python auditor.py --help
```

Common modes:

```
# Manual config file
python auditor.py --input config.yaml --format json --out ./reports

# Interactive prompts (no file)
python auditor.py --interactive --format txt

# Entra ID automated collection (uses env vars)
python auditor.py --entra --format html --out ./reports

# HTTPS probe of endpoints
python auditor.py --probe https://login.example.com --probe https://app.example.com --format js

# Combine Entra ID with HTTPS probe
python auditor.py --entra --probe https://login.microsoftonline.com --format html --out ./repor
```

Output formats:

- `json` machine-readable findings
- `txt` console-friendly summary
- `html` shareable report

Understanding Results

Each control is evaluated with a status and rationale:

- **PASS**: Meets NIST SP 800-63-4 expectation
- **WARN**: Partially meets or ambiguous
- **FAIL**: Does not meet expectation

Findings include category, evidence, expected criteria, gap analysis, and suggested remediation. Ensure target IAL/AAL/FAL are set via CLI or YAML.

Security and Privacy Guidance

- Store secrets in a vault; never commit secrets.
- Grant only read-only Graph scopes and obtain admin consent.
- Limit distribution of reports; they may include sensitive details.
- Use allowlists for probes; avoid scanning without authorization.

- Sanitize reports before external sharing.

Troubleshooting

- **Auth errors (401/403)**: verify tenant/client IDs, secret, and admin consent for scopes.
- **Missing data**: ensure appropriate read scopes (e.g., ConditionalAccess.Read.All).
- **Rate limiting (429)**: retry later; add backoff in pipelines.
- **TLS probe failures**: check hostname/SNI and proxies (set HTTPS_PROXY if needed).
- **HTML report missing**: ensure your version supports --format html and dependencies are installed.

Operational Best Practices

- Pin versions in requirements.txt; update quarterly.
- Rotate client secrets regularly; prefer managed identities.
- Log centrally with redaction.
- Maintain a changelog for control logic updates.
- Validate Graph schema changes periodically.

Example Commands

```
# Baseline Entra ID assessment
python auditor.py --entra --format html --out ./reports

# Production web app probe with manual config
python auditor.py --input prod-app.yaml --probe https://app.example.com --format json --out ./r

# Quick interactive check
python auditor.py --interactive --format txt
```

Sample Output (JSON excerpt)

```
{
  "system": "Payment-Portal",
  "target_levels": {"IAL": 2, "AAL": 2, "FAL": 2},
  "findings": [
    {
      "control": "PasswordPolicy.MinLength",
      "status": "PASS",
```

```
      "observed": 12,
      "expected": ">= 8 (higher recommended for AAL2+)",
      "rationale": "Meets minimum; aligns with strong password guidance."
    },
    {
      "control": "MFA.RequiredForAllUsers",
      "status": "FAIL",
      "observed": false,
      "expected": true,
      "remediation": "Enforce MFA for all users or all interactive sign-ins via Conditional Acc
    }
  ],
  "summary": {
    "pass": 25,
    "warn": 6,
    "fail": 3,
    "risk_rating": "Medium"
  }
}
```

Appendix A — Example .env

```
ENTRA_TENANT_ID=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
ENTRA_CLIENT_ID=yyyyyyyy-yyyy-yyyy-yyyy-yyyyyyyyyyyy
ENTRA_CLIENT_SECRET=your-secret
HTTP_PROXY=
HTTPS_PROXY=
NO_PROXY=localhost,127.0.0.1
```

Load it with:

```
export $(grep -v '^#' .env | xargs)
```

Appendix B — Example CI Usage (GitHub Actions)

```
name: NIST Auditor
on: [workflow_dispatch]
jobs:
  run-audit:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with:
          python-version: '3.11'
      - run: pip install -r requirements.txt
      - run: |
```

```
        python auditor.py --entra --format json --out reports
      env:
        ENTRA_TENANT_ID: ${{ secrets.ENTRA_TENANT_ID }}
        ENTRA_CLIENT_ID: ${{ secrets.ENTRA_CLIENT_ID }}
        ENTRA_CLIENT_SECRET: ${{ secrets.ENTRA_CLIENT_SECRET }}
    - uses: actions/upload-artifact@v4
      with:
        name: nist-auditor-reports
        path: reports
```
