/firebase-apk-scanner

Source: `~/.claude/skills/tob-firebase-apk-scanner/skills/firebase-apk-scanner/SKILL.md`

---

name: firebase-apk-scanner
description: Scans Android APKs for Firebase security misconfigurations including open databases, storage buckets, authentication issues, and exposed cloud functions. Use when analyzing APK files for Firebase vulnerabilities, performing mobile app security audits, or testing Firebase endpoint security. For authorized security research only. argument-hint: [apk-file-or-directory] allowed-tools:

Bash({baseDir}/scanner.sh:*), *Bash(apktool:*), Bash(curl:*), Read, Grep, Glob disable-model-invocation: true

Firebase APK Security Scanner

You are a Firebase security analyst. When this skill is invoked, scan the provided APK(s) for Firebase misconfigurations and report findings.

When to Use

- Auditing Android applications for Firebase security misconfigurations
- Testing Firebase endpoints extracted from APKs (Realtime Database, Firestore, Storage)
- Checking authentication security (open signup, anonymous auth, email enumeration)
- Enumerating Cloud Functions and testing for unauthenticated access
- Mobile app security assessments involving Firebase backends
- Authorized penetration testing of Firebase-backed applications

When NOT to Use

- Scanning apps you do not have explicit authorization to test
- Testing production Firebase projects without written permission
- When you only need to extract Firebase config without testing (use manual grep/strings instead)
- For non-Android targets (iOS, web apps) - this skill is APK-specific
- When the target app does not use Firebase

Rationalizations to Reject

When auditing, reject these common rationalizations that lead to missed or downplayed findings:

- **"The database is read-only so it's fine"** - Data exposure is still a critical finding; PII, API keys, and business data may be leaked
- **"It's just anonymous auth, not real accounts"** - Anonymous tokens bypass `auth != null` rules and can access "authenticated-only" resources
- **"The API key is public anyway"** - A public API key does not justify open database rules or disabled auth restrictions
- **"There's no sensitive data in there"** - You cannot know what data will be stored in the future; insecure rules are vulnerabilities regardless of current content
- **"It's an internal app"** - APKs can be extracted from any device; "internal" apps are not protected from reverse engineering
- **"We'll fix it before launch"** - Document the finding; pre-launch vulnerabilities frequently ship to production

Reference Documentation

For detailed vulnerability patterns and exploitation techniques, consult:

- Vulnerability Patterns Reference

How to Use This Skill

The user will provide an APK file or directory: `$ARGUMENTS`

Workflow

Step 1: Validate Input

First, verify the target exists:

```
ls -la $ARGUMENTS
```

If `$ARGUMENTS` is empty, ask the user to provide an APK path.

Step 2: Run the Scanner

Execute the bundled scanner script on the target:

```
{baseDir}/scanner.sh $ARGUMENTS
```

The scanner will:

1. Decompile the APK using apktool
2. Extract Firebase configuration from all sources (google-services.json, XML resources, assets, smali code, DEX strings)
3. Test authentication endpoints (open signup, anonymous auth, email enumeration)
4. Test Realtime Database (unauthenticated read/write, auth bypass)
5. Test Firestore (document access, collection enumeration)
6. Test Storage buckets (listing, write access)
7. Test Cloud Functions (enumeration, unauthenticated access)
8. Test Remote Config exposure
9. Generate reports in text and JSON format

Step 3: Present Results

After the scanner completes, read and summarize the results:

```
cat firebase_scan_*/scan_report.txt
```

Present findings in this format:

Scan Summary

Metric	Value
APKs Scanned	X
Vulnerable	X
Total Issues	X

Extracted Configuration

Field	Value
Project ID	`extracted_value`
Database URL	`extracted_value`
Storage Bucket	`extracted_value`
API Key	`extracted_value`
Auth Domain	`extracted_value`

Vulnerabilities Found

Severity	Issue	Evidence
CRITICAL	Description	Brief evidence
HIGH	Description	Brief evidence

Remediation

Provide specific fixes for each vulnerability found. Reference the Vulnerability Patterns for secure code examples.

Manual Testing (If Scanner Fails)

If the scanner script is unavailable or fails, perform manual extraction and testing:

Extract Configuration

Search for Firebase config in decompiled APK:

```
# Decompile
apktool d -f -o ./decompiled $ARGUMENTS

# Find google-services.json
find ./decompiled -name "google-services.json"

# Search XML resources
grep -r "firebaseio.com\|appspot.com\|AIza" ./decompiled/res/

# Search assets (hybrid apps)
grep -r "firebaseio.com\|AIza" ./decompiled/assets/
```

Test Endpoints

Once you have the PROJECT_ID and API_KEY:

Authentication:

```
# Test open signup
curl -s -X POST -H "Content-Type: application/json" \
  -d '{"email":"test@test.com","password":"Test123!","returnSecureToken":true}' \
  "https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=API_KEY"

# Test anonymous auth
curl -s -X POST -H "Content-Type: application/json" \
  -d '{"returnSecureToken":true}' \
  "https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=API_KEY"
```

Database:

```
# Realtime Database read
curl -s "https://PROJECT_ID.firebaseio.com/.json"

# Firestore read
curl -s
"https://firestore.googleapis.com/v1/projects/PROJECT_ID/databases/(default)/documents"
```

Storage:

```
# List bucket
curl -s "https://firebasestorage.googleapis.com/v0/b/PROJECT_ID.appspot.com/o"
```

Remote Config:

```
curl -s -H "x-goog-api-key: API_KEY" \
  "https://firebaseremoteconfig.googleapis.com/v1/projects/PROJECT_ID/remoteConfig"
```

Severity Classification

- **CRITICAL**: Unauthenticated database read/write, storage write, open signup on private apps
- **HIGH**: Anonymous auth enabled, storage bucket listing, collection enumeration
- **MEDIUM**: Email enumeration, accessible cloud functions, remote config exposure
- **LOW**: Information disclosure without sensitive data

Important Guidelines

1. **Authorization required** - Only scan APKs you have permission to test

2. **Clean up test data** - The scanner automatically removes test entries it creates
3. **Save tokens** - If anonymous auth succeeds, use the token for authenticated bypass testing
4. **Test all regions** - Cloud Functions may be deployed to us-central1, europe-west1, asia-east1, etc.
5. **Multiple instances** - Some apps use multiple Firebase projects; test all discovered configurations

---

Revision #4
Created 2026-02-18 08:40:06 UTC by John
Updated 2026-05-31 20:01:50 UTC by John