MACRO PRAXIS | Research Institute

The Hidden Costs & Risks of “Free” Deception Tools

RESEARCH PAPER TITLE: Why DIY honeytokens, roll-your-own decoys, and public-LLM-named traps underperform in the enterprise—and how to evaluate an “enterprise-grade” deception strategy

ANALYSIS AUTHOR: Bobby Boughton, MacroPraxis Research Institute Fellow
PUBLICATION DATE: July 21, 2025

Abstract

Organizations under budget pressure often begin their deception journey with free tools or small, roll-your-own experiments. These approaches can be useful in training labs or limited pilots, yet our analysis shows that they rarely scale to enterprise environments without imposing significant hidden costs and material coverage gaps along real attacker paths. We examine where and why destination-centric traps (stand-alone decoys and simple tokens) tend to alert late; we analyze operational fragility in environments with endpoint churn and cloud drift; we discuss governance risks introduced by public-model–assisted decoy naming; and we quantify total cost of ownership across three enterprise size bands. Taken together, the evidence suggests that “free” deception frequently costs more than expected while delivering a lower probability of early, reliable detection (especially with the advent of AI-assisted adversaries). We conclude with a practical evaluation rubric and a data-driven decision framework for leaders.

Key Takeaways

1. **Coverage on attacker paths beats destinations.** Early, reliable detection comes from breadcrumbs and honeytokens spread across endpoints, identities, and cloud paths—not from a handful of destination traps.
2. **DIY looks cheap but scales poorly.** Manual design and rotation create operational debt; at enterprise scale, labor alone can exceed a platform subscription while delivering later alerts.
3. **Automation is the unlock.** EDR/XDR-assisted deploy and refresh compress per-asset touch time, enable consistent rotation, and reduce stale-lure risk.
4. **Public-model naming adds risk, not certainty.** LLM-styled labels can become class-predictable and raise governance questions; name realism is not a substitute for path coverage and rotation.
5. **Measure outcomes, not artifacts.** Track time-to-trip, alert precision, coverage, rotation health, and indistinguishability; report quarterly and scale what empirically improves these metrics.

1. Introduction

The promise of cyber deception is compelling: plant believable tripwires so that intrusion attempts are revealed early, with high signal quality, and routed quickly into investigation and response. In practice, however, many programs start with a small number of isolated lures—perhaps a tokenized document, a honeypot VM, or a cloud key left in a repository—and then declare the strategy complete. This tendency to **“check the box”** creates a **structural mismatch between AI-assisted attacker behavior and defender coverage.** Adversaries traverse endpoints, identities, and cloud resources in chains; destination-centric traps that merely wait at the end of those chains frequently fire too late to prevent meaningful impact.

2. Path-Centric vs. Destination-Centric Deception

Real attackers advance through graphs: they enumerate identities, harvest and reuse credentials, laterally move across endpoints, and touch cloud services on the way to data stores and business systems. Deception that primarily watches the destination—e.g., a single bucket token or a conspicuous honeypot—may capture opportunistic activity, but it leaves the lateral movement phase comparatively uninstrumented. By contrast, path-centric deception distributes breadcrumbs on endpoints and identities (for example, mapped-drive artifacts, RDP/SSH/DB credentials, and service-account honeytokens), and augments cloud paths with keys and URLs that are consistent with an environment’s own policies. When an adversary encounters such lures during routine discovery or credential replay, the signal is both earlier and cleaner. The figure below visually illustrates this tradeoff.

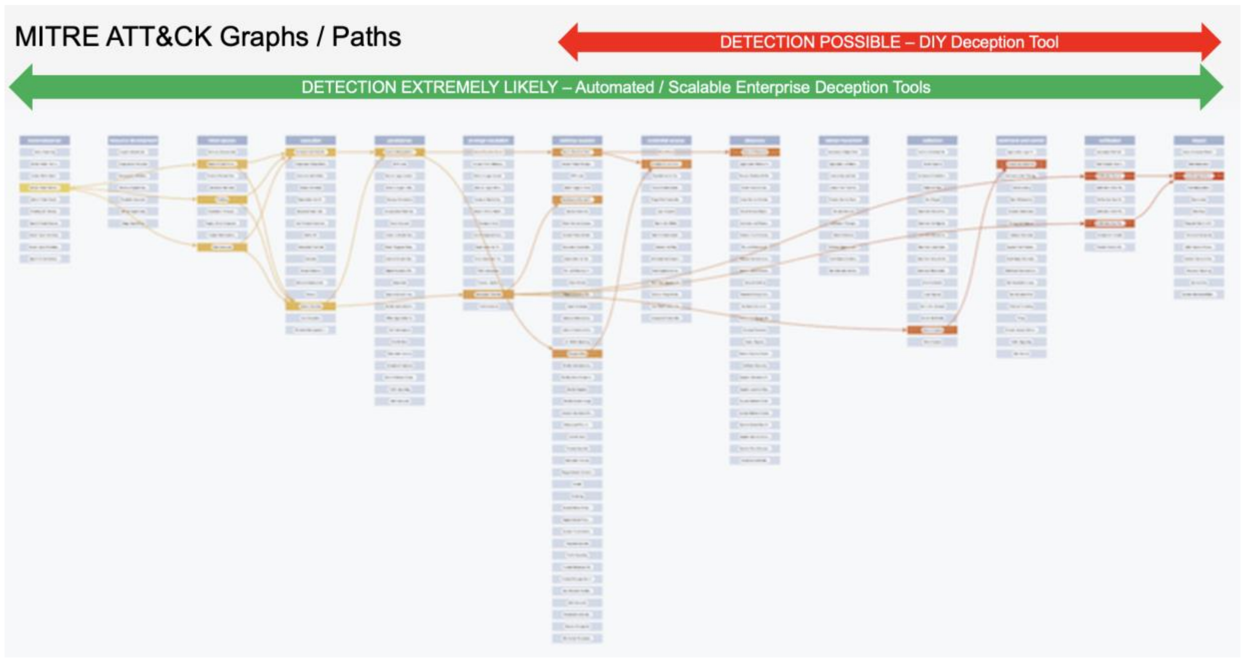
MITRE ATT&CK Graphs / Paths
DETECTION POSSIBLE – DIY Deception Tool
DETECTION EXTREMELY LIKELY – Automated / Scalable Enterprise Deception Tools

Figure 1 — MITRE ATT&CK paths: why scalable, path-centric deception trips earlier

This diagram maps an attacker’s progress across MITRE ATT&CK (left→right) and shows where different deception strategies typically trigger. The green bar reflects automated, path-centric deception: scalable breadcrumbs on endpoints, identity honeytokens, and cloud-path lures that make early detections likely during discovery, credential access, and lateral movement. The red bar reflects DIY/destination-heavy approaches: while a DIY program may plant some breadcrumbs, they’re usually few, hard to maintain at scale, and more easily identifiable, so they seldom trip early and tend to alert later near collection/exfiltration/impact. The takeaway: instrument the paths broadly and automatically—not just a handful of destinations—if you want time to contain.

3. Operational Fragility of DIY Programs

DIY deception efforts often rely on manual placement and periodic refresh of lures. In environments with thousands of endpoints and frequent image updates, this manual cadence struggles to keep pace. Breadcrumbs decay as machines are reimaged or renamed; honeypots drift out of configuration; cloud lures lose alignment with changing IAM and networking policies. Even when teams initially plan for regular rotation, the cycle time elongates as other priorities compete for staff attention. Over time, the apparent low price of **initially "free" tools (they introduce upsell licensing costs once low thresholds are reached)** is offset by substantial labor hours that are hard to sustain—and by the risk that stale, inconsistent lures fail to trip at all.

4. Public-Model Naming and Governance Considerations

Some modern token workflows use public large language models (LLMs) to suggest decoy names that blend with a customer's existing assets. While this can accelerate design, it introduces two concerns. First, public models tend to generate plausible but typical names, which means a large family of such decoys may share latent stylistic artifacts. Sophisticated adversaries can adopt filtering heuristics that reduce interactions with those families, particularly when traps sit only at destinations. Second, when name generation relies on a public model, inventory metadata leaves the tenant during design time; many organizations treat this as a **governance and vendor-risk question** that must be addressed explicitly.

5. Total Cost of Ownership (TCO): A Quantitative View

To make the trade-offs concrete, we model annualized costs for DIY deception versus an enterprise-grade platform that automates deployment and rotation via existing EDR/XDR tooling. We include labor (fully-loaded security engineering time) and, for the enterprise option, a representative software subscription. Our intent is not to price any specific vendor but to capture realistic orders of magnitude that decision-makers can adapt.

5.1 Modeling Approach

Let:

- **N_e** = number of endpoints and servers instrumented with breadcrumbs.
- **N_c** = number of cloud services/projects receiving path lures.
- **R** = monthly rotation frequency.
- **t_d** = hours required to design and place a lure.
- **t_r** = hours required to rotate a lure.
- **C_FTE** = fully-loaded hourly rate for a security engineer.

The **DIY annual labor** cost is:

$$C\_DIY{,}labor = (N_e + N_c) \times (t\_d + (12R \times t\_r)) \times C\_FTE$$

We then add an annual operations constant (C_ops) for playbooks, QA, and integration upkeep. For the enterprise option, we assume automation reduces per-asset hours substantially and include a platform subscription (C_tool).

5.2 Assumptions by Enterprise Size

We define three size bands and fix explicit assumptions to avoid ambiguity. These can be tuned per reader.

- **Small Enterprise:** N_e=900, N_c=30; R=0.5 for DIY, R=1.0 for enterprise; C_FTE=$120/hr.
- **Medium Enterprise:** N_e=5,000, N_c=120; same C_FTE.
- **Large Enterprise:** N_e=25,000, N_c=300; same C_FTE.

DIY: t_d=0.50 hr, t_r=0.25 hr, limited ops support.
Enterprise: t_d=0.05 hr, t_r=0.02 hr, more frequent rotation, plus subscription and smaller ops constant.

- **Small Enterprise**: (N_e=900) assets (workstations + servers), (N_c=30) cloud services; rotation (R=0.5) (every two months) for DIY, (R=1.0) (monthly) for enterprise. Fully-loaded rate (C_{FTE}=$120/hr).
  - DIY effort: (t_d=0.50) hr, (t_r=0.25) hr.
  - Enterprise effort: (t_d=0.05) hr, (t_r=0.02) hr; subscription (C_{tool}=$75,000/yr).
  - Ops constants: (C_{ops}=$36,000) (DIY), $12,000 (enterprise).
- **Medium Enterprise**: (N_e=5,000), (N_c=120); same (C_{FTE}=$120/hr).
  - DIY effort: (t_d=0.50) hr, (t_r=0.25) hr, (R=0.5).
  - Enterprise effort: (t_d=0.05) hr, (t_r=0.02) hr, (R=1.0); subscription (C_{tool}=$225,000/yr).
  - Ops constants: (C_{ops}=$96,000) (DIY), $36,000 (enterprise).
- **Large Enterprise**: (N_e=25,000), (N_c=300); same (C_{FTE}).
  - DIY effort: (t_d=0.50) hr, (t_r=0.25) hr, (R=0.5).
  - Enterprise effort: (t_d=0.05) hr, (t_r=0.02) hr, (R=1.0); subscription (C_{tool}=$600,000/yr).
  - Ops constants: (C_{ops}=$240,000) (DIY), $90,000 (enterprise).

Rationale: The DIY parameters reflect manual design and limited rotation sustained by small teams; the enterprise parameters reflect EDR/XDR-assisted rollout and health monitoring that compress per-asset touch time while enabling more frequent rotation.

5.3 Results

Using the assumptions above, we compute annual labor hours and costs for both approaches. The DIY approach shows rapidly escalating labor at scale; the enterprise approach trades a subscription for much smaller, sustainable touch time.

5.3.1 Calculations (per band)

- **Small ((N_e+N_c=930))**
  DIY hours: (930 x (0.50 + 6 imes0.25) = 930 x 2.00 = 1,860) → $223,200; DIY total with ops: **$259,200**.
  Enterprise hours: (930 x (0.05 + 12 x 0.02) = 930 x 0.29 = 269.7) → $32,364; plus $75,000 tool and $12,000 ops → **$119,364**.

- **Medium ((N_e+N_c=5,120))**
  DIY hours: (5,120 x 2.00 = 10,240) → $1,228,800; DIY total with ops: **$1,324,800**.

Enterprise hours: (5,120 x 0.29 = 1,484.8) → $178,176; plus $225,000 tool and $36,000 ops → **$439,176**.

- **Large ((N_e+N_c=25,300))**
  DIY hours: (25,300 x 2.00 = 50,600) → $6,072,000; DIY total with ops: **$6,312,000**.
  Enterprise hours: (25,300 x 0.29 = 7,337) → $880,440; plus $600,000 tool and $90,000 ops → **$1,570,440**.

5.4 TCO Summary Table

Enterprise Size	Assets in Scope ((N_e+N_c))	DIY Hours	DIY Total Cost	Enterprise Hours	Enterprise Tool Cost	Enterprise Ops	Enterprise Total	Annual Savings
Small	930	1,860	$259,200	270	$75,000	$12,000	$119,364	**$139,836**
Medium	5,120	10,240	$1,324,800	1,485	$225,000	$36,000	$439,176	**$885,624**
Large	25,300	50,600	$6,312,000	7,337	$600,000	$90,000	$1,570,440	**$4,741,560**

Note: Savings exclude the expected-loss reduction from earlier detection; incorporating breach-loss avoidance would increase the enterprise advantage further.

5.5 TCO Graphs

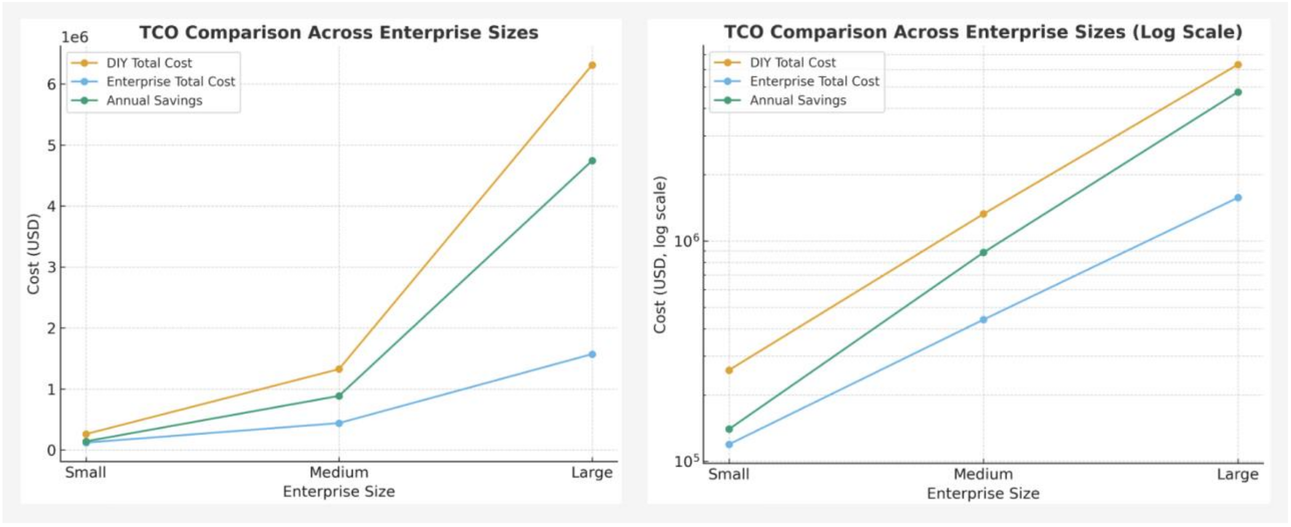

Figure 2 — TCO comparison (linear and log scales)

Side-by-side charts show annual costs for DIY deception (orange), enterprise deception (blue), and resulting annual savings (green) across small, medium, and large enterprises. On the linear plot (left), DIY costs ramp sharply—driven by manual design/rotation labor—while the enterprise total grows more moderately. The log plot (right) highlights the same trend across orders of magnitude: the cost gap and savings widen with scale, indicating that automation/rotation efficiency dominates as environments grow.

6. Measuring What Matters (Program Metrics)

Rather than counting traps, mature programs measure outcomes. Coverage metrics focus on the percentage of endpoints, servers, privileged identities, and cloud entry points instrumented with breadcrumbs and honeytokens. Signal metrics track median time-to-trip from initial access and the precision of deception alerts relative to total alerts. Operational metrics capture rotation health (median lure age, drift rate) and indistinguishability (how well a classifier can separate real vs. decoys; values near chance are preferred). Publishing these figures quarterly builds credibility and helps leadership understand risk reduction in concrete terms.

7. Evaluation Rubric for Enterprise Deception

When evaluating solutions, prioritize controls that instrument attacker paths across endpoints, identity, and cloud; that orchestrate deployment and refresh through your existing endpoint tooling; and that provide health telemetry and API-level integrations into SIEM/SOAR and ticketing. Seek identity deception primitives (service-account honeytokens, SPN lures, and group-based triggers), cloud-path lures with scoped policies, and features that resist fingerprinting. Insist on explicit data-handling disclosures and role-based controls, and require that vendors support measurement: coverage, time-to-trip, precision, rotation health, and indistinguishability tests.

8. Case Snapshots (Anonymized)

- A global retailer that relied on document/API tokens reported frequent late alerts that did not change response outcomes. After instrumenting endpoints and identities with automated breadcrumbs and honeytokens, the median time-to-trip fell from multiple days to hours, and analysts reported a marked improvement in alert quality.
- A regional healthcare provider with a small red-team function observed that a dedicated honeypot was ignored during lateral movement; adding service-account honeytokens produced actionable detections within a single pivot.
- A mid-market financial services firm replaced manual cloud tokens that had drifted out of policy with an automated rotation tied to patch cadence, cutting maintenance hours by more than half while improving consistency.

9. Conclusion

Deception often enters the enterprise as a low-cost experiment—an ad-hoc honeypot, a handful of tokens, a clever document—meant to prove a point or satisfy an audit. At small scale, these artifacts can produce occasional wins. At enterprise scale, however, the physics change. Endpoints churn, identities evolve, and cloud services proliferate. Destination-centric traps become islands, rotation lags, and the operational burden shifts to already overtaxed teams. The result is a control that looks inexpensive on paper but behaves like unfunded technical debt: high touch, brittle, and—most importantly—late to signal along real attack paths.

Our quantitative model highlights why the economics flip at scale. Even with conservative assumptions, manual design and rotation if done correctly would consume thousands to tens of thousands of engineering hours annually in medium to large estates, often surpassing the cost of enterprise platforms that automate deployment and refresh. That differential excludes the more consequential variable: the expected loss associated with late or missed trips. When lures sit only at destinations, adversaries can traverse endpoints and identities largely unobserved; when breadcrumbs and honeytokens instrument those paths, detections occur earlier, with higher precision, and response has room to matter. In practice, shifting signal left yields outsized risk reduction relative to line-item tool spend.

The governance dimension reinforces this conclusion. Public-model-assisted naming may accelerate design, but it introduces data-handling questions and a subtle predictability risk that sophisticated actors can exploit. Sustainable programs minimize exposure of inventory metadata, measure indistinguishability of decoys, and treat deception artifacts as living objects with explicit health, rotation, and coverage targets—not as static, clever names.

Leaders evaluating deception should therefore apply the same standards they use for any Tier-1 control: automate relentlessly, instrument the paths adversaries actually use (endpoints, identities, cloud), integrate with existing EDR/XDR and SOC workflows, and publish program metrics—coverage, time-to-trip, precision, rotation health, indistinguishability—quarterly. **DIY approaches can seed learning, but they should not anchor strategy unless they can prove parity on these measures over time.**

In short: the organizations that win with deception don't "check the box." They operationalize it. **They move detection left, compress manual effort with automation, and hold themselves to measurable outcomes.** Do that, and deception stops being a clever trick and becomes a durable advantage—one that turns lateral movement into your early-warning system and converts would-be breaches into rapid, contained events.

References (selected)

1. IBM Security. *Cost of a Data Breach Report*. Annual editions. IBM, 2020–2024.
2. National Security Agency (NSA) & Partners. *Active Directory: Techniques for Detecting and Mitigating Malicious Activity*. U.S. Government Guidance.
3. Juels, A., & Rivest, R. (2013). *Honeywords: Making Password-Cracking Detectable*. Proceedings of the ACM Conference on Computer and Communications Security.
4. Thinkst Canary. (2025). *Introducing the AWS Infrastructure Canarytoken*. Thinkst Product Blog.
5. Google Threat Intelligence. (2024). *Threat Actor Use of Generative AI*. Google Security Blog.
6. Acalvio Technologies. (2025). Acalvio Recognized by Gartner as a Tech Innovator. Analyst Report summary.
7. Survey articles on cyber deception efficacy and operationalization in enterprise networks. Peer-reviewed journals, 2019–2024.
8. Boughton, B. (2025, Jan 14). *TBT Breach Analysis — First American Financial Corp. (2023 Ransomware Cyberattack)*. MacroPraxis Research Institute.
9. Boughton, B. (2024, Nov 4). *TBT Breach Analysis — Change Healthcare (2024)*. MacroPraxis Research Institute.
10. Boughton, B. (2023). *TBT Breach Analysis — MGM Resorts (2023)*. MacroPraxis Research Institute.
11. Boughton, B. (2020). *TBT Breach Analysis — SolarWinds Supply Chain Attack (2020)*. MacroPraxis Research Institute.
12. Boughton, B. (2022–2024). *TBT Breach Analysis — Salt Typhoon Telco Attack*. MacroPraxis Research Institute.

Disclaimer