

2025

State of Attack Path Management

Identity Risk Through a New Lens

Table of Contents

Introduction	3
Executive Summary	4
Access Graphs vs. Attack Graphs: Seeing Identity Risk Through an Adversary's Eyes	6
Identities at Rest vs. Identities in Transit	11
Detection in Context: Escaping the Tyranny of the Atomic Alert	17
From Theory to Control: The Future of Identity Security	21
Operationalizing Attack Path Management: Turning Visibility into a Complete Practice	40
Updates to Graph Expansion: A Review	44
Trends in Identity Tradecraft from Operations	54
Community Contributions to the BloodHound Ecosystem and the Evolving State of Attack Path Management	67

Introduction

The 2025 State of Attack Path Management report is SpecterOps's inaugural attempt to highlight one of the biggest growing problems that we see facing the security community: Identity Security.

Identity Security impacts organizations big and small and is not something that can just be patched away. This problem has only grown more complex over the last several years, with increased adoption of new technologies, each with their own set of identities and ways of handling privilege. Across all these new developments, there has been little ability to see how all the individual pieces of privilege come together to form what we call an **attack path**. Between sophisticated attacks, leaked credentials, and social engineering, compromised identities are a problem every organization faces, and addressing attack paths is only going to become more important over time.

Identities are an integral part of how every company's network functions. Each user, group, computer, etc. all require some type of identity and access to perform their duties. Further complicating the challenges of securing identities, the management of identities often lives somewhere between the security teams and the IT/operations team. The tug of war between maintaining function and secure implementation is a constant struggle, requiring an understanding of risk acceptance and business requirements to fully address.

SpecterOps is more than just a consulting company or just a security software company, we are a blend of the best aspects of each part. Our unique blend of highly technical offensive security assessments and insights gleaned from both BloodHound Community and BloodHound Enterprise provide us with a complete view of the attack path problem and a detailed understanding of both the impacts attack paths have in any network.

Our intent with the inaugural State of Attack Path Management report is to highlight the problem of attack paths, providing our understanding and perspective of the security concerns and risks it represents for an organization.

SpecterOps was founded on the principles of open-source and transparency, values that we still hold to today. We want to use this report as an opportunity to provide insights we have gathered across our consulting, research, and product efforts to our customers and the security community. Our hope is to help better educate and equip others with an understanding of how to effectively address the challenges of identity security and attack paths.

We won't claim that we can stop breaches from happening or solve all identity problems—that is something which can never be achieved. However, we do think that we can work to help everyone minimize impactful breaches and understand the attack paths that exist before an attacker does.

In this report, we hope to impart the idea of Attack Path Management as a practice. **To gain an edge on the adversary, we must shift the paradigm from one of static identities and defensive checklists to one that embraces adversarial perspectives.**

Executive Summary

Identity Security is a complex and difficult security challenge for every organization. In this report, we introduce Attack Path Management (APM) to explain the significance of the Attack Graph versus Access Graph perspective and the dangers that come from improperly configured identities.

The typical security approach for dealing with identities emphasizes the importance of passwords, multi-factor authentication (MFA), or other authentication tools, with a focus on protecting the wrong person from logging in at all costs. However, what this approach neglects to consider is there are other ways an attacker can move freely through the organization without ever needing to log in. **The focus for Identity Security should extend beyond preventing credential compromise and become a complete understanding of what could happen when (not if) an attacker gains access.**

The identity space is evolving, both with the addition of new technologies as well as the general identity sprawl that occurs as an organization grows and technical debt mounts.

When BloodHound was first released in 2016, it supported a limited number of relationships and only supported Active Directory. As of the release of this report, BloodHound supports a large increase of coverage across Active Directory, as well as including support for Active Directory Certificate Services (ADCS), NTLM Relay, Azure/Entra ID, and hybrid paths. Our intent is to continue to improve the coverage for a more complete graph while working to add support for new technologies like SCCM and Intune.

Once an organization understands the significance of the Attack Graph and the need for an APM program to manage this risk, the next step is to implement an effective program to eliminate attack paths and prevent future paths from emerging. We will discuss some of the challenges organizations face in building an APM program and some best practices we have seen in successful program implementations; ensuring that we are not just identifying problems but remediating them.

APM is more than closing attack paths, it is understanding what causes attack paths to exist in the first place. Understanding the actual implementation of privilege through graphs lets an organization take proactive steps in their identity security, such as helping prevent ransomware spread and better controlling privileged accounts. Understanding how and why attack paths happen in an environment gives credence to actual identity risk and prevents abuse before it has a chance to happen.

When using APM to secure identities, the primary focus is almost always on restricting access to the most highly privileged identities that can pose the most risk. However, it is possible for an attacker to achieve a devastating breach of an organization's critical assets without the need for a high level of control. We introduce "Privilege Zones" which will allow you to better define what important systems and identities should also be tracked as a security boundary. With this added capability, organizations have even more ability to map out attack paths and understand the risks their network would face during an attack.



The risks represented by attack paths in an environment are not just an academic concern but a real problem waiting to happen if not addressed.

Think like an attacker

Attackers don't go through your tools—they go around them, leveraging gaps in your program and exploiting identities to reach your critical assets. Our mission is to demystify tradecraft and stop adversaries in their tracks.



While performing Red Team assessments for SpecterOps customers, we routinely see the impact that identity attack paths have on an organization. We will cover a few interesting “War Stories” of actual attack paths that we encountered in environments over the last year, detailing the security impact that various small configuration decisions can become when combined into an attack path.

Finally, we want to highlight the amazing community that BloodHound brought together, with over 20,000 members in our community Slack.

This year, we celebrated many exciting contributions to the APM space, a few of which we highlight in the report, from community perspectives of attack paths in new technologies to quality-of-life improvements to make tasks easier, and finally some work to integrate BloodHound with large language models (LLMs).

SpecterOps was founded with an emphasis on transparency and the open-source community, and we are amazed at the work we have seen in this space from the community this year and excited for what is to come!

ACCESS GRAPHS VS. ATTACK GRAPHS

Seeing Identity Risk Through an Adversary's Eyes

Introduction: The Wrong Lens

In cybersecurity, defenders and attackers often operate with profoundly different mental models. A recurring pattern we at SpecterOps see across breach reports, red team operations, and internal investigations is the gap between how secure an environment appears from a policy or audit perspective and how vulnerable it proves to be in practice.

Organizations frequently assume their configurations are effective because access is “locked down” according to entitlement reviews or group policy audits; however, attackers can still find paths to slip through the cracks—paths made possible not by misconfiguration but an incomplete mental model of control.

The root of this problem is perspective. Organizations often train their defensive teams to view identity through an auditor's lens (e.g., “Who has access to what?”, “Is that access justified?”, etc.) This perspective supports governance but fails to capture how access can be chained, escalated, or abused. It treats permissions as endpoints rather than potential avenues of compromise.

To truly manage identity risk, defenders must adopt the adversary's perspective. They must understand how access and relationships combine to form paths of escalation. This is the essential difference between **access graphs** and **attack graphs**: between what looks secure on paper and what an adversary could exploit in practice.

Access Graphs: Modeling Control Over Resources

An access graph models the relationships between identities and the resources they can control. It maps users to groups, groups to roles, and roles to assets; answering who has access to what. These graphs are central to identity governance programs and support least privilege enforcement, access certifications, and compliance audits. In short, they help prove that policy matches intent.

In this model, control flows from identities to resources. If a user is a member of the SQL Admins group, and that group has administrative rights over a database server, the access graph shows a direct control relationship. This makes the environment legible to defenders and auditors alike.

Access graphs, however, have a critical limitation: They are static and individualized. They are predicated on a well-behaved actor and do not account for what an adversary might do with the access once they obtain it. They also do not account for the granularity of control (i.e., treating access categories broadly) when, in practice, attackers exploit very specific and often overlooked permissions. They tell you who can turn the key, but not what doors they can open once inside.

Attack Graphs: Modeling Control Over Identities

Attack graphs build on the access graph but shift the focus to how control accumulates. They show how adversaries can combine seemingly benign relationships (which access reviews often overlook) into chains of escalating privilege. Instead of simply mapping permissions, attack graphs reveal how an attacker can move through the environment, progressing from one identity to the next.

One of the most important differences is how access is interpreted. Access graphs typically model broad categories like **GenericWrite** or **GenericAll** or simply show group memberships and privileged roles. Adversaries, however, operate on more granular terms. For example, an attacker might not have **GenericWrite** over a user or computer account; however, if they have **WriteProperty** access to the **msDS-KeyCredentialLink** attribute, they can register a rogue authentication key. This tactic is known as “Shadow Credentials”¹ and grants control over the identity without requiring a password or Kerberos ticket. Most access graphs wouldn't even represent this path because it lives beneath the level of abstraction they are designed to model. The attack graph, by contrast, surfaces this fine-grained permission as an active path to identity compromise.

The attack graph also adds an important semantic: Resource control can imply identity control. If an identity has **AdminTo** rights over a workstation, and another user has an active session on that system (i.e., **HasSession**), an adversary could steal tokens, inject into processes, or extract credentials from memory to impersonate the logged-in user.

The identity is accessible not because of formal access rights but because its authentication material is present in a vulnerable state on a system the attacker controls.

Each edge in the attack graph represents an actionable primitive drawn from real adversary tradecraft. These aren't abstract relationships but known opportunities for adversary progression. The attack graph doesn't just show what access exists; it shows where an attacker is likely to go next.

Here are a few examples of common attack graph primitives and how adversaries use them:

- **AdminTo + HasSession** → Credential dumping or token impersonation of logged-in users
- **AddMember** → Escalation via group membership modification (e.g., adding self to *Domain Admins*)
- **GenericWrite (user object)** → Modification of **ServicePrincipalName** (SPN) to enable Kerberoasting
- **WriteOwner or WriteDACL** → Take ownership and rewrite permissions on critical objects
- **WriteProperty to msDS-KeyCredentialLink** → Shadow Credentials attack, enabling silent impersonation

These edges do not represent hypothetical concerns; they reflect well documented attacker techniques observed across real-world breaches and offensive operations.

The strength of the attack graph lies in how it composes these building blocks to reveal paths that accumulate power in an adversary's hands.

1. <https://specterops.io/blog/2021/06/17/shadow-credentials-abusing-key-trust-account-mapping-for-account-takeover/>



Identity Accumulation: The Snowball Effect

Attackers rarely stop at the first compromised identity. Instead, they move through the environment, leveraging that identity to compromise others, and accumulating access as they go. This progression, from initial access to lateral movement to privilege escalation, is what makes identity risk so dangerous. It is not about any one account or permission, but what an attacker can become through the graph.

This is where the attack graph diverges most sharply from the access graph. The access graph asks, “What can this identity access?” The attack graph asks, “What paths exist from this identity to other, more powerful ones?” That difference in framing reveals a dangerous conundrum: Defenders often assess exposure based on an identity’s intended use or what the user should do with it. Attackers, conversely, have no such constraints.

We’ve encountered environments where *Domain Users* were unknowingly members of *Local Administrators* on every workstation in the environment; sometimes for years. No one noticed because the users themselves never exercised the access. However, when an attacker compromised one of those accounts, they immediately capitalized on it. Within moments,

the attacker had *Local Administrators* access across the estate, allowing them to dump credentials, pivot to privileged accounts, and escalate their control.

What makes identity accumulation especially dangerous is that most defensive telemetry or governance models do not capture it. Organizations may monitor for initial compromise, and they may audit privileged group memberships, but they rarely track the intermediate steps an attacker takes as they snowball access across dozens of identities. Many of these steps occur below the radar, pivoting via local admin rights, leveraging cached credentials, or exploiting overlooked delegation chains. Without a path-based model, these escalation chains look disconnected and harmless. The attack graph connects them into a coherent narrative: one that reflects how attackers actually operate and where defenders must focus their attention.

Identity risk is path dependent. It depends not just on what any one identity can do but on how multiple identities relate to each other (directly or indirectly) through exploitable relationships. The attack graph reveals this chain of progression, but the access graph does not.

CASE STUDY



ESX Admins

Case Study: The Hidden Risk in “ESX Admins”

In July 2024, Microsoft disclosed a critical vulnerability, CVE-2024-37085, that affected VMware ESXi hypervisors integrated with Active Directory. This flaw allowed attackers to exploit a naming convention in group management and gain full administrative access to ESXi hosts. Specifically, ESXi hypervisors joined to an Active Directory domain that automatically granted administrative privileges to any domain group named *ESX Admins*, regardless of whether this group was intentionally created or not. This behavior does not rely on the group's security identifier (SID) but solely on its name. Consequently, an attacker with sufficient permissions could create a group named “ESX Admins” and add themselves to it, thereby obtaining administrative control over the ESXi hosts.

From an access graph perspective, such a group might appear to have limited and controlled membership, often including only a few IT administrators. However, the attack graph reveals something more dangerous. In multiple real environments, SpecterOps observed attack paths from broader groups like *Authenticated Users* to the *ESX Admins* group, not because of direct membership but through misconfigured ACLs, inherited delegation rights, or subtle edge conditions that allow for privilege escalation. These relationships are often invisible in traditional access reviews, but they make the group and the cluster it controls reachable by thousands of users.

The implications are severe. ESXi clusters often host vital services, including virtualized domain controllers (DCs). With administrative access, an attacker could extract sensitive data, such as the `ntds.dit` database, leading to a complete compromise of the domain.

VMware addressed this vulnerability in ESXi version 8.0 Update 3 (ESXi80U3-24022510). However, no patch is available for ESXi 7.0; instead, VMware provided a workaround involving changes to advanced host settings to mitigate the risk.

This case underscores the importance of understanding the difference between access and attack graphs. While access graphs may suggest a secure configuration, attack graphs can reveal hidden pathways that attackers might exploit, emphasizing the need for a more comprehensive approach to security analysis.

Why This Matters: Rethinking Identity Risk

Access graphs are valuable as they help organizations understand entitlements, support compliance initiatives, and visualize access relationships in ways that are easy to audit; however, graphs are descriptive, not diagnostic. They show what access exists but not what can happen as a result of that access. Instead, they focus on policy alignment versus adversarial opportunity.

Attack graphs, by contrast, allow organizations to model risk in operational terms. They expose the paths that attackers can take (i.e., how one identity leads to another, how control escalates, and how blast radius grows). This is critical for prioritization.

Without the attack graph, defenders are left guessing which users represent real risk. With it, they can rank identities by how close they are to critical assets, how easily an attacker could compromise them, and how much control they unlock.

In practice, this enables a more strategic approach to hardening. For example, a security architect may run a query across the attack graph and discover that a non-privileged user account has a three-hop path to *Domain Admin*: one that includes a writable group membership, a session exposure, and a misconfigured ACL. With this context, the team can:

- Remove the user from the vulnerable group
- Apply session isolation or credential hygiene controls
- Fix the ACL to close the escalation path

These remediations are not abstract. They reduce the actual number of viable attack paths and shrink the effective blast radius of identity compromise. Rather than treating all privileged accounts equally, teams can focus their efforts on breaking the most exploitable chains first.

This changes how identity risk is quantified. It's no longer about who has a sensitive role. It's about how many paths lead to that role and how feasible those paths are to walk. This perspective enables a more precise, graph-based approach to hardening (remediating chokepoints, removing exploitable links, and eliminating unnecessary privilege relationships based on actual attacker movement) instead of guesswork.

Trade the Checklist for the Adversary's Map

Identity risk is not a static property of an account or group but rather a path-dependent phenomenon. Reducing that risk requires more than reviewing who has access; it requires understanding how an adversary operating with intent and creativity can *chain, abuse, and escalate* it.

The access graph is the auditor's tool. It validates policy, ensures compliance, and keeps organizations aligned with regulatory expectations. The attack graph, however, is the adversary's map. It reveals opportunity, intent, and reachability. It models the real threat.

To effectively manage identity risk, defenders must use both lenses but prioritize the one that predicts compromise.

The attacker doesn't care about the checklist; they care about the path.

Identities at Rest vs. Identities in Transit

Introduction: Borrowing the Paradigm

In cybersecurity, the concept of data at rest versus data in transit is foundational. It defines how we approach encryption, storage, and secure transmission, and (more importantly) teaches us that the state of a resource changes how we protect it.

The same distinction should apply to identities.

In today's enterprise environments, identities don't live solely in directories or vaults. They move, authenticate, and persist in sessions. And yet, our industry continues to treat identity as a static object: a record in an Identity Provider (IdP), protected by password policies and multi-factor authentication (MFA).

We challenge that model. We propose a new framing, identities at rest versus identities in transit, and argue that until defenders treat both states as first-class citizens, attackers will continue to exploit the blind spots in between.

Defining the States: At Rest and In Transit

Identities at Rest

An identity at rest is the representation of an account in a system, typically registered in an IdP like Active Directory, Entra ID, or Okta. This includes:

-  Stored credentials (e.g., passwords, SSH keys, biometric factors)
-  Group memberships or role assignments
-  Access policies tied to the identity object

The defining feature of the at-rest identity is that it represents potential control. Compromising an identity at rest allows an attacker to attempt authentication, but it doesn't guarantee success. That depends on the ability to bypass controls like MFA, password rotation, vaulting, and network restrictions.

Identities in Transit

In contrast, an identity in transit is an active session. This is what exists after authentication:

-  Kerberos tickets and access tokens in Windows environments
-  Browser cookies and OAuth tokens in federated/cloud platforms
-  Primary refresh tokens (PRTs) and their derivatives in hybrid systems

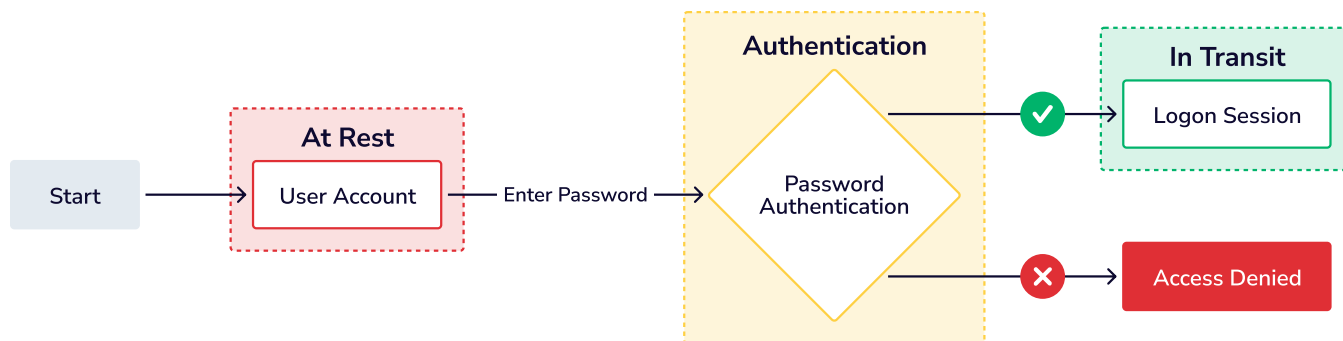
An identity in transit represents realized access.

It no longer needs to authenticate because it is the authenticated session. From an attacker's perspective, compromising a machine where an identity is active allows them to bypass all the protections aimed at the at-rest state.

Importantly, these states coexist. An attacker without access to the environment sees identities at rest, such as accounts behind login prompts. An attacker with a foothold on a workstation, however, can access identities in transit that are present in memory or session storage and ready for hijacking.

Visualizing the Transition

The diagram below represents a simplified authentication flow that illustrates the boundary between an identity at rest and an identity in transit:



In this flow:

- The user account exists in a passive state, at rest, until the authentication process is initiated
- The password authentication step functions as a gate
- If the login is successful, the identity transitions into a live logon session, where it becomes in transit, an active, authenticated presence in the system
- If authentication fails, the identity remains at rest, and access is denied

This model highlights the key insight: The authentication process is the inflection point where an identity shifts from being a static object to a dynamic, operational construct. While defenders often focus on protecting the object at rest, attackers increasingly target the session in motion or the state that begins after this transition.

The Attacker's Perspective

For adversaries, identities in transit are often more valuable and more accessible than identities at rest. While passwords may be hashed, vaulted, or protected by MFA, session tokens are typically treated as bearer artifacts so anyone with the token can act as the user.

This has led to an evolution in attacker tradecraft:

- **Token theft** via Mimikatz, LSASS scraping, or Windows APIs
- **Pass-the-Ticket** and **Pass-the-Hash/Key** attacks in Active Directory environments
- **Browser session hijacking** by extracting cookies or local storage artifacts
- **OAuth token replay** across CLI tools and SDKs
- **Adversary-in-the-middle (AitM)** proxies that capture both credentials and sessions

Attackers often begin by compromising a low-privilege machine and then use User Hunting tactics to locate higher value identities in transit. Once they find a session on a workstation, browser, or application, they could steal that session and reuse it to bypass every traditional control.

This tactic, known as “User Hunting,” is explored in depth later in this essay. It refers to the attacker’s ability to locate active sessions of high-value users — a key strategy for compromising identities in transit.

CASE STUDY 1



Snowflake — The Breach We Saw (and the One We Didn't)

In 2024, a financially motivated threat actor compromised multiple customer environments in Snowflake. The breach originated from infostealer malware on a third-party contractor's device: a machine that had administrative access to various Snowflake tenants. From that machine, the attacker harvested and sold static credentials (passwords) tied to Snowflake accounts.

Snowflake's response was textbook identity-at-rest thinking:

- Enable MFA
- Rotate passwords
- Implement IP-based access restrictions

All of these are sound recommendations *if* you assume the attacker must log in. In this case, the attacker already had access to the contractor's machine, where active sessions almost certainly existed. Browser cookies, OAuth tokens, and refresh tokens would have provided access without ever triggering a login prompt.

The post-incident recommendations, while valid, addressed what did happen and entirely missed what could have happened. Because they only considered the identity at rest, they ignored the risk of the identity in transit. It was a plan for the past, not a defense for the future.

Visualizing the Identity State Transition

Recall the simplified identity lifecycle we introduced earlier.

In this model, authentication is the transition point. The identity begins at rest, and, if successful, becomes a session in transit.

Now compare this with the defensive interventions recommended in response to the Snowflake breach.

Every control, MFA (1), network policy rules (2), and password rotation (3), is located before the authentication boundary. These are valid measures to protect the identity at rest; however, once a session is established, these defenses no longer apply. The attacker who compromises a legitimate endpoint inherits the in-transit identity and, with it, the access to sensitive data without encountering these controls.

CASE STUDY 2



Void Blizzard

Void Blizzard — Trading the Login Page for the Browser Session

In May 2025, Microsoft Threat Intelligence reported on a Russia-affiliated threat actor, tracked as Void Blizzard, conducting credential theft and session hijacking campaigns against government, transportation, defense, NGO, and healthcare targets across Europe and North America. This campaign represents another clear example of how modern threat actors exploit identities in transit while defenders continue to fixate on identities at rest.

Void Blizzard's operations began with the theft of credentials at rest, usernames and passwords obtained via infostealers or purchased from dark web marketplaces. They did not, however, stop there. Their phishing infrastructure included AitM proxies and typosquatted authentication portals that mirrored Microsoft Entra ID logins. These tools allowed Void Blizzard to intercept session tokens and bypass MFA, sidestepping every defense tied to the act of authentication.

These aren't novel tactics but rather modern refinements of well-known techniques.

- Instead of logging in with stolen credentials and triggering IP-based or behavioral analytics (e.g., "impossible travel"), they proxied traffic through the victim's legitimate session.
- Instead of cracking passwords or brute-forcing logins, they captured the post-authentication state and reused it invisibly.
- Instead of relying on persistence via malware implants, they simply rode the user's identity as long as the session remained valid.

Yet the detection guidance released alongside the disclosure focused on familiar themes, enabling MFA, rotating passwords, and monitoring for anomalous login activity such as impossible travel. These detections assume a world where attackers authenticate from unusual places or devices. But modern adversaries have evolved. They use AitM infrastructure to blend in, proxying traffic through known-good machines and IPs. The result: Detection logic built for authentication events simply doesn't trigger.

Void Blizzard didn't just steal identities at rest; they stole identities already in transit. They didn't log in—they moved in.

The Defender's Weakness: Controls Anchored to Identities at Rest

Despite major advances in identity management, most enterprise controls remain fixated on defending identities at rest. Identity and access management (IAM) policies, password vaults, privilege access management (PAM) workflows, and MFA challenges all assume that the primary risk is in the attempt to authenticate.

But attackers don't need to authenticate if they already possess the session.

Conditional Access: Powerful but Not Panacea

Modern conditional access policies (CAPs) can restrict access based on location, device, or risk, but in most systems, these conditions are enforced only at token issuance. Once an access token is granted, it can be replayed from any location unless continuous access evaluation is in place. Conditional access works at the gate, not within the session.

Vaulting and PAM: Still Fighting Yesterday's War

Password vaulting and PAM tools focus on securing credentials before use. But attackers often arrive after the identity has authenticated. Vaults are irrelevant when the attacker doesn't need the key, as they've already slipped through the door.

Identity-in-Transit in Action: Two Examples

- **Pass-the-Ticket** (Active Directory): Extract a ticket-granting ticket (TGT) from memory and reuse it. No password, no MFA, no vault.
- **Browser Cookie Hijacking**: Steal authenticated session cookies and replay them into software-as-a-service (SaaS) apps like Salesforce or Entra. The session is valid, so the attacker inherits access silently.

Detection Engineering: Fighting Blind

Most detection logic is built to catch authentication attempts, but token theft and session hijacking don't leave those breadcrumbs. They generate no failed logins, no brute-force indicators, and no authentication spikes.

As early as 2017, SpecterOps's Chief Technology Officer (CTO) Jared Atkinson and Chief Services Officer (CSO) Robby Winchester presented "A Process Is No One"¹ at Black Hat Europe, highlighting how token impersonation in Windows could be detected through cross-process correlation. Yet, years later, most environments still lack the telemetry or analytics to catch such attacks in real time.

User Hunting: From Admin Pivoting to Session Targeting

The concept of User Hunting was first popularized in offensive security circles by SpecterOps security researcher Will Schroeder (i.e., harmj0y),² who presented it at ShmooCon 2015 in his talk "I Hunt Sys Admins." The phrase, and the philosophy behind it, drew inspiration from leaked NSA documents³ that described U.S. intelligence efforts to compromise system administrators as a means of accessing sensitive foreign networks. These state-level operations treated sysadmins not just as gatekeepers but as vulnerabilities. A logic Schroeder adapted into a formalized methodology for red teams: Locate where high-value identities are active, then exploit their sessions to move laterally and escalate privileges.

This tactic laid the groundwork for both red team tradecraft and tools like BloodHound, where the **HasSession** edge formalizes session presence as an attack path.

Today, User Hunting has expanded well beyond AD.

1. <https://www.blackhat.com/docs/eu-17/materials/eu-17-Atkinson-A-Process-Is-No-One-Hunting-For-Token-Manipulation.pdf>

2. <https://www.youtube.com/watch?v=yhuXbkY3s0E>

3. <https://theintercept.com/2014/03/20/inside-nsa-secret-efforts-hunt-hack-system-administrators/>

Modern attackers don't just look for who's logged in. They look for which identities are in transit on which machines.

- Kerberos tickets in memory
- Browser cookies and OAuth tokens in session storage
- PRTs on hybrid-joined Windows devices
- Federated SAML assertions and OpenID tokens in browser contexts

These session artifacts are often spread across workstations used for daily tasks. A single endpoint may hold simultaneous in-transit identities from the domain, the cloud, and even personal services. Attackers harvest this information to move laterally, escalate privileges, or exfiltrate data all without reauthenticating.

Defenders can adopt this strategy too:

- Hunt for where sensitive identities are active
- Monitor session sprawl and privilege exposure
- Use **HasSession** data to prioritize hardening and containment

User Hunting isn't just a precursor to compromise; it's a lens into where your most valuable identities are currently vulnerable.

Toward State-Aware Identity Defense

Identities do not exist in a single state. Like data, they move from static credentials in a directory to active sessions on workstations, browsers, and cloud platforms. They transition from potential to power the moment a user logs in and yet most of our security

investments treat identity as if it lives only in the vault, authentication log, and policy engine.

This is the fundamental flaw in how we protect identity today.

The breaches at Snowflake and the espionage campaigns of Void Blizzard demonstrate that attackers understand this dichotomy better than defenders do.

They steal credentials and harvest active sessions. They bypass MFA by inheriting authentication. They sidestep conditional access by riding trusted paths.

And they do all this invisibly because our tools are still watching the front door long after the adversary has come in through the side.

If we are to make meaningful progress, we must adopt an identity security strategy that is state-aware:

- One that protects not just the credential but the session
- One that monitors not just the login but the inheritance of trust
- One that maps not just access but reachability

Identity-in-transit must become a first-class citizen in both our detection strategies and our control models. This means tracking where sessions exist, understanding their scope, and hardening the endpoints where they live. It means embracing techniques like User Hunting not just as red team tradecraft but also as blue team situational awareness.

The next generation of identity security won't just be about controlling who can log in. It will be about understanding what identities are doing once they're active and who else might be watching.

This is where Attack Path Management (APM) becomes essential. By modeling not just access rights but the paths an attacker can take through in-transit identities, Identity APM transforms session exposure from an invisible liability into a visible, measurable, and remediable risk.

It gives defenders the tools to see how an identity's presence on a single machine can become the key to an entire environment and to take proactive steps before that key is copied and reused.

DETECTION IN CONTEXT

Escaping the Tyranny of the Atomic Alert

Introduction: The Emissary's World

Modern detection engineering has become a triumph of narrow focus. We chase precision—atomic alerts, minimal false positives, clean queries—and, in doing so, we often reduce complex situations into discrete, isolated facts. This mindset feels efficient and technical, but it can blind us to the broader landscape in which threats unfold.

This dynamic is captured powerfully in Iain McGilchrist's book *The Master and His Emissary*, which explores how the two hemispheres of the human brain interpret the world in fundamentally different ways. The left hemisphere (the Emissary) is diligent, detail-oriented, and analytic. It specializes in breaking the world down into parts. The right hemisphere (the Master), by contrast, understands context. It sees wholes, relationships, ambiguity. The Emissary is brilliant at distinguishing things from one another, but the Master is what allows those things to have meaning in a larger frame.

To illustrate the difference, McGilchrist tells the story of a bird searching for food. The bird's focused attention, what allows it to spot and distinguish a seed from a bed of pebbles, is the mode of the Emissary. However, while the bird focuses on the pebbles, something must stay aware of the wider environment: predators, shadows, movement in the periphery. That's the Master's role. Without it, the bird might succeed in finding a seed, only to be eaten in the process.

In cybersecurity, we often find ourselves operating like that bird—obsessively focused on finding the “seed” of malicious activity in a noisy bed of benign events. We hone detection logic to distinguish the signal from the noise, but too often we do so at the expense of broader situational awareness. What's the context of this alert? How does it fit into the attacker's path? What environment is this happening in and what could it mean?

This loss of context has real consequences, especially when it comes to prioritization. In the absence of environmental understanding, alerts are flattened into a single plane of urgency.

**Everything feels important and nothing does.
Analysts are left guessing which alerts truly matter while attackers
quietly follow paths that remain invisible in atomic views.**

What Detection Has Become

Modern detection engineering operates under the Emissary's gaze. We define success by how precisely a query can isolate a single behavior. We build detections around technical signatures, individual log lines, or isolated actions that map neatly to tactics and techniques. The more atomic and self-contained the alert, the more we tend to trust it.

This approach isn't inherently wrong (in fact, it's often necessary); however, when it becomes the dominant or only mode of thinking, it distorts our understanding of threats. An alert becomes a fact unto itself, detached from the environment that gives it meaning. A suspicious PowerShell command might raise a flag, but is it part of a scripted deployment, a red team exercise, or an unfolding attack? A login to a sensitive system might look anomalous, but is it abuse or just an admin responding to a legitimate outage?

The danger lies in how these atomized detections shape the workflows of analysts. Instead of following a narrative, they triage alerts as if each were a separate case, each requiring a yes/no decision based solely on its internal evidence. The result is a world of detection without context.

Worse still, this Emissary-led model trains us to overvalue clarity and undervalue ambiguity. It favors what is easily observable and discourages exploration into the surrounding environment. Analysts become hesitant to trust their intuition or ask, "What else is happening here?" because the system rewards precision instead of understanding.

This isn't just a philosophical problem but rather an operational one. Attackers don't think in atomic actions; they move through environments and build context. If our detections can't do the same, we're always a step behind.

Reclaiming Context: The Master's Perspective

To counterbalance the Emissary's atomization, we need to return to the Master's way of seeing: detection rooted in relationships, context, and narrative. This doesn't mean abandoning precision but rather framing precision inside a broader awareness of how attackers operate and environments actually function.

The Master doesn't look for isolated events but, instead, asks, "How does this fit into the flow of action?" A PowerShell invocation is not just a process; it is a move someone made from somewhere toward some objective. That move exists in relation to other moves, some of which have already occurred and others are yet to come.

This is where graph-based thinking becomes indispensable. Graphs provide a native language for expressing relationships between identities, devices, permissions, behaviors, and sessions. They allow us to ask questions that assume context:



Is this account part of a known path to Tier Zero?



Does this session create a new control relationship that didn't exist before?



Is this behavior anomalous in a way that matters, or is it occurring in a dead end?

With this perspective, a detection is no longer a fixed point. It becomes a signal node embedded in a living system: A thread you can pull to reveal structure, intention, and threat.

Right-brain detection doesn't abandon telemetry. It enriches it. It recognizes that telemetry tells you what happened but not what it means. Meaning is constructed in context, through a mental model of the environment, through understanding how parts fit into wholes, and through a willingness to navigate ambiguity.

The challenge is that this kind of detection is harder to operationalize. It doesn't always produce neat alerts or clear labels, but it does produce better analysts who can recognize when something doesn't make sense even if the query doesn't say so.

A Context-Rich Example: DCSync as a Boundary Violation

Let's ground this philosophy in something concrete. Consider the well-known DCSync technique,¹ which allows an attacker to impersonate a DC and request replication of credentials; including the hashes of privileged accounts. On its face, detecting DCSync seems straightforward: Watch for specific calls to the `DsGetNCChanges` API. Many environments do just that.

However, when viewed through the Emissary's lens, this detection is brittle. You might catch the technique, but you'll also generate noise from legitimate replication activity. The alert is treated as an isolated event. This API call occurred: yes or no?

Let's view DCSync through the Master's perspective. The question shifts from, "Did DCSync occur?" to, "Who was allowed to perform DCSync, and should they have been?" It becomes a matter of environmental state, not just behavior. Suddenly, the real detection opportunity isn't the replication itself; it's the precondition that made it possible.

In most domains, organizations should only grant a handful of identities (typically DCs) the privileges required to perform DCSync. When another identity gains those rights, whether by misconfiguration, mistake, or compromise, that's not just a misstep. It's a Tier Zero boundary violation.

The graph makes this shift legible. It shows which identities have replication rights. It shows where those rights came from (i.e., direct assignment, group membership, shadow delegation) and it allows us to continuously monitor changes in control relationships, not just for moments of malicious activity.

This changes the detection game. Instead of hunting for a needle in a haystack of legitimate traffic, we surface the haystacks that shouldn't exist in the first place. The signal becomes more precise because it's tied to something inherently suspicious, a non-DC identity with replication rights.

And critically, it's a signal with a story, not just what happened but what made it possible and what could happen next if left unresolved.

Enriching the Atom: Context-Aware Alerts

Context doesn't have to live only in the analyst's head. We can inject it directly into our detections by augmenting atomic alerts with structured information from the graph. This is where the philosophy of the Master becomes a set of concrete, operational practices.

Take credential dumping from LSASS as an example. At the atomic level, a detection might simply flag a suspicious process accessing `lsass.exe` memory. But that alert, on its own, leaves the analyst in the dark. Is this a known red team simulation? Is the user behind the process privileged? Who was actually logged onto the machine at the time?

Now imagine that same alert enriched with graph-derived context. The system queries the environment and returns:

- The list of users with active sessions on the targeted system
- Whether any of those users have direct or indirect paths to Tier Zero
- The usual behavior patterns for the source identity and system

Suddenly, two alerts from the same rule can carry radically different implications. If the only logged-on user is an IT intern with no access to anything sensitive, the alert may warrant a lower triage level. But if the session belongs to a domain admin, or to an identity sitting on a privilege escalation path, the same base alert becomes high priority, even urgent.

We can think of these enrichments as falling into two categories:

1. **Global Enrichments:** Applied to many detections, these provide broad indicators of blast radius or risk. For example, "How many hops is this identity or machine from Tier Zero?" or "Is this entity part of a known attack path?"
2. **Rule-Specific Enrichments:** Tailored to the detection logic itself. In the LSASS example, that might mean asking "Who is logged in to the system being targeted?" or "Does this behavior represent a new relationship not previously seen?"

1. <https://specterops.io/blog/2023/10/24/domain-of-thrones-part-i/>

But this isn't just enrichment in the traditional sense. Most enrichment simply appends metadata to make alerts more readable—usernames, asset tags, geolocation, maybe a known threat label. What we're talking about here is structural. It's not adding context around the alert; it's altering the meaning of the alert itself by resituating it inside a dynamic model of the environment. In this way, graph context doesn't decorate detections; it transforms them.

This layered approach turns detections into narrative events. Each one is a chapter in a story about control, movement, and risk. Instead of forcing analysts to recreate the environment from scratch, we give them a compressed version of it, enough to interpret what the alert means in its real context.

The result isn't just better alerts but better learning. Each triage teaches the analyst something about the environment and over time, that context-awareness becomes second nature: part of how the security operations center (SOC) thinks, not just how it reacts.

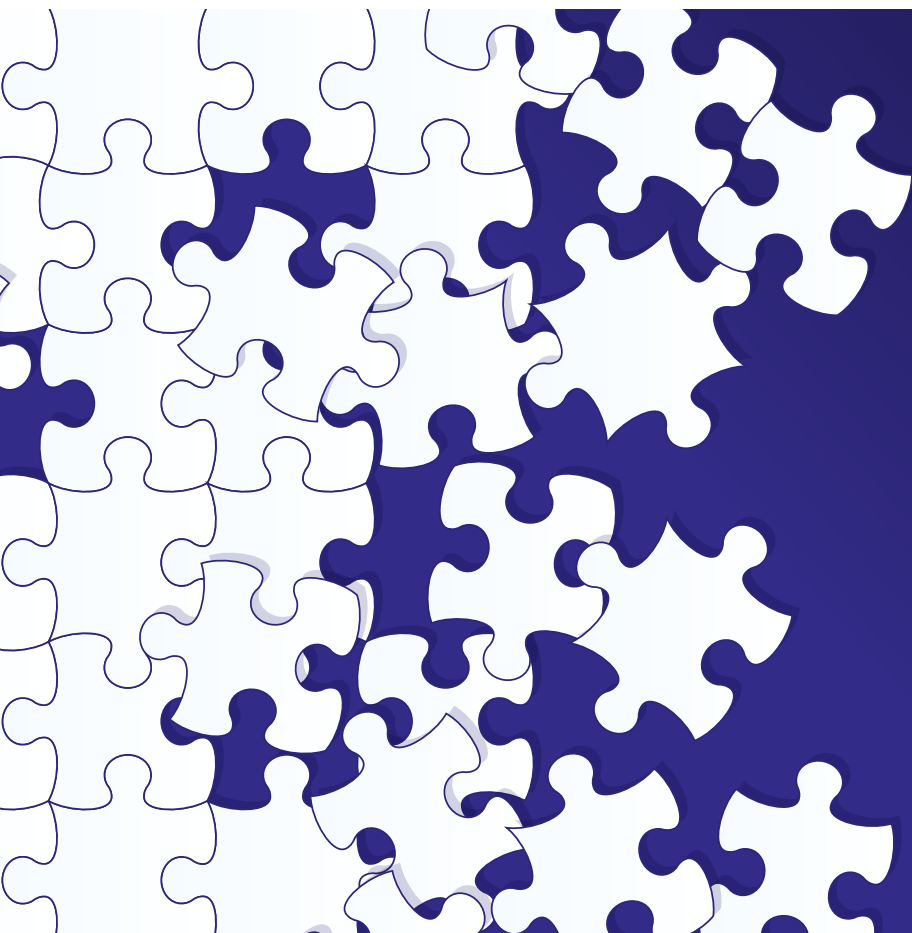
Toward a More Human Detection Practice

The cybersecurity industry built detection programs with good intentions. They optimized for clarity, measurability, and precision; however, somewhere along the way, they began to mistake detail for understanding. They started treating detections like isolated events rather than environmental signals and in doing so, empowered the Emissary while leaving the Master behind.

The result is a detection landscape overrun with alerts, starved of meaning. Analysts drown in precision without perspective while attackers move fluidly through the seams of our environments.

But it doesn't have to stay that way.

By reclaiming the Master's mode of perception (through graph-based context, environmental modeling, and narrative-aware detection), we can restore a sense of coherence to our work. We can turn detections into stories. We can give analysts the ability not just to respond, but to understand. And perhaps most importantly, we can begin to prioritize—not based on what looks suspicious in isolation but rather on what matters in the grander scheme of risk and movement.



This is what it means to
enrich detection with context.
Not just a technical add-on,
but a philosophical shift:

**from fragments to
wholes, from activity
to intent, from noise
to meaning.**

FROM THEORY TO CONTROL

The Future of Identity Security

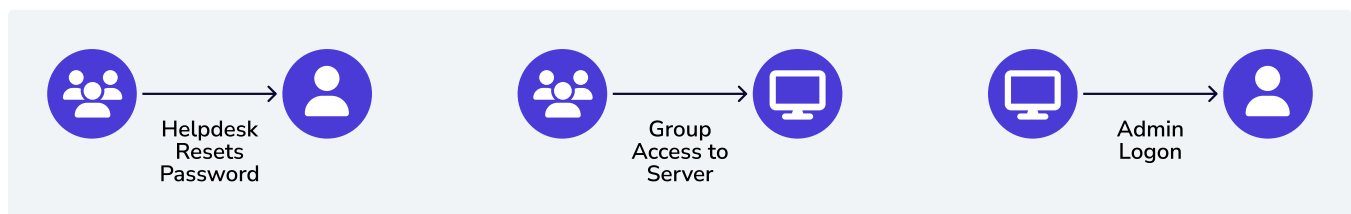
Identity-based attack paths are involved in most breaches (up to 95%¹) and now represent the primary threat to most organizations. These paths form through a combination of user behavior and structural privilege misconfigurations and the problem continues to grow every year. For decades, the industry has tried to address this risk, but most solutions have focused on symptoms versus root causes. The security community's reliance on least privilege wasn't wrong, but it was incomplete. To make real progress, we need to approach this problem in a fundamentally different way.

The Problem at Scale

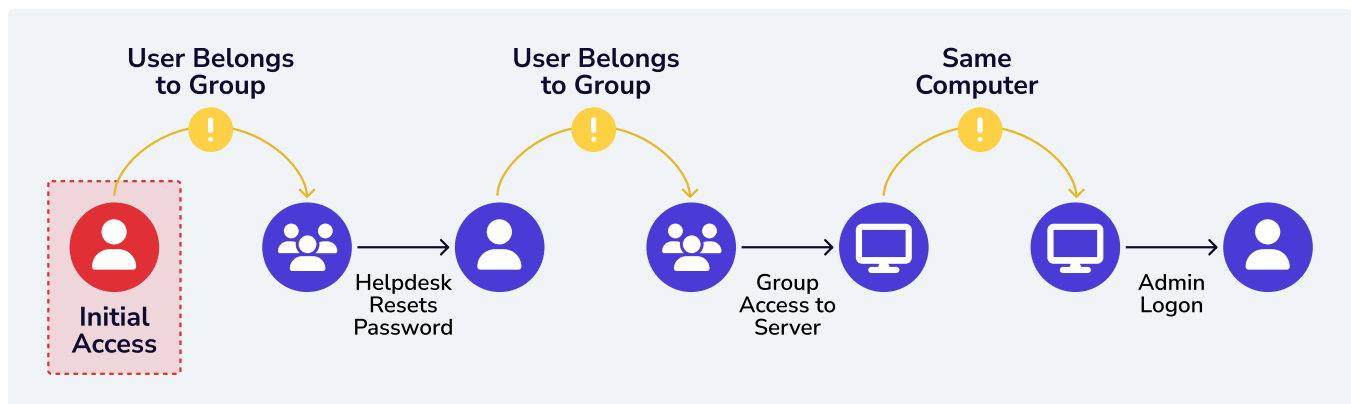
What is an Attack Path (Really)?

Attack paths are not rare edge cases but rather a fundamental property of modern identity environments. They form when an attacker can chain together legitimate privileges, misconfigurations, and leftover access to move laterally or escalate rights. No exploit is required. No malware is necessary.

Most attack paths look innocuous in isolation: a helpdesk user with password reset rights, a legacy group with access to a file server, or an admin session left behind on a shared machine.

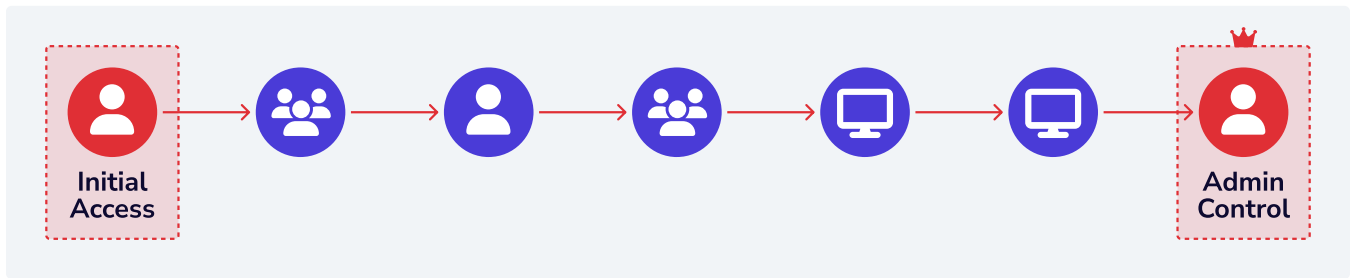


But when connected, these permissions create a hidden route to compromise.



1. <https://files.scmagazine.com/wp-content/uploads/2022/02/CyberRisk-Alliance-State-of-Ransomware-Report-February-2022.pdf>

Attackers operate by logging in and moving laterally, chaining together identities until they reach something that matters.



The Numbers Are Against Us

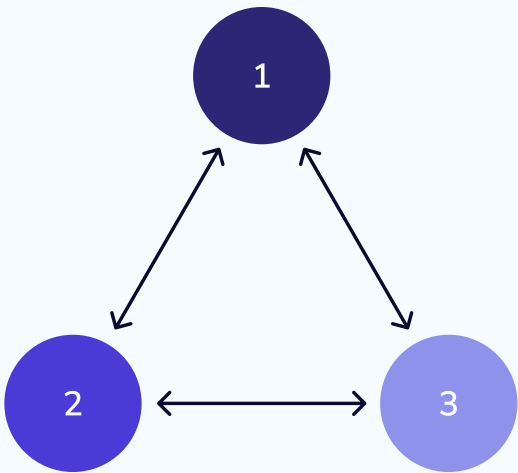
Attack paths don't scale linearly but rather exponentially. Real-world data from BloodHound Enterprise deployments shows just how quickly the problem grows with identity sprawl:



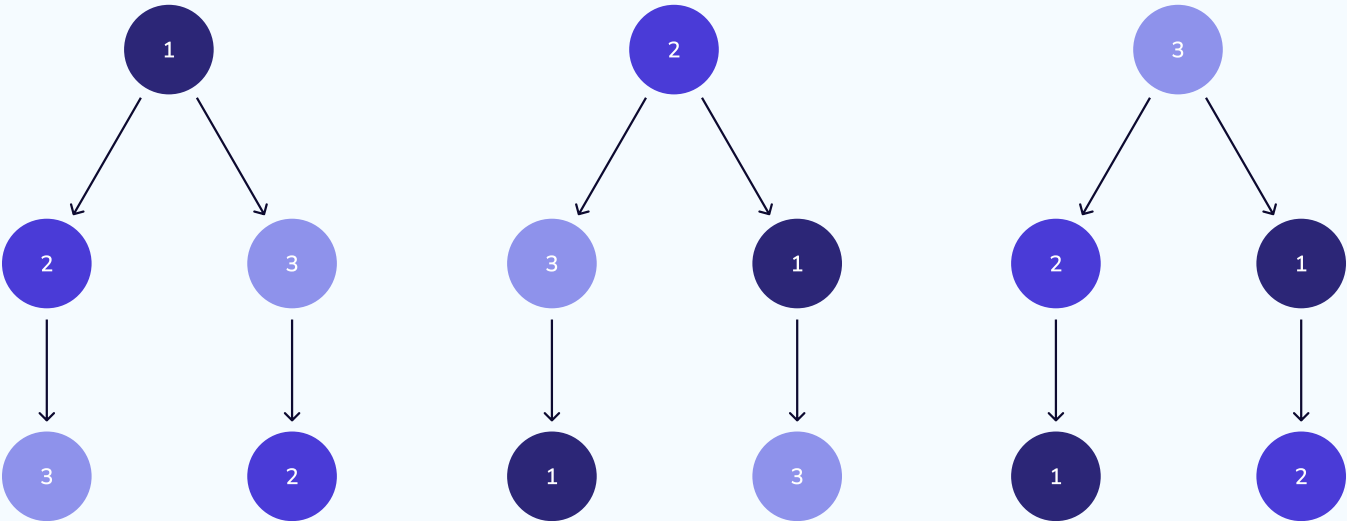
Keep in mind that this is the identity count, not the employee count. Most industry estimates expect a 1:20 human to non-human identity ratio which makes sense when you factor service accounts, service principals, certificates, machine identities, and so on.

If you want to understand why these numbers explode in this way, we'll use a story from Andy Robbins, co-creator of BloodHound and Principal Product Architect, to illustrate.

Imagine three cities connected by roads as a demonstration of a strongly connected graph.



Now, how many different paths are there between these three cities using those roads? To calculate this, you must start from every city and find every path to every other city from the starting city. For example:



Look on the left-hand side. We have two trees descending from the first city with these paths:

- 1,2,3
- 1,3,2

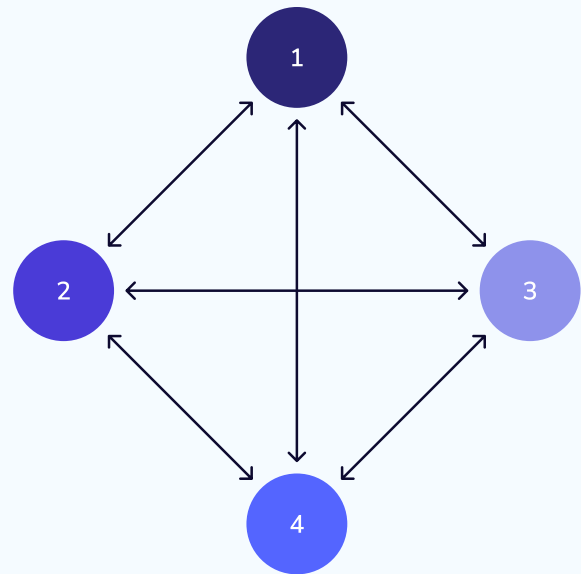
Then you can see the same when we originate from City 2 in the center and City 3 on the right.

Let's start keeping track:

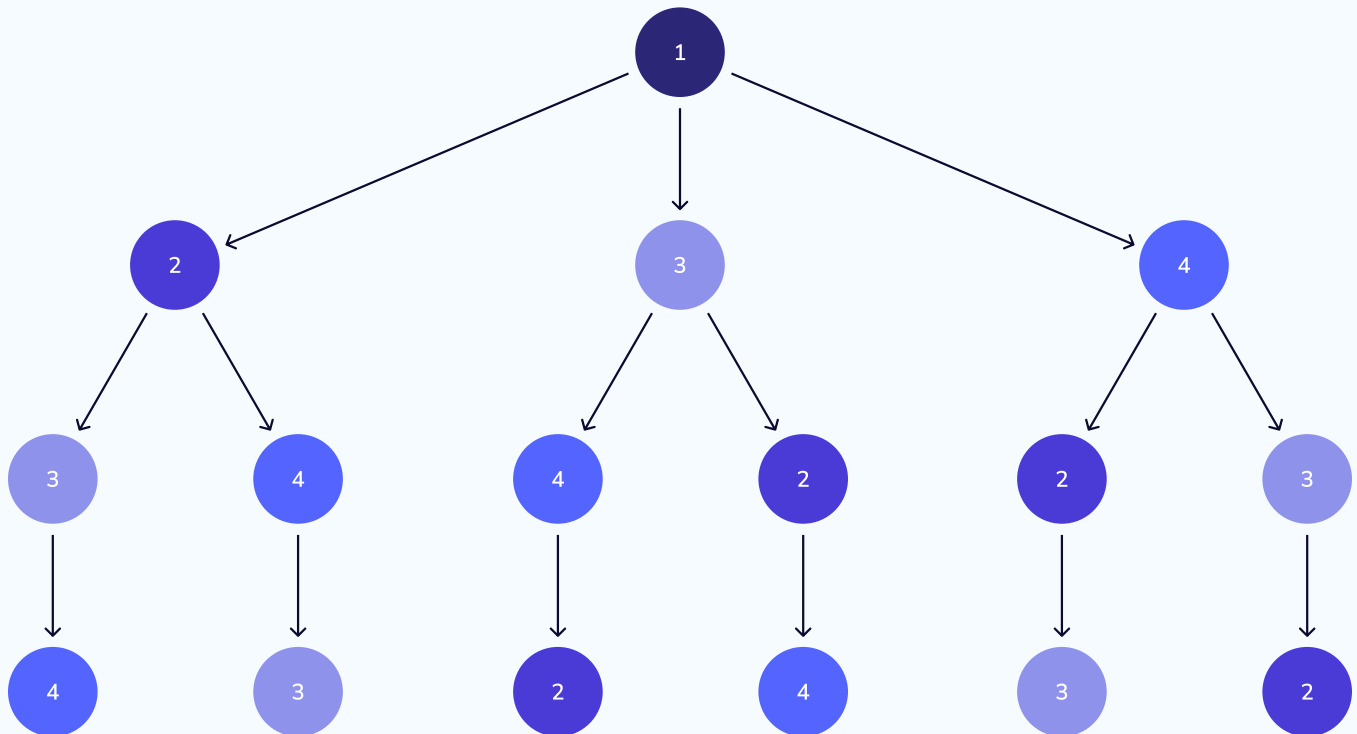
Number of Nodes	Number of Possible Paths
3	6

What happens when we have four cities?

Again, we start from each city and find each path to every other city in the graph.



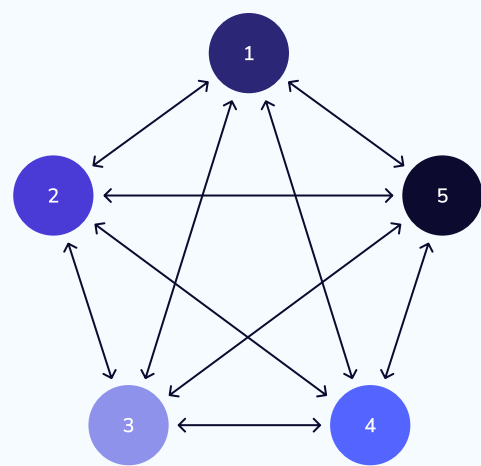
Here's what that looks like from City 1:



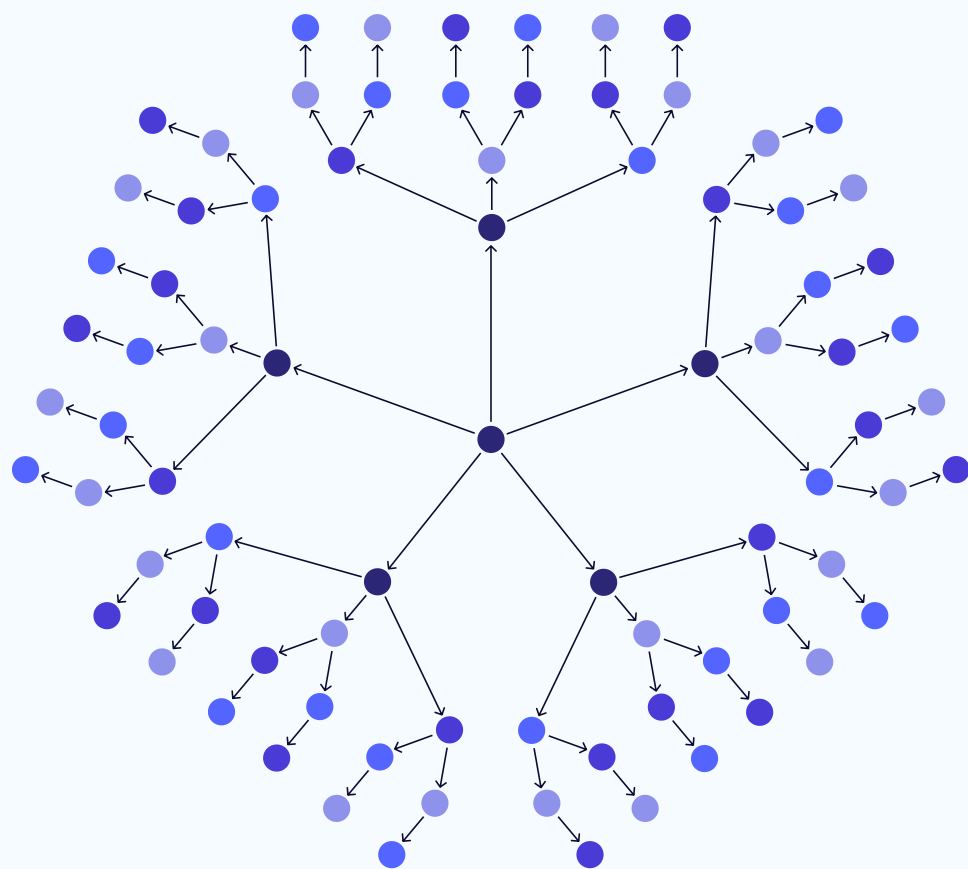
In this graph with four cities, the total number of all paths is 24:

Number of Nodes	Number of Possible Paths
3	6
4	24

Five cities, strongly connected:



Enumerating the paths would look like this:



The total number of **all paths** is now 120:

Number of Nodes	Number of Possible Paths
3	6
4	24
5	120

You may start to notice a pattern: The total number of all paths in a strongly connected graph is the factorial of the number of nodes. The factorial is found by multiplying all integers “under” an integer, so the factorial of 3 is found by:

$$3 \times 2 \times 1 = 6$$

The factorial of 5 is found like this:

$$5 \times 4 \times 3 \times 2 \times 1 = 120$$

If we keep going, we will see that this number explodes quickly:

Number of Nodes	Number of Possible Paths
3	6
4	24
5	120
6	720
7	5,040
8	40,320
9	362,880
10	3,628,800

This hopefully starts to demonstrate how this problem scales out of control. Keep in mind that this is calculating the number of paths from any city to any other city. When calculating identity attack paths in BloodHound Enterprise, we’re specifically looking at the number of attack paths connecting lower privilege identities to critical identities and resources.



Why AI Is Making the Problem Even Worse

Today’s enterprise already faces a 1:20 ratio of humans to identities, thanks to service accounts, automation tools, hybrid environments, and third-party integrations. But that ratio is rapidly climbing with the rise of Artificial Intelligence (AI) as every new agent or bot needs an identity. In many environments, it’s headed toward 1:40.¹

And more identities mean more attack paths.

At the same time, attackers are using AI to exploit those paths with greater speed, stealth, and creativity. Techniques that once required expert-level skill can now be generated with a prompt, making advanced tradecraft accessible to anyone with a keyboard.

We’ve seen this firsthand. In a recent adversary simulation with one of our more mature clients—an organization that’s invested heavily in people, process, and technology—SpecterOps consultants were constrained only by their imagination. Using AI, our operators generated a custom Command and Control (C2) framework written in Perl (a coding language none of our team actively uses), and successfully evaded the client’s defenses. AI removed the skill barrier entirely. What mattered was creativity, not expertise.

That same power is in the hands of every adversary.

Even More Identities	Enhanced Attackers
 Every AI agent needs an identity	 Automated, continuous, agentic
 NHIs outnumber humans 20-to-1	 Nation-state tradecraft, for anyone
 150% growth in NHIs this year	 Too fast. Too many. Too late.

The result? More identities, more privilege relationships, more exploitable paths, and less time to react. Without a scalable way to eliminate attack paths, the problem will continue to grow faster than any security team can respond.

But do any of these problems matter if we adhere to least privilege?

1. <https://www.appviewx.com/blogs/key-takeaways-from-the-2024-esg-report-on-non-human-identity-nhi-management/>

How We Got Here

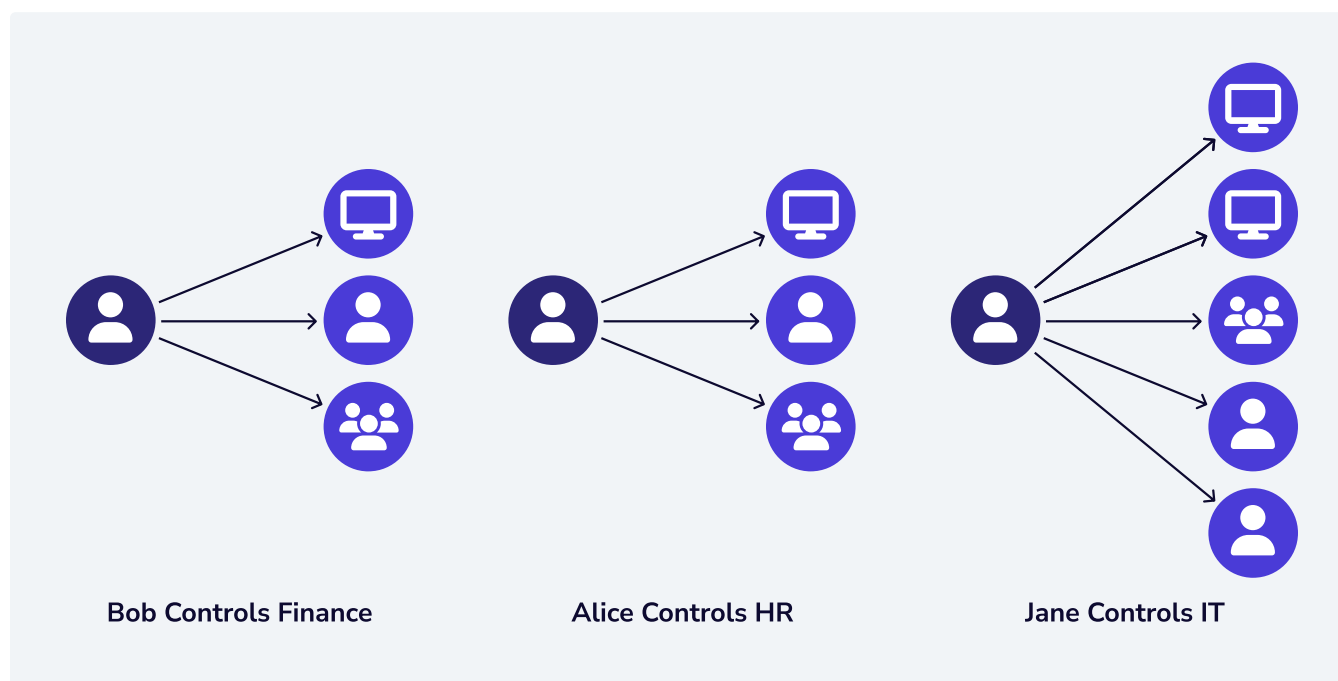
The Myth (and Limitation) of Least Privilege

“Every program and every privileged user of the system should operate using the least amount of privilege necessary to complete the job.”

- Jerome Saltzer

The principle of least privilege, the idea that users and systems should only have the access they absolutely need, has been a staple of security best practices since 1975.¹ It's cited in every framework, audit checklist, and incident postmortem. While the principle is sound, the way organizations apply it in practice falls short.

Organizations typically interpret least privilege in terms of direct assignment (i.e., what permissions a given identity has been explicitly granted). This can be applied individually or through group/role delegation. These assignments are typically done through the use of Identity Governance and Administration (IGA) tools like SailPoint. These tools provision users at scale for what they should access and nothing more (ideally). For example:



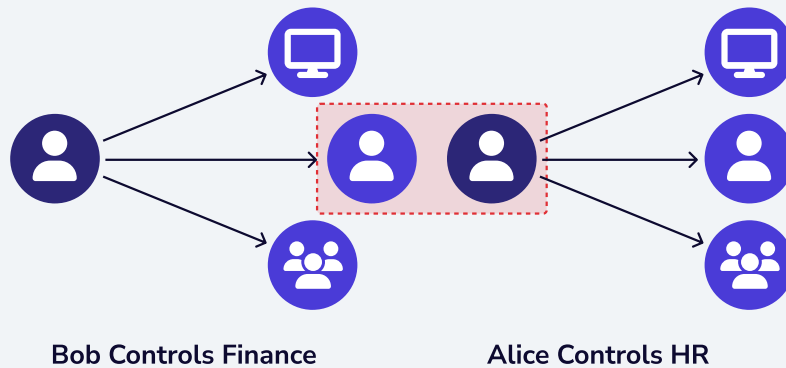
Bob only has access to Finance, Alice to HR, and Jane to IT. Everything is organized, easy to audit, clean, and **woefully incomplete**.

Before I go further, it's important to note that this is not an IGA problem; these tools are doing exactly what they were designed to do. Provision identities at scale with the permissions they need to do their job. It's what they're not doing, and never designed to do, that creates attack paths.

Take the previous image above. That's how the auditor sees the environment, but what happens if Alice is both of these users?

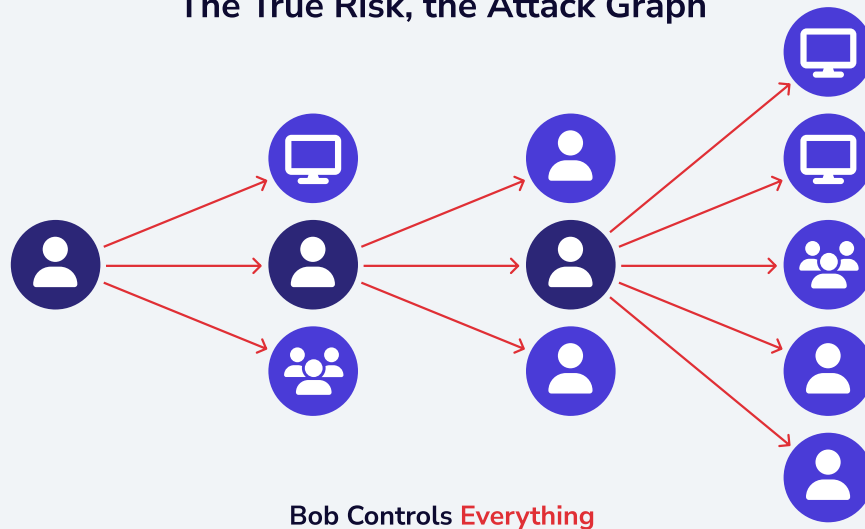
1. https://en.wikipedia.org/wiki/Principle_of_least_privilege

Privilege Collision Across an Access Graph



These are privilege collisions and they happen in every environment.

The True Risk, the Attack Graph



No policy was violated. No excessive permissions were granted. On paper, everyone is operating under least privilege. **But when those isolated permissions are combined, they create viable attack paths: Routes an attacker can exploit without breaking a single rule.**

These hidden overlaps, often called shadow entitlements, are where intended privilege and effective privilege diverge. They're not violations of policy; they're artifacts of complex, interconnected access. And they're exactly what adversaries use to escalate, pivot, and ultimately compromise critical systems. Attackers aren't interested in what they have now; they care about what they can reach.

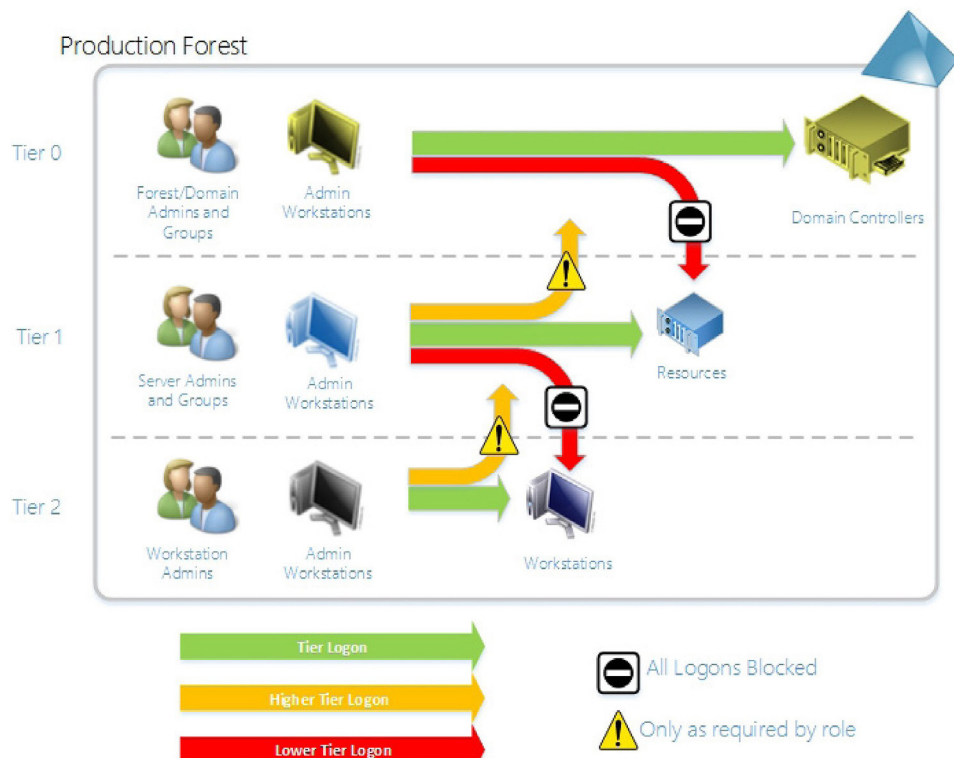
Least privilege was never wrong, but it was incomplete. It focused on assigning access in isolation, not understanding how privilege accumulates and interacts. Until we account for how privilege composes across the environment, not just how it's assigned, we'll keep creating attack paths.

That's what the industry tried to fix with architectural tiering models and "best practice." But as we'll see next, intent without visibility only gets you so far.

Best Practice to Save the Day?

As attack paths became more understood across the industry, SpecterOps started developing more in-depth guidance and best practices on how to combat this threat. At the time, and largely still today, Active Directory was the backbone of identity for nearly every company.

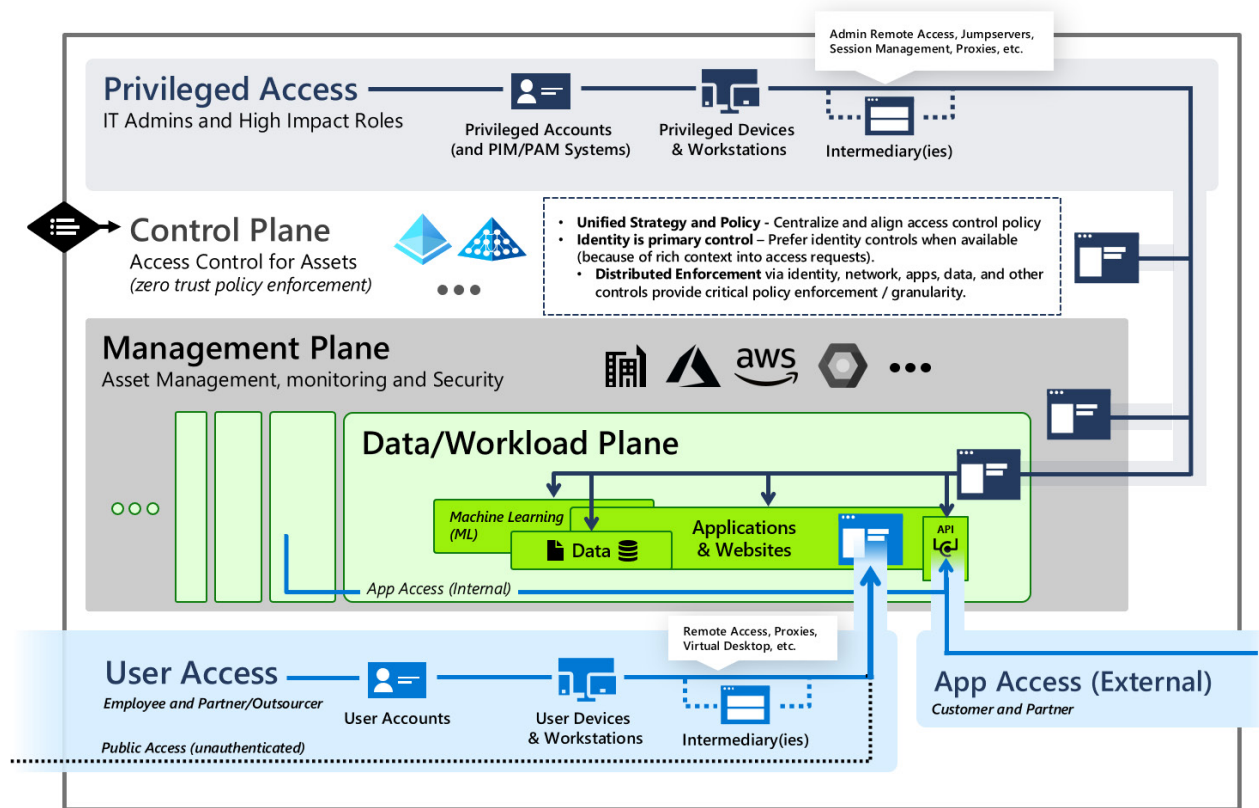
The first guidance came in 2012 with Microsoft's "Mitigating Pass-the-Hash (PtH) Attacks and Other Credential Theft Techniques" whitepaper,¹ and again in 2014 with version 2.² This then evolved into a Tiered Administration Model, which was formally introduced in 2014 and followed up quickly with the Enhanced Security Admin Environment (ESAE)³ or "Red Forest" model.



These all provided a great framework on how to structure your environment, but there was one crucial problem: How did you ever know if you did it right?

Organizations spent millions of dollars and years of effort in the pursuit but faltered and failed hard, leaving only a couple of ESAE environments in active use today.

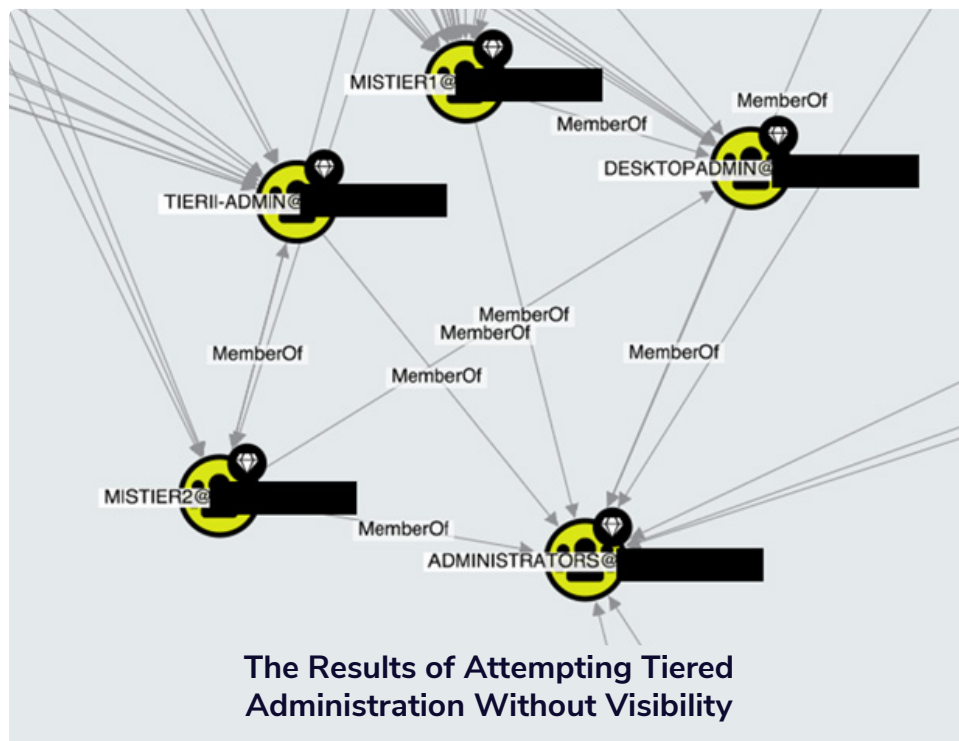
1. [http://download.microsoft.com/download/7/7/A/77ABC5BD-8320-41AF-863C-6ECFB10CB4B9/Mitigating_Pass-the-Hash \(PtH\) Attacks and Other Credential Theft Techniques_English.pdf](http://download.microsoft.com/download/7/7/A/77ABC5BD-8320-41AF-863C-6ECFB10CB4B9/Mitigating_Pass-the-Hash_(PtH)_Attacks_and_Other_Credential_Theft_Techniques_English.pdf)
2. <https://download.microsoft.com/download/7/7/a/77abc5bd-8320-41af-863c-6ecfb10cb4b9/mitigating-pass-the-hash-attacks-and-other-credential-theft-version-2.pdf>
3. <https://learn.microsoft.com/en-us/security/privileged-access-workstations/esae-retirement>



Microsoft now recommends the Enterprise Access Model¹ with fairly prescriptive guidance for administrators, but the problem still stands.

Without visibility, implementation is like drawing floor plans for a maze while you're already lost within.

People tried, very hard, to do the right thing but these are extremely complex systems.



1. <https://learn.microsoft.com/en-us/security/privileged-access-workstations/privileged-access-access-model>



You name it, SpecterOps has seen it. It affects small companies, large companies, mature companies, new companies, old companies, etc. At this point, when someone tells me they’ve “separated their administrators from their users,” I say back, “Well, you’d be the first.”

Remember the scale we’re dealing with? This isn’t anyone’s fault, nor is it a Microsoft problem. Any identity directory will have the same problems. We’ve seen this as organizations try to move from Active Directory to Entra ID, Okta, AWS, GCP, you name it. The problem manifests again and again. It’s a complexity and visibility problem.

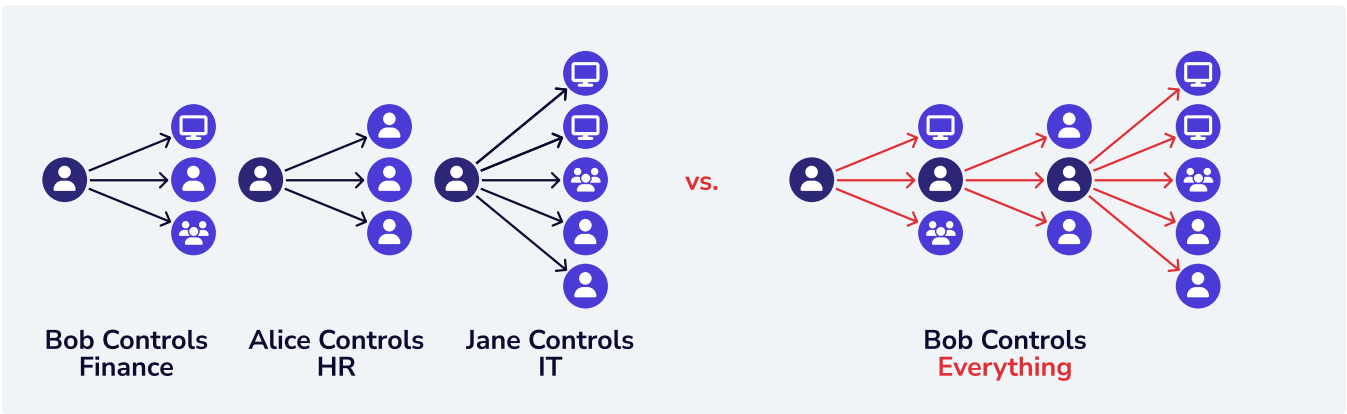
There has never been a technical control to visualize these boundaries. It’s all been best practices, spreadsheets, and intent. However, that didn’t stop us from trying to throw all our tools at the problem.

The Tools We Turned To (and Why They Fell Short)

Without the ability to directly visualize or enforce privilege boundaries and stop attack paths, we leaned into other tools: ones that could control how access was granted or detect when something looked suspicious. This gave rise to broad reliance on IGA, PAM, Endpoint Detection and Response (EDR), and Identity Threat Detection and Response (ITDR) as indirect responses to identity risk.

But these tools were built to solve different problems. They help manage access and detect abuse. They do not address the structure of privilege itself. They don’t stop attack paths.

**At best, they treat the symptoms of attack paths.
At worst, they create a false sense of security
while the problem continues to grow.**



IGA: Assigned, Not Effective

As covered earlier, IGA helps organizations assign and track access, but it doesn't show what that access **enables**. It's focused on what should exist, not what actually exists when access is inherited, nested, or combined. In short, IGA governs permissions but does not map privilege relationships or identify attack paths.

PAM: Controlled Entry, Uncontrolled Aftermath

PAM is excellent at securing how privileged access is issued. It enables practices like just-in-time assignments, MFA enforcement, and conditional access. But once access is granted, PAM steps out of the picture. Jared Atkinson described the problem effectively by introducing the concept of identities at rest versus in transit.

	Identity at Rest	Identity in Transit	
Definition	A privileged account with a credential in a vault	- Sessions - Tokens	- Processes - Created post authentication
Attacks	- Password dumping - Brute forcing	- Token impersonation - Session hijacking - Process injection	- Cookie theft - Pass-the-Ticket

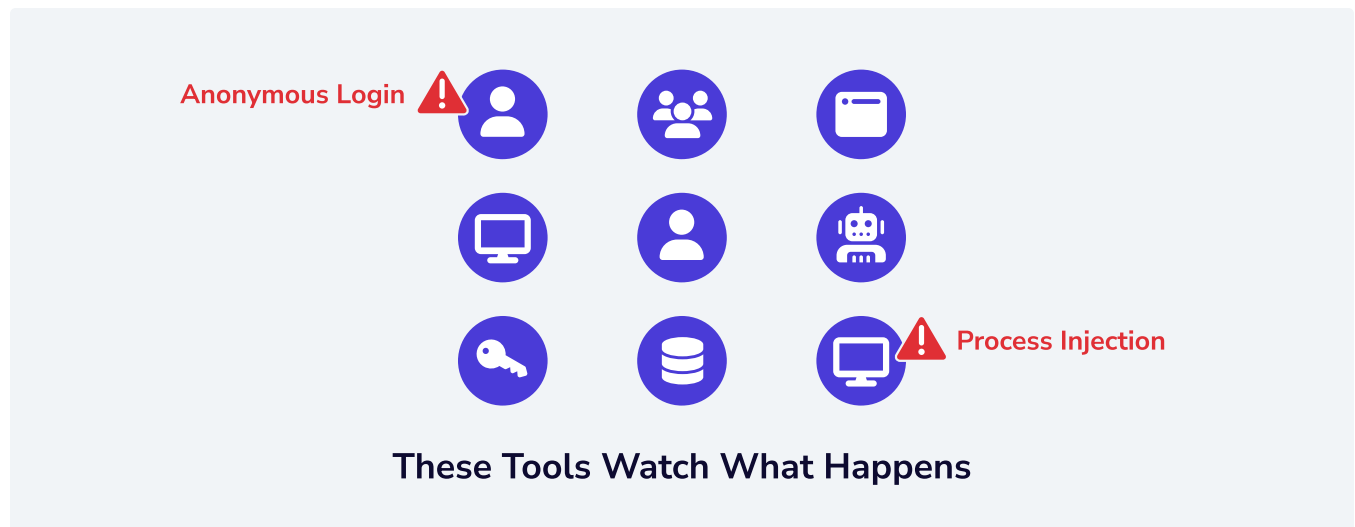
After authentication, new auth material is generated that attackers can and will steal. Attackers can steal session tokens, harvest Kerberos tickets, or hijack access tokens all without ever touching a password. These credential artifacts persist across sessions and systems. PAM doesn't clean them up, monitor their spread, or prevent their reuse.

It's too late for MFA, password rotation, or conditional access. I find myself explaining this to around 80% of our clients who initially may an attack path off because "that account is vaulted."

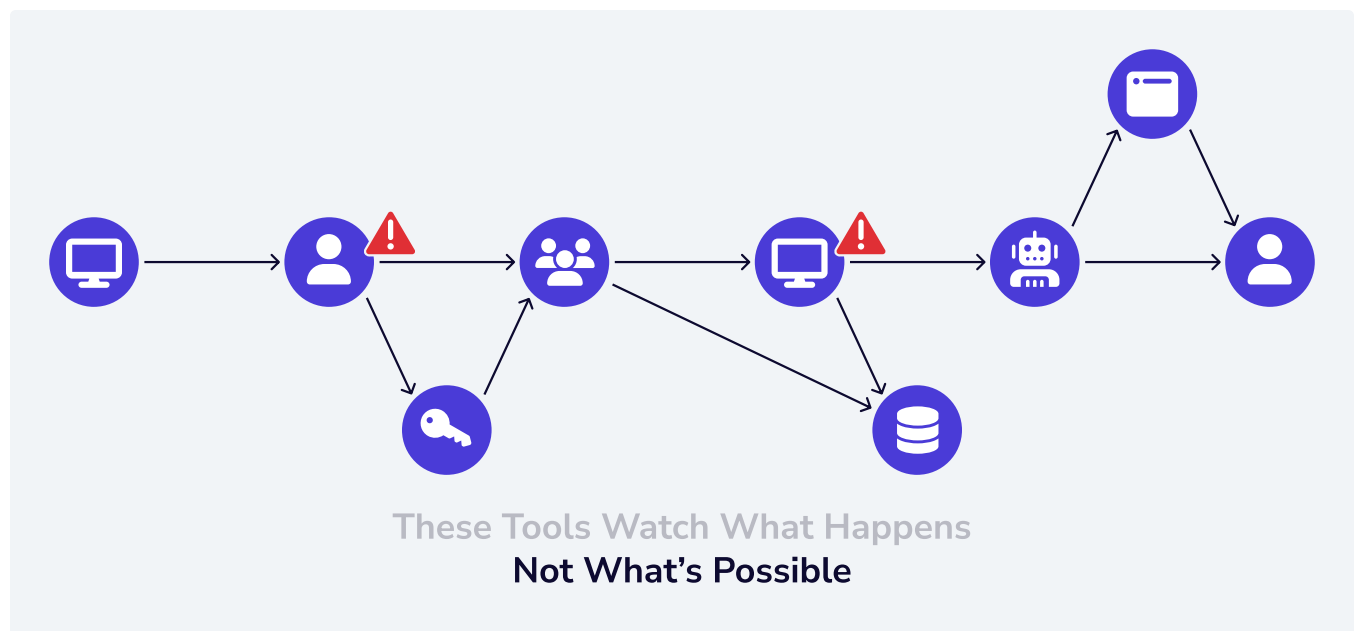
PAM is great at controlling initial credential use (MFA, just-in-time, conditional access, etc.), but attackers don't need to break into this initial use of the credential. They just need to follow the trail of what PAM left behind. PAM governs the front door and attackers follow you in.

EDR and ITDR: Watching Behavior, Not Structure

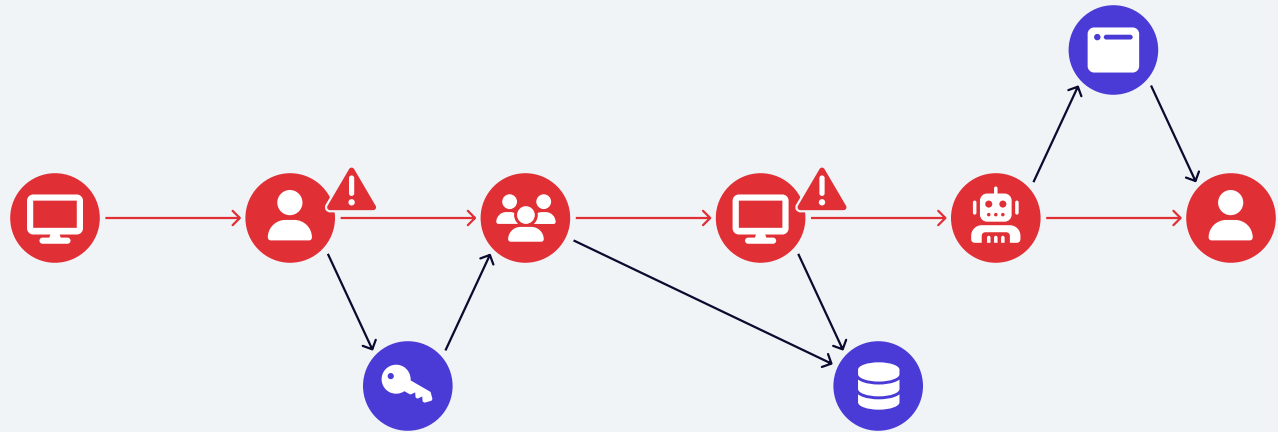
Behavioral detection tools like EDR and ITDR are designed to catch malicious activity. They look for anomalies, such as strange logon patterns, suspicious processes, or lateral movement tactics.



But identity attack paths often don't look anomalous. They rely on valid users accessing valid systems with valid credentials, just not in the combinations defenders expected. If there's no malware and no behavioral outlier, there's nothing to detect.



These tools operate at the layer of events. Attack paths exist at the layer of architecture. And architectural risk doesn't generate logs, until it's too late.



They Detect the *Symptom*, Not the Condition

EDR is very good at detecting malware execution or persistence mechanisms. ITDR excels at identifying misuse (e.g., brute force attempts, anomalous logons, impossible travel). They're fantastic tools but we're relying on them to fix a problem they were never designed to combat. Neither addresses the problem of attack paths and with an ever-growing identity problem, they're going to fail to keep pace.

We're relying on alerting to stop a problem we could have prevented.

We're throwing cameras on every entryway but leaving all the doors wide open. This is why we created Attack Path Management to directly solve the problem of attack paths.

From Tier Zero to Privilege Zones

Tier Zero: The Starting Point

Attack Path Management is the continuous discovery, mapping, and risk assessment of attack path choke points.

BloodHound Enterprise was built to tackle a core problem: If an attacker can reach the highest level of privilege, "Tier Zero" or "Privileged Access," it's game over. Domain admins, global administrators, application owners: These are the systems and identities that control everything else. They're not just high-value targets. They are the blast radius.

Our early mission was to help organizations eliminate attack paths to Tier Zero, and it worked. Customers often reduced their Tier Zero exposure by over 30% in the first 30 days. Six months in, we had our first customer who had eliminated all Tier Zero attack paths. They were removing decades of technical debt and continuously monitoring for any changes to preserve their improved identity risk posture. For the first time, they had visibility into exactly which paths led to their most critical assets, and the ability to shut them down surgically. We were thrilled.

But Tier Zero is only the top of the pyramid. And really, no one **really** cares about Tier Zero beyond the administrators and security professionals that realized what it meant. It's really about securing business critical resources that matter, such as regulated data, applications that produce revenue, intellectual property (IP) that represents the value of our company, or the MRI machine that can't stop working in the middle of an incident.

Privilege Zones: Expanding the Defense

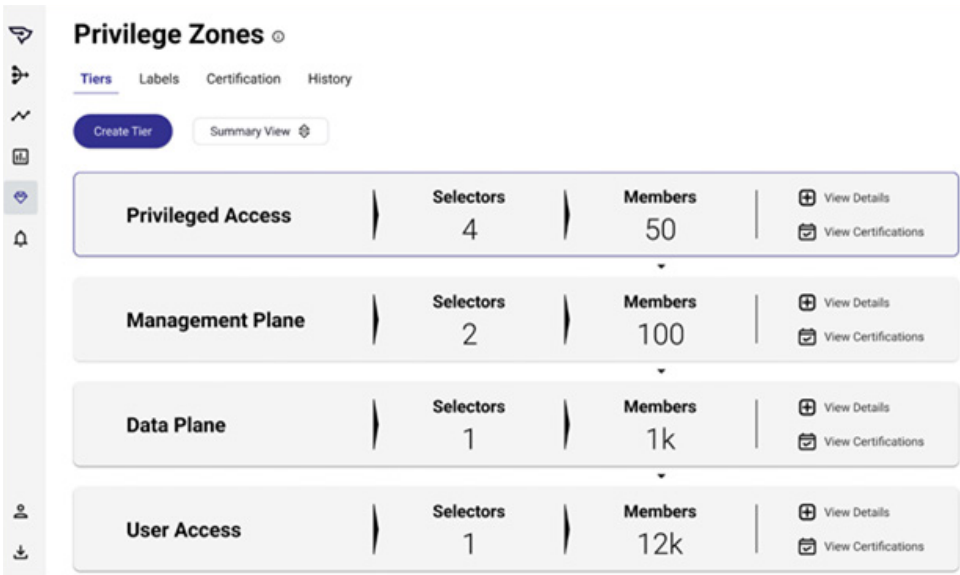
We had to start with Tier Zero first, but it was always our intent to move to what really mattered. As soon as we launched BloodHound Enterprise, we were thinking how to tackle this problem and four years later, we are excited to introduce Privilege Zones.

Privilege Zones is a new capability in BloodHound Enterprise that transforms the way organizations define and enforce security boundaries. Built on the same attack graph-based engine that powers visibility in BHE, Privilege Zones extends control beyond Tier Zero and makes attack paths across zones actionable by giving security and identity and access management teams a way to sever them with precision.

It's no longer just about "Who can reach Tier Zero?"
It's now about "Who can reach where they shouldn't—anywhere?"

Privilege Zones give you the power to define logical security boundaries and enforce them at scale. Whether you're aligning to Microsoft's Enterprise Access Model or your own internal segmentation strategy, BloodHound Enterprise makes it real.

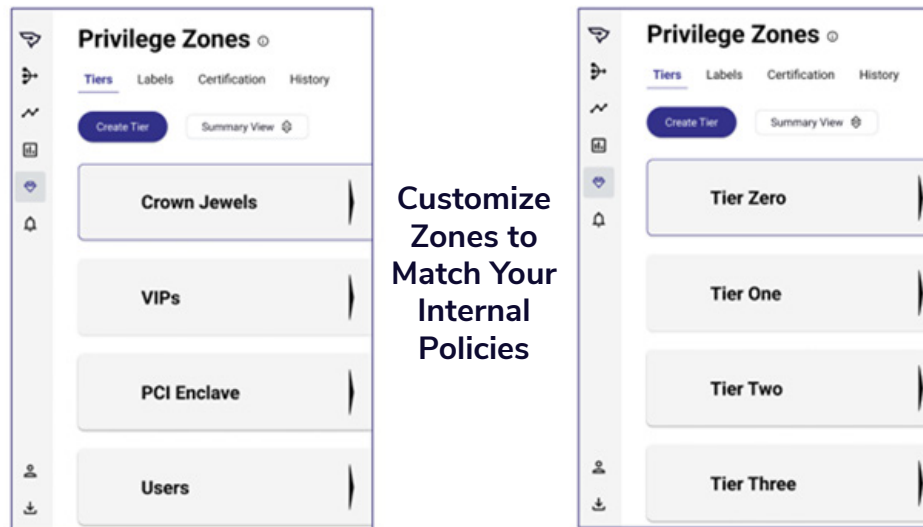
- ✓ Define Zones based on tiers, sensitivity, or business function
- ✓ Prevent escalation or lateral movement between zones
- ✓ Ensure your secure identity architecture design matches reality



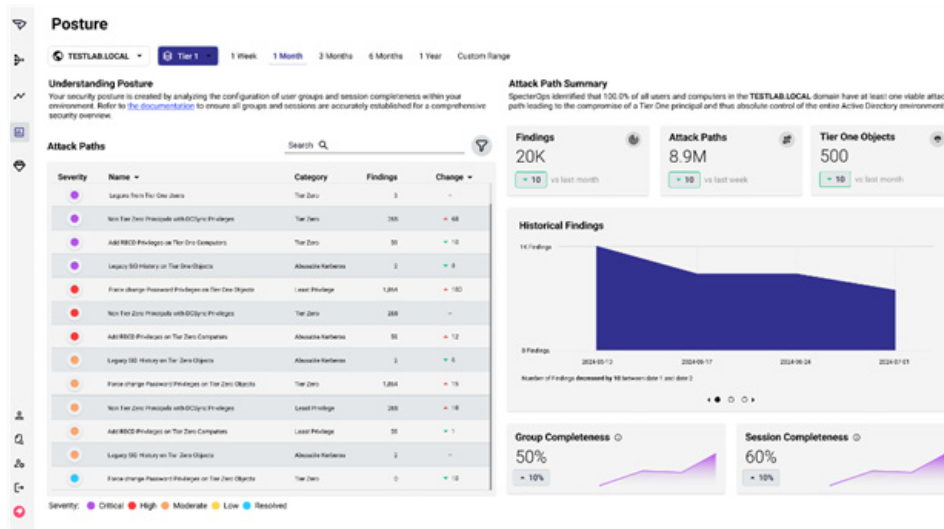
Privilege Zones			
Tiers	Labels	Certification	History
Create Tier Summary View			
Privileged Access	Selectors 4	Members 50	View Details View Certifications
Management Plane	Selectors 2	Members 100	View Details View Certifications
Data Plane	Selectors 1	Members 1k	View Details View Certifications
User Access	Selectors 1	Members 12k	View Details View Certifications

The New Privilege Zone Management Interface
in BloodHound Enterprise

With Privilege Zones, you can establish logical security boundaries based on how your organization actually works, not just how legacy tier models suggest you should. These zones can represent anything: cloud and on-premises workloads, different trust levels, business units, compliance scopes, or sensitivity tiers.



Once zones are defined, Privilege Zones continuously analyzes identity and permission data to detect and eliminate attack paths that violate those boundaries. That means you can spot and sever unintended links between, say, a developer account in Entra ID and production infrastructure or between a contractor's legacy Active Directory account and the *Domain Admins* group.

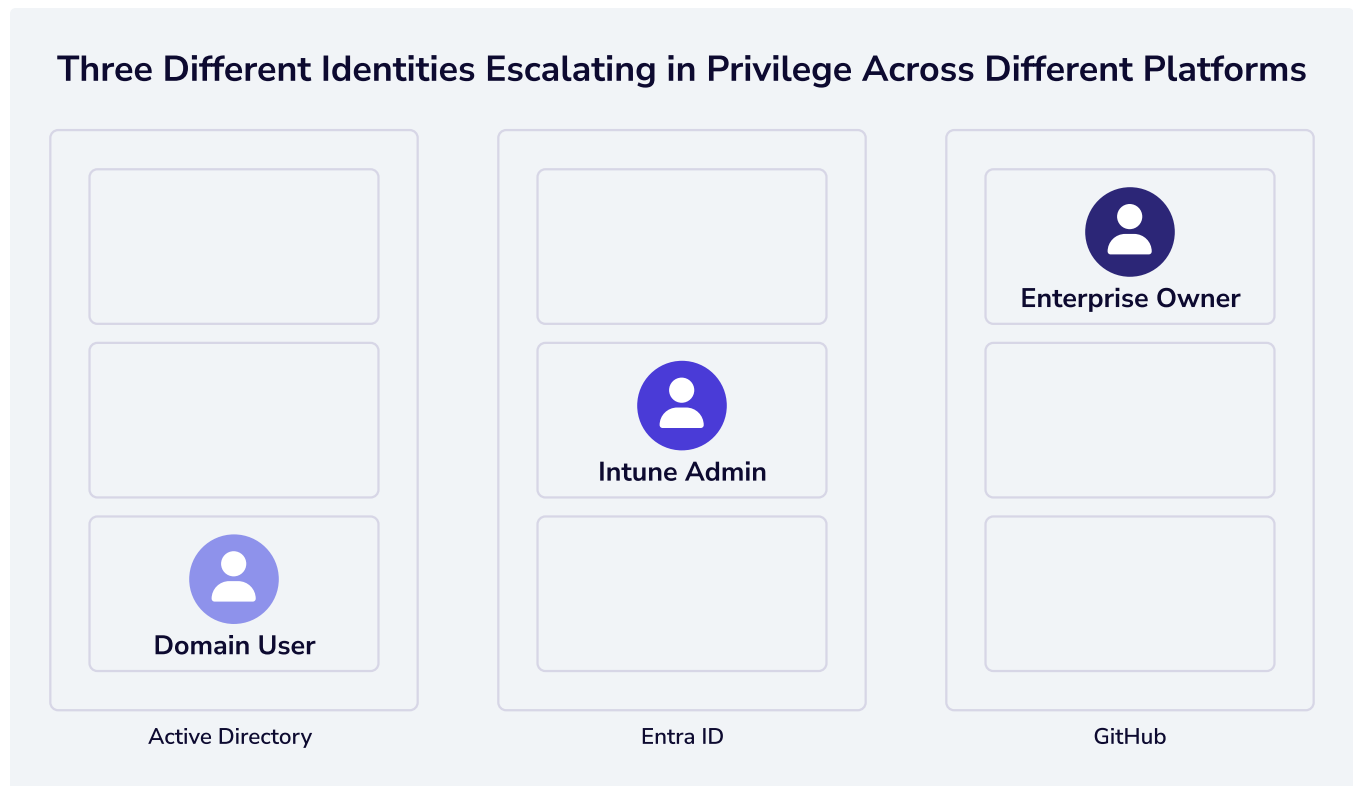


Continuously Monitor and Eliminate Attack Paths Crossing Your Privilege Zone Boundaries

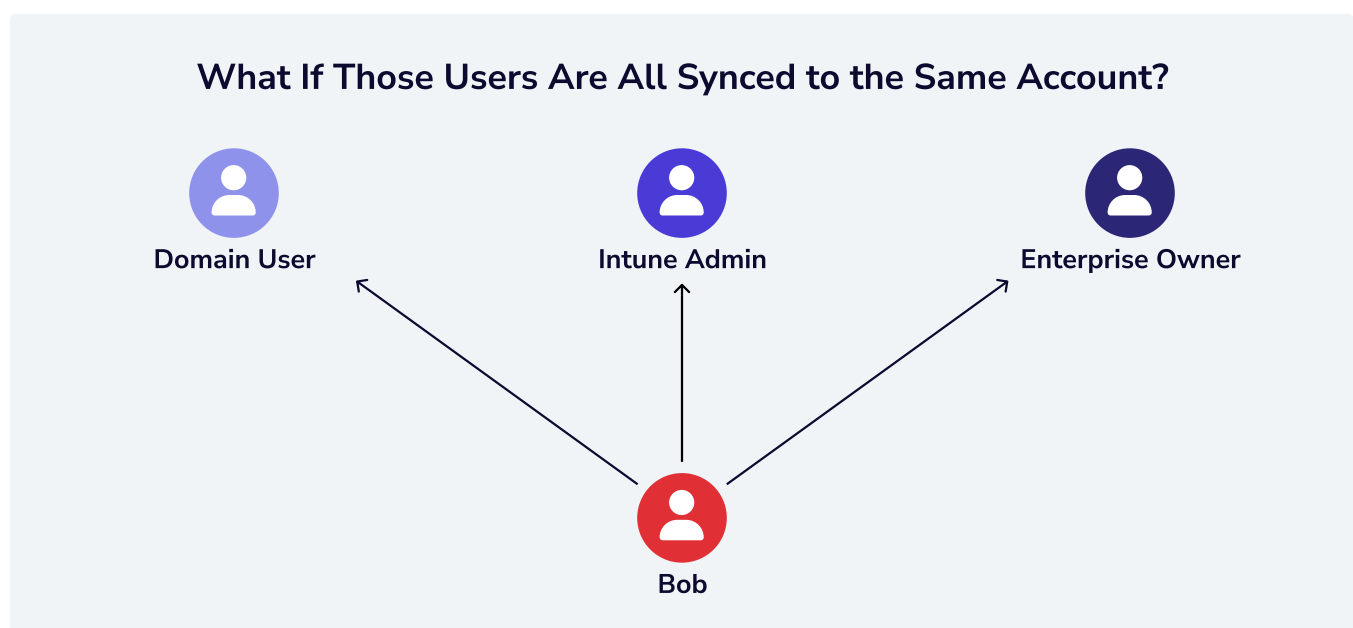
And because it's all built on the BloodHound Analysis, you're getting total visibility. You see the paths, you understand the risk, and now for the first time, you can draw the line and make it stick. This is the missing control the industry has needed for decades: a way to enforce least privilege structurally, not just individually.

Built for Hybrid Identity

The modern enterprise identity landscape is fragmented by design. Most users have multiple accounts tied to the same person: one in AD, another in Entra ID, maybe more in GitHub, or internal systems. These aren't edge cases (*pun intended*), but the norm.

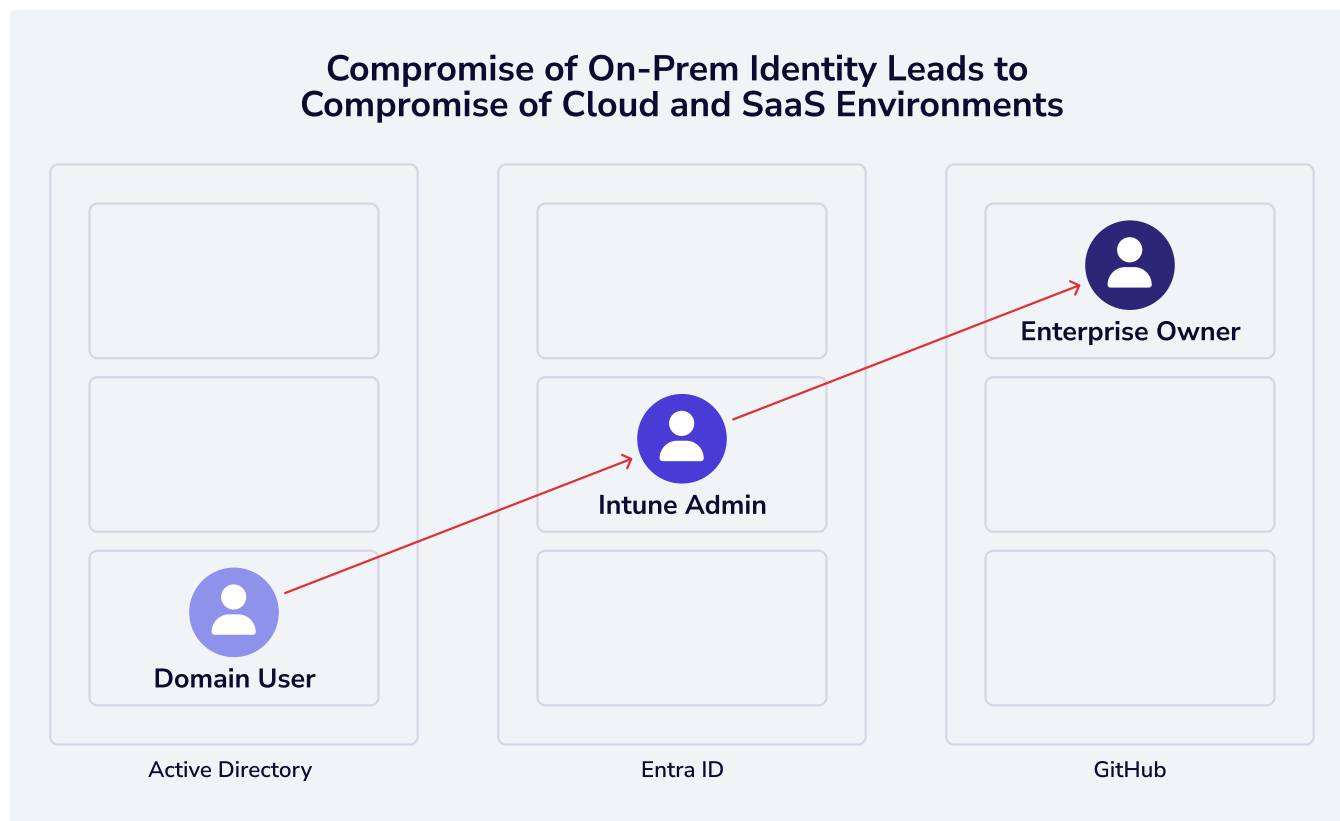


Identity sprawl creates hidden connections between environments that attackers are quick to exploit. A user with limited access in one system might have privileged access in another. Or worse, their accounts might chain together to form a path to critical assets. These links are almost never documented, let alone governed.



This isn't just poor hygiene that's not recommended,¹ it's a privilege zone violation across systems. But how would you know if you're violating this and creating Attack Paths?

Spoiler, you won't. In reality, everyone has this problem. **In an analysis conducted in 2024, we found that 100% of environments were syncing privileged roles and 70% were syncing super admin roles.**



Privilege Zones takes identity sprawl head-on. By mapping and enforcing boundaries across linked identities, regardless of where they live, Privilege Zones eliminate the hidden chains that make sprawl so dangerous. Visualize and stop the risk.

Least Privilege, Realized

Least privilege was never just about giving each identity the right level of access. It was about ensuring no one could go further than they should. That's always been the real goal, but we've never had a way to verify it at scale.

Privilege Zones completes that picture. It shifts least privilege from an aspirational design principle to an enforceable control. It maps the true reach of privilege across the environment and cuts the hidden threads that connect identities across boundaries.

This isn't an evolution of best practices. It's the realization of what those best practices were trying to achieve all along.

1. <https://learn.microsoft.com/en-us/entra/identity/role-based-access-control/security-planning#ensure-separate-user-accounts-and-mail-forwarding-for-global-administrator-accounts>

OPERATIONALIZING ATTACK PATH MANAGEMENT

Turning Visibility into a Complete Practice

As organizations mature in their identity security posture, they inevitably encounter a crossroads. **It is no longer a matter of whether identity-based attack paths exist in their environments; it is a matter of how to consistently find and remove them while maintaining operational stability.**

This acceptance of an uncomfortable truth acknowledges that determined attackers will eventually find their way around even carefully considered defenses. No perimeter is perfect, no user training is foolproof, and no patch management system catches every vulnerability before it's exploited. This isn't defeatism; it's realism that should direct our defensive strategies.

At this stage, the conversation shifts from awareness to action. The real challenge lies in turning that visibility into something operational—an ongoing, structured, and accountable effort to reduce identity-based risk across the enterprise.

This is where Attack Path Management (APM) becomes a practice, not a point-in-time project. It is where identity graphs and privilege maps evolve from helpful visuals to functional inputs across detection, response, remediation, and governance. The organizations that succeed here are not only identifying problems, they are also solving them at scale.

Consider the strategic advantage this approach offers. While attackers must operate covertly, typically working with incomplete information and limited time, defenders have unfettered access to their own environments. When operationalized, identity APM provides the framework and ability to identify all of an organization's identity attack paths and programmatically eliminate them based on their criticality. The resulting environmental improvements don't just mitigate attack paths; they address the underlying conditions and behaviors that form new attack paths, creating an identity security baseline for identifying net new threats as they arise.

Embedding APM Into Existing Security Workflows

Operationalizing APM begins by meeting teams where they already work. Most organizations don't need a new process, they need better data feeding into existing ones. The fastest path to maturity often comes from integrating attack path intelligence into the systems already in use:



Other identity security tools



SIEMs



Asset inventories



Ticketing tools



SOAR platforms



Risk registers

Expanding Privileged Access Management

A foundational need in an identity security program is to manage the accounts in an environment that require elevated access to perform privileged tasks and activities. PAM solutions provide systematic management and protection of privileged accounts, credentials, and commands used to administer critical systems and applications across an identity environment. When properly deployed, these solutions significantly limit an attacker's ability to move laterally through your network and escalate privileges and provide an essential level of defense.

However, there are limitations to what PAM tools can secure. While organizations can typically identify and protect explicitly granted privileges, it's the incidental privilege—unintended access rights granted through complex group memberships, inheritances, and role assignments—where critical security gaps emerge across Identity infrastructures. This oversight can leave open doors for attackers.

Before you can protect your organization's critical identities and resources, you must first understand where those assets are within your environment and how they can be accessed, directly or indirectly, by any identity. This means visibility beyond the obvious traditional Tier Zero principals like *Domain Admins* and *Enterprise Admins*, and into accounts that have elevated privileges by delegation. Visualizing identity attack paths throughout identity systems enables defenders to uncover the identity attack paths that attackers could exploit to move laterally, cross into hybrid environments, escalate privileges, and create persistent backdoors for future access.

Add Prevent to Detect and Respond

ITDR is a security discipline combining threat intelligence, tools, and processes to protect identity systems. ITDR solutions have become essential for identifying, investigating, and responding to advanced threats targeting user identities and Identity and Access Management (IAM) systems. These solutions provide critical insights into user activities, patterns, and behaviors while automating responses to disrupt and contain threats, once detected. However, organizations are discovering significant gaps in their ITDR coverage that leave them vulnerable to sophisticated attacks.

The problem is that ITDR solutions are inherently reactive. They address threats that are already in motion, but they don't provide a framework for addressing identity risk at its foundation.

Identity defense requires understanding the pathways available to an attacker, and how to remove them proactively and safely. By continuously measuring and eliminating the underlying identity risk in an environment, defenders can force attackers to choose from limited options, risking detection and exposure. The problem comes down to potential energy versus kinetic energy. It is far easier to manage risk at rest when options and planning are possible than it is once an attack is in motion, and options are fewer, far less appealing, and of greater consequence.

Attack Path Management, the Proactive Approach

When attack path data is treated as another layer of organizational telemetry, like a CVE feed or misconfiguration alert, it can trigger actions automatically. For example, when a new path to a critical asset is discovered, a ticket can be created and routed via ITSM. That same path may also update the enterprise risk register, feeding into GRC dashboards and compliance reports. This ensures that attack path risk doesn't live in isolation and instead becomes part of the operational fabric.

Defining Ownership for Remediation

Visibility without ownership breeds inaction. Attack paths, by their nature, often cross domains. A path might originate with an endpoint misconfiguration, pass through nested security groups, and end with a misconfigured service account running a critical application. Without clear handoffs and expectations, it's easy for teams to assume "someone else" is responsible.

Effective APM programs establish defined ownership models, mapping specific classes of paths to specific teams, whether that's IAM, infrastructure, application security, or incident response.

They also define what constitutes a critical exposure (e.g., a broad path to Tier Zero) and what the expectations are for response. Ideally, these expectations are backed by predefined remediation playbooks that reduce ambiguity and encourage timely, repeatable actions.

Validating Operational Impact: Balancing Risk Reduction With Business Continuity

Attack path remediation is a security action and it's a change management event. Many paths are deeply embedded in business-critical systems and removing them without understanding operational consequences can cause outages, disrupt workflows, or introduce new risks.

This is why validating the impact of remediation must be a core component of any APM program. It's not enough to know that a path leads to a high-value target. Organizations must also know what business applications rely on the identities or access paths involved, what teams own and operate those systems, and what risks would be introduced by breaking or altering access.

Answering these questions requires coordination across multiple stakeholders, including:

- **Business application owners** who can validate usage patterns and dependencies
- **Helpdesk and support teams** who may identify past service tickets tied to the identity or access in question
- **Change advisory boards** who govern the timing and scope of operational changes
- **Compliance and audit stakeholders** who must understand the trade-offs and document decision-making

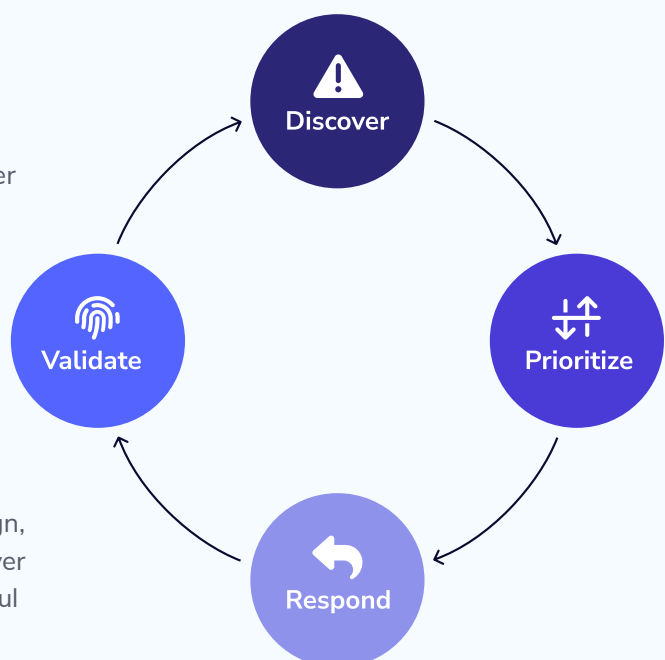
A mature APM program does not treat remediation as a purely technical fix. It treats it as a business decision, informed by security context and governed by operational impact. This ensures that risk reduction does not come at the cost of reliability, and that cross-functional stakeholders are engaged in a structured, predictable way.

Organizations that adopt this mindset see higher success rates in remediation efforts and reduced rollback incidents. They also build stronger trust between security and operational teams, transforming APM from a source of friction into a source of alignment.

Creating a Feedback Loop for Remediation Validation

No remediation effort is complete without validation. In APM, this means verifying that an identified path no longer exists after a fix is applied and reopening the issue if it does. In practice, this requires a continuous or scheduled reassessment cycle to monitor for regressions and ensure long-term success.

An effective APM program doesn't stop at individual fixes. It also tracks metadata about the paths that reappear, revealing deeper structural issues. If the same types of paths are being reintroduced repeatedly, the problem likely lies in provisioning automation, group design, or access governance—not in individual user behavior. Over time, this feedback loop becomes one of the most powerful diagnostic tools for systemic identity risk.



Measuring Program Impact

Like any security discipline, APM needs metrics to track progress, justify investment, and drive accountability. Mature programs typically focus on a blend of tactical and strategic indicators. These include:

- Total number of attack paths to Tier Zero or other crown jewel assets
- Mean time to remediate (MTTR) for new or regressed paths
- Percentage of paths remediated versus accepted or deferred
- Distribution of remediation effort across operational teams
- Recurrence rate of previously remediated path types

Together, these metrics not only show whether the program is working, they reveal where it is stalling, where improvements are most needed, and what kind of organizational change is required to improve posture over time.

Anchoring APM in Policy and Governance

One of the most overlooked elements of operational APM is the role of policy. While automation can surface risks, and metrics can drive accountability, policy is what gives the program its staying power.

Leading organizations formalize APM expectations within broader identity and risk policies. These policies define what an attack path is, how it is classified, what response times are expected based on severity, and who owns resolution. Policy also connects APM to audit and compliance, ensuring that attack path exposure is not just treated as a technical issue, but as a business risk that demands transparency and rigor.

In this context, governance is not a barrier to speed but a mechanism for sustainability. It ensures that as teams shift, budgets change, and priorities evolve, the core practices of attack path remediation remain intact.

Addressing Identity Security Debt

No Attack Path Management program starts with a blank slate. Most organizations have years (or even decades) of accumulated identity complexity: stale permissions, legacy group structures, and ad-hoc exceptions. This identity security debt is the raw material that attack paths are built from.

Operationalizing APM means acknowledging that some of this debt must be addressed strategically, over time. It means segmenting remediation efforts into manageable projects, whether by business unit, privilege zone, or path criticality, and working through them in an intentional, coordinated fashion. It also means accepting that in many cases, the goal is not to eliminate every possible path immediately, but to prioritize and chip away at the ones that pose the most risk.

When handled this way, APM becomes not just a security program, but a lever for improving architectural hygiene across the organization.

Execution Is the Differentiator

Attack Path Management is not a new idea, nor is it a niche capability. What separates high-performing security organizations from the rest is not that they can see their attack paths, but that they've built the muscle to act on them.

By integrating attack path intelligence into existing workflows, defining clear ownership, establishing validation loops, tracking meaningful metrics, and reinforcing it all with policy, organizations can transform APM from insight to impact.

That's where the real work begins and where risk reduction actually happens—not when a path is found, but when an organization takes action.

UPDATES TO GRAPH EXPANSION

A Review

The BloodHound graph is always expanding. It started with just three node types and a few edges in 2016; now we have over 30 nodes and more than 130 edges. This expansion is a result of two primary factors:

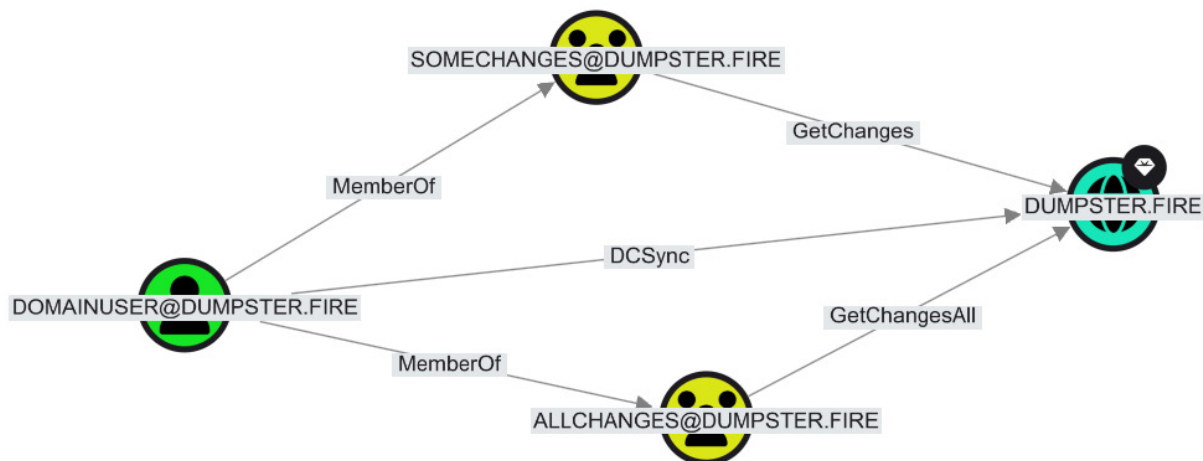
- A dynamic threat landscape with new attack primitives discovered regularly by our research team and the infosec community at large
- Decades' worth of adversary tradecraft that we distill to high-fidelity nodes and edges

In this article, I will review some of the notable additions to the graph from the past year, encompassing hybrid paths, Active Directory trusts, and NTLM relay attacks.

There Is More Than Meets the Eye

Most BloodHound users primarily interact with the “Explore” view to run pathfinding queries or analyze inbound or outbound access, which displays traversable edges. These edges represent a control relationship that an attacker can abuse to take over the destination node. However, our graph expansion efforts also include non-traversable edges and node properties that aren't immediately visible in path finding.

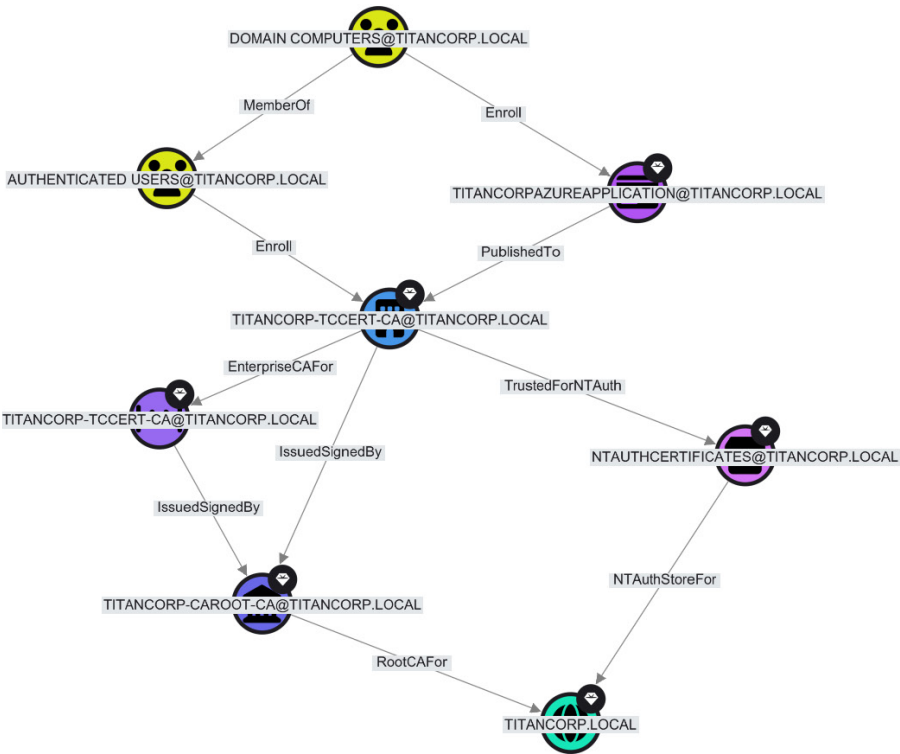
Many attack techniques have more than one prerequisite, and these non-traversable edges and properties enable us to represent those individual prerequisites and to subsequently determine whether all the conditions for executing the attack are met. The classic example of such an attack technique is DCSync, which exploits the Active Directory replication API to dump credential material directly from a DC. The DCSync attack requires two permissions: (1) **GetChanges** and (2) **GetChangesAll**. When a security principal holds both permissions, the attack is viable. BloodHound represents the individual **GetChanges** and **GetChangesAll** permissions as non-traversable edges and creates a traversable DCSync edge only when both non-traversable edges exist.



DCSYNC Prerequisites

BloodHound creates such traversable edges in Post Processing, after the ingestion process finishes creating the simpler nodes and edges. The AD CS attacks that were added to the graph over the past couple of years heavily rely on creating traversable edges in Post Processing. As we model increasingly complex attacks, non-traversable edges and post-processed edges become some of the graph’s most important pillars.

Some of the post-processed edges have a composition view that you can explore by clicking the “Composition View” button in the edge’s entity panel. The composition view displays the nodes and edges that represent all the prerequisites for abusing the edge. This provides both attackers and defenders with everything they need to know.



ADCSESC1

Relationship Information

Source Node:

DOMAIN COMPUTERS@TITANCORP.LOCAL

Target Node:

TITANCORP.LOCAL

Last Seen by BloodHound:

2025-06-04 08:16 CDT (GMT-0500)

General

Windows Abuse

Linux Abuse

OPSEC

References

Composition

The relationship represents the effective outcome of the configuration and relationships between several different objects. All objects involved in the creation of this relationship are listed here:

DOMAIN COMPUTERS@TITANCORP.LOCAL

TITANCORP.LOCAL

TITANCORP-CAROOT-CA@TITANCORP.LO...

TITANCORP-TCCERT-CA@TITANCORP.LO...

TITANCORP-TCCERT-CA@TITANCORP.LO...

AUTHENTICATED USERS@TITANCORP.LO...

TITANCORPAZUREAPPLICATION@TITAN...

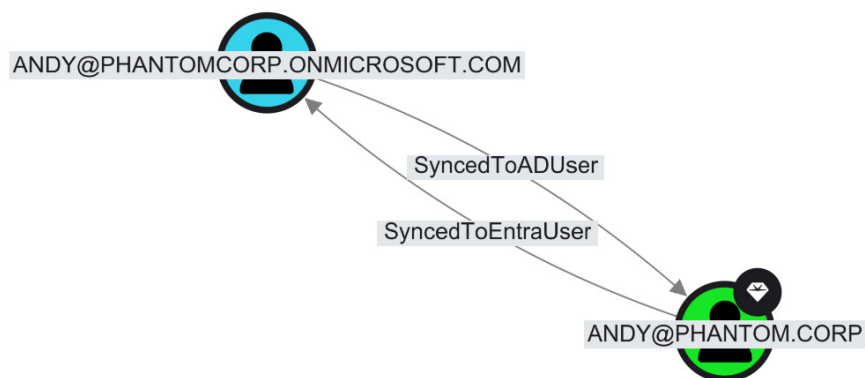
NTAUTHCERTIFICATES@TITANCORP.LOC...

ADCS ESC1 Composition View

As Above, So Below

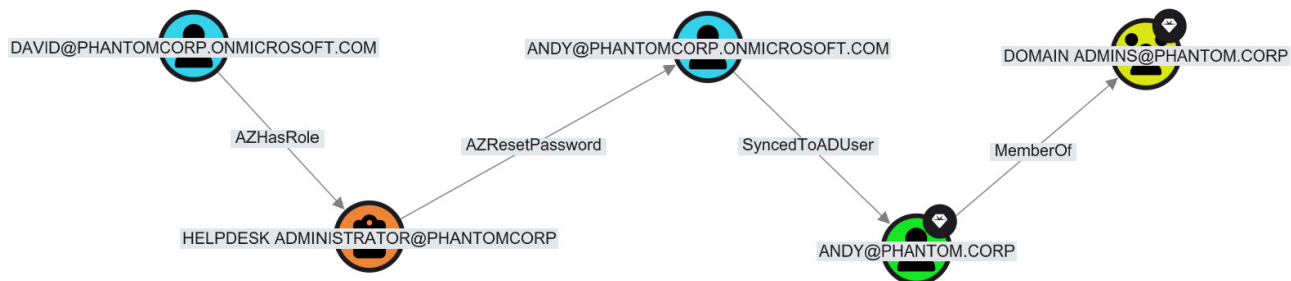
When we initially introduced Entra/Azure elements into BloodHound, they formed a separate subgraph, with no edges connecting the on-premises Active Directory subgraph and the cloud Entra/Azure subgraph. While the Active Directory subgraph almost always contained critical attack paths due to decades of technical debt, the Entra/Azure subgraph was often less interesting. Until we introduced the first two “hybrid paths” representing user synchronization from Active Directory to Entra (**SyncedToEntraUser**) and vice versa (**SyncedToADUser**).

The **SyncedToEntraUser** edge represents the ability to authenticate to the Entra account using the corresponding Active Directory account credentials, provided password hash synchronization, pass-through authentication, or seamless single sign-on (SSO) is enabled. The **SyncedToADUser** represents the ability to authenticate to the Active Directory account using the corresponding Entra account credentials, provided password write-back is enabled.



Active Directory ↔ Entra ID Account Synchronization

The moment we ran pathfinding queries with these new edges, the results were beautiful (beauty is in the eye of the beholder, right?). We identified paths that go from on-premises Active Directory to privileged, synchronized Entra accounts, and then back down to privileged, synchronized Active Directory accounts, and so on. Environments with no path to *Domain Admin* suddenly had many. Environments with a rather boring Entra/Azure graph now had paths to *Global Admin*. In some environments, we found “there and back again” kind of paths that even crossed Active Directory forest security boundaries when multiple disparate forests were synchronized to the same Entra tenant.



Hybrid Attack Path

These discoveries have piqued our appetite for more, and we are now planning to introduce additional edges that abuse hybrid-joined/Intune-managed devices, edges that further abuse the synchronization mechanism, and perhaps edges that abuse Cloud Kerberos Trust.

A key lesson we learned from this development is that whenever we expand the graph to new platforms or technologies, we must always strive to include at least one hybrid path to avoid creating disparate subgraphs in the future.

Relationships Built on Trust

Recently, we completely redesigned how BloodHound models Active Directory trusts. Active Directory trusts allow principals from a trusted domain or forest to authenticate to resources in the trusting domain or forest. Authentication alone, though, is not enough for taking over a resource. It also requires authorization (permissions) to do so, or the ability to impersonate someone with such access.

The previous way of modeling Active Directory trusts used traversable edges to represent trust relationships, which often resulted in false positives. As explained above, trust relationships enable authentication, and authentication alone is not sufficient for taking over a resource. After all, the ability to authenticate to a service does not grant the ability to completely control it. So, should trusts be non-traversable edges?

The Domain Is Not a Security Boundary

It has long been established that trust relationships between domains within the same forest are not a security boundary.¹ The domain, in a sense, is merely a container. If an attacker compromises one domain in the forest, it is trivial to escalate privileges to *Enterprise Admin* and compromise the entire forest through attacks such as SID Hopping,² abusing unconstrained delegation in conjunction with authentication coercion or different ways to abuse the Configuration naming context.

1. <https://blog.harmj0y.net/redteaming/the-trustpocalypse/>

2. <https://specterops.io/blog/2017/10/30/a-guide-to-attacking-domain-trusts/>

Therefore, trust relationships between domains within the same forest, also known as intra-forest trusts, represent more than just authentication; they do indeed represent a control relationship and should be traversable. In the new model, we introduced a new traversable edge called **SameForestTrust** to represent these trusts.

Good Fences Make Good Neighbors

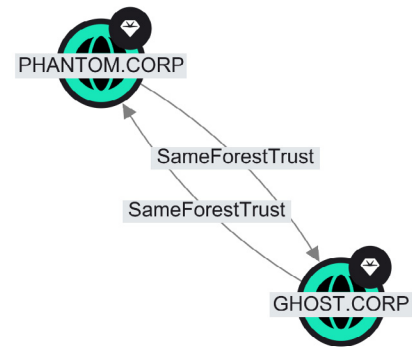
If trust relationships between domains within the same forest aren't a security boundary, then what is? The question almost gives away the answer: The forest is the security boundary. At least by default, trust relationships between forests allow no more than authentication, which is interesting, but by itself, it is not enough for takeover. Therefore, in the new model, we introduced a new non-traversable edge called **CrossForestTrust** to represent trusts that cross a forest boundary.

A Hole in the Fence

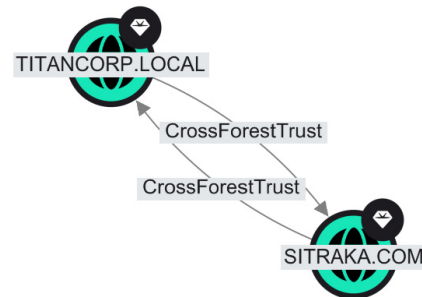
Forest trusts are secure by default and neutralize the known attack techniques that could cross them. Notably, SID history spoofing is neutralized by SID filtering, and unconstrained delegation is disabled across forest trust authentication. However, these defaults can be modified.

When SID filtering is "relaxed," if attackers compromise the trusted forest, they can spoof the SID history of an account in the trusting forest to impersonate privileged accounts in the trusting forest, thereby compromising the trusting forest. This is the most well-known attack technique that abuses forest trusts. In the new model, we introduced a new traversable edge called **SpoofSIDHistory** to represent forest trusts with SID filtering disabled.

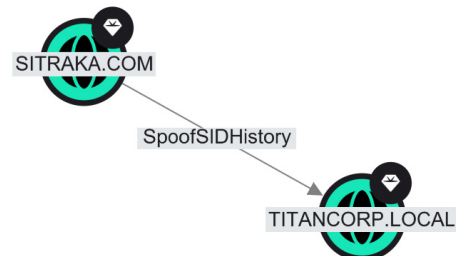
Another less-known, cross-forest trust attack technique abuses unconstrained delegation. If the default configuration is changed to allow unconstrained delegation across the forest trust, attackers can coerce Kerberos authentication from a DC in the target forest to a compromised host configured with unconstrained delegation in the foreign forest to steal the TGT of the DC and take over the entire forest. In the new model, we introduced a new traversable edge called **AbuseTGTDlegation** to represent forest trusts with unconstrained delegation enabled.



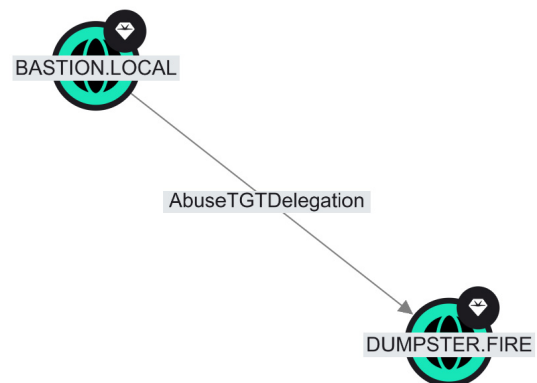
Intra-Forest Trust Relationship



Inter-Forest Trust Relationship



**SpoofSIDHistory Attack
Across Forest Trust**

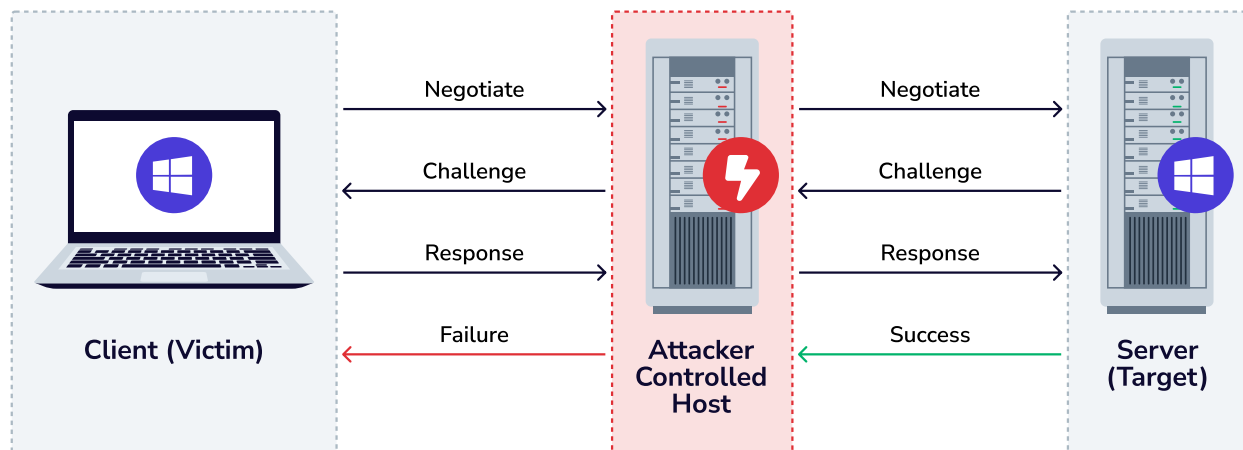


**Unconstrained Delegation Abuse
Across Forest Trust**

NTLM Relay in the Spotlight

NTLM relay attacks have been around for a long time. While many security practitioners think NTLM relay is a solved problem (or at least a not-so-severe one), it is, in fact, alive and kicking and arguably worse than ever before. Relay attacks have become the easiest way to compromise domain-joined hosts, paving the way for lateral movement and privilege escalation.

In a nutshell, instead of cracking passwords or hashes, attackers can simply relay the NTLM exchange between a client and a server until authentication is successful and then abuse the established session to perform any action the client would be allowed to perform on the server.



Over the years, Microsoft has introduced several settings that can mitigate NTLM relay attacks, including signing and channel binding enforcement on various protocols, as well as disabling NTLM altogether. However, many organizations avoid applying these settings at scale due to concerns about compatibility issues that could disrupt operations.

We decided to add NTLM relay attacks to BloodHound to put a spotlight on the problem, highlight the real risks organizations face, and help them develop more precise and effective remediation strategies than “enforce everything, everywhere.”

From Zero to Hero

NTLM relay attacks come in different variations. We chose to focus, at least initially, on the combination of authentication coercion and NTLM relay to SMB, LDAP/LDAPS, and AD CS Web Enrollment. *Authenticated Users* can coerce authentication from any Windows host using primitives such as The Printer Bug¹ and PetitPotam,² triggering an authentication attempt from the host's computer account to an attacker-controlled listener. From there, the attacker can relay the NTLM messages to a vulnerable server that the coercion victim is permitted to access.

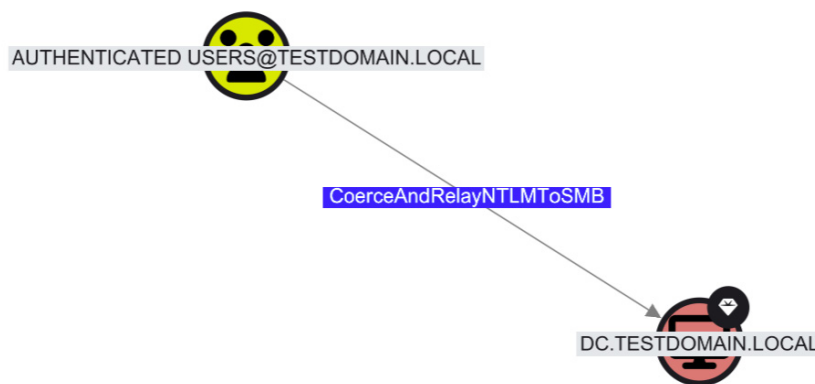
This pattern means that attackers with virtually the lowest level of access (i.e., *Authenticated Users*) can compromise affected domain-joined hosts. We often see Tier Zero hosts vulnerable to such attacks, meaning that anyone can compromise the entire environment, resulting in 100% exposure and 100% impact.

CoerceAndRelayNTLMToSMB

The new **CoerceAndRelayNTLMToSMB** edge is the simplest of the new NTLM relay edges. The edge always originates from the *Authenticated Users* node and leads into the target computer node. It represents a combination of computer account authentication coercion against the relay victim (the client) and an NTLM relay attack against the relay target (the server).

1. <https://www.thehacker.recipes/ad/movement/mitm-and-coerced-authentications/ms-rprn>

2. <https://www.thehacker.recipes/ad/movement/mitm-and-coerced-authentications/ms-efsr>

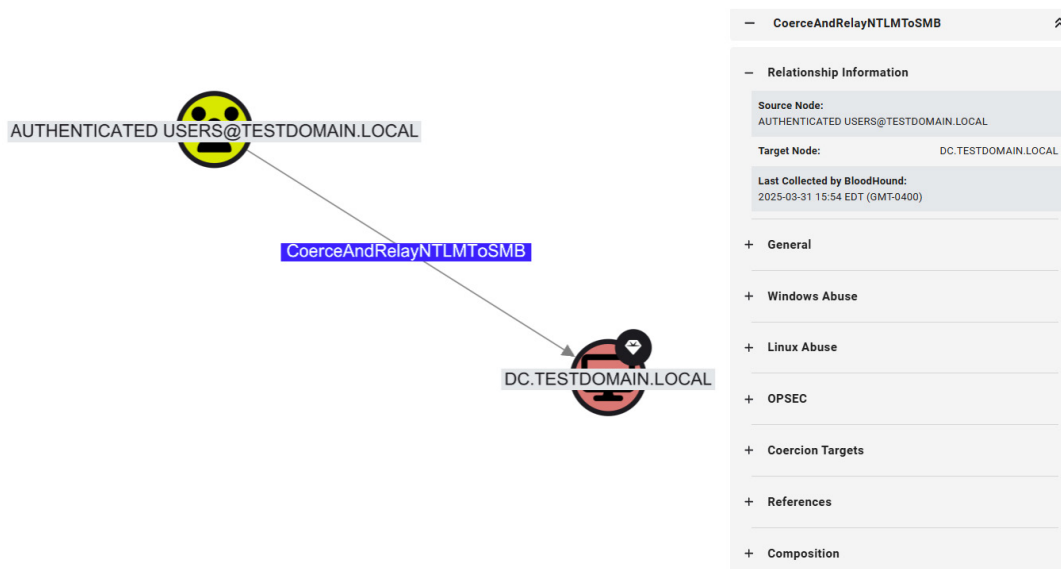


Authentication Coercion and NTLM Relay to SMB

This edge represents NTLM relay to SMB. If the relay victim has privileged access to the relay target, the attacker can gain access to the C\$ or ADMIN\$ share, dump LSA secrets from Remote Registry, including the computer account password, or move laterally via the Service Control Manager.

The prerequisites for this attack are minimal: only that SMB signing is not required on the target server, and at least one computer in the environment, any computer, to have admin access to the target and act as the relay victim (the client).

Expanding the Coercion Targets accordion in the entity panel shows the clients from which the attacker needs to relay to the target server.



NTLM Relay Coercion Targets

This scenario is very common with SCCM deployments and represents the attack known as TAKEOVER-2.¹

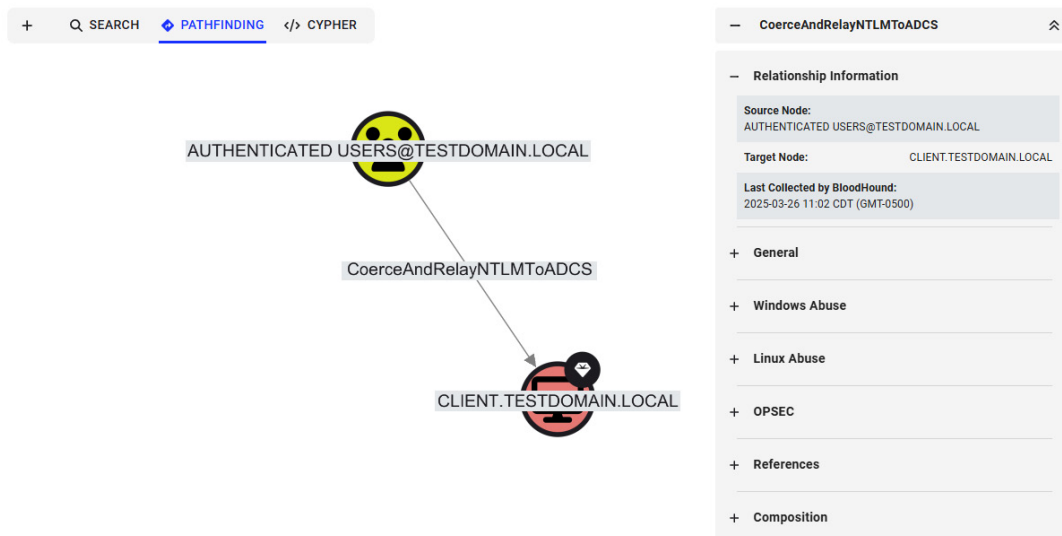
CoerceAndRelayNTLMToADCS

This is the infamous ADCS ESC8 attack, which Will Schroeder and Lee Chagolla-Christensen disclosed in their “Certified Pre-Owned” white paper.²

The new **CoerceAndRelayNTLMToADCS** edge originates from the *Authenticated Users* node and leads into the victim computer node (the client), unlike **CoerceAndRelayNTLMToSMB**, which leads into the relay target computer node (the server). The reason for the difference is that the attack compromises the relay victim rather than the relay target.

1. https://github.com/subat0mik/Misconfiguration-Manager/blob/main/attack-techniques/TAKEOVER/TAKEOVER-2/takeover-2_description.md

2. <https://posts.specterops.io/certified-pre-owned-d95910965cd2>

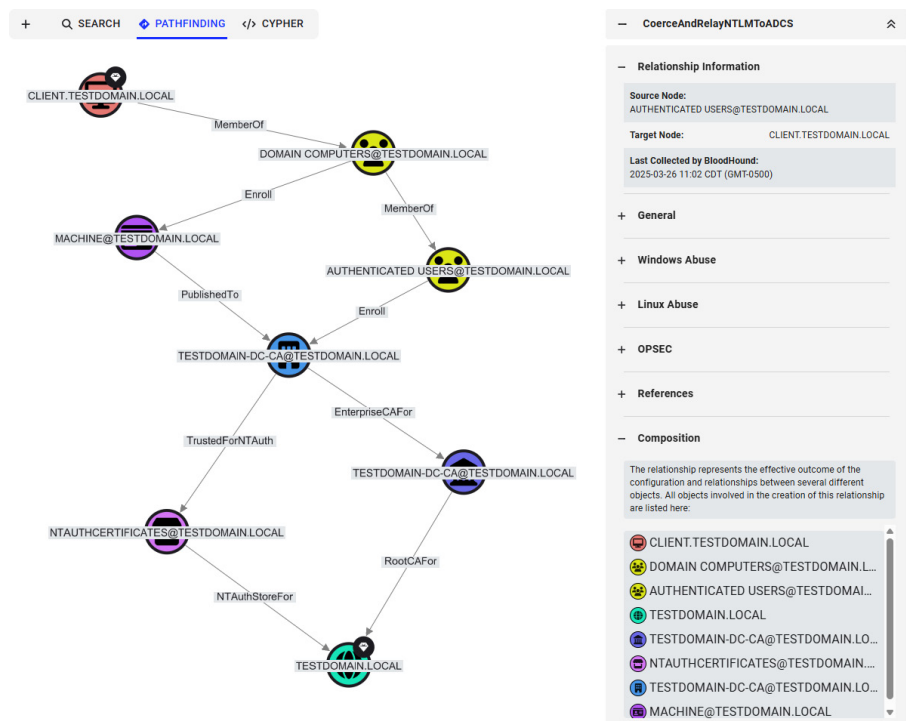


Authentication Coercion and NTLM Relay to ADCS

This edge represents NTLM relay to the ADCS Web Enrollment endpoint. If the relay victim is permitted to enroll a certificate that allows Kerberos/Schannel authentication, the attacker can use such a certificate to compromise the relay victim host via S4U2Self¹ abuse or a silver ticket,² or access any resource that the computer account is authorized to access.

The prerequisites for this attack are many and involve permissions and settings on the certificate authority (CA), the certificate template, and even the domain. The secret ingredient for the relay component specifically is that the affected enterprise CA must have a Web Enrollment endpoint available over HTTP, or over HTTPS with Extended Protection for Authentication (EPA) disabled.

Expanding the composition view shows these prerequisites.



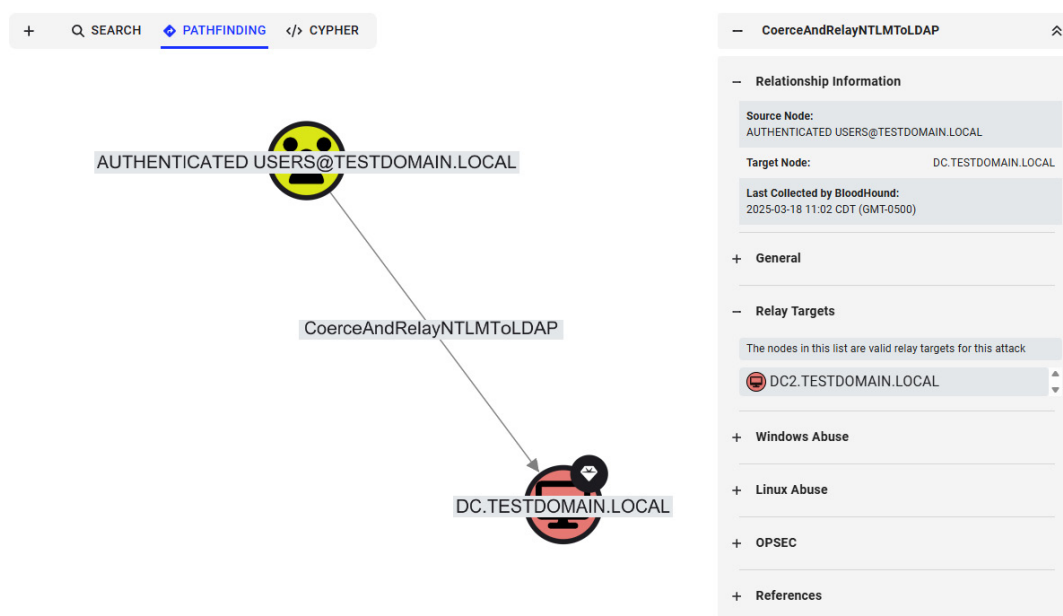
Authentication Coercion and NTLM Relay to ADCS Composition View

1. <https://www.thehacker.recipes/ad/movement/kerberos/delegations/s4u2self-abuse>

2. <https://www.thehacker.recipes/ad/movement/kerberos/forged-tickets/silver>

CoerceAndRelayNTLMToLDAP/LDAPS

The new **CoerceAndRelayNTLMToLDAP** and **CoerceAndRelayNTLMToLDAPS** edges are like the **CoerceAndRelayNTLMToADCS** edge, which originates from the *Authenticated Users* node and leads into the victim computer node (the client). These edges are far more complicated to identify because they require not only the server to be vulnerable, but also the client. The gist of it is that the client must have the WebClient service running to coerce authentication without session security, and the server must not require either LDAP signing or LDAP channel binding (either one is sufficient).



Authentication Coercion and NTLM Relay to LDAP

These edges represent the ability to establish an LDAP session as the affected computer account and then take over the host via resource-based constrained delegation (RBCD), as explained in detail in this post,¹ or take over the host via the Shadow Credentials attack, as explained here.²

The entity panel shows the affected DC under the Relay Targets accordion.

Better Remediation Strategies

The remediation guidance for NTLM relay attacks is often “enforce everything, everywhere,” which is not very practical in a large environment that requires backward compatibility. However, BloodHound now helps defenders identify what is actually viable in their environments and prioritize high-impact/high-exposure targets. BloodHound has a set of pre-built cypher queries that can get you started with that.

1. <https://shenaniganslabs.io/2019/01/28/Wagging-the-Dog.html>

2. <https://posts.specterops.io/shadow-credentials-abusing-key-trust-account-mapping-for-takeover-8ee1a53566ab>

NTLM Relay Attacks
All coerce and NTLM relay edges
ESC8-vulnerable Enterprise CAs
Computers with the outgoing NTLM setting set to Deny all
Computers with membership in Protected Users
DCs vulnerable to NTLM relay to LDAP attacks
Computers with the WebClient running
Computers not requiring inbound SMB signing

NTLM Relay Pre-Built Queries

More Relay is Coming

We plan to cover additional relay scenarios in the near future, including targeting MS SQL Server, which will reveal SCCM TAKEOVER-1¹ exposures, and Kerberos relay² to highlight that disabling NTLM and switching to Kerberos is not the solution.

In The Works

Our graph expansion efforts are in full force. We are committed to continually improving our coverage for Active Directory and Entra/Azure, while also expanding our focus to new platforms and technologies. We have already completed the design work for SCCM and Intune, and these are now in the implementation pipeline. We are also researching technologies that so far haven't gotten much attention from the offensive security community, such as Ping Identity and Jamf.

These efforts require meticulous work to ensure no false positives make their way into the graph, giving you confidence that when you see a traversable edge in BloodHound it is indeed abusable.

1. https://github.com/subat0mik/Misconfiguration-Manager/blob/main/attack-techniques/TAKEOVER/TAKEOVER-1/takeover-1_description.md
2. <https://googleprojectzero.blogspot.com/2021/10/using-kerberos-for-authentication-relay.html>

Trends in Identity Tradecraft from Operations

The security landscape has fundamentally shifted, and identity is undeniably the new perimeter. As organizations race to secure their identities, a critical security gap often persists. The on-premises environment and its interconnections with cloud and SaaS systems remain a weak and critical point.

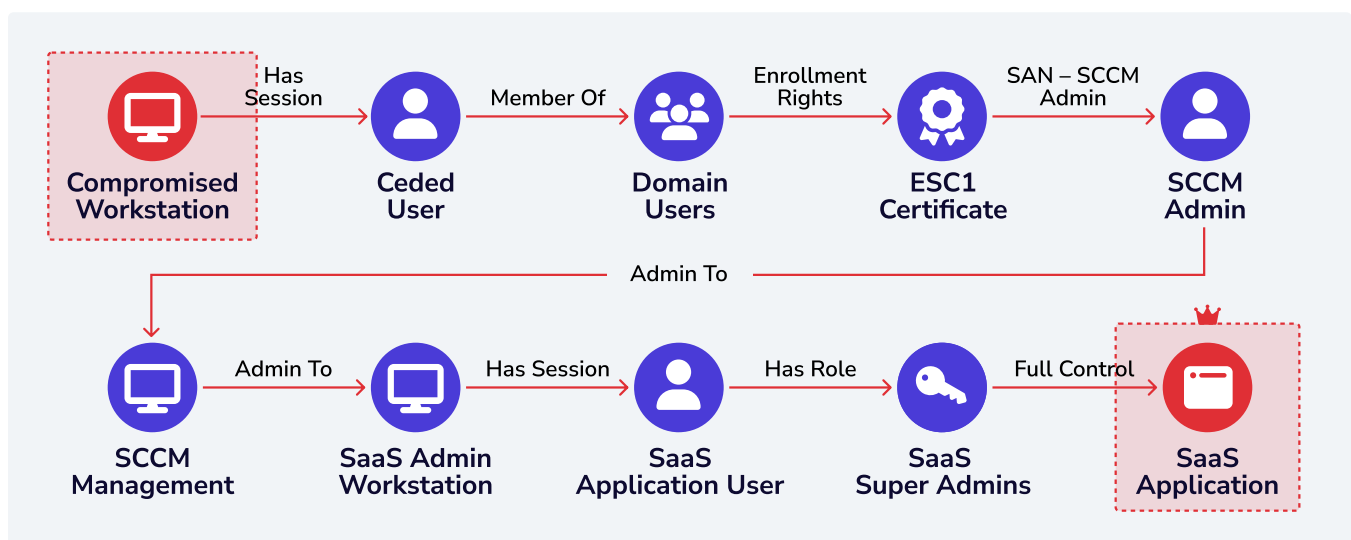
SpecterOps engagements demonstrate that these on-premises systems, far from being relics of a bygone era, serve as potent launchpads for sophisticated attacks that bridge the gap into cloud and SaaS environments. Even as the “perimeter” itself becomes more abstract, on-premises environments remain a juicy target.

It’s within these on-premises complexities and their hybrid connections that attackers thrive. It is often said that attackers live in the gaps between what “should be” and what “is.” Our consultants consistently find these gaps hidden in the complex web of identity relationships; they’re often overlooked, misconfigured, and violating foundational security concepts (such as the clean source principle¹).

As red team operators, we increasingly find ourselves targeting not only identities “at rest” (e.g., passwords or hashes), but also “in transit” (e.g., cookies, sessions, tokens, etc.), leveraging techniques like browser cookie theft to hijack active sessions. This is foundational tradecraft in today’s hybrid environments, allowing attackers to bypass protections like MFA and CAPs to seamlessly pivot from compromised on-premises endpoints to highly privileged cloud and SaaS roles (and back).

The below war stories are not theoretical; they are real attack paths from SpecterOps’s own offensive security engagements. Each of these stories illustrates a successful compromise that was built on a foundation of identity attack paths.

War Story: Unprivileged Active Directory User to SaaS Administrator



1. [The Security Principle Every Attacker Needs to Follow](https://specterops.io/blog/2024/07/17/the-security-principle-every-attacker-needs-to-follow/), Elad Shamir, <https://specterops.io/blog/2024/07/17/the-security-principle-every-attacker-needs-to-follow/>

Our objective was to infiltrate a critical SaaS application responsible for offline network storage backups. The challenge was heightened by the organization's security measures: The SaaS application's credentials were deliberately segregated from Active Directory, eschewing any SSO convenience, and robust MFA was enforced. Our attack hypothesis was simple. Could a compromise of Active Directory pave the way to a SaaS administrator, ultimately leading to the application's compromise? We aimed to prove that even with these layers, the path from internal domain control to cloud application compromise was not only possible but plausible.

Active Directory Certificate Services Exploitation

As it often does, it started with a single foothold via a Mythic C2¹ agent, a standard user account on a domain-joined workstation found via the result of a successful phishing campaign. Standard AD recon revealed ADCS was in play. A quick query with Certify² and we hit our first jackpot: A certificate template was misconfigured with "Client Authentication," "Enrollee Supplies Subject," and, crucially, enrollment rights for all domain users. This was the classic ESC1³ vulnerability and an easy win for us.

Certify Output Indicating an ESC1 Vulnerability

```

Template Name                               : Cisco
Display Name                               : Cisco
Certified Authorities                       : Domain CA 2
Enabled                                     : True
Client Authentication                       : True
Enrollment Agent                           : False
Any Purpose                                : False
Enrollee Supplies Subject                   : True
Certificate Name Flag                       : EnrolleeSuppliesSubject
Enrollment Flag                             : IncludesSymmetricAlgorithms
Private Key Flag                           : ExportableKey
Extended Key Usage                         : Client Authentication
Requires Manager Approval                   : False
Requires Key Archival                      : False
Authorized Signatures Required              : 0
Validity Period                            : 5 years
Renewal Period                             : 6 weeks
Minimum RSA Key Length                     : 20248
Permissions
    Enrollment Permissions
        Enrollment Rights                   : TARGETDOMAIN.COM\\Domain Admins
                                           TARGETDOMAIN.COM\\Domain Users
                                           TARGETDOMAIN.COM\\Enterprise Admins

```

Privilege Escalation to SCCM

Our next objective was to gain broader administrative access. SCCM admins are always a juicy target, given their ability to manage a vast swath of hosts. An LDAP query quickly identified the SCCM Admins group and its members. We picked our target (let's call it SCCM_Admin_User1) and our compromised low-privilege user utilized Certipy⁴ to request a certificate via the vulnerable "UniversalSigner" template but with a twist. We supplied "SCCM_Admin_User1" as the alternate name. Just like that, we had a certificate that allowed us to impersonate an SCCM administrator.

1. <https://github.com/its-a-feature/Mythic>

2. <https://github.com/GhostPack/Certify>

3. <https://posts.specterops.io/certified-pre-owned-d95910965cd2>

4. <https://github.com/ly4k/Certipy>

Pivoting to the SaaS Application Administrator Workstation

With SCCM admin rights, the network started to open. Our ultimate target was a critical SaaS application responsible for sensitive data storage. Internal documentation scraped from an anonymously accessible Confluence page pointed us to the “Storage and Backup Engineers” group as the gatekeepers. Another LDAP search gave us a list of their usernames.

Now it was time to reach one of them. We accessed the SCCM administration console and using the “Devices” overview, we located the active workstation of a `Backup_Admin_User1` (let’s call it `BU1-WORKSTATION`). We needed to get our C2 running in their user context. A new Mythic payload was staged on an open network share. Then, leveraging SharpSCCM,¹ we deployed our payload to `BU1-WORKSTATION`, ensuring it executed as the currently logged-on `Backup_Admin_User1`.

SaaS Application Compromise via Session Hijacking

We were operating as “`Backup_Admin_User1`” on the user’s own machine. Local host enumeration showed they were actively using the target SaaS application via Microsoft Edge. This was our moment. We programmatically killed their existing Edge processes and relaunched the browser, but this time with the remote debugging port enabled and instructing it to restore the last session.

Relaunching Edge with the Remote Debug Port

```
> exec_process "C:\PROGRA~2\Microsoft\Edge\Application\msedge.exe"
  --args "--remote-debugging-port=9222 --remote-allow-origins=* --restore-last-session" --ppid 16096
[+] Process created successfully
    ProcessId:      29908
    ProcessName:    C:\PROGRA~2\Microsoft\Edge\Application\msedge.exe
    ProcessArgs:    --remote-debugging-port=9222 --remote-allow-origins=* --restore-last-session
    ParentProcId:   16096 (explorer.exe)
```

To avoid direct network traffic from our C2 infrastructure to the SaaS app (which might trigger alerts), we established a SOCKS proxy through our agent on `BU1-WORKSTATION`. All our subsequent traffic to the cloud application would now appear to originate from the legitimate admin’s workstation. Using scripts from the Cuddlephish² framework, specifically `smooth_criminal.js`, we connected to Edge’s remote debugging port and siphoned off the session cookies and local storage data for the SaaS application.

Finally, with another Cuddlephish script, `stealer.js`, we loaded these stolen session cookies into our own Chromium browser. Because the cookies already contained a valid MFA claim, we bypassed all MFA prompts and gained direct, authenticated access to the critical SaaS application, inheriting the “super admin” privileges of `Backup_Admin_User1`. At this point, we had achieved our objective to obtain full control of the cloud backup solution from an unprivileged AD user.

Wrap Up

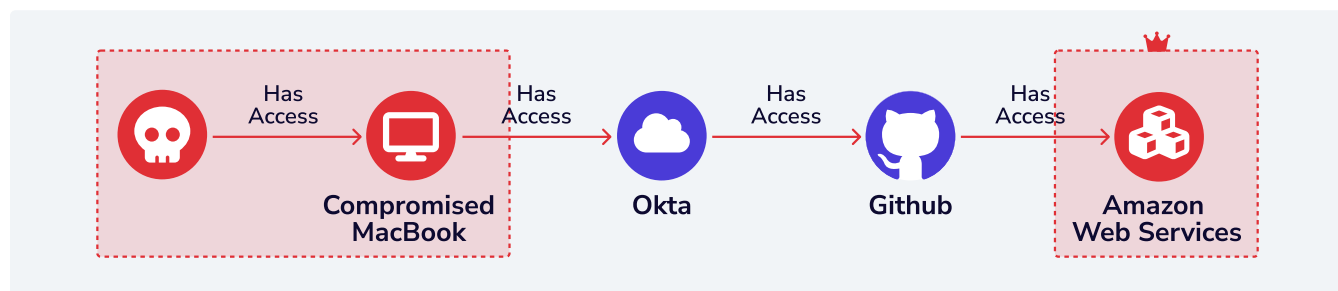
This engagement perfectly illustrated how chained identity vulnerabilities (from an overly permissive AD certificate template to leveraging privileged SCCM console access for lateral movement, and finally to browser session hijacking) can unravel an organization’s security posture and lead to the compromise of critical cloud-based assets. You can dive deeper into this attack path by watching our recorded presentation³ from SO-CON 2025.

1. <https://github.com/Mayhem/SharpSCCM>

2. <https://github.com/fkasler/cuddlephish>

3. <https://www.youtube.com/watch?v=hQk2Eb9ZsnM>

War Story: Pivoting Through Okta and GitHub to AWS



During a recent red team assessment, we were amazed to see just how far pulling on the thread of integrated identity systems could take us.

Let's set the scene. We had gained access to a developer's MacBook and the objective of the assessment was to access the AWS organization which housed sensitive data for the target organization. The usual search for stored credentials in `~/.aws/credentials` failed to surface anything useful and we needed to look for further options.

Accessing Okta

Extracting cookies from the running Google Chrome session was simple enough. By opening a debug port¹ and using the `Storage.getCookies` method, we had a snapshot of session cookies² which quickly provided access to the organization's Okta tenant that our victim user authenticated to.

This is where things got challenging. Although we could interact with the Okta dashboard, attempting to access any integrated web application resulted in a request for MFA. This left us at a fork in the road. We could either attempt to socially engineer the victim user to consent to an MFA request or spend some time looking for a workaround. Thankfully, time was on our side, so we decided to look for alternate paths before taking the high-risk social engineering route. As any seasoned operator knows, there is usually more than one path to achieving your objective.

Ultimately, we turned our sights on the Okta Verify application.³ We knew from our previous research that, given a certain configuration, this could provide a workaround for MFA. After all, nobody likes picking up their phone to accept a push authentication attempt, so many organizations have incorporated solutions such as Okta Verify's macOS application to make life a little easier. Using the tool OktaRealFast,⁴ we intercepted the MFA request and verified that no authentication prompt would be shown to the user. This was enough to access any of the Okta integrated applications.

Reviewing GitHub Enterprise

Of the observed available web applications, GitHub Enterprise stood out as a viable candidate due to our team's history of finding incorrectly committed AWS credentials in source code repositories. Unfortunately, despite a comprehensive few days of enumeration, the available repositories didn't reveal anything useful for moving us closer to the objective.

What stood out, however, was the use of AWS in GitHub Actions via the `aws-actions/configure-aws-credentials` action. While this initially led us to believe that an AWS Access Key was being used (and potentially stored in a secret within the GitHub environment), a separate Git repository containing the organizations DevOps configuration revealed that OpenID Connect (OIDC) had been configured to allow GitHub Actions to assume roles within AWS.

1. <https://posts.specterops.io/hands-in-the-cookie-jar-dumping-cookies-with-chromiums-remote-debugger-port-34c4f468844e>

2. <https://attack.mitre.org/techniques/T1539/>

3. <https://help.okta.com/en-us/content/topics/mobile/okta-verify-overview.htm>

4. <https://github.com/xpn/OktaPostExToolkit>

Using GitHub Actions to Access AWS

For anyone unaware, OIDC is supported by GitHub¹ as a method of accessing AWS by assuming a configured role. Within AWS, the GitHub OIDC provider must be added as an identity provider and configured with a trust policy.

The following is an example trust policy:

```
“Condition”: {
  “StringEquals”: {
    “token.actions.githubusercontent.com:aud”: “sts.amazonaws.com”,
    “token.actions.githubusercontent.com:sub”: “repo:octo-org/octo-repo:ref:refs/heads/octo-branch”
  }
}
```

However, in the case of our target environment, the **sub field** was set to include a wildcard:

repo:organization/repo-name:*

The wildcard within the policy meant that any branch within a repository had permission to assume a preconfigured role within AWS. At first glance, this made sense, because branches of Git repositories also require the ability to execute GitHub Actions. However, the risk is that any user with permission to create a new branch also can assume the AWS role by modifying the GitHub action workflow. The only requirement was for the attacker to identify a GitHub repository with access to an AWS role worth compromising.

Luckily for us, we found such a repository. We used this repository to modify access permissions within AWS across each of its accounts. Although the compromised account didn't have permission to push to the “main” branch, the Git pull-request workflow of the organization allowed our account to create a new Git branch.

Next, to exploit this, we created a new branch (after a day of reviewing existing branch names and coming up with an inconspicuous name) and modified the GitHub action. Since this was a red team assessment, we wanted to mirror the legitimate GitHub action as closely as possible (and certainly didn't want our interactions with AWS to come from an IP address not tied to GitHub), so we decided to move forwards by creating a remotely accessible shell within the running container.

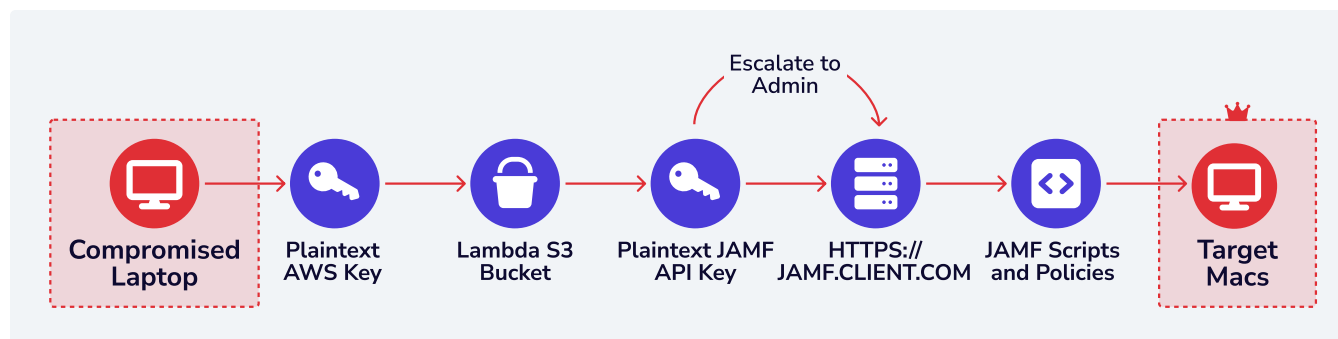
Finally, we pushed the commit, waited for our reverse shell connection, and BINGO! We listed the environment variables and revealed a new AWS session token for an account privileged enough to access the target's AWS organization root account. We had achieved the objective and could now operate without restriction in the AWS organization.

Wrap Up

What we loved about this attack path was how each of the different identity systems played a part. Each system had its strengths and weaknesses, but taking the time to understand how each was misconfigured led to achieving the objective without ever needing an undisclosed exploit. Reading your way around misconfigurations is often all it takes to have a devastating impact on your target and is the go-to strategy for modern red teams operating beyond the bounds of traditional AD environments. This is the reality of modern offensive operations—victory is often found not in an exploit but in the documentation.

1. <https://docs.github.com/en/actions/security-for-github-actions/security-hardening-your-deployments/configuring-openid-connect-in-amazon-web-services>

War Story: Cloud Backup to Jamf Enterprise Compromise



Sometimes, the key to an enterprise isn't a zero-day exploit but a forgotten file. In a recent engagement with a global client, we turned a single, hardcoded AWS key found on a developer's Mac into administrative control over their entire fleet of macOS devices. We regularly assess this client, and this time we were interested in identifying any new attack paths in their environment.

Obtaining Jamf Access

We started with access to one of their macOS laptops via Mythic C2. Upon performing local system enumeration, we quickly discovered a globally readable AWS access key and secret combination being used for centralized log forwarding. We were off to a good start.

Using the newly recovered AWS key, we enumerated our access in the client's AWS environment. We discovered the keys belonged to a principal with permissions to write to an AWS Session Queue Service (SQS) queue used for logging but could not read from it. However, we eventually found the key could be used to enumerate a handful of other S3 buckets, including one labeled for use with AWS Lambda functions.

We began exfiltrating ZIP compressed archives from the AWS Lambda S3 bucket, extracting, and reviewing the contents of the files. The archives contained Python code to perform asynchronous maintenance functions and in some cases API keys for software technologies used by the client. The client later confirmed that they had previously used the S3 bucket for periodic backups of their Lambda codebase and configurations but had since stopped and had not yet deleted the bucket. Buried in one of these old backups, we hit paydirt. We noticed that one of the archives contained Python code which would evaluate macOS attributes within a client hosted Jamf tenant and contained a JSON file with an embedded API token for the associated service account. Next, we copied the API token out of the file and used it to obtain a new JWT token to enumerate the client's Jamf tenant via the API. We were in.

Privilege Escalation via Jamf

Upon issuing an API request to the Jamf Pro tenant, we discovered the service account was restricted to a custom privilege set which possessed 27 update permissions. Within the collection of update permissions was the sensitive "Update Accounts" permission that had been scoped to the global Jamf Pro tenant.

```
“account”: {  
  “id”: 81,  
  “username”: “REDACTED”,  
  ...  
  “privilegeSet”: “CUSTOM”,  
  “privilegesBySite”: {  
    “-1”: [  
      ...  
      Update Accounts,  
      Read Computers,  
    ]  
  }  
}
```

A quick check of Jamf’s developer documentation confirmed our suspicion. We had found the holy grail of IAM misconfigurations: a service account with the power to promote itself to a full-blown administrator. It was a single permission that unraveled everything. We quickly scripted an HTTP “PUT” request to the Jamf API, and just like that, we updated our compromised service account to a full Administrator. The compromised account could now create new scripts and policies using the “Create Script” and “Create Policy” permissions in Jamf.

Lateral Movement

With our new admin rights, we leveraged the Jamf API to enumerate all macOS systems in the environment, identifying users who likely had access to the confidential data we were after. We then pushed malicious scripts and policies targeting these systems that were configured to establish command-and-control (C2) upon execution. We didn’t have to wait long. The next time our target hosts checked in with Jamf, we received new C2 callbacks.

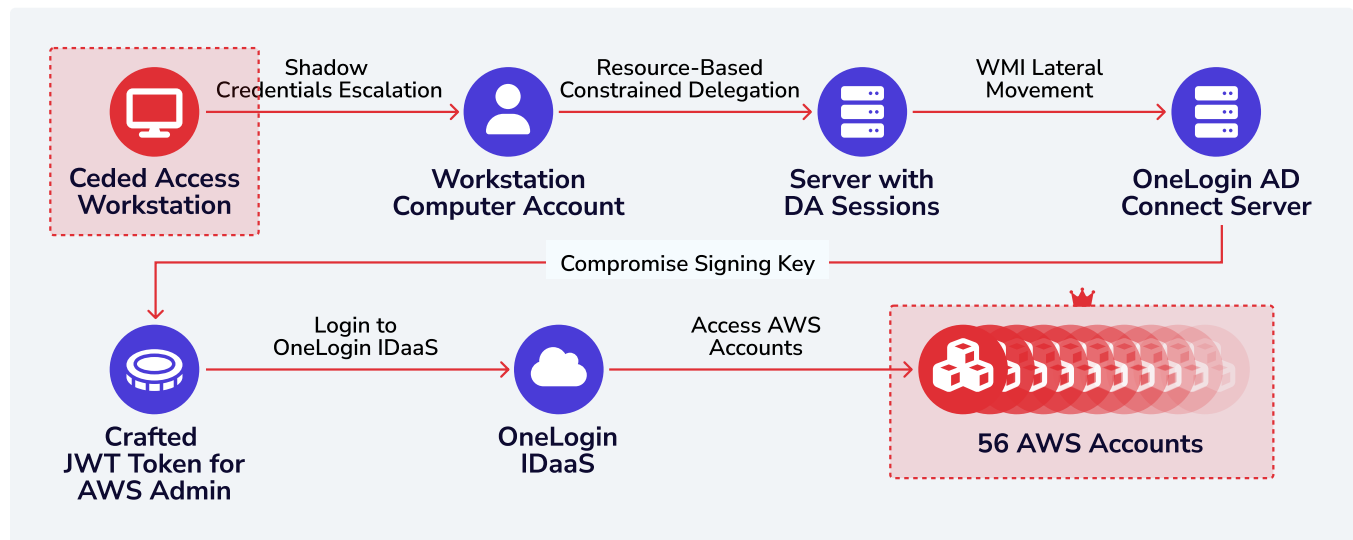
Crucially, because our malicious scripts were deployed via Jamf, they were executed by a trusted management agent installed on the endpoint. This allowed our C2 to establish itself while flying completely under the radar of their fleet-wide EDR solution. From there, we leveraged our new access to enumerate shared documents and identify the confidential materials that fulfilled our assessment objectives.

Wrap Up

This attack path is a powerful illustration of how digital exhaust (in this case, a forgotten S3 backup) can provide the critical link needed to compromise an enterprise. The success of this engagement hinged on two distinct but related failures: improper secrets management in a legacy cloud asset and an overly permissive service account in Jamf that could escalate its own privileges.

It serves as a reminder that effective Attack Path Management must account for these dormant risks and extend beyond user identities to scrutinize the permissions of every service principal, no matter how minor it may seem.

War Story: From Active Directory Compromise to OneLogin to 56 AWS Accounts



How does a standard, unprivileged user on an internal network gain administrative control over an entire cloud environment? On a recent engagement, we demonstrated exactly that, tracing an attack path from a single unprivileged Active Directory account to the full compromise of the client's cloud infrastructure. The cornerstone was not a flaw in the cloud provider itself, but in the third-party service trusted to bridge the on-premises and cloud worlds: OneLogin's AD Connect.

Ceded Access and Local Privilege Escalation

This assessment began with ceded access on a standard windows workstation as a low privileged user on the client's AD domain that was generated explicitly for this assessment.

Enumerating the ceded access workstation and the AD domain showed that an NTLM relay¹ attack was possible because DCs were configured to disable two critical security controls: LDAPS channel binding and SMB signing. With this knowledge, we executed an NTLM relay attack to enroll a new set of attacker-controlled Shadow Credentials² for the ceded access workstation.

Setting Shadow Credentials for Privilege Escalation

```

# clear_shadow_creds DC0: [redacted]
Found Target DN: CN=DC [redacted]
Target SID: S-1-5-21-891 [redacted]

Shadow credentials cleared successfully! ←

# set_shadow_creds DC011 [redacted]
Found Target DN: CN=DC0 [redacted]
Target SID: S-1-5-21-89 [redacted]

KeyCredential generated with DeviceID: 0f078464-8101-3c2f-d433-faad779daf1f
Shadow credentials successfully added!
Saved PFX (#PKCS12) certificate & key at path: tn4IvcaQ.pfx ←
Must be used with password: G4i [redacted]
  
```

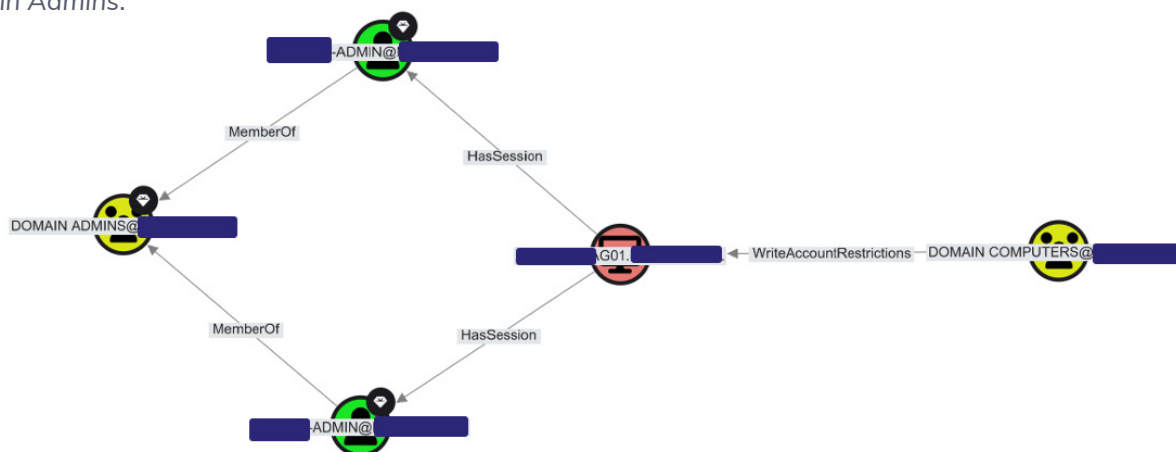
1. <https://specterops.io/blog/2025/04/08/the-renaissance-of-ntlm-relay-attacks-everything-you-need-to-know/>

2. <https://specterops.io/blog/2017/06/17/shadow-credentials-abusing-key-trust-account-mapping-for-account-takeover/>

Using the shadow credentials, we obtained a Kerberos TGT and decrypted it to recover the NT hash for the workstation's AD computer account. The NT hash allowed us to forge valid ticket-granting service (TGS) tickets and use Impacket's WMIexec.py¹ to execute our Mythic agent payload with SYSTEM privileges, gaining full control of the workstation.

Domain Compromise and Escalation to Domain Administrator

After achieving local privilege escalation, we shifted to analyzing the AD environment with BloodHound Community Edition (BloodHound CE). BloodHound quickly identified an attack path from Domain Computers to Domain Admins:



BloodHound Graph Detailing Attack Path from Domain Computers to Domain Administrators

Our analysis revealed a critical privilege escalation path. The Domain Computers group held **WriteAccountRestrictions** permissions on a server where two Domain Admins were logged in. This specific permission allowed us to configure a resource-based constrained delegation (RBCD) attack.² Using **SharpAllowedToAct**,³ we modified the target server's Active Directory object, granting our workstation the ability to impersonate users authenticating to it via the **msDS-AllowedToActOnBehalfOfOtherIdentity** property. We then moved laterally to the server as an administrator and extracted the credentials for both Domain Admin users, leading to full domain compromise.

Compromising AWS through OneLogin's AD Connect Service

With full control of Active Directory, we pivoted to targeting the client's AWS environment. Our early reconnaissance showed they used OneLogin for federated authentication, and BloodHound immediately highlighted a service account with OneLogin in its name. This account had active sessions on three servers, which we discovered ran the OneLogin AD Connect agent.

The OneLogin AD Connect agent synchronizes users in the Active Directory domain with OneLogin's IDaaS product. It installs as a service called "OneLogin Active Directory Connector" and the service executes a .NET executable (i.e., ConnectorService.exe.) This was fortuitous for us since .NET executables are trivial to reverse engineer.

Next, we analyzed the disassembled code of ConnectorService.exe and found a few interesting things:

- It wrote sensitive data, including obfuscated secrets, to world-readable log files on disk at: C:\ProgramData\OneLogin, INC\Logs\
- It validated user identities by taking credentials directly from the OneLogin API and passing them to the local LogonUserW Windows API.

1. <https://github.com/fortra/impacket>

2. <https://shenaniganslabs.io/2019/01/28/Wagging-the-Dog.html>

3. <https://github.com/pkb1s/SharpAllowedToAct>

OneLogin Service Log Entry with Obfuscated Secrets

```
2024-11-22 00:29:52,895|INFO |159|ConnectorConfigurationService - Configuration <=
https://api.onelogin.com/api/adc/v4/configuration?version=5.1.8&token=2057*****
*****0a3h&mux=1&api_key=3a29*****f797&
directory_id=16746&directory_token=2057*****0a3b
&adcVersion=5.1.8
<configuration><api_key>3a29*****f797</api_key><ldap_
server_host>Azuks idp01</ldap_server_host><ldap_server_port></ldap_server_port>
<base_dn></base_dn><directory_id>167
```

At this point, we hypothesized that if the secrets are being obfuscated in memory, then logically a memory dump should yield both the value of the secret before and after obfuscation. Next, we obtained a memory dump of the service binary and shortly found the cleartext API key in memory.

ClearText API Key in Memory for OneLogin ADC Connect Service

```
Connection: upgrade
Upgrade: websocket
Sec-WebSocket-Accept: 0IxBN[REDACTED]
GET /api/adc/v4/configuration?version=5.1.8&token=20577dc6[REDACTED]
[REDACTED]&mux=1&api_key=3a29be16322a[REDACTED]&directory_
id=[REDACTED]&directory_token=20577[REDACTED]&adcVersion=5.1.8
HTTP/1.1
User-Agent: ADC 5.1.8
Accept: application/xml
Host: api.onelogin.com
Accept-Encoding: gzip
ncoding: gzip
```

Inside the memory dump it was clear that the service was querying an undocumented AD Connect API to pull the tenant's configuration from OneLogin directly. Mocking up this request in Burp Repeater revealed the tenant's configuration details including a base64 encoded signing key.

Configuration API Response Part 1

```

HTTP/2 200 OK
Date: Tue, 10 Dec 2024 20:38:51 GMT
Content-Type: application/xml; charset=utf-8
Content-Length: 5198
Cache-Control: no-cache no-store max-age=0 must-revalidate private s-maxage=0
Etag: "41862548267983f8a15a7301e4f35a82"
Expires: 0
P3p: CP="CAO DSP COR CURa ADMa DEVa OUR IND PHY ONL UNI COM NAV INT DEM PRE"
Pragma: no-cache
Status: 200 OK
X-Correlation-Id: 00b9e751-b3d5-423a-b8cd-2051c5b5c982
X-Frame-Options: DENY
X-Request-Id: 6758A6DB-0A0905C6-79F2-0A090322-24E3-49CE87-21A58B
Strict-Transport-Security: max-age=63072000; includeSubDomains;
X-Content-Type-Options: nosniff

```

```

<configuration>
  <api_key>
    3a29be1[REDACTED]
  </api_key>
  <ldap_server_host>
    [REDACTED]
  </ldap_server_host>
  <ldap_server_port>
    [REDACTED]
  </ldap_server_port>
  <base_dn>
    [REDACTED]
  </base_dn>
  <directory_id>
    [REDACTED]
  </directory_id>
  <connector_id>
    [REDACTED]
  </connector_id>
  <authentication_attribute>
    userprincipalname
  </authentication_attribute>
  <provisioning_enabled>
    false
  </provisioning_enabled>
  <sync_disabled_users>

```

Configuration API Response Part 2

```
<sso_idp enabled="true">
  <audience>
    urn:onelogin:account:[REDACTED]
  </audience>
  <endpoints appBaseUrl="/onelogin/idp">
    <endpoint url="https://*:8080/" />
  </endpoints>
  <issuer>
    urn:onelogin:directory:[REDACTED]
  </issuer>
  <signing_key format="base64">
    kbNbJi[REDACTED]
  </signing_key>
  <consumer_url>
    https://[REDACTED]onelogin.com/trust/onelogin-sso/jwt?account_id=[REDACTED]
  </consumer_url>
</sso_idp>
/configuration>
```

After identifying a base64 signing key, we returned to the disassembled service binary to identify how to craft valid JSON web tokens (JWTs) for the OneLogin IDaaS. Based on the disassembled code, it was easy to identify that the fields the JWT token required:

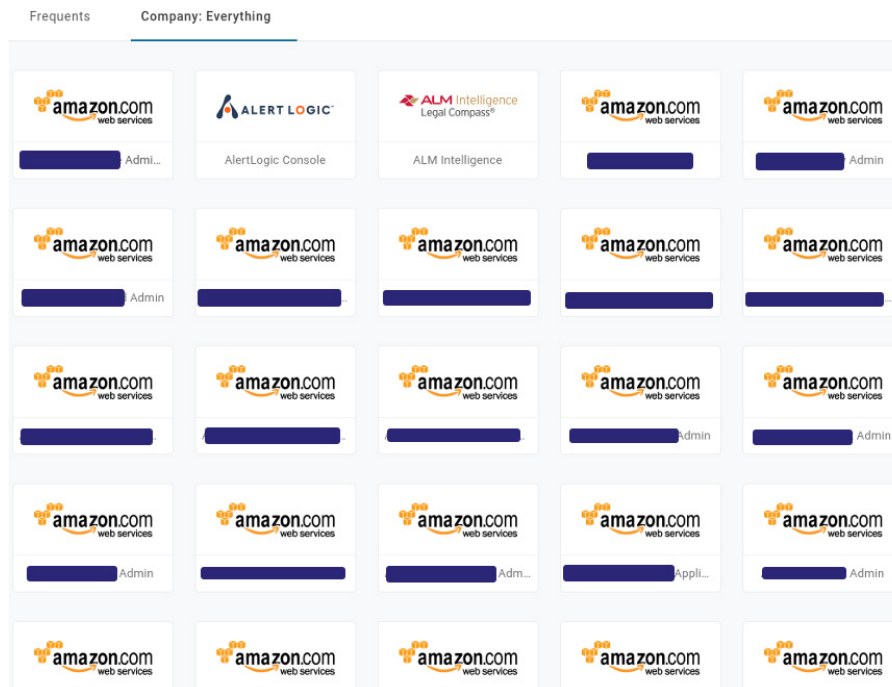
- **Exp:** (Token Expiration)
- **iss:** (Token Issuer) In this case, the directory the token was crafted for
- **aud:** (Audience) the audience for this token is the OneLogin account number
- **sub:** (Subject) this is a custom identifier for each user synced with OneLogin

Based on these requirements we were missing one critical piece of information, the sub value. The subject value is a custom identifier for the user being authenticated.

To find a list of all the users in the tenant, we can reach back out to that undocumented AD Connect API. The <https://api.onelogin.com/api/adconnect/v4/users> endpoint revealed a list of all registered users in the tenant along with their respective subject values. Putting it all together, we could finally craft valid JWTs for any arbitrary user in the OneLogin IDaaS.

With a valid crafted JWT, all that was left was to use that JWT to authenticate to the cloud login portal for OneLogin. Operators submitted the token to the `/trust/onelogin-sso/jwt` endpoint and successfully impersonated an AWS admin in the client organization.

With the authentication process complete, we were greeted with the impersonated user's OneLogin application directory, which included administrative access to 56 AWS Accounts.



OneLogin Application Menu for AWS Admin

Wrap Up

This attack path represents the reality of modern cloud-focused attacks: The most devastating paths often traverse multiple identity systems, exploiting the seams where they connect. What began as a traditional Active Directory compromise evolved into a cloud security breach, demonstrating how on-premises identity systems often serve as the perfect pivot point to cloud environments.

By deciphering how OneLogin's AD Connect service handled authentication behind the scenes, we could forge valid JWTs for any user in the organization, ultimately compromising 56 AWS accounts. Organizations deploying identity management solutions must recognize that abstracting away complexity doesn't eliminate risk and sometimes transforms it into something potentially more dangerous.

The Benefits of Continuous Visibility

The attacks leveraged in these stories were often the result of seemingly low-risk issues compounding into catastrophic attack paths. An MFA bypass, a session token, a forgotten key, or even a misconfigured wildcard character can be all that stands between business as usual and total compromise. One might assume these attack paths only exist in less mature environments, yet we consistently uncover them in even our most security-conscious clients. Something is missing.

Our red team experience highlights the critical need for continuous visibility into not just individual vulnerabilities or misconfigurations but rather the chained identity attack paths they create.

Community Contributions to the BloodHound Ecosystem and the Evolving State of Attack Path Management

Introduction to Community Contributions

Attack Path Management (APM) is a discipline focused on understanding, prioritizing, and disrupting the paths attackers follow to compromise systems and data. No organization, product, or methodology can solve APM's challenges in isolation.

Initially released by SpecterOps as a free and open-source tool for analyzing Microsoft's Active Directory relationships, BloodHound quickly became a cornerstone of APM practices before APM was common. A defining aspect of BloodHound and its community is a deliberate commitment to transparency. SpecterOps's 2018 post "A Push Toward Transparency"¹ outlines how we have consciously shared the attacker's perspective with defenders. Instead of hoarding offensive tradecraft, we have sought to democratize it.

John Lambert, Corporate Vice President and Security Fellow at Microsoft, penned a line that we have often used to discuss BloodHound and its mission:

"Defenders think in lists. Attackers think in graphs. As long as this is true, attackers will win."

This quote captures the asymmetry that we created BloodHound to address, but the tool itself is only part of the story. A global BloodHound community shares our values and mission and has contributed to enriching the BloodHound project and APM research.

BloodHound started with Active Directory. While AD remains a foundational part of BloodHound, the security landscape is ever evolving, and BloodHound has grown to include technologies like EntraID and the Microsoft Graph API. However, new attack paths are being discovered every day. Transparency and open source have inspired community innovation: tools built by security practitioners worldwide to extend, enhance, and adapt BloodHound to their needs. Whether it's identifying and extending the graph with misconfigured file shares, triaging vulnerable group policy objects (GPOs), or algorithmically surfacing critical attack paths, the community recognized and filled gaps in the APM lifecycle—often faster and more creatively than traditional enterprise software development can.

These contributions embody a shared security ethic. As SpecterOps Chief Executive Officer (CEO) David McGuire noted in the manifesto mentioned above, "Just as we reject security through obscurity as a strong control to protect client environments, we should also reject tradecraft efficacy through obscurity as well." Community-driven tools reinforce this philosophy. They transform detection into understanding and understanding into control.

1. <https://specterops.io/blog/2018/05/04/a-push-toward-transparency/>

The BloodHound community is not just an audience but a force multiplier. This section recognizes its contributions and provides a roadmap for what's possible when the security community shares freely, thinks offensively, and acts collaboratively.

Community Projects Advancing APM

BloodHound is the successor to research and tooling that came before it. Before BloodHound, we had tools like PowerView¹ and TrustVisualizer² to help red teamers understand the complicated relationships inside Active Directory networks. BloodHound met a need for something that could handle the larger and more complex environments the creators encountered on red team assessments.

BloodHound's continued growth is not only a story of innovation and research from within—it's one of evolution through community. While BloodHound laid the groundwork for graph-based security analysis in Active Directory environments with an open-source tool, community contributions have extended and enriched it.

Community projects have tackled initial data collection to advanced attack path prioritization and Cypher query composition. Many were born from operational needs: real problems encountered during assessments, red team operations, or incident response. Others reflect emerging ideas, such as incorporating large language models (LLM) and automation to make path analysis more effective and scalable.

This section will spotlight several community-developed tools directly contributing to the BloodHound ecosystem or APM in 2025. We will examine what they do, who built them, how they work, and, most importantly, how they empower defenders through transparency, automation, and actionable insight.

PowerHuntShares



Author: Scott Sutherland of NetSPI

Repository: <https://github.com/NetSPI/PowerHuntShares>

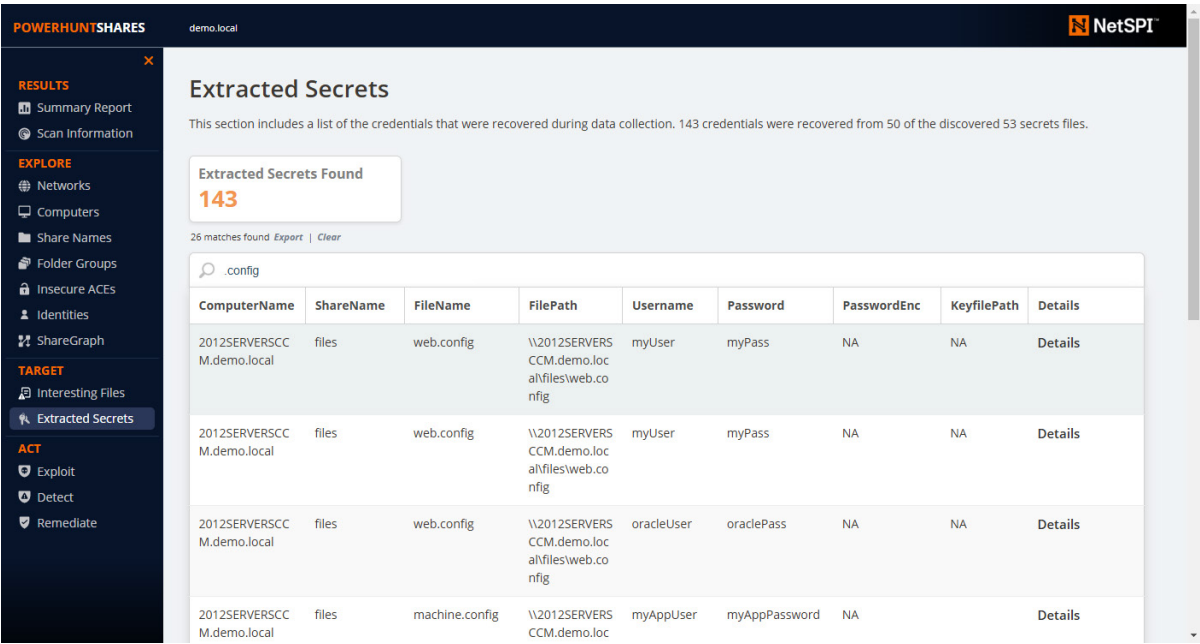
PowerHuntShares is a PowerShell-based tool NetSPI developed to identify and triage excessive SMB share permissions across an enterprise network. It enables defenders to identify file shares that may expose sensitive data or serve as stepping stones for lateral movement.

SMB shares have long been a common and often overlooked vector for adversaries. Misconfigured or overly permissive shares can allow attackers to access sensitive files or drop and execute malicious payloads on remote systems. These shares frequently serve as transit points or data staging locations in red team operations. From an APM perspective, every accessible share represents a potential path, and every writable share can become a launch point for compromise.

PowerHuntShares addresses this by scanning systems for open SMB shares, cataloging them, and reviewing their contents to classify interesting files and extract potential secrets.

1. <https://github.com/PowerShellEmpire/PowerTools/tree/master/PowerView>

2. <https://github.com/HarmJ0y/TrustVisualizer>



POWERHUNTSHARES demo.local **NetSPI**

RESULTS

- Summary Report
- Scan Information

EXPLORE

- Networks
- Computers
- Share Names
- Folder Groups
- Insecure ACES
- Identities
- ShareGraph

TARGET

- Interesting Files
- Extracted Secrets**

ACT

- Exploit
- Detect
- Remediate

Extracted Secrets

This section includes a list of the credentials that were recovered during data collection. 143 credentials were recovered from 50 of the discovered 53 secrets files.

Extracted Secrets Found
143

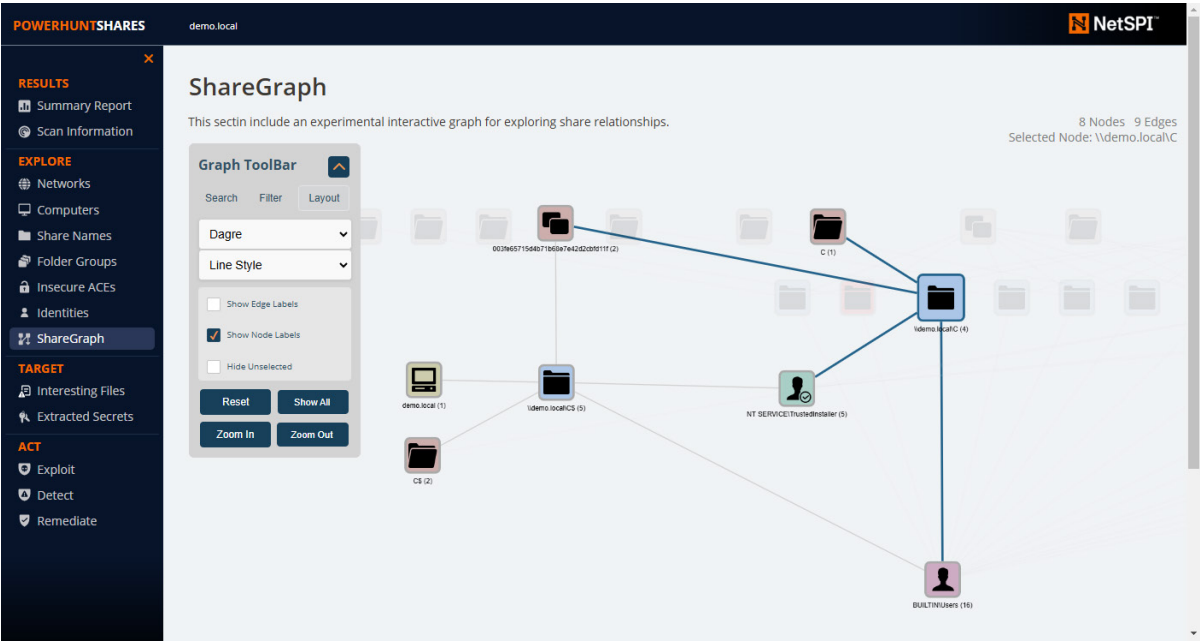
26 matches found [Export](#) | [Clear](#)

Search: .config

ComputerName	ShareName	FileName	FilePath	Username	Password	PasswordEnc	KeyfilePath	Details
2012SERVERSCC M.demo.local	files	web.config	\\2012SERVERSCC CCM.demo.loc ahfiles\web.co nfig	myUser	myPass	NA	NA	Details
2012SERVERSCC M.demo.local	files	web.config	\\2012SERVERSCC CCM.demo.loc ahfiles\web.co nfig	myUser	myPass	NA	NA	Details
2012SERVERSCC M.demo.local	files	web.config	\\2012SERVERSCC CCM.demo.loc ahfiles\web.co nfig	oracleUser	oraclePass	NA	NA	Details
2012SERVERSCC M.demo.local	files	machine.config	\\2012SERVERSCC CCM.demo.loc	myAppUser	myAppPassword	NA		Details

The tool generates detailed CSV and HTML output that is easier to browse than older tools that provide only console output. There is also a pending feature to generate a BloodHound-compatible import file that might one day add a “high-risk share” edge to the graph.

In the meantime, the PowerHuntShares v2.0 release contains an experimental feature for rendering a graph view of the share data. This graph view supports some basic functionality such as search, filtering, and adjustable layouts.



SMB shares remain fertile ground for attackers, and continued research into this attack path will remain important. PowerHuntShares also cites PowerView as an inspiration. Much like BloodHound, PowerHuntShares has evolved existing tooling and research to meet modern needs and make the data more accessible.

GPOHound



Author: Cogiceo

Repository: <https://github.com/cogiceo/GPOHound>

GPOHound is a Python tool Cogiceo developed to pull and analyze GPO from an Active Directory domain's SYSVOL and uncover potential security risks in these environments. While BloodHound supports GPO-related data, GPOHound seeks to enhance this capability by extracting and enriching BloodHound data.

Mining SYSVOL is another classic attack path. GPOHound identifies misconfigurations and privilege escalation paths tied to GPOs, such as excessive permissions and privilege escalation paths. It outputs structured data as JSON and in a BloodHound-compatible format.

GPOHound integrates with BloodHound by enriching its Neo4j database with additional node properties and edges derived from its Active Directory analysis. The project also includes custom queries for these new edges and properties that operators can import into BloodHound.

Further, GPOHound can use its connection to BloodHound to tap into the graph for what the project aptly calls an "LDAP-like source for Active Directory information." It's an excellent example of an integration that adds and pulls data for better analysis and APM efforts.

BloodHound-MCP-AI



Author: Mor David

Repository: <https://github.com/MorDavid/BloodHound-MCP-AI>

BloodHound-MCP-AI (BH-MCP-AI) is a project by Mor David and one of the first to bring LLM integration into the BloodHound ecosystem. As a Model Context Protocol (MCP) server, it follows the open MCP standard, which allows large language models to access and interact with external data sources. In this case, BH-MCP-AI serves as a bridge between an LLM and BloodHound's graph data, enabling new ways to query and analyze AD environments through natural language.

This project marks an early and promising step toward AI-assisted APM. The community has used LLMs like ChatGPT for a long time, but this was limited to asking the LLM to construct Cypher queries to pull up attack paths or extract specific pieces of data. BH-MCP-AI enables operators to ask BloodHound questions using natural language, such as "show me attack paths to high-value targets" or "identify DCs vulnerable to NTLM relay attacks."

BH-MCP-AI exemplifies how community projects can push APM into the future by making path management more accessible.

As Mor David points out in the project documentation, everyone should handle their BloodHound data as sensitive data. Take care when exposing sensitive information to any LLM.

bloodhound_mcp



Author: Matthew Nickerson

Repository: https://github.com/mwnickerson/bloodhound_mcp

Bloodhound_mcp (BHMCP) is an experimental and promising project by Matthew Nickerson, marking the second effort to bring an MCP server to BloodHound. It's exciting to see momentum building around this idea, with multiple projects exploring how LLMs can interact with BloodHound data.

Like BH-AI-MCP, BHMCP acts as a bridge between BloodHound's graph database and an LLM, enabling users to query complex AD relationships using natural language. Matthew is currently developing BHMCP with Anthropic's Claude Desktop client in mind, making it easy to set up in a small desktop lab or virtual machine (VM).

He recommends using the current build for training labs, demos, and research—perfect environments for experimenting with natural language interfaces and showcasing BloodHound's capabilities in an accessible, intuitive way.

A Cumulative Look at the Community Projects

The tools the BloodHound community developed (e.g., PowerHuntShares, GPOHound, and the BloodHound MCP servers) are more than discrete utilities. Together, they form a composite advancement of APM practices, demonstrating how community insight leads to better, more relevant security tooling. This section explores key themes and shared impacts across these projects.

Themes and Contributions Across Projects

The strength of the BloodHound community lies not only in its size but also in its diversity of thought and background. The core BloodHound development team is not all security practitioners, so insight from SpecterOps red teamers and researchers and the external community of researchers, red teamers, and defenders is immensely valuable.

Several core themes emerged across the highlighted community projects:

Discovery and Accessibility

The tools identify often overlooked relationships or misconfigurations, like excessive file permissions or vulnerable GPO settings, and provide structured output to help teams prioritize the findings. The output is easy to digest and act upon. This focus on making sense of complexity is a hallmark of effective APM.

Further, misconfigured file shares and GPOs have long been a staple of penetration testing and red teaming. While these are not new problems, they are not solved problems and often directly contribute to attack paths. Making it easier to enumerate and tag these misconfigurations remains as crucial as ever.

Automation and Graph Expansion

The tools automate tasks that otherwise require tedious manual efforts, such as enumerating file shares or tagging misconfigurations. It's also exciting to see these tools experimenting with enriching the

BloodHound graph with additional data and new edge types, potentially transforming data that was one-dimensional for so long into the BloodHound graph data with context for APM work.

The most impactful tools don't just generate data but contribute to and enrich existing data to help define relationships and prioritize action. These tools help shape how defenders model risk in their environments.

Intelligent Analysis and Natural Language

From the beginning, mastering Cypher was one of the biggest hurdles a BloodHound user might face when trying to become a power user. Learning to compose custom Cypher queries to dive deeper into the graph and extract statistics and new attack paths could be difficult. Several of the most popular early community projects were wikis and GitHub repositories of custom Cypher queries or BloodHound interfaces like Walter Legowski's CypherDog.¹

Projects like the BloodHound MCP servers bring intelligent automation into play, offloading the burden of manual graph analysis to the computer and enabling operators to use natural language to query the graph. These projects herald the beginning of a shift toward machine-assisted APM, where operators and defenders are less burdened with crafting Cypher queries. This potentially enables more people to get the most value out of BloodHound.

Projects like these continue to push APM forward as a repeatable discipline that any organization can adopt, adapt, and scale. They bring innovation through diversity. This diversity ensures that APM tooling remains grounded in operational reality and avoids stagnation, and it fosters a natural product-market fit where the "market" is the security community and its practitioners.

The community is vital to BloodHound's continued relevance and growth. As identity becomes the new perimeter, BloodHound's flexible graph model, enriched by community input, can evolve to match the growth of offensive tradecraft and research.

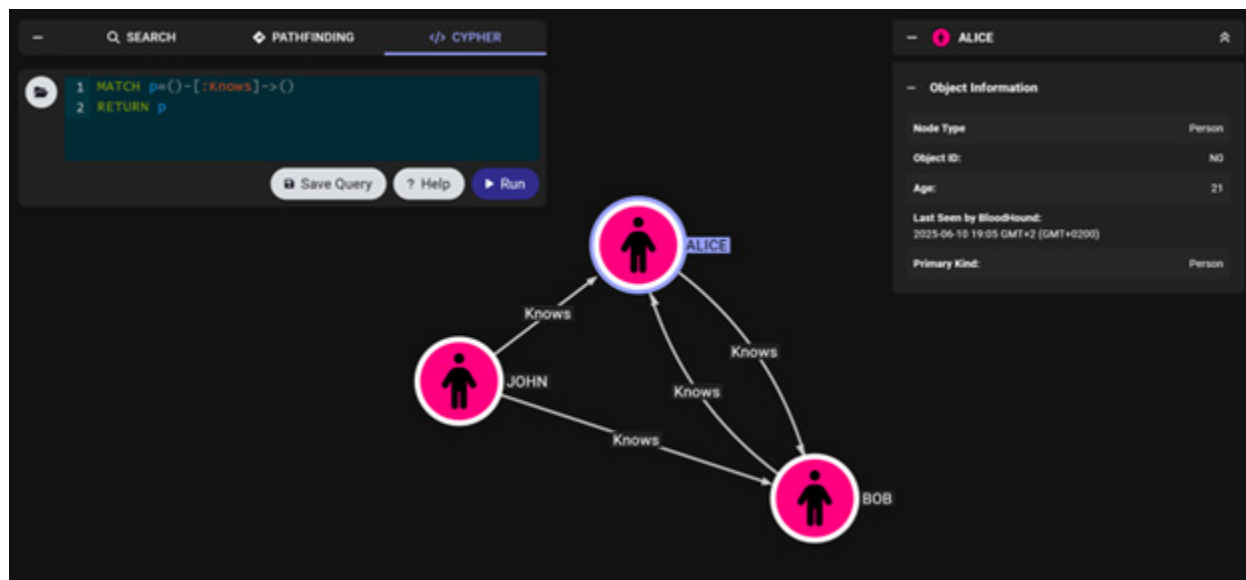
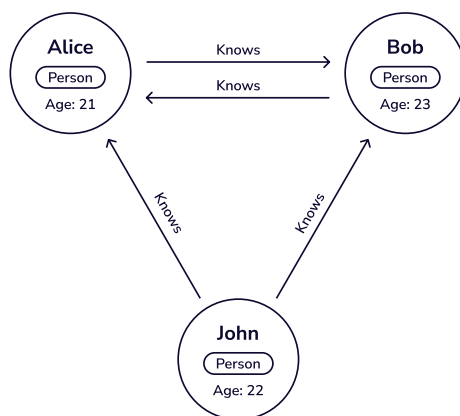
1. <https://github.com/SadProcessor/CypherDog>

Future of Community Contributions in APM

Looking ahead, the role of the community in APM will remain critical. For a few years, community projects and BloodHound practices have involved AI and machine learning. The community has published blog posts about using ChatGPT and similar LLMs to assist with crafting Cypher queries. This year, LLM-assisted querying evolved into a direct interface between LLMs and the BloodHound graph. We expect to see more of these projects in the future.

We also expect to see more projects pushing data and new edges to the BloodHound graph. Today, projects are manipulating and expanding the graph, but it's not the easiest thing to do. Future community projects will be able to take advantage of one of BloodHound's newest features, the BloodHound OpenGraph (BHOg).

BloodHound can now ingest generic data in a structured format instead of only from traditional collectors like SharpHound. BHOg opens the door for projects that enrich the BloodHound graph with new nodes and edges translated from sources like identity providers and cloud platforms.



The above figure is an example of an Arrows model translated into BloodHound using the OpenGraph. By lowering the integration barrier, we create an ecosystem where any tool can contribute value.

Celebrating Community as a Strategic Asset in APM

Community contributions are not ancillary to BloodHound; they are foundational. The most exciting innovations in APM are being driven by people closest to the problems: security practitioners, analysts, and operators who understand the nuance of Attack Path Management.

The BloodHound community has taken the platform further than any roadmap alone could have. Its members have built tools to add data, dig deeper, and expand what the graph can show.

As the BloodHound ecosystem continues to grow, we invite all defenders to:

- Use these community tools to strengthen your APM program.
- Support their creators by sharing feedback, ideas, and visibility.
- Contribute your perspectives, code, or challenges.



At SpecterOps, we believe we can create a more secure world through demystifying adversary tradecraft and promoting actionable approaches that are accessible to all.

A secure future is a shared one and the community is what makes it possible.

Acknowledgements

Thank you to our contributors: Jared Atkinson, Lance Cain, Julian Catrambone, Adam Chester, Andrew Chiles, Kate Dawson, Justin Kohler, Christopher Maddalena, Matthew Merrill, Elad Shamir, Zachary Stein, Robby Winchester, and Jason Wolfe

About SpecterOps

SpecterOps is a leader in identity risk management. Possessing deep knowledge of adversary tradecraft, the company enables global organizations to detect and remove critical attack paths before sophisticated attackers can take advantage of them – a practice called Attack Path Management. SpecterOps built and maintains widely used open-source security toolsets, including BloodHound, the company's foundational tool that enables attack path management in Active Directory, Entra ID and hybrid environments. BloodHound has been recommended by the U.S. Department of Homeland Security,¹ PricewaterhouseCoopers² and many others. BloodHound Enterprise is the company's managed SaaS for identity and security teams, allowing for attack path prioritization, remediation guidance and reporting to show improvements over time. **For more information on the benefits of an Attack Path Management practice, as well as SpecterOps and BloodHound, visit <https://specterops.io/>**

1. <https://www.cisa.gov/news-events/directives/ed-21-02-mitigate-microsoft-exchange-premises-product-vulnerabilities>

2. <https://www.pwc.co.uk/cyber-security/pdf/responding-to-growing-human-operated-ransomware-attacks-threat.pdf>