ARTICLE

Detecting Kerberoasting Attacks in Active Directory Networks

Kerberoasting detection depends on spotting abnormal service ticket requests, correlating them with account and host context, and validating whether the signal is strong enough to alert on without overwhelming operations. This article explains what to look for, how to confirm it, and what to verify before production use.

Security - July 09, 2026

Key takeaways

Kerberoasting detection is possible, but it is rarely reliable from a single event or a single field. The strongest detections combine service-ticket request patterns, account context, and host behavior so that normal Kerberos traffic is not mistaken for an attack.

The operational question is not just whether an attacker can request Kerberos service tickets. It is whether your environment can distinguish legitimate application traffic from a pattern that indicates password-hash harvesting for offline cracking.

After reading this article, you should be able to decide whether Kerberoasting detection applies to your environment, understand the signals that matter, use a compact validation workflow, and verify what must be in place before any alert is promoted to production.

Why Kerberoasting detection matters

Kerberoasting is attractive to attackers because it abuses normal Kerberos behavior. A domain user can request a service ticket for an SPN-bound account, capture the ticket, and try to crack it offline. That means the attack can happen without obvious authentication failure noise, and the value of the signal is often in the pattern rather than in a single event.

This matters operationally because Active Directory environments often include many service accounts, scheduled tasks, middleware systems, and legacy applications. Those systems create legitimate Kerberos volume that can mask abuse. If detection is too broad, you drown in alerts. If it is too narrow, you miss the attack until an attacker has already recovered credentials.

The practical goal is to identify service-ticket request behavior that looks unusual for the requesting principal, the host, the time of day, or the target service account. Good Kerberoasting detection is therefore a correlation problem, not just a log-search problem. If you are also reviewing privilege exposure, it helps to pair this work with How to Audit Active Directory Privileged Group Memberships so you can see which service accounts are actually worth protecting first.

How Kerberoasting shows up in logs

The attack usually begins with Kerberos service ticket requests for SPNs associated with user accounts. In Windows environments, the most useful starting point is the event stream associated with service ticket issuance. The exact event IDs and available fields depend on your audit configuration, collector, and log pipeline, so verify what is present in your environment rather than assuming every domain controller exposes the same view.

A detection rule often starts with one of these ideas:

- A single source host requests many service tickets across different SPNs in a short interval.
- A user or workstation that rarely interacts with a service suddenly requests tickets for many service accounts.
- A request pattern appears from a non-standard system, such as a workstation instead of a known application server.
- Ticket requests target accounts with weak password hygiene or older service-account practices.

None of these is sufficient on its own. A batch job, inventory scanner, patching tool, or application server can look similar if you do not apply context. That is why the signal should be evaluated against historical baselines and known service inventories.

Compact detection workflow

Use the workflow below as a validation framework rather than as a rigid recipe. It helps determine whether a suspected pattern is worth alerting on.

The key point is that a useful detection has to survive more than one question. A high number of service-ticket requests may be normal on a busy app server. The same activity from a user workstation at 02:00 on a Friday may deserve investigation. The context determines whether the event is informational or suspicious.

What to look for in practice

A practical Kerberoasting detection strategy usually relies on a small set of stable observations.

First, look for unusual concentration of service-ticket requests. Attack tools often enumerate SPNs and request tickets in a way that is broader than normal user behavior. That does not always mean a single burst; some operators throttle requests to stay under the radar. The important point is the deviation from the host’s normal profile.

Second, review the source principal. A workstation account or low-privilege user that requests many service tickets is more interesting than a known application server that does the same. If the source is a server, check whether it is a documented application host, jump box, or management node. If it is not, the signal gets stronger.

Third, inspect the target accounts. Kerberoasting remains especially relevant when service accounts have weak password rotation, long-lived credentials, or broad entitlement. That is one reason least privilege still matters; reducing the blast radius of service accounts lowers the consequence of a successful roast. Hardening Active Directory with Least Privilege Access Controls is useful context here because exposure reduction and detection quality work better together than separately.

Finally, evaluate timing. Requests that occur during normal application windows may be less suspicious than the same pattern during off-hours. Timing alone is not decisive, but it can improve confidence when combined with host and account context.

Practical scenario: when the alert should feel familiar

Imagine a mid-sized environment with several line-of-business applications, a handful of SQL servers, and a mixed fleet of employee workstations. Most Kerberos service ticket traffic comes from two application servers and a small set of managed administration hosts.

One morning, a workstation used by a finance employee suddenly requests tickets for a wide spread of SPN-backed accounts: legacy file services, middleware, database services, and a few rarely used application accounts. There is no obvious change ticket, no corresponding application deployment, and the workstation is not on the list of approved admin or integration systems.

That is the kind of pattern where Kerberoasting detection becomes operationally useful. The request volume alone is not proof, but the combination of unusual source, broad target set, and weak business justification makes the event worth investigation. In a real environment, the next check would be whether the account involved is a known management tool, whether the behavior matches a patch job, and whether the target service accounts have weak password practices that increase risk.

Implementation trade-offs

Kerberoasting detection is a good example of a high-signal idea that can be implemented badly.

The first trade-off is sensitivity versus noise. If you alert on any account that requests more than a small number of tickets, you will catch benign application activity. If you require a very large spike, you may miss slow-and-steady enumeration. The right threshold depends on your baselines, not on a universal number.

The second trade-off is coverage versus cost. Richer telemetry, such as detailed domain controller auditing plus endpoint context, improves detection quality. But it also increases collection, storage, and triage effort. In some environments, the best result comes from alerting only on suspicious source hosts and leaving the high-volume analytics to hunting workflows.

The third trade-off is precision versus interpretability. A highly tuned behavioral model may reduce false positives, but if analysts cannot explain why it fired, it becomes difficult to operationalize. For most teams, a simpler rule with strong supporting context is easier to sustain.

What this means in practice

In practice, Kerberoasting detection works best as a layered control.

At the first layer, you need visibility into service-ticket requests and the ability to group them by source and target. At the second layer, you need asset and identity context so that you know which hosts are expected to generate that traffic. At the third layer, you need a way to judge whether the target service accounts are sensitive, stale, or poorly managed.

That means the alert logic should answer questions such as:

- Is the source host a known application server or management system?
- Does the user normally access these services?
- Are the target accounts expected SPN-bearing service accounts?
- Is the request pattern consistent with historical behavior?
- Would a compromise of these accounts materially affect the environment?

If the answer to most of these is unknown, the alert is not ready for production. A detection rule without context usually creates either excess noise or blind spots.

Decision guidance: when this approach applies

Kerberoasting detection is appropriate when you have at least basic domain-controller auditing and enough asset context to separate normal service activity from suspicious enumeration. It is especially useful in environments with shared service accounts, legacy applications, or high-value AD tiers.

It is less effective if your telemetry is incomplete, if SPN ownership is poorly documented, or if every workstation behaves like a server because of automation and heavy scripting. In those environments, you may need to improve inventory and service-account governance before relying on detection.

A practical decision rule is simple: if you cannot explain the top five sources of Kerberos service-ticket volume in your environment, do not assume a volume-based alert will be stable. Build the baseline first, then decide whether the detection belongs in real-time monitoring or in a hunting report.

Common mistakes

One common mistake is treating any service-ticket request as suspicious. Kerberos is normal infrastructure behavior, and many legitimate systems generate the same kind of traffic that attackers abuse.

Another mistake is ignoring source identity. Without host context, a workstation and an application server can look identical in the log stream even though the risk is very different.

A third mistake is failing to inventory service accounts. If you do not know which accounts own SPNs, you cannot judge whether the target of the request is expected or sensitive. That also makes it hard to prioritize remediation.

A fourth mistake is promoting a detection without testing it against known-good examples. Before going live, validate it against scheduled jobs, application startup periods, patch cycles, and remote administration activity. The goal is to know what normal looks like before you interpret abnormal.

A fifth mistake is focusing only on detection while leaving service accounts weakly protected. If the environment still relies on static passwords, outdated service-account practices, or excessive privilege, the attack remains viable even if you catch some of the attempts.

Production readiness checklist

Before you use a Kerberoasting detection rule in production, verify the following:

- Domain-controller auditing captures the service-ticket request events you intend to monitor.
- Source host, user, and target account are available in the log pipeline.
- You have a baseline for normal service-ticket activity by key servers and business hours.
- Known application servers, jump hosts, and automation hosts are documented.
- SPN-bearing service accounts are inventoried and owned.
- Alert thresholds account for legitimate bursts from real workloads.
- Analysts know how to distinguish a noisy baseline from a suspicious pattern.
- The rule has been tested against at least a few known-good service workloads.
- High-value service accounts are tracked for weak passwords, stale usage, or poor privilege boundaries.

If several of these items are missing, detection will still be possible, but the risk of false positives and missed incidents will be high.

Final takeaway

Kerberoasting detection is most effective when you treat it as a contextual anomaly problem: request volume matters, but source identity, target account quality, and expected workload patterns matter more. If you can baseline your Kerberos traffic, correlate it with asset knowledge, and validate the signal against real operational behavior, you can build detections that are both practical and defensible.