# Research: Decoy Ideas — Deception Playbook

**Author:** Ahmed Emad Eldeen  
**Purpose:** A practical, prioritized research brief to inspire deception (decoy) controls against insider threats and internal attackers. This document contains decoy classifications, concrete trap ideas, deployment guidance, SIEM/IR playbook suggestions, risks and legal considerations, KPIs, and a 30/60/90-day MVP plan.

## 1) Decoy Classification (Quick)

* **Honeyfiles / Honeydocuments:** Fake documents (PDF, Word, Excel) that contain enticing content (payroll, contracts, credentials). Monitor access, downloads, and copies.
* **Honeytokens (Canary tokens):** Tokens that phone home when used — examples: clickable URLs, DNS requests, fake API keys, fake cloud credentials, fake email addresses.
* **Decoy Accounts / Fake Identities:** Service or user accounts created with believable names but not used in reality. Any authentication attempts are suspicious.
* **Honeypots / Decoy Hosts & Services:** Emulated hosts/services (SSH, database, file shares) that look like valuable assets but contain no real data.
* **Database Traps / Fake DB Rows:** Poisoned rows or tables containing attractive fake PII or secrets; any read/export triggers alerting.
* **Application-Level Decoys:** Fake pages or API endpoints inside internal apps, wikis, or SharePoint containing enticing links or resources.

## 2) Why Organizations Use Deception

* **Early detection of insiders:** Internal attackers or compromised accounts often access internal resources; decoys detect lateral movement and data snooping quickly.
* **Enriched forensics:** Decoys provide timelines, attacker IPs, user-agents, and command sequences that improve incident investigations.
* **Diversion & damage limitation:** Decoys distract attackers away from real assets and buy time for containment and investigation.

## 3) Practical Decoy Ideas (Initial List)

Each entry includes: placement, rationale, and suggested alert/response.

**Payroll\_Q4\_2025.xlsx (Fake Payroll File)**

1. **Payroll\_Q4\_2025.xlsx (Fake Payroll File)**
   * Placement: \\Shared\Finance\Reports\2025\
   * Rationale: Attractive to attackers looking for financial info.
   * Alert / Response: File open/download/copy → immediate SOC notification + isolate endpoint or user session.
   * Implementation tip: Embed a canary token (e.g., remote image or hosted link) inside the document to detect access.
2. **Honeypot DB row with ssn = '999-99-9999'**
   * Placement: Read-only DB replica or a dedicated table visible to some roles.
   * Rationale: Any SELECT on that row/table indicates unauthorized data discovery/export.
   * Alert / Response: SELECT on the row → SIEM event, collect session details and origin.
   * Implementation tip: Mark the row as fake in metadata but keep it indistinguishable for queries.
3. **Fake AWS key inside an internal repo**
   * Placement: Internal Git repo or build config file in a plausible path.
   * Rationale: Detects attempts to use leaked keys or automated scanning for credentials.
   * Alert / Response: Any attempt to use the key or access the token endpoint → webhook trigger + escalate.
   * Implementation tip: Route the key use to a harmless canary endpoint that logs and alerts.
4. **Decoy AD account svc\_finance\_backup**
   * Placement: Active Directory / Azure AD with zero real privileges and disabled for normal use.
   * Rationale: Authentication attempts show reconnaissance or credential stuffing.
   * Alert / Response: Any login → disable account, capture authentication source, open IR ticket.
   * Implementation tip: Configure account to require MFA (but do not enable real access) and log all auth attempts.
5. **Internal Wiki page 'Executive passwords' with a clickable canary URL**
   * Placement: Confluence / SharePoint page styled like a private doc.
   * Rationale: Employees or attackers searching for sensitive data may click links.
   * Alert / Response: Click triggers webhook → SIEM event and immediate enrichment (IP, UA).
   * Implementation tip: Use a short, unique tracking URL that returns a 404 to avoid accidental use.
6. **Network honeypot (SSH service imitating a DB host)**
   * Placement: Segmented VLAN where it can be reached from internal networks but no production data.
   * Rationale: Captures lateral movement, commands, and tools used by attackers.
   * Alert / Response: Connection attempts → full session logging, network capture, and automated analysis.
   * Implementation tip: Ensure honeypot has no connectivity to production systems and is monitored separately.

**New / Additional Decoy Types**

1. **Email honeytoken (fake invoice or HR request)**
   * Placement: Hosted internally or sent to a small test group as a controlled bait email.
   * Rationale: Detects data exfiltration via email or attackers looking for attachments/links.
   * Alert / Response: Attachment open or link click → SIEM + email trace + notify owner + escalate if external forwarding observed.
   * Implementation tip: Host the tracked asset on a unique domain to capture referrer and IP.
2. **Fake S3 bucket / cloud storage object (company-backups-prod)**
   * Placement: Cloud console with object listing visible to some roles, or public-sounding bucket name in code comments.
   * Rationale: Detects scanning/enumeration and attempts to access cloud storage.
   * Alert / Response: List/get attempts → CloudTrail/S3 access alert + block source IP and rotate associated keys.
   * Implementation tip: Make object metadata enticing but ensure contents are fake and non-sensitive.
3. **Fake API admin endpoint (/api/v1/admin/override)**
   * Placement: API gateway or reverse proxy configuration in staging that appears in route lists.
   * Rationale: Finds attackers probing APIs for admin functionality or privilege escalation endpoints.
   * Alert / Response: Any call → log full payload and headers, generate high-priority alert, add automated WAF rules.
   * Implementation tip: Ensure endpoint is read-only and returns informative-but-harmless errors while logging details.
4. **Phantom SaaS admin console (fake SSO app entry)**
   * Placement: Intranet apps list or SSO application catalog displaying a fake "Admin Console".
   * Rationale: Attracts attackers or curious insiders trying to access admin interfaces.
   * Alert / Response: Login attempts or password resets → capture credential attempts and source metadata; open IR ticket.
   * Implementation tip: Integrate the fake console with canary links rather than legitimate services.
5. **Fake CI/CD secret / token in pipeline variables**
   * Placement: Visible in CI config or pipeline definition in a test repo.
   * Rationale: Detects attempts to harvest secrets or abuse CI runners.
   * Alert / Response: Usage or retrieval attempts → pause the pipeline, collect evidence, alert SecOps.
   * Implementation tip: Have the token call a logging-only service to capture requester details.
6. **Fake package name in requirements (npm/pypi decoy)**
   * Placement: package.json/requirements.txt referencing a believable-but-fake package.
   * Rationale: Detects dependency scanning, typosquatting attempts, or attempts to fetch external libs.
   * Alert / Response: Registry lookups or package fetch attempts → webhook and registry logs capture requester.
   * Implementation tip: Host the decoy package on a private registry with webhooks for telemetry.
7. **Deceptive log entries with fake secrets (SIEM honeytokens)**
   * Placement: Application logs, syslog, or centralized log store entries that include fake API keys or PII.
   * Rationale: Reveals actors searching logs or exfiltrating log stores.
   * Alert / Response: Read/export of logs containing honeytoken → trigger SIEM alert and investigate access patterns.
   * Implementation tip: Place tokens in logs that appear in search results (common keywords) to increase chances of discovery.
8. **Fake internal support ticket with attachment link**
   * Placement: Helpdesk/ticketing system (ticket visible to appropriate teams).
   * Rationale: Threat actors or insiders who scrape tickets for useful artifacts will follow links.
   * Alert / Response: Attachment open or link click → ticketing system webhook + SOC ticket creation.
   * Implementation tip: Mark the ticket as low-priority but ensure the link is unique and trackable.
9. **Decoy network share with tempting file names (Salary\_Adjustments.xlsx)**
   * Placement: SMB/NetBIOS share that appears in network discovery lists.
   * Rationale: Detects internal scanning and lateral movement looking for shares.
   * Alert / Response: File access/listing → endpoint and network telemetry captured; isolate machine if suspicious.
   * Implementation tip: Place read-only decoy files and instrument them with remote resources to track access.
10. **Fake privileged IAM role in cloud (prod-db-admin)**
    * Placement: Cloud IAM console as a dormant role with no real privileges but discoverable.
    * Rationale: Identifies attempts to enumerate or assume high-privilege roles in the cloud.
    * Alert / Response: AssumeRole or call attempts → CloudTrail alert, revoke suspicious keys, and notify cloud ops.
    * Implementation tip: Attach an inline policy that only allows calling a canary endpoint (for logging) and nothing else.
11. **Honeypot printer / IoT device on network**
    * Placement: Simulated printer, NAS, or IoT device with common ports open on internal network.
    * Rationale: Detects scanning and device-targeted attacks used for lateral movement.
    * Alert / Response: Access attempts → network capture + device session logging + block origin.
    * Implementation tip: Make banner strings and device names realistic to improve believability.
12. **Fake code comment with "password: Admin123!" (honeystring)**
    * Placement: Comments inside code snippets in internal repos or docs.
    * Rationale: Attracts attackers searching repos for hard-coded secrets.
    * Alert / Response: Repository clones/searches that return hits on the honeystring → alert + collect git clone metadata.
    * Implementation tip: Use a unique, easily searchable string and monitor repo search logs.
13. **Decoy user profile / org-chart entry with contact links**
    * Placement: Internal org-chart, LinkedIn-style internal pages, or departmental pages.
    * Rationale: Social-engineering bait to detect attempts to contact fake privileged users.
    * Alert / Response: Messages, password-reset requests, or calendar invites to the fake profile → SOC/HR notified and interaction logged.
    * Implementation tip: Use an alias account that forwards nothing externally but logs inbound messages.
14. **Fake metadata access point in cloud instances (imitated IMDS call)**
    * Placement: A controlled endpoint that mimics instance metadata behavior for a decoy instance.
    * Rationale: Detects attempts to abuse instance metadata service for credential harvesting.
    * Alert / Response: Any call to the decoy metadata → high-priority alert and forensic snapshot of the instance.
    * Implementation tip: Ensure decoy metadata never provides real credentials — only an instrumented canary response.
15. **Deceptive database schema names (creditcards\_staging)**
    * Placement: DB admin panels, schema listings, or migration files visible in test environments.
    * Rationale: Attracts curiosity-based discovery and SQL-oriented attackers.
    * Alert / Response: Schema listing or query activity → audit the origin and collect session logs.
    * Implementation tip: Use realistic column names but ensure no real PII exists.
16. **Fake vulnerability scan results / bug tracker entry**
    * Placement: Internal vuln tracker or ticketing system showing “unpatched critical” entries.
    * Rationale: Lures attackers to attempt exploitation of an apparent critical hole.
    * Alert / Response: Exploitation attempts → trap, capture exploit payloads, and raise IR.
    * Implementation tip: Keep the entry suspiciously vague so admins don’t act on it accidentally

## 4) Deception Design & Deployment Best Practices

* **Blend-in (make decoys believable):** Use real naming conventions, realistic timestamps, file metadata, and permission models consistent with the environment.
* **Placement & access control:** Put decoys where they are plausible but should not be routinely accessed by normal users to reduce noise.
* **Comprehensive logging:** Capture as much telemetry as possible (file access, process execution, network connections, session recordings) to support forensic analysis.
* **SIEM & IR integration:** Link decoy alerts to high-priority SIEM rules and automated playbooks (disable accounts, isolate endpoints).
* **Routine refresh & anti-fingerprinting:** Rotate and update decoys regularly so advanced attackers can’t easily fingerprint and avoid them.
* **False-positive tuning:** Start small and expand; measure noise and adjust thresholds to keep team trust high.

## 5) Suggested SIEM Rules & IR Playbook Samples (Templates)

### Example SIEM Rules

* **Rule 1 — High priority:** Access to Payroll\_Q4\_2025.xlsx outside business hours from internal network.  
  Action: P1 alert, auto-assign to SOC analyst, capture endpoint and user context, isolate endpoint if criteria met.
* **Rule 2 — Critical:** Successful authentication to decoy AD account.  
  Action: Immediately disable account, capture session logs, create incident in IR platform.
* **Rule 3 — Medium:** Outbound DNS request to canary domain.  
  Action: Tag incident, enrich with external IP reputation and context, block domain if malicious.

### Example IR Playbook (On decoy activation)

* Triage: Verify decoy activation and check for false positive.
* Enrichment: Collect endpoint ID, process tree, user context, network origin, and timeline.
* Containment: Temporarily isolate affected endpoint(s) and disable implicated accounts.
* Investigation: Retrieve session recordings, packet captures, and correlate with other telemetry.
* Remediation: Remove attacker persistence, rotate credentials if needed, patch access controls.
* Post-incident: Update deception artifacts and SIEM thresholds; document lessons learned.

## 6) Risks, Legal & Ethical Considerations

* **Employee privacy:** Ensure deception usage complies with local labor laws, contracts, and privacy policies. Avoid invasive monitoring beyond what is legally permitted.
* **False positives & erosion of trust:** Poorly designed decoys produce noisy alerts. Start small, tune, and communicate to stakeholders where appropriate.
* **Detection by attackers:** Sophisticated attackers might fingerprint decoys; maintain and vary decoys over time.
* **Operational safety:** Ensure decoys cannot be used as stepping stones to attack real assets (isolate them properly).

## 7) KPIs to Measure Effectiveness

* Mean Time To Detect (MTTD) for decoy-triggered alerts.
* Number of incidents first detected by deception per quarter.
* False positive rate of deception alerts.
* Quality of intelligence gained (actionable IPs, TTPs, IOCs used to strengthen defenses).

## 8) 30 / 60 / 90-Day MVP Implementation Plan

* **Week 1–2 (MVP):** Deploy 3 simple canary tokens: a fake payroll file, a canary URL embedded in an internal wiki, and a fake cloud API key in a test repo.  
  Wire token webhooks to SIEM or a SOC notification channel.  
  Create simple SIEM rules and a basic IR checklist.
* **Month 1:** Add a decoy AD account and a honeypot DB row.  
  Configure automated playbook actions (disable accounts, collect logs).  
  Run false-positive tuning exercises.
* **Month 2–3:** Deploy honeypot hosts/services to capture lateral movement activity.  
  Build dashboards and KPIs, refine playbooks, and run tabletop or red-team exercises against the decoys.

## 9) Quick Naming & Content Templates (Ready to use)

* Honeyfile names: Payroll\_Q4\_2025.xlsx, Executive\_Compensation\_2025.pdf, Customer\_DB\_Export\_2025.csv
* Decoy AD names: svc\_finance\_backup, archiver\_service\_01
* Fake DB row example: { email: 'ceo\_sensitive@example.com', ssn: '999-99-9999', tag: 'honey' }
* Tip: Match names and metadata with your organization’s conventions for better stealth.

## 10) Recommended Tools & Services

* Canarytokens (Thinkst) — quick canary token generation for files, URLs, DNS, and cloud keys.
* Commercial deception platforms (if you need scale): consider solutions that provide automated deployment and analytics.
* SIEM integration: Ensure your SIEM (or cloud-native SIEM) receives and prioritizes decoy alerts, with automated enrichment and incident creation.

End of document.