Contents

[3.6] Unity Principle Exploitation

1. Operational Definition: The tendency to favor requests from individuals perceived to be part of one's "in-group" (e.g., same team, company, or background), leading to relaxed security scrutiny based on a false sense of shared identity.

2. Main Metric & Algorithm:

- **Metric: In-Group Preferential Treatment Index (IPTI)**. Formula: `IPTI = (N_bypass_in_group / N_requests_in_group) / (N_bypass_out_group / N_requests_out_group)`.
- **Pseudocode:**

  python

```
def calculate_ipti(access_logs, request_logs, hr_data):
    """
    Compares bypass rates for in-group vs. out-group requests.
    """
    # 1. Define in-group (e.g., same department) and out-group
    user_departments = get_department_mapping(hr_data)

    in_group_requests = 0; in_group_bypass = 0
    out_group_requests = 0; out_group_bypass = 0

    for request in request_logs:
        requester = request.requester_id
        grantor = request.grantor_id
        # Check if requester and grantor are in the same in-group (department)
        if user_departments[requester] == user_departments[grantor]:
            in_group_requests += 1
            if request.was_bypassed: # e.g., no ticket, policy exception
                in_group_bypass += 1
        else:
            out_group_requests += 1
            if request.was_bypassed:
                out_group_bypass += 1

    bypass_rate_in = in_group_bypass / in_group_requests if in_group_requests > 0 else 0
    bypass_rate_out = out_group_bypass / out_group_requests if out_group_requests > 0 else

    IPTI = bypass_rate_in / bypass_rate_out if bypass_rate_out > 0 else float('inf')
    return IPTI
```

- **Alert Threshold:** `IPTI > 2.0` (Bypass rate for in-group requests is twice that of out-group requests).

3. Digital Data Sources (Algorithm Input):

- **HR Database API:** To determine group memberships (department, team, location). Fields: `user_id`, `department_id`, `team_id`.
- **Access Request Logs (ServiceNow, Jira, custom IAM):** To get a record of requests and their approval status. Fields: `requester`, `grantor`, `approval_status`, `timestamp`, `policy_exception_flag`.
- **Access Logs (as above):** To infer bypasses not captured in request logs.

4. Human-to-Human Audit Protocol: Present auditors with a series of hypothetical access request scenarios from "in-group" and "out-group" requesters. Measure the difference in the perceived legitimacy of the request and the likelihood of granting it. Alternatively, analyze historical access approval tickets for correlation between grantor/requester department and the speed of approval.

5. Recommended Mitigation Actions:

- **Technical/Digital Mitigation:** Implement attribute-based access control (ABAC) where possible, automating decisions based on user attributes and resource sensitivity, removing human in-group bias from the equation.
- **Human/Organizational Mitigation:** Conduct training on in-group bias, making teams aware of the tendency and encouraging them to apply the same security standards to everyone, regardless of affiliation.
- **Process Mitigation:** Blind the grantor to certain requester attributes (like department) during the approval process for certain low-risk requests, forcing a focus on the request's merit.