ARTICLE

Detecting Active Directory Privilege Escalation Paths with BloodHound

BloodHound turns Active Directory relationships into a graph that helps security teams identify privilege escalation paths before they are used. This article explains how to interpret the findings, validate whether a path is real, and decide which risks matter most in production.

Security - July 18, 2026

Why privilege escalation paths matter

The practical problem is not whether an Active Directory environment has weaknesses; it is whether those weaknesses connect into a path an attacker can actually use to move from a low-privilege account to a high-value target. In many environments, no single misconfiguration looks severe on its own. The risk appears when permissions, group membership, delegated rights, local admin access, session data, and ACLs line up into a chain.

BloodHound is useful because it models those relationships as a graph instead of a flat list of findings. That makes it easier to answer a more operational question: *Can this account reach domain admin-level control, either directly or through a small number of hops?* After reading this article, you should be able to recognize the kinds of escalation paths BloodHound exposes, decide whether the results apply to your environment, validate a suspicious path, and determine what to verify before using the findings in production risk work.

Key takeaways

- Privilege escalation in Active Directory is usually a chain of relationships, not a single vulnerability.
- BloodHound helps identify those chains by mapping users, groups, sessions, local admin rights, ACLs, and delegation.
- A path in the graph is not automatically a real exploit path; validation matters.
- The most useful output is a short list of actionable paths ranked by business impact and exposure.
- Production use requires clean collection scope, current data, and a clear remediation owner.

How BloodHound helps detect escalation paths

BloodHound ingests directory and host relationship data and turns it into queries over a graph. In practice, that means you can ask questions such as whether a user can control a machine that has a privileged session, whether a group has rights to modify another group, or whether an object ACL allows privilege-bearing changes.

The value is not just visualization. The graph exposes relationships that are easy to miss in traditional reviews:

- nested group membership that crosses team boundaries
- local administrator rights on servers or workstations
- delegated rights to modify users, groups, or computers
- privileges over service accounts or management groups
- sessions on high-value systems that create lateral movement opportunities
- misconfigured ACLs that allow write, reset, or ownership changes

These are the ingredients of escalation paths. A common example is a standard user who has control over a workstation, where a privileged user logs on intermittently. That user can then reach the server, change group membership, or pivot into a system that grants further rights. The chain matters more than any single node.

If your environment already tracks exposure and remediation by business risk, How to Prioritize Vulnerabilities Using Risk Scoring Models can help frame whether a discovered path should be treated as a high-priority control gap or a lower-impact hardening issue.

The workflow that produces useful findings

A useful BloodHound workflow is less about clicking through every path and more about collecting evidence that a path is credible, current, and operationally meaningful.

This workflow avoids a common failure mode: treating every displayed path as equally urgent. A path that crosses a stale session or a disabled account is different from a path that runs through active privileges on a Tier 0 system. The graph is a starting point, not the final verdict.

What the graph is actually telling you

BloodHound findings are best understood as relationship evidence. Each edge in the path usually represents a permission or condition that could be leveraged under the right circumstances. That might include write access, group management rights, local admin rights, DCSync-related permissions, or session-based opportunities.

When reviewing a path, ask four questions:

- Is the relationship still current?
- Is the target object or host in active use?
- Does the edge represent a permission that can realistically be used from the source context?
- Does the path reach a meaningful privilege boundary, or only a mildly sensitive account?

This is where many teams overread the output. A graph can show that a user has rights over an object, but it does not prove the relevant credential is available, that endpoint protections are absent, or that a control cannot interrupt the chain. Verification is essential.

A practical complement is to tie the path to asset criticality. If the destination is a low-impact server, the path may still matter, but it should not be treated the same as a path into domain-wide administrative control. The same logic used for vulnerability scoring applies here: context determines priority.

Practical scenario: when the path looks familiar

Imagine a mid-sized enterprise with separate workstation, application, and server teams. A help desk group has delegated rights to reset passwords for a subset of users. An application support group can administer service accounts for a line-of-business platform. One of those service accounts is also a local administrator on an application server. A privileged operator occasionally logs onto that server for maintenance.

None of these assignments may look dramatic in isolation. Help desk reset rights are common. Application support needs some management capability. Local admin on a server is normal for operations. But once the relationships are modeled together, a path emerges from a low-privilege foothold to a higher-privilege account or system.

This is the kind of environment where BloodHound is most valuable. The graph does not tell you that compromise is certain; it tells you that the organization has created an attack path worth investigating. The operational question becomes whether the path is real, current, and reachable from the places you assume are safe.

Validating a suspected escalation path

Validation is the difference between a useful finding and noise. A path should be checked for freshness, asset state, and practical use of each hop. For example, session data may be stale if data collection was incomplete or if the privileged user no longer logs onto that host. Group membership may be accurate while the effective permission is blocked by policy, scoping, or a disabled account.

A compact validation check is to confirm the source object, the edge type, and the destination impact. If one of those three elements is weak, the path may not deserve immediate action. If all three are strong, the path deserves a remediation plan.

Useful validation questions include:

- Is the source account enabled and actively used?
- Is the privilege-bearing relationship direct or inherited?
- Is the target account, group, or host still in service?
- Does the path depend on a one-time session or a persistent permission?

- Would breaking this edge materially reduce attacker movement?

This is also where environment hygiene matters. Stale directory objects, unmanaged local admin sprawl, and inconsistent group ownership can create false confidence in the graph. The better your directory governance, the more trustworthy your path analysis will be.

Implementation trade-offs you should expect

BloodHound is powerful, but it comes with practical trade-offs. The first is data freshness. Graphs are only as good as the collection window, so a path may disappear or appear depending on when data was gathered.

The second trade-off is coverage. If host session data, local group membership, or ACL collection is incomplete, the graph may understate risk. On the other hand, broader collection increases operational overhead and can create more data to review. You need enough coverage to detect meaningful paths without turning analysis into a maintenance burden.

The third trade-off is interpretation. Some paths are technically valid but operationally hard to use because of segmentation, additional controls, or limited attacker reach. Others are easy to exploit because they chain together common administrative habits. Your process should distinguish between possible and probable.

Finally, there is the remediation trade-off. Closing a path often means changing long-standing access patterns, not just fixing a single object. That can affect support workflows, service dependencies, and change windows. In practice, some paths are easier to sever by adjusting delegation boundaries, removing local admin rights, or eliminating privileged sessions on shared hosts.

How to decide whether a finding is actionable

A BloodHound path becomes actionable when it meets three conditions: the path is current, the destination matters, and there is a feasible remediation owner.

Use this decision rule:

- Act now when the path reaches Tier 0 assets, domain-wide rights, or high-value service accounts and the relationship is current.

- Track and validate when the path is plausible but depends on stale sessions, uncertain asset state, or partial collection.
- Deprioritize when the path is real but only reaches low-impact assets and there is no meaningful escalation beyond that point.

This is one of the places where business context matters as much as technical detail. A path that reaches a file server may not be urgent; a path that reaches an account used for management tooling may be. If you already use risk-based prioritization for security work, apply the same discipline here: impact, exposure, and exploitability should decide priority, not graph complexity alone.

Common mistakes when using BloodHound for escalation analysis

A frequent mistake is overtrusting the shortest path. The shortest path is often the most elegant graph result, but not always the most realistic operational path. A longer path through stable, active relationships may be more important than a shorter path that depends on stale data.

Another mistake is treating every admin-like edge as equivalent. Local administrator rights on a single workstation are not the same as rights that let you change a high-value group or manage a privileged service account. Edge type matters.

Teams also often overlook collection gaps. If you cannot see sessions, ACLs, or local admin relationships, your graph may miss the exact path an attacker would use. That is especially important in environments with segmented administrative models or multiple management tiers.

A final mistake is remediating one edge while leaving the underlying pattern intact. If help desk, server support, and privileged operations all depend on broad delegation and shared admin behavior, one closed path can be replaced by another. Fix the control pattern, not just the symptom.

What this means in practice

In production, BloodHound should be used as a control validation tool, not just an offensive testing aid. The practical output is a set of evidence-backed paths that show where your Active Directory design allows unnecessary privilege movement.

That means the best teams do not just ask, ?Can an attacker reach domain admin?? They also ask:

- Which permissions create the path?
- Which objects are part of the path repeatedly?
- Are privileged sessions landing on systems that should be hardened or isolated?
- Is delegation broader than the business need?
- Which fixes reduce multiple paths at once?

This approach helps you move from noisy findings to structural improvements. Often the highest-value fixes are the ones that reduce graph connectivity: removing standing privileges, tightening delegation, separating administrative tiers, and minimizing privileged logons to shared systems.

If DNS is part of your directory validation or attack surface monitoring workflow, DNSSEC Deployment Best Practices for Secure Domain Validation can help you harden name resolution trust boundaries that often sit alongside directory-dependent services.

Production readiness checklist

Before you rely on BloodHound findings for remediation planning, verify the following:

- Data collection is current enough to represent live relationships.
- The collection scope includes the object types needed for your analysis.
- High-value targets are clearly identified and agreed with stakeholders.
- Path results are validated against current directory and host state.
- Remediation owners exist for each class of issue, not just each individual finding.
- Changes can be tested and re-collected to confirm the path is gone.
- Exception handling is defined for business-required delegation and admin access.

If any of these are missing, treat the graph as directional intelligence rather than authoritative proof.

Final takeaway

BloodHound is most effective when you use it to detect privilege escalation paths as connected, testable relationships rather than isolated misconfigurations. The graph tells you where control can flow; validation tells you whether that flow is real; business context tells you what to fix first. If you can answer those three questions, you can turn BloodHound from a visualization tool into a practical control for reducing Active Directory attack paths.

Use this guidance together with DNSSEC and DNS tunneling detection to connect the workflow with related operational context already available on the site.