TUTORIAL

Active Directory Kerberos Delegation Abuse and Detection Tutorial

This tutorial shows how Kerberos delegation abuse works in Active Directory, how to identify the risky delegation types, how to validate exposure, and how to build practical detection and follow-up checks before production use.

Security - July 27, 2026

Why Kerberos delegation abuse matters

Kerberos delegation abuse is a practical Active Directory risk because delegation can let one account act on behalf of another service or user. If that trust is configured too broadly, an attacker who compromises a delegated service account can often move laterally, impersonate higher-value users, and reach systems that should never have been in scope.

This tutorial shows how to recognize the delegation types that matter, how to validate whether an environment is exposed, and how to build a detection workflow that is useful in production. By the end, you should be able to decide whether a delegation path is actually risky, test it safely, and confirm whether your logging can catch abuse or only gives you false confidence.

Prerequisites and stop-here checks

Before you begin, make sure you have a test plan and authorization to inspect Active Directory configuration and security logs. Some of the validation steps below can reveal sensitive trust relationships and service account behavior, so treat them like privileged administrative work.

Stop here if:

- You do not know which accounts are allowed to change delegation settings.
- You cannot read directory attributes such as msDS-AllowedToDelegateTo, userAccountControl, or the relevant service principal names.
- Security logging on domain controllers is not enabled at a useful level.
- You have no way to test changes in a lab or with a controlled service account first.

You will also need a way to query directory objects and review domain controller events. PowerShell is usually enough for the configuration checks, and event log access or centralized log collection is needed for detection validation. If you are also trying to understand whether the delegation path is part of a larger privilege escalation chain, Detecting Active Directory Privilege Escalation Paths with BloodHound can help you confirm whether the delegation is isolated or connected to other paths.

What you are trying to build

The finished state should look like this:

- You know which delegation type is present for a given account or service.
- You can identify whether that delegation is constrained, unconstrained, or resource-based.
- You can verify whether the account is allowed to impersonate high-value identities.
- You have at least one alert or hunting query that can show suspicious delegation use.
- You can distinguish between normal service behavior and an abuse pattern.

The goal is not to memorize every Kerberos field. The goal is to produce a repeatable workflow that answers a simple operational question: can this delegated identity be abused, and would we notice if it were?

Understand the delegation types that create risk

Unconstrained delegation

Unconstrained delegation is the highest-risk model because a service can potentially receive and reuse delegated credentials for users who authenticate to it. In practice, this creates a large blast radius if the service host or account is compromised.

Goal: Identify whether any account or computer object is marked for unconstrained delegation.

Action: Review the userAccountControl flags on service accounts and computer accounts, and inspect delegation-related settings in directory management tools.

A quick PowerShell check for computer objects with unconstrained delegation can start like this:

Expected output: A list of computer objects that are configured for unconstrained delegation.

Validation: Confirm that each result is expected and approved by application owners. Unconstrained delegation should be rare in modern environments.

Common failure: Reading the flag and assuming the object is safe because it is ?just a server.? If that server is compromised, the delegation model is still dangerous.

Constrained delegation

Constrained delegation limits which backend services can be accessed on behalf of a user. It is safer than unconstrained delegation, but still risky if the allowed target list is wider than necessary or if the delegated account has more privilege than required.

Goal: Determine which services an account can delegate to.

Action: Review the msDS-AllowedToDelegateTo attribute on service accounts.

Expected output: Service accounts with a populated list of backend services.

Validation: Confirm the target list matches the real application architecture. The shortest safe list is usually the right one.

Common failure: Treating any constrained delegation as acceptable without checking whether the service account can impersonate privileged users or reach administrative systems.

Resource-based constrained delegation

Resource-based constrained delegation changes the trust direction: the target resource decides which principals may delegate to it. This is operationally useful, but it still requires strong control over who can edit the target object?s delegation settings.

Goal: Identify whether the target computer objects allow resource-based delegation from unexpected accounts.

Action: Inspect the msDS-AllowedToActOnBehalfOfOtherIdentity security descriptor on resource computers.

Expected output: A controlled list of objects that are explicitly permitted to act on behalf of users.

Validation: Verify that the setting exists only for intended application relationships and that object write permissions are tightly scoped.

Common failure: Reviewing only the application account and ignoring who can modify the resource object itself. If an attacker can write the attribute, the trust can be abused.

Map the delegation path before you test abuse

Goal

Before you attempt any validation, identify the service account, the backend target, and the privilege level of the users that might be impersonated. Delegation is only dangerous when the trust path reaches something meaningful.

Action

Check these points for each candidate object:

- What account is the front-end service running as?
- Which backend SPNs or computers are permitted?
- Can the service impersonate ordinary users only, or administrative identities as well?
- Is the service account itself privileged, nested into privileged groups, or allowed to log on interactively?

If permission review is part of your process, How to Audit Active Directory Permissions for Excess Access is useful for validating whether the account has more directory power than the application truly needs.

Expected output

A simple inventory row per object is usually enough:

- Object name
- Delegation type
- Allowed backend services
- Can impersonate privileged users: yes or no
- Owner or application team
- Required remediation if misconfigured

Validation

Sanity-check the path by asking whether a compromise of the source account would actually help an attacker. If the answer is no, the configuration may still be sloppy but it is less urgent. If the answer is yes and the target is a tier-0 or sensitive system, treat it as high priority.

Common failure

A frequent mistake is focusing only on the delegation attribute and ignoring the surrounding permissions model. A service account with broad admin rights is already risky; delegation just makes the impact worse.

Validate exposure safely

Goal

Confirm that the delegation path behaves as expected without attempting destructive actions or relying on guesswork.

Action

Use a controlled, low-impact test account and a known service path to validate whether authentication flows through the expected backend service. The exact test method depends on the application, but your validation should answer three questions:

- Does delegation occur at all?
- Does it reach the intended backend?
- Does it permit only the intended identity scope?

A practical control is to compare a normal service request with a request made by a test principal that should not be able to reach the backend directly. If the backend still accepts the delegated identity, you have confirmed the trust path is functioning.

Expected output

Evidence that the service receives delegated authentication under the conditions it is supposed to support, and no evidence that it reaches unauthorized backends.

Validation

Validate with the smallest possible scope. If the service supports only one backend endpoint, test only that endpoint. Do not expand the test to privileged targets just because the delegation technically allows it.

Common failure

The most common failure is assuming that ?works as designed? means ?safe.? Delegation can be functioning exactly as intended and still be too permissive.

Build detections that catch abuse, not just configuration

Configuration checks tell you whether risky delegation exists. Detection tells you whether someone is using it suspiciously.

What to look for

Useful detections usually focus on one of these behaviors:

- A privileged user authenticates to an unusual service host.
- A service account requests backend access that does not match normal application flow.
- A delegated identity suddenly accesses systems outside its known dependency chain.
- A new SPN or delegation setting appears on an object that should be stable.

Kerberos delegation abuse often becomes visible only when you correlate authentication with service activity. If you are already building telemetry for related identity attacks, the logging principles in Detecting Active Directory DCSync Attacks with Event Logs are a good model for correlating security events with the action that followed them.

Goal

Create detections that flag abnormal delegation use with enough context to investigate quickly.

Action

Start with these detection ideas:

- Alert when delegation-related attributes change on sensitive accounts or computers.
- Alert when a service account accesses a backend system it does not normally use.
- Alert when a high-value user is authenticated through a service host that is not part of the normal application path.
- Alert when a computer or user account is enabled for delegation and the change was not accompanied by a change request.

The exact event IDs and fields vary depending on what you are collecting, but the principle is consistent: log the change, the source, the actor, and the affected object.

Expected output

A hunting query or alert rule that returns a small number of well-explained events instead of a noisy list of generic Kerberos activity.

Validation

Test the detection with a known-good configuration change in a lab or maintenance window. You should see the change event, the actor account, and the affected object. If you cannot connect those three elements, the alert is not ready for production use.

Common failure

A common mistake is alerting on all Kerberos traffic or all service ticket requests. That approach usually creates noise and makes analysts ignore the rule.

Practical validation checklist

Use this checklist when reviewing a candidate delegation path:

- The delegation type is known and documented.
- The source account is owned by a specific team.
- The backend targets are minimal and justified.
- No privileged users are expected to authenticate through the path unless there is a documented business reason.
- The object permissions that control the setting are restricted.
- A change detection rule exists for the attribute or policy that enables the trust.
- A hunting query exists for unusual backend access by the delegated account.

If any of those points is missing, do not treat the configuration as ready.

Operational follow-up after you find a risky path

Goal

Convert the finding into an actionable remediation plan rather than leaving it as a one-time security report.

Action

Prioritize fixes in this order:

- Remove unnecessary delegation entirely.
- Reduce the backend target list.
- Prevent privileged users from traversing the service path.
- Tighten who can modify the delegation-related object attributes.
- Add monitoring for both configuration changes and abnormal service use.

Where remediation is not immediately possible, document the exception with an owner, expiration date, and monitoring requirement.

Expected output

A tracked item with remediation scope, business owner, and a concrete date for review or removal.

Validation

Recheck the directory object after the change to confirm the risky setting is actually gone or narrowed. Then confirm the alert or hunting query still works and is not dependent on the old configuration.

Common failure

The common operational miss is fixing the wrong layer. Teams remove one backend target but leave a broader delegation setting or excessive write permission in place, which allows the same abuse to return later.

Final check before production use

Before you rely on this process in production, confirm the following:

- You can inventory delegation without administrative guesswork.
- You can explain why each delegated service is allowed to exist.
- You know which accounts could be impersonated and why that matters.
- You have at least one useful detection for suspicious delegation use.
- You have a remediation owner for every risky path you found.

If you can answer those points, you have moved beyond theory and built a practical workflow for Kerberos delegation abuse and detection. That is the real operational win: less uncertainty, faster investigation, and fewer delegated paths that can be quietly turned into an access compromise.

Use this guidance together with DNSSEC validation failures and detect ransomware to connect the workflow with related operational context already available on the site.