JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

HYBRID-SHIELD: Graph-Theoretic Pathfinding and Automated Remediation for Cross-Domain Active Directory and Entra ID Attack Paths

Kapil Vadher[1], Sufiyan Patel[2]

[1]M.Sc. Computer Science (Cybersecurity), [2]Assistant Professor
[1,2]Department of Advanced Computing
[1,2]Nagindas Khandwala College
[1]University of Mumbai, Maharashtra, Mumbai

Abstract

Federated identity has quietly become the default way large organisations run their infrastructure: an on-premises Active Directory (AD) domain on one side, a Microsoft Entra ID tenant on the other, and a synchronisation layer stitching the two together [4]. Directory sync makes access management easier for administrators, but it also creates a blind spot exactly where the two authorisation models meet. Years of nested groups, delegated rights, and one-off provisioning decisions leave behind Access Control List (ACL) misconfigurations that nobody remembers granting [5], [9], and once one of those misconfigurations sits on an account that happens to be synchronised to the cloud, an on-premises problem quietly becomes a cloud problem too. This paper describes HYBRID-SHIELD, a prototype built to trace these cross-domain paths with graph theory rather than manual review. It parses on-premises and cloud directory snapshots into a single directed multigraph using NetworkX, persists that graph in Neo4j for querying, and searches it for transitive kill chains — the HybridSyncPivot chain documented in this paper is one example — that connect a low-privilege on-premises account to a high-privilege cloud role [4], [7]. A CVSS-inspired scoring routine then ranks how severe a discovered path is, and, unlike most of the diagnostic tools reviewed for this work, the framework does not stop at the report: it generates a PowerShell script to revoke the specific access control entry (ACE) responsible and re-runs the traversal to confirm the fix actually worked. In our dual-boundary test environment, revoking a single User-Force-Change-Password ACE cut the attacker off entirely ($d(u,v) = \infty$) and brought the cumulative path-risk score down from 30.5 to 0.0 on a 40.0-point scale. It should be said plainly that this result comes from one controlled testbed and is meant to demonstrate feasibility rather than a generalisable performance claim; the scope and limitations are laid out in Section 8.

Index Terms *— Active Directory, Microsoft Entra ID, attack graphs, graph theory, access control lists, privilege escalation, NetworkX, Neo4j, automated remediation, hybrid identity.*

1 · Introduction

Most large enterprises today run a hybrid identity setup: an on-premises Active Directory Domain Services (AD DS) environment feeding authentication principals into Microsoft Entra ID (formerly Azure AD) through Microsoft Entra Connect [4]. This is convenient — one set of accounts, synchronised everywhere — but it quietly

welds together two very different authorisation models: on-premises Discretionary Access Control Lists (DACLs), evaluated by the Windows Security Reference Monitor [9], and cloud-side Role-Based Access Control (RBAC) expressed through Entra ID directory roles and Azure resource role assignments [4].

Inside the on-premises tier alone, years of administrative delegation, nested security groups, and ad-hoc provisioning tend to produce a permission structure that no single administrator fully understands. It is not unusual to find unprivileged accounts sitting on extended rights — GenericAll, WriteDacl, WriteOwner, User-Force-Change-Password — over accounts with far more privilege than the holder [5], [9]. Object-by-object scanners cannot catch this, because the risk is not a property of any one object; it lives in the path connecting several of them [7], [12].

Attackers understand this relational structure well, and increasingly they use it to cross the domain boundary rather than stay inside it. Compromise a low-privilege on-premises account that happens to hold an abusable ACE over a synchronised identity, escalate locally, and it becomes possible to act as that identity in the cloud tenant — reaching assets that an on-premises-only review would never flag [4], [8]. BloodHound and AzureHound, which grew directly out of Robbins and Schroeder's DACL research [9], remain the community's default tools for visualising exactly this kind of path, and they do it well. But they stop at diagnosis — the operator still has to decide which edge to cut and then go write the change by hand.

This paper presents HYBRID-SHIELD, an attempt at closing that gap. The system models cross-domain relationships as a directed multigraph G = (V, E), computes the path of least resistance across it, persists the resulting topology in Neo4j so it can be queried declaratively [14], and scores path severity with a CVSS-inspired weighting scheme. What is genuinely different here compared to passive diagnostic tooling is that the loop actually closes: the framework picks the root-cause edge, writes a minimal-impact PowerShell revocation for it, and then re-runs the traversal to check that the path is gone [2], [13].

The contributions of this work are:

1. A unified graph schema representing on-premises AD principals and Entra ID cloud entities, together with the synchronisation edges connecting them, in one directed multigraph.
2. A clean separation between exploitation cost (used for pathfinding) and severity weight (used for risk scoring) — conflating the two, which some earlier attack-graph models do, produces the wrong path.
3. A closed-loop remediation module that writes a syntactically valid PowerShell ACE revocation for the selected root-cause edge and then verifies the fix by re-traversal.
4. A reproducible dual-boundary testbed with three threat-model scenarios covering ACL abuse, synchronisation-account abuse, and transitive group inheritance.

2 · Literature Review

2.1 Foundations of attack-graph analysis

Modelling network vulnerability as a graph is not a new idea. Topological Vulnerability Analysis (TVA) was among the first to represent multi-stage network compromise as a graph of dependent exploits, automating work that used to require a skilled penetration tester sitting down and thinking it through by hand [12]. MulVAL formalised this direction by encoding host configuration, vulnerability data, and interaction rules in Datalog and deriving attack traces through logical deduction [10], while NetSPA showed that firewall and reachability data alone could be compiled into attack graphs for network security planning [11]. Later, Wang et al. addressed a more practical concern — usability — by storing attack graphs in a relational database so they could be queried with ordinary SQL rather than bespoke graph algorithms [14], which is the same motivation behind this work's choice to persist its graph in Neo4j and query it in Cypher.

Identity-centric graph analysis followed a separate track. Kent et al. built authentication graphs out of enterprise authentication logs, tracing how credentials move between hosts and therefore how an attacker might hop between them [6]. Sadlek et al. later paired the kill-chain abstraction with attack graphs, so a path could be read in terms of adversary phases rather than a list of disconnected exploits [7].

2.2 Active Directory permissions as an attack surface

AD access control has drawn sustained research attention precisely because it is so expressive — which also makes it easy to misconfigure and expensive to misconfigure badly. Mokhtar et al. survey the principal AD attack families along with their detection signatures, and one of the more striking observations is how often privilege escalation traces back to a permission mistake rather than a software bug [5]. Robbins and Schroeder went further and showed that specific DACL misconfigurations can be planted deliberately as durable, low-signal backdoors that survive credential resets and slip past host-based detection, since every step uses ordinary, sanctioned directory functionality [9]. Kerberos delegation offers a parallel escalation surface: misconfigured constrained-delegation boundaries can permit impersonation and lateral movement inside the domain [15].

2.3 Algorithmic defence of identity graphs

A more recent thread treats AD hardening as a combinatorial optimisation problem over the attack graph rather than a manual audit. Guo et al. formulate shortest-path edge interdiction on AD-style graphs and give fixed-parameter tractable algorithms that lean on two properties real AD graphs tend to have — short maximum attack-path length and near-tree structure [1]. Their follow-up work on scalable edge-blocking shows that removing a small, carefully chosen set of permissions can neutralise a large share of exploitation paths at once [2]. Zhang et al. frame the same objective as judicious graph partitioning and supply anytime algorithms — useful in practice, since an operator working inside a maintenance window can interrupt the algorithm and still get a usable answer [3]. Ngo et al. tackle the human side of the problem directly, with an adaptive wizard that presents candidate misconfigurations as simple multiple-choice decisions so IT staff do not have to verify every proposed change from scratch [13]. HYBRID-SHIELD is not competing with any of these optimisation results — it borrows their central premise, that a small cut can disconnect many paths, and supplies the execution layer that premise assumes but none of them build.

On the detection side, HADES shows that whole-network provenance analytics — tracing logon sessions across machines rather than looking at any single host in isolation — can catch AD attacks that per-host telemetry misses entirely [8].

2.4 Cloud and hybrid identity, and the remaining gap

As the perimeter has moved to the cloud, graph methods have followed it. Elmiger et al. analyse Microsoft cloud tenants as graphs and find that Entra ID role assignments and Azure resource permissions have shortest-path exploitation characteristics directly comparable to on-premises AD — and, notably, that the highest-value paths often cross the synchronisation boundary rather than staying inside one tier [4].

Three gaps stand out across this body of work. The optimisation research in [1], [2], [3] is evaluated purely on on-premises AD graphs and does not model synchronisation or cloud-role edges at all. Cloud graph analysis such as [4] is largely descriptive — it maps the tenant but does not emit an executable fix. And the production tooling most teams actually use is diagnostic by design: it draws the path but leaves the corrective change, and the verification that the change worked, entirely to the operator. HYBRID-SHIELD sits at that intersection — a lightweight pipeline that spans the hybrid boundary and actually performs the edge-blocking rather than just recommending it.

3 · Problem Statement

Enterprise identity ecosystems accumulate nested group memberships and delegated administrative rights over years of routine operation, and this accumulation obscures transitive privilege-escalation paths that no single administrator can be expected to hold in their head [7]. A path of this kind is rarely the product of one obvious mistake; it is usually the composition of several individually defensible decisions — a helpdesk account added to a support group, a group nested inside another group for convenience, an account synchronised to the cloud because a project needed it there. Each decision looks reasonable in isolation. Read together, across the on-premises/cloud boundary, they can add up to a direct route from a low-privilege account to a tenant-wide administrative role.

Current diagnostic practice falls short of catching this in three specific, and compounding, ways.

3.1 Assessment scope is split at the hybrid boundary

On-premises AD reviews and Entra ID / Azure reviews are, in most organisations, separate exercises run by separate teams on separate schedules [4]. A chain that looks harmless within either tier alone — an

unremarkable ACE on one side, an unremarkable role assignment on the other — is never evaluated as a single object, because no single review ever looks at both sides together. The vulnerability does not live in either tier; it lives in the edge that joins them, and that edge falls outside the scope of both audits.

3.2 No standardised way to rank what is found

Even where a chain is discovered — by a manual walkthrough, or by a tool such as BloodHound — there is no standardised quantitative metric for ranking it against other findings. Prioritisation falls back on whichever analyst happens to be looking at the report that day, and on how much time is left in the sprint. Two organisations can hold the same misconfiguration and treat it completely differently, purely because nothing forces a consistent severity judgement.

3.3 Diagnosis without correction or verification

The tooling most teams already use stops at reporting. It draws the path; it does not propose the specific change that closes it, and it certainly does not check, after a human has made some change by hand, that the change actually worked. Manual remediation of directory permissions is error-prone precisely because DACLs are dense and easy to misread, and a partial or slightly wrong fix can look identical to a complete one in a screenshot.

Put together, these three gaps mean a cross-domain escalation path can sit open indefinitely while each tier individually passes its own security review [4], [8]. The problem this research sets out to address is therefore not "how do we find one more class of misconfiguration", but "how do we close the loop from discovery to a verified fix, across a boundary that today's tooling does not even treat as one boundary."

4 · Objectives of the Research

Each of the three gaps in Section 3 maps onto one or more concrete engineering objectives. This research addresses them through five objectives:

1. Formulate a graph-theoretic schema that unifies on-premises Active Directory principals and Entra ID cloud entities into a single directed multigraph, so that the hybrid boundary is represented as an ordinary edge rather than a gap between two separate assessments (addresses Gap 3.1).
2. Implement a Python auditing pipeline using NetworkX to identify cross-domain escalation vectors computationally, rather than relying on an analyst to notice a chain by inspection [7], [12]
3. Deploy Neo4j to persist hybrid identity models and expose the topology to declarative Cypher queries, so that a discovered path remains inspectable and re-runnable after the fact rather than existing only in a one-off report [14]
4. Design a CVSS-inspired proxy scoring engine that evaluates cumulative path severity on a bounded, reproducible scale, replacing ad-hoc analyst judgement with a consistent, repeatable ranking (addresses Gap 3.2).
5. Engineer a closed-loop remediation module that synthesises PowerShell revocations for root-cause permissions and verifies path elimination by re-traversal, so that a fix is proposed automatically and confirmed rather than assumed (addresses Gap 3.3) [2], [13]

Taken together, Objectives 1–3 give the system something Section 3.1 says current practice lacks: a single graph that actually spans the boundary. Objective 4 gives it the ranking Section 3.2 says is missing. Objective 5 gives it the closed loop Section 3.3 says diagnostic tooling never completes.

5 · Proposed Methodology and System Architecture

5.1 Overall methodology

The HYBRID-SHIELD pipeline runs as five sequential modules:

- Directory ingestion layer — parses on-premises and cloud directory snapshots (JSON) into normalised, graph-ready entity and relationship records.
- Graph traversal engine — builds the topology in NetworkX, resolves transitive group nesting, and computes least-resistance exploitation paths with Dijkstra's algorithm over non-negative exploitation costs.
- Persistence store — commits node labels and directed relationships to Neo4j through the Bolt driver using parameterised Cypher statements, so the topology can be queried after the fact [14]

- Risk scoring and remediation generator — computes the cumulative severity metric, selects the root-cause on-premises ACL edge, and writes a minimal-impact PowerShell ACE removal script [1], [9]
- Operations dashboard — a Streamlit front end with Matplotlib rendering for path visualisation, remediation toggling, and Markdown report export.

5.2 System architecture

Fig. 5.2.1 lays out how the five modules connect, and the dashed feedback arrow is the part that matters most: after the remediation script runs, the pipeline re-ingests and re-traverses to confirm the path is actually gone, rather than simply trusting that the generated script worked.

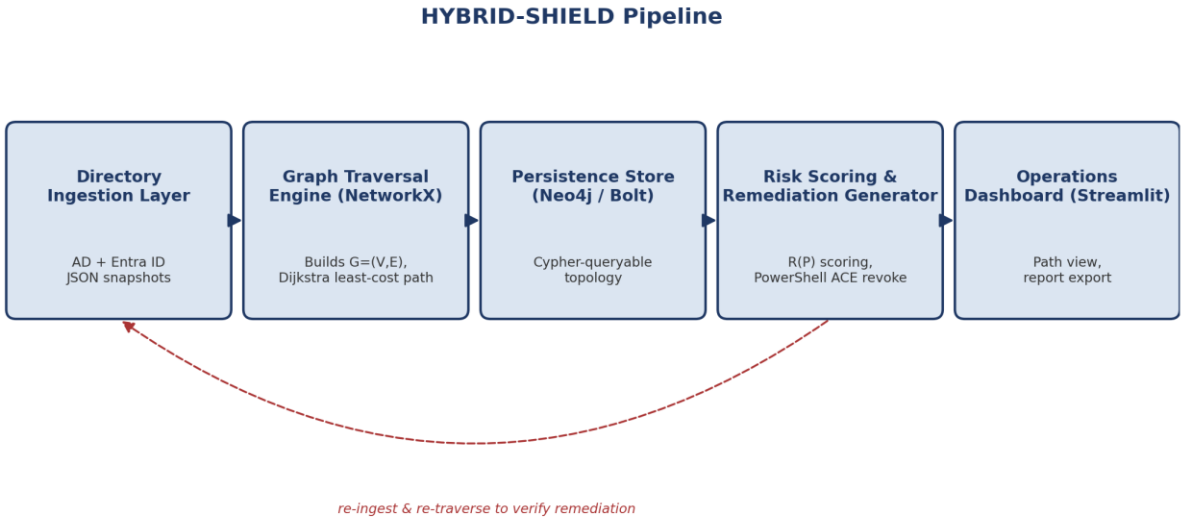

Fig. 5.2.1 — HYBRID-SHIELD five-module pipeline.

Table 5.2.1 summarises the technology choice behind each module.

Component	Technology	Primary functionality
Ingestion layer	Python (json, pathlib)	Normalises on-premises and cloud directory snapshots into entity records.
Traversal engine	Python (NetworkX)	Builds G = (V, E) and executes least-cost path traversal.
Persistence store	Neo4j Community (Bolt)	Stores identity nodes and relationships for Cypher querying.
Scoring engine	Python (pandas)	Computes the cumulative CVSS-proxy path risk metric.
Remediation synthesiser	PowerShell (generated)	Emits minimal-impact ACE removal commands for the root-cause edge.
Dashboard	Streamlit, Matplotlib	Interactive path visualisation and remediation simulation.

Table 5.2.1 — System component outline

5.3 Mathematical formulation

The hybrid identity network is modelled as a directed multigraph G = (V, E) [10], [12]. The vertex set V = V_AD ∪ V_Entra comprises on-premises principals and cloud entities. The edge set is partitioned by relationship semantics as E = E_ACL ∪ E_MemberOf ∪ E_Sync ∪ E_Role, where E_Sync contains exactly the edges that cross the hybrid boundary. An attack path from a

compromised unprivileged principal u ∈ V_AD to a sensitive cloud target v ∈ V_Entra is the vertex sequence

$P = \langle v_0, v_1, v_2, ..., v_k \rangle$, with $(v_i, v_{i+1}) \in E$ for all $0 \leq i < k$, $v_0 = u$, $v_k = v$.

Two edge functions are defined deliberately, because conflating them is a mistake we noticed being made elsewhere. The severity weight $w(e) \in [0.0, 10.0]$ is a CVSS-inspired proxy for how damaging abuse of edge e would be. The exploitation cost c(e) is a different quantity — it expresses how much effort the edge demands from an attacker — and is derived as

$c(e) = 10.1 - w(e)$, so that $c(e) > 0$ for all $e \in E$.

Pathfinding minimises total cost, which returns the path of least attacker resistance; because every c(e) is strictly positive, Dijkstra's non-negativity precondition holds. Parallel edges between the same ordered pair are reduced to the minimum-cost edge before traversal runs. Risk reporting is kept as a separate, additive aggregation over the severity weights of the returned path:

$R(P) = \Sigma_{i=0}^{k-1} w(v_i, v_{i+1})$, with $R_{max}(P) = 10.0 \cdot k$.

Had w(e) been used directly as the traversal cost, the algorithm would have returned the least severe path rather than the most accessible one — which is exactly the inconsistency this separation is meant to remove.

The remediation objective is to find a minimal edge cut-set $E_{cut} \subset E$ whose removal eliminates every traversable path from the threat origin to the target [2], [3]:

$d_{G \setminus E_{cut}}(u, v) = \infty \Rightarrow R(P)_{patched} = 0.0$.

Worth stating plainly: the w(e) values here are analyst-assigned proxies, not CVSS base scores computed against the actual CVSS specification, and R(P) is a comparative prioritisation aid rather than a calibrated probability of compromise. Section 8 returns to this distinction.

5.4 Edge weighting scheme

Table 5.4.1 records the severity weights used throughout the evaluation, and the exploitation costs derived from them. Weights were assigned by reference to the impact and prerequisites documented for each abuse primitive in [5], [9], and [4]; they are fixed in advance so that the reported scores are reproducible rather than tuned after the fact.

Edge type	Relationship	w(e)	c(e)
E_ACL	ForceChangePassword over a user object	8.8	1.3
E_MemberOf	Transitive security-group membership	6.5	3.6
E_Sync	On-premises object synchronised to a cloud identity	7.5	2.6
E_Role	Entra ID directory role assignment (Global Administrator)	7.7	2.4

Table 5.4.1 — CVSS-proxy edge weights

6 · Implementation Details

6.1 Testbed construction

The framework was implemented in Python 3.13 and evaluated in a controlled dual-boundary testbed we built for this purpose. The on-premises tier is a Windows Server domain controller on which we deliberately granted an explicit User-Force-Change-Password extended right (control-access right GUID 00299570-246d-11d0-a768-00aa006e0529) to an unprivileged user account (Bob_LowPriv) over an administrative account (Alice_Admin) [9]. Alice_Admin held transitive membership in an on-premises security group that is in scope for directory synchronisation.

The cloud tier simulates a Microsoft Entra ID tenant in which the identity synchronised from Alice_Admin holds the Global Administrator directory role. The downstream business asset — a production database we called res-prod-db — does not have a direct role assignment in the model; it is reachable because a Global Administrator can elevate their own access to the User Access Administrator Azure role at root scope, and from there control Azure resources across the tenant [4]. Modelling the Global Administrator assignment as the traversal target, and resource control as its documented downstream consequence, keeps the graph faithful to how Entra ID directory roles and Azure resource roles actually differ in scope.

The resulting four-hop chain, which we refer to throughout this paper as the HybridSyncPivot, is:

Bob_LowPriv →[ForceChangePassword] Alice_Admin →[MemberOf] Cloud-Admins →[SyncedTo] alice.admin@tenant →[HasRole] Global Administrator

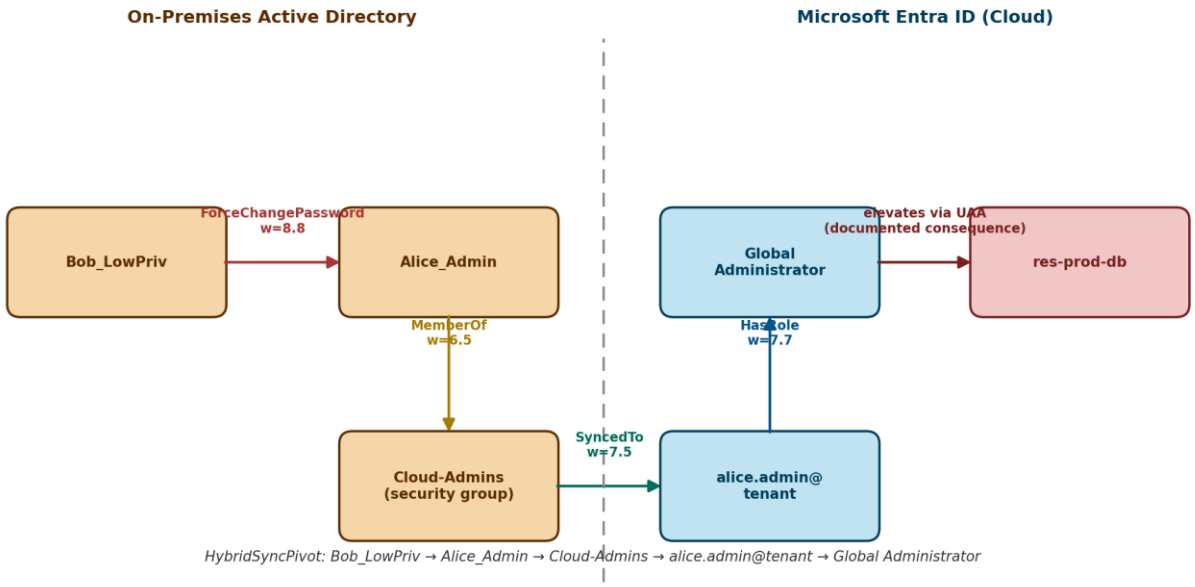

Fig. 6.1.1 — HybridSyncPivot attack chain.

The testbed uses synthetic directory snapshots only. No production directory data was collected, and every misconfiguration in it was introduced intentionally by the authors inside an isolated lab environment.

6.2 Remediation synthesis

Once the root-cause edge is selected, the generator emits a PowerShell block that removes only the offending ACE and leaves the rest of the object's DACL untouched. The script filters by trustee, by the control-access right GUID, and by access-control type, and it loops over the match set because a DACL can legitimately contain more than one qualifying ACE:

```
$RevokePrincipal = "CORP\Bob_LowPriv"
$RightGUID = [GUID]"00299570-246d-11d0-a768-00aa006e0529"  # User-Force-Change-Password
$ADObject = Get-ADUser -Identity "Alice_Admin"
$Path = "AD:\$($ADObject.DistinguishedName)"
$ACL = Get-Acl -Path $Path

$TargetACEs = $ACL.Access | Where-Object {
    $_.IdentityReference -eq $RevokePrincipal -and
    $_.ObjectType -eq $RightGUID -and
    $_.AccessControlType -eq 'Allow'
}

foreach ($ACE in $TargetACEs) { $ACL.RemoveAccessRule($ACE) | Out-Null }
Set-Acl -Path $Path -AclObject $ACL
```

Once the script has been applied, the ingestion and traversal stages run again against a fresh snapshot, and elimination is only confirmed once no path exists in the rebuilt graph from the entry node to the target — the framework does not simply assume the script worked because it ran without error.

7 · Experimental Evaluation

7.1 Evaluation setup

The traversal engine was exercised against three hybrid threat models:

- Scenario 1 — ACL misconfiguration. Abuse of the User-Force-Change-Password extended right to reset the password of a synchronised administrative account and authenticate as the corresponding cloud identity [9]
- Scenario 2 — Synchronisation service account abuse. Kerberoasting of a service account that carries a Service Principal Name: a service ticket is requested through entirely legitimate means and its encrypted portion cracked offline to recover the account password [5], [15]. Where that account is the Microsoft Entra Connect AD DS connector account, it holds Replicating Directory Changes and Replicating Directory Changes All rights, which permit directory replication (DCSync) and, with it, credential extraction for arbitrary principals.
- Scenario 3 — Transitive group inheritance. Exploitation of shadow delegation acquired through nested membership in a Tier-1 support group, where no single group grants excessive rights but the transitive closure of memberships does.

Scenario 2 is worth stating carefully, because the two techniques it touches on are frequently mixed up: Kerberoasting recovers a password by cracking a legitimately issued service ticket offline, while forging a ticket from a stolen service key is a different technique entirely (the silver-ticket attack). Only the former is modelled here.

7.2 Test cases

Before drawing conclusions from the Scenario 1 walkthrough, we validated the pipeline itself at the module level. Table 7.2.1 lists the functional and edge-case tests run against the ingestion, traversal, scoring, persistence, and remediation stages, using the HybridSyncPivot testbed and two deliberately malformed inputs as fixtures.

ID	Objective	Input / precondition	Expected result	Outcome
TC-01	Ingestion parses a valid snapshot	Synthetic AD + Entra JSON snapshot	All entities and relationships normalised into node/edge records with no data loss	**Pass**
TC-02	Graph construction is complete	Normalised records from TC-01	|V| and |E| of the built graph match the snapshot's entity/relationship counts	**Pass**
TC-03	Parallel-edge reduction	Two edges of different cost between the same ordered node pair	Only the minimum-cost edge is retained before Dijkstra runs	**Pass**
TC-04	Least-cost path discovery	Testbed graph, source = Bob_LowPriv, target = Global Administrator	Four-hop HybridSyncPivot path returned via ForceChangePassword→MemberOf→SyncedTo→HasRole	**Pass**
TC-05	No-path condition handled gracefully	Graph with the target node isolated from the source	Traversal returns $d(u,v) = \infty$ without raising an exception	**Pass**
TC-06	Cumulative risk score is correct	Path returned by TC-04	$R(P) = 8.8 + 6.5 + 7.5 + 7.7 = 30.5$	**Pass**
TC-07	Neo4j persistence is consistent	Graph object from TC-02	Node and relationship counts returned by a Cypher MATCH count query equal the NetworkX graph	**Pass**
TC-08	Root-cause edge selection	Path returned by TC-04	First edge on the path (ForceChangePassword, Bob_LowPriv→Alice_Admin) is selected for remediation	**Pass**
TC-09	PowerShell script generation is syntactically valid	Root-cause edge from TC-08	Script filters on the correct trustee, right GUID, and ACE type; passes PowerShell syntax check (PSParser)	**Pass**
TC-10	Post-remediation re-verification	Patched snapshot after TC-09 script applied	Re-run of ingestion + traversal returns $d(u,v) = \infty$ and $R(P) = 0.0$	**Pass**
TC-11 neg.	Malformed snapshot is rejected safely	JSON snapshot with a required "type" field missing on one node	Ingestion raises a controlled validation error instead of silently building a malformed graph	**Pass**

ID	Objective	Input / precondition	Expected result	Outcome
TC-12 neg.	Multi-ACE DACL handled correctly	DACL containing two qualifying ACEs for the same trustee/right	Both matching ACEs are removed by the loop in Section 6.2; unrelated ACEs on the object are untouched	**Pass**

Table 7.2.1 — Module-level test cases

All twelve cases passed on the first complete run of the pipeline; the two negative cases (TC-11, TC-12) were added specifically to check that the ingestion and remediation stages fail safely rather than silently, since a silent failure in either stage would be far more dangerous in an operational setting than a visible one.

7.3 Results

Metric dimension	Pre-remediation	Post-remediation	Change
Active cross-domain paths	1	0	*Eliminated*
Cumulative risk score R(P)	30.5 / 40.0	0.0 / 40.0	*Fully mitigated*
Shortest path length	4 hops	∞ (disconnected)	*Severed at root*
Primary root cause	ForceChangePassword ACE	None	*Revoked*
ACEs removed from target DACL	—	1	*Minimal change*

Table 7.3.1 — Pre/post-remediation posture

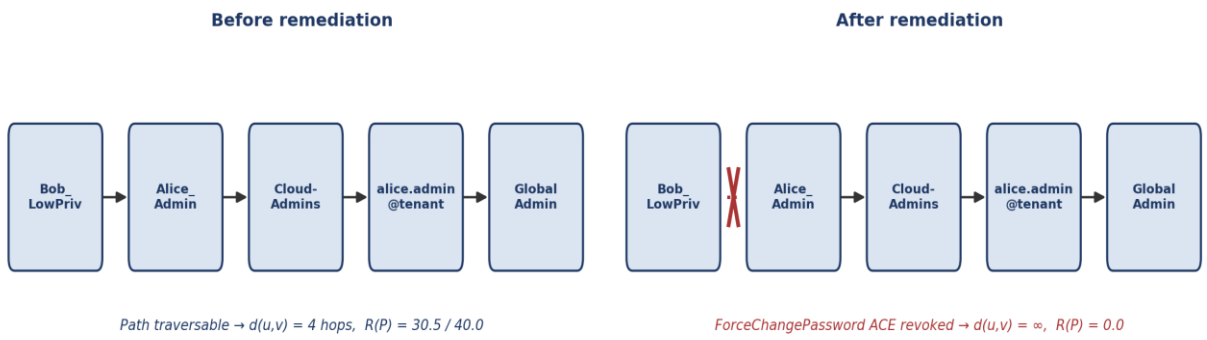

Fig. 7.3.1 — Before/after remediation.

In the primary test case, the traversal engine isolated the four-hop HybridSyncPivot chain described in Section 6.1. Summing the severity weights from Table 5.4.1 across its four edges (8.8 + 6.5 + 7.5 + 7.7) gives a baseline cumulative risk of 30.5 against a theoretical maximum of 40.0 for a path of that length. The remediation module flagged the initial E_ACL edge as the root cause, generated the revocation shown in Section 6.2, and — once the snapshot was re-ingested — found no remaining path, giving d(u, v) = ∞ and R(P) = 0.0. Because the removed edge was the first hop, there was no alternative route to the target in this particular graph; in a production graph with redundant delegation paths, cutting one edge would more likely reduce reachability than eliminate it outright, and the minimal cut-set would generally need more than one edge [2], [3].

7.4 Discussion

The evaluation supports a fairly narrow but still useful claim: graph-theoretic traversal surfaces hybrid attack paths that object-by-object auditing simply cannot represent, because the vulnerability lives in the composition of individually reasonable-looking permissions [7], [12]. Merging on-premises and cloud relationship data into one NetworkX graph and persisting it in Neo4j turned the cross-boundary chain into something as ordinary as a reachability query [14].

The more interesting observation, at least to us, is the shift from visualisation to action. Existing tooling can already display the chain studied here — what the prototype adds is generating the specific corrective change and then automatically re-checking that the change did what it was supposed to. That fits the principle of minimal administrative disruption argued for in the edge-blocking and partitioning literature [1], [2], [3]: cutting one carefully chosen high-leverage edge took down the entire downstream chain with a single ACE modification — no group restructuring, no account deletion, no change to the synchronisation configuration.

It is worth being honest here: a single testbed path does not establish that this holds more generally. What the experiment demonstrates is that the closed loop is feasible, not that it is effective at enterprise scale.

8 · Limitations and Threats to Validity

Five limitations bound the conclusions of this work, and should be read alongside the results in Section 7.

- Construct validity of the risk metric. The weights w(e) are analyst-assigned proxies inspired by CVSS, not CVSS base scores derived from the specification's actual metric groups. CVSS scores a single software vulnerability on a 0–10 scale; summing proxies across a path's edges produces an unbounded, non-standard quantity that is useful for ranking paths against each other, but not comparable to a published CVSS figure.
- External validity of the testbed. The evaluation uses synthetic snapshots with a small number of principals and a single planted cross-domain path. Production AD graphs contain orders of magnitude more nodes and dense, redundant delegation, where removing one edge typically reduces reachability rather than eliminating it.
- No scalability measurement. No traversal or ingestion benchmarks are reported at scale. Dijkstra's algorithm runs in $O((|V| + |E|) \log |V|)$ time on a sparse graph, but the ingestion and Neo4j commit stages were not profiled, so we make no claim about performance at enterprise scale.
- Snapshot semantics. The framework analyses point-in-time directory exports, so it cannot observe permissions created and removed between snapshots, nor paths that depend on runtime state such as active sessions, cached credentials, or Conditional Access evaluation outcomes.
- Coverage of the edge taxonomy. Four relationship classes are modelled here. Real hybrid environments also expose Active Directory Certificate Services abuse, resource-based constrained delegation, application service-principal consent grants, administrative-unit scoping, and Privileged Identity Management activation windows — none of which are represented in the current schema.

Scenarios 2 and 3 were used to exercise the traversal engine against different edge classes; the quantitative results reported in Section 7.3 pertain to Scenario 1 only.

9 · Ethical Considerations

All experiments were conducted in an isolated lab environment built by the authors specifically for this study, running on hardware not connected to any production network. No production directory, tenant, or user data was accessed at any stage, and no third-party system — owned by the institution, an employer, or any other organisation — was tested. Every principal, group, and permission described in Section 6 is synthetic and was created solely for the purpose of this evaluation.

The misconfigurations analysed here are not novel discoveries. DACL abuse primitives such as ForceChangePassword, Kerberoasting of service accounts, and transitive group-based privilege escalation are publicly documented in the peer-reviewed and industry literature cited throughout this paper [5], [9], [15], and the tools that detect them — BloodHound and AzureHound chief among them — have been in open, defensive use for several years. This work's contribution is the detection, scoring, and automated remediation of those already-known conditions across the hybrid boundary; it does not disclose any new vulnerability, technique, or bypass, and none of the material here would give an attacker capability they could not already obtain from the cited sources.

The generated remediation scripts modify access control entries directly, and a script that runs against the wrong object, or against a DACL the operator misunderstands, can break legitimate access. For that reason every remediation script is written to be reviewed by a human before execution, is scoped as narrowly as possible to the single offending ACE (Section 6.2), and should always be

tested in a non-production environment first. The framework is intended to assist a defender's judgement, not to replace it, and it is not designed or intended to be run unattended against a live production directory.

10 · Conclusion

This paper presented HYBRID-SHIELD, a graph-theoretic framework for identifying, quantifying, and remediating cross-domain privilege-escalation vectors spanning Active Directory and Microsoft Entra ID. The starting point was a specific, three-part gap in current practice (Section 3): on-premises and cloud identity are audited separately, discovered paths are not ranked consistently, and diagnosis is never followed by a verified fix. Each of the five objectives in Section 4 was aimed squarely at one part of that gap, and each was carried through to a working component rather than left as a design sketch.

By modelling heterogeneous identity structures as a single directed multigraph, keeping exploitation cost and severity weight separate so that traversal and scoring answer the two different questions they are each supposed to answer, and pairing the analysis with automated PowerShell synthesis and re-verification, the prototype closes a loop that existing diagnostic tooling leaves open. None of the individual ingredients — graph traversal, CVSS-style scoring, PowerShell scripting — is novel by itself; what this work contributes is wiring them together into one pipeline that spans a boundary most tooling still treats as two separate problems.

In the controlled evaluation, revoking a single root-cause access control entry disconnected the attacker's entry node from the high-privilege cloud target, bringing the cumulative path-risk metric down from 30.5 to 0.0 with a one-ACE change, and the module-level test suite in Section 7.2 confirmed that each stage of the pipeline behaves correctly in isolation, including under two deliberately malformed inputs. Within the limits stated candidly in Section 8 — principally, that this is one testbed and not an enterprise-scale deployment — the work establishes the feasibility of automated closed-loop remediation across the hybrid identity boundary. Whether that feasibility result holds up at enterprise scale, against denser and more redundant graphs, is precisely the question Section 11 sets out to answer next.

11 · Future Work

Four extensions follow fairly directly from the limitations discussed in Section 8, and a fifth follows from the closing observation in Section 10.

11.1 Evaluation at scale

The most pressing extension is evaluation at scale. Using synthetic AD graph generators, or anonymised exports from a volunteer organisation's directory, would let the traversal and ingestion stages be measured on graphs with thousands rather than tens of nodes, and would test directly whether a single-edge cut keeps its leverage once the graph is dense and redundant, or whether — as is likely — a proper minimal cut-set of several edges becomes necessary [1], [2]. This would also be the point at which to profile the Neo4j commit stage, which was not benchmarked in this work.

11.2 Dynamic, telemetry-driven weighting

The second is dynamic telemetry ingestion [8]. Pulling in live signals from Microsoft Defender for Identity and Entra ID audit logs would let the engine assign contextual weights that reflect current conditions — an anomalous Conditional Access outcome, a multi-factor authentication gap, an open Privileged Identity Management activation window — rather than the static severity constants used here, which are the same on a quiet Tuesday as they are during an active incident.

11.3 A wider edge taxonomy

The third is broadening the edge taxonomy to cover Active Directory Certificate Services abuse, resource-based constrained delegation, and application service-principal consent grants, each of which introduces cross-tier reachability the present four-edge-type model simply cannot express, and each of which has its own published abuse primitives that could be scored in the same w(e)/c(e) scheme used here.

11.4 A proper minimal cut-set solver

The fourth is replacing the current greedy, first-edge root-cause heuristic with a proper minimal cut-set solver, drawing on the fixed-parameter and anytime algorithms of [1], [2] and [3], paired with the adaptive decision-elicitation model of [13] so that, on a graph where cutting one edge is not enough, operators can adjudicate a small set of proposed removals without unbounded manual review effort.

11.5 Operator-facing deployment

Finally, the Streamlit dashboard used for this evaluation is a research instrument, not an operations tool. Turning it into something a security team could run continuously — with role-based access to the graph itself, scheduled re-ingestion, and integration with existing ticketing systems so a discovered path becomes a tracked work item rather than a one-off report — is the natural next step once the scale and taxonomy questions above have been answered.

References

[1] M. Guo, J. Li, A. Neumann, F. Neumann, and H. Nguyen, "Practical Fixed-Parameter Algorithms for Defending Active Directory Style Attack Graphs," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, no. 9, pp. 9360–9367, 2022. doi:10.1609/aaai.v36i9.21167 https://doi.org/10.1609/aaai.v36i9.21167

[2] M. Guo, M. Ward, A. Neumann, F. Neumann, and H. Nguyen, "Scalable Edge Blocking Algorithms for Defending Active Directory Style Attack Graphs," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, no. 5, pp. 5649–5656, 2023. doi:10.1609/aaai.v37i5.25701 https://doi.org/10.1609/aaai.v37i5.25701

[3] Y. Zhang, M. Ward, and H. Nguyen, "Practical Anytime Algorithms for Judicious Partitioning of Active Directory Attack Graphs," Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI-24), pp. 7074–7081, 2024. doi:10.24963/ijcai.2024/782 https://doi.org/10.24963/ijcai.2024/782

[4] M. Elmiger, M. Lemoudden, N. Pitropakis, and W. J. Buchanan, "Start thinking in graphs: using graphs to address critical attack paths in a Microsoft cloud tenant," International Journal of Information Security, vol. 23, no. 1, pp. 467–485, 2024. doi:10.1007/s10207-023-00751-6 https://doi.org/10.1007/s10207-023-00751-6

[5] B. I. Mokhtar, A. D. Jurcut, M. S. ElSayed, and M. A. Azer, "Active Directory Attacks—Steps, Types, and Signatures," Electronics, vol. 11, no. 16, art. 2629, 2022. doi:10.3390/electronics11162629 https://doi.org/10.3390/electronics11162629

[6] A. D. Kent, L. M. Liebrock, and J. C. Neil, "Authentication graphs: Analyzing user behavior within an enterprise network," Computers & Security, vol. 48, pp. 150–166, 2015. doi:10.1016/j.cose.2014.09.001 https://doi.org/10.1016/j.cose.2014.09.001

[7] L. Sadlek, P. Čeleda, and D. Tovarňák, "Identification of Attack Paths Using Kill Chain and Attack Graphs," Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS), 2022. arXiv:2206.10272 https://arxiv.org/abs/2206.10272

[8] Q. Liu, K. Bao, W. U. Hassan, and V. Hagenmeyer, "HADES: Detecting Active Directory Attacks via Whole Network Provenance Analytics," IEEE Transactions on Dependable and Secure Computing, vol. 23, no. 1, pp. 936–953, 2026. doi:10.1109/TDSC.2025.3611866 https://doi.org/10.1109/TDSC.2025.3611866

[9] A. Robbins and W. Schroeder, "An ACE Up the Sleeve: Designing Active Directory DACL Backdoors," Black Hat USA 2017 whitepaper, Las Vegas, NV, 2017. https://blackhat.com/docs/us-17/wednesday/us-17-Robbins-An-ACE-Up-The-Sleeve-Designing-Active-Directory-DACL-Backdoors-wp.pdf

[10] X. Ou, S. Govindavajhala, and A. W. Appel, "MulVAL: A Logic-based Network Security Analyzer," Proceedings of the 14th USENIX Security Symposium, Baltimore, MD, 2005. https://www.usenix.org/conference/14th-usenix-security-symposium/mulval-logic-based-network-security-analyzer

[11] M. L. Artz, "NetSPA: A Network Security Planning Architecture," M.Eng. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2002. https://hdl.handle.net/1721.1/29899

[12] S. Jajodia and S. Noel, "Topological Vulnerability Analysis," in Cyber Situational Awareness, Advances in Information Security, vol. 46, Springer, 2010, pp. 139–154. doi:10.1007/978-1-4419-0140-8_7 https://doi.org/10.1007/978-1-4419-0140-8_7

[13] H. Q. Ngo, M. Guo, and H. X. Nguyen, "Adaptive Wizard for Removing Cross-Tier Misconfigurations in Active Directory," Proceedings of the 34th International Joint Conference on Artificial Intelligence (IJCAI-25), pp. 7661–7669, 2025. doi:10.24963/ijcai.2025/852 https://doi.org/10.24963/ijcai.2025/852

[14] L. Wang, C. Yao, A. Singhal, and S. Jajodia, "Implementing interactive analysis of attack graphs using relational databases," Journal of Computer Security, vol. 16, no. 4, pp. 419–437, 2008. doi:10.3233/JCS-2008-0327 https://doi.org/10.3233/JCS-2008-0327

[15] J. R. Dora, "Attacks on Active Directory — Kerberos Delegation: Exploitation of Active Directory Using Kerberos Constrained Delegation," IEEE, 2024. Available at IEEE Xplore, document 10820404. https://ieeexplore.ieee.org/document/10820404