M5: Análisis de Infraestructuras Empresariales

1. Fundamentos Teóricos de Análisis de Seguridad en Active Directory

Active Directory (AD) es la implementación de Microsoft de un **servicio de directorio** con un modelo distribuido y jerárquico que soporta autenticación, autorización y descubrimiento de recursos en entornos Windows empresariales. Entender su arquitectura lógica (espacio de nombres, particiones, esquema), su modelo de replicación multimáster, el comportamiento y propiedades de las relaciones de confianza (trusts) y el motor de políticas (GPO) es esencial para evaluar riesgos, diseñar controles y analizar incidentes a nivel de dominio/forest.

1.1 Arquitectura conceptual de Active Directory

1.1.1 Concepto: directorio, base de datos distribuida y espacio de nombres

- **Directorio**: AD actúa como un almacén estructurado (directorio LDAP-like) que guarda objetos (usuarios, equipos, grupos, OU, controladores de dominio, etc.) y sus atributos. El directorio sirve tanto para **identidad** (quién es) como para **autorización** (qué puede hacer).
- **Base de datos distribuida y jerárquica**: La información se organiza en *naming contexts* o *particiones* (domain, configuration, schema, application) y se replica entre Domain Controllers (DCs). Esta separación permite replicación selectiva y escalabilidad.

1.1.2 Teoría del espacio de nombres y estructura lógica

Namespace particionado (dominios, árboles y bosques):

- Un **dominio** es una partición administrativa: contiene los objetos (cuentas, grupos, equipos) del ámbito. Los DCs dentro del mismo dominio comparten réplicas de la partición de dominio.
- Un **árbol** es un conjunto de dominios con un espacio de nombres contiguo (p. ej. `sales.contoso.com` hijo de `contoso.com`).
- Un **bosque** (forest) agrupa árboles y representa la unidad de seguridad/administración más amplia (incluye el esquema y la configuración comunes). El **Global Catalog** facilita búsquedas parciales en todo el bosque.

Implicaciones de seguridad: el bosque define la frontera mayor de confianza administrativa — cambios forest-wide (p. ej. schema) afectan a todos los dominios. Por eso, operaciones forest-level requieren controles estrictos (FSMO roles, permisos, auditoría).

1.1.3 Nombres y contexto de nomenclatura (DN, RDN, CN — Estructura LDAP)

- **Distinguished Name (DN)**: identificador único de un objeto en el directorio, formado por una secuencia de *Relative Distinguished Names* (RDN) desde el objeto hasta la raíz (ej.: `CN=Juan Perez,OU=Usuarios,DC=corp,DC=contoso,DC=com`). Cada RDN es un par `atributo=valor`.
- **Relative Distinguished Name (RDN)**: nombre relativo (el componente más a la izquierda del DN) — por ejemplo, `CN=Juan Perez`.
- **Common Name (CN)**: atributo común usado a menudo en RDNs.
- **Consecuencia práctica:** distinguir entre *identidad lógica* (DN) y *identificadores utilizables para autenticación* (sAMAccountName, UPN). El DN es único y definitivo, pero no es la credencial que los usuarios introducen en un login.

1.1.4 Esquema extensible: clases, atributos y riesgos

- **Qué es el esquema:** el esquema define *qué* tipos de objetos y *qué* atributos son válidos en el bosque (p. ej. `user`, `group`, `computer`) y las reglas (cardinalidad, sintaxis) de cada atributo. El esquema es compartido por todo el bosque.
- **Extensibilidad:** el esquema puede extenderse (añadir atributos/clases) para integrar aplicaciones (Exchange, SCCM, aplicaciones propietarias) mediante LDIF u otras herramientas; dichas extensiones requieren privilegios del **Schema Master** (FSMO) y su replicación es forest-wide.
- **Riesgo de seguridad y gobernanza:** extender el esquema es una operación de alto impacto: errores o extensiones maliciosas pueden persistir y otorgar superficies de abuso o de fallos (p. ej. atributos que contengan información sensible o comportamientos distintos). Por tanto, cambios deben controlarse (review, backups, auditoría) y realizarse sólo con justificación.

1.1.5 Particiones (naming contexts) — dominio, esquema, configuración, aplicación y Global Catalog

- **Domain partition:** contiene cuentas y objetos propios de ese dominio; replica sólo entre DCs del dominio.
- **Schema partition:** contiene las definiciones (no datos) de clases/atributos; replica a todos los DCs del bosque.
- **Configuration partition:** contiene topología forest-wide (sites, subnets, NTDS settings, connection objects). Replica a todos los DCs del bosque.
- **Application partitions:** particiones opcionales que almacenan datos de aplicaciones (por ejemplo: DNS integrados en AD — `DomainDNSZones`, `ForestDNSZones`) y permiten replicación selectiva (réplicas sólo en DCs elegidos). Útil para limitar tráfico y alcance de datos dinámicos.

Consecuencia práctica: comprender particiones es clave para interpretar por qué un cambio (p. ej. a schema) se propaga a todo el bosque, mientras que otros cambios (p. ej. a datos de DNS en una application partition) pueden replicarse sólo a un subconjunto de DCs.

1.1.6 FSMO (Flexible Single Master Operations)

- **Problema:** AD es *multimaster* (cada DC puede aceptar escrituras), pero ciertas operaciones forest/domain-wide requieren una autoridad única para evitar conflictos (p. ej. asignación de RID pools, actualizaciones del esquema).
- **Solución:** Cinco roles FSMO (Schema Master, Domain Naming Master, RID Master, PDC Emulator, Infrastructure Master) asignan responsabilidades únicas para operaciones concretas; algunos son forest-level (Schema, DomainNaming), otros domain-level.

Implicación de seguridad/operativa: la protección de los DCs que ostentan roles FSMO es crítica: corrupción o compromiso del Schema Master o Domain Naming Master puede tener impacto forest-wide; por eso se ubican y protegen con políticas y respaldo.

1.1.7 Modelo teórico de replicación multimáster

1.1.7.1 Propiedades generales

- **Multimaster:** cualquier DC (excepto RODC en su caso) puede aceptar cambios originarios; esos cambios deben replicarse a otras réplicas del mismo naming context. La Knowledge Consistency Checker (KCC) genera la topología de replicación (connection objects) automáticamente según sites y costos de enlace.
- **Replicación a nivel de atributo y unidad de réplica:** AD registra metadatos por atributo (no sólo por objeto) y replica cambios en atributos, lo que permite granularidad en detección de cambios y en resolución de conflictos.

1.1.7.2 Metadatos de replicación: USN, invocationId, originator, versión de atributo

- **USN (Update Sequence Number):** contador local (por DC) que incrementa por cada cambio originado en ese DC; se usa para calcular qué cambios necesita traer un DC de sus socios. El USN es **local** a cada DC (no comparable entre DCs sin más información).
- **InvocationId / DC-GUID:** identifica la "instancia" del almacén de datos del DC (cambia si el DB se restaura desde backup incorrectamente); permite que réplicas detecten restauraciones/USN rollbacks y forzar nuevos epoch para evitar pérdida de cambios. El uso correcto del invocationId evita que un DC restaurado "oculte" operaciones ya ejecutadas.
- **Metadatos por atributo (msDS-ReplAttributeMetaData / msDS-ReplValueMetaData):** AD mantiene para cada atributo la información de la última modificación *originante* (timestamp, USN originador, versión de atributo y nombre del DC que la originó). Esto permite aplicar un criterio determinista para aceptar/rechazar actualizaciones entrantes.

1.1.7.3 Resolución de conflictos y consistencia eventual

- **Consistencia eventual (eventual consistency):** cuando múltiples DCs aceptan cambios, hasta que las réplicas convergen existe un periodo en el que distintos DCs pueden ofrecer visiones distintas del directorio. AD está diseñado para converger (replicable) pero no para consistencia inmediata.

- **Detección / resolución de conflictos:** AD compara *stamps* de atributo que incluyen versión (contador de cambios en ese atributo) y, en empates, usa timestamps como tiebreaker — en la práctica el comportamiento se aproxima a "last-writer-wins" según criterios de metadatos (pero la lógica real usa versión+origen+timestamp). Por esa razón, prevenir conflictos (diseño topológico, buenas prácticas en FSMO) es preferible.
- **Fallos típicos y riesgos operativos:** restauraciones/restores mal hechas pueden producir USN rollbacks que rompen la convergencia (requieren pasos correctivos: metadata retired, invocationId change); Microsoft documenta procedimientos de detección y recuperación.

1.2 Teoría de Trust Relationships y seguridad transitiva

1.2.1 Qué es una relación de confianza (trust)

Trust relationship es un mecanismo por el cual un dominio (o forest) acepta autenticar/aceptar tokens de usuarios del otro dominio; las trusts delegan la posibilidad de que un usuario autenticado en A acceda recursos en B según permisos. En entornos Windows estas relaciones se usan para permitir acceso cross-domain y compartir recursos.

1.2.2 Propiedades matemáticas (modelado mediante grafos)

Se puede modelar el conjunto de dominios como un **grafo dirigido** G=(V,E), donde los vértices V son dominios/forests y las aristas E representan trusts dirigidas. Las propiedades estándar que Microsoft implementa y administra son:

1. **Reflexividad (autoconfianza):** cada dominio confía en sí mismo naturalmente (hay un path trivial de longitud 0). En términos de grafo, existe una identidad; esto corresponde a que las cuentas locales del dominio son reconocidas dentro de él. *(Concepto general; Microsoft crea automáticamente trusts child<->parent al crear dominios hijos.)*
2. **Simetría (bidireccionalidad cuando es two-way):** una **two-way trust** indica que la relación permite autenticaciones en ambos sentidos (las autenticaciones pueden fluir de A→B y B→A). En el grafo, esto corresponde a una arista en ambas direcciones entre dos nodos. Microsoft crea trusts bidireccionales automáticamente entre dominio padre e hijo.
3. **Transitividad:** una **transitive trust** permite extender la confianza a través de caminos: si A confía en B y B confía en C (ambas transitive), entonces A confía en C (hay camino A→B→C). Microsoft genera trusts transitive en estructuras de árbol y forest, lo que simplifica administración, pero amplia el perímetro de acceso. En el grafo, la transitividad significa que la relación de confianza es cerrada por composición de aristas (siempre que las aristas sean transitive).

Nota práctica: la combinación de transitividad y bidireccionalidad extiende fuertemente el alcance: una concesión de acceso desde un dominio de frontera puede, si no se controla, permitir el movimiento lógico a través de varios dominios. Por eso, el diseño de trusts exige análisis de minimización de superficie y controles en reinos heterogéneos.

1.2.3 Implementación concreta y tipos de trusts (resumen)

- **Child / Parent trusts**: creados automáticamente en la jerarquía de dominios; son dos-way y transitive.

- **Tree-root trusts**: entre árboles dentro de un bosque (transitive).
- **External trusts / Realm trusts / Forest trusts**: para escenarios de integración con dominios no-Windows o con otros forests; según tipo pueden ser non-transitive o transitive y one-way o two-way.

1.2.4 SIDHistory, SID filtering y "quarantining" — Teoría y seguridad

- **SID (Security Identifier):** el identificador interno de un principal de seguridad en Windows; un SID incluye un *domain identifier* y un *RID* (Relative ID). Cuando se migra un objeto de un dominio a otro (por ejemplo, durante migraciones), la antigua identidad se guarda en `sIDHistory` para mantener acceso a recursos que aún usan el SID antiguo en ACLs.
- **Riesgo de SIDHistory:** si `sIDHistory` circula libremente a través de trusts (p. ej. trusts forest-wide con SIDHistory habilitado), un atacante en el dominio que suministra la SIDHistory podría obtener acceso a recursos en el dominio que la acepta (elevando su privilegio por la presencia de SIDs en el token). Este vector ha sido usado históricamente en ataques cross-forest.
- **SID Filtering (default behavior & quarantining):** por defecto Microsoft habilita SID filtering al crear trusts entre forests nuevos (limita qué SIDs entran en tokens a través del trust). SID filtering actúa en el borde de la trust: **filtra/descarta SIDs foráneos** (incluidos Valores en sIDHistory) para prevenir que un principal en la zona confiada lleve SIDs que el lado confiado no espera. Existe además la opción de "enable sid history" si la migración lo requiere; pero activarlo reduce la seguridad.
- **Quarantining:** término usado a veces en runbooks y documentación para describir el proceso de aplicar SID filtering o bloquear la propagación de SIDHistory hasta que se valide (es una medida de "cuarentena" sobre SIDs foráneos). Microsoft y guías de seguridad recomiendan mantener SID filtering activado salvo que una migración controlada lo necesite.

Implicación práctica importante: desde la perspectiva defensiva, las trusts (y cualquier configuración que permita transportar SIDs/identidades) amplían la superficie de privilegios. Revisar y minimizar trusts, aplicar SID filtering, y auditar cuentas con sIDHistory son controles recomendados.

1.3 Marco teórico de Group Policy Objects (GPOs)

1.3.1 Concepto y propósito

GPOs son la plataforma de Microsoft para administrar configuración (configuración de equipo y usuario, scripts, políticas de seguridad, modelos de restricción) de forma centralizada. Un GPO es un contenedor que incluye un conjunto de *settings* y un enlace (link) a un contenedor de AD (sitio, dominio u OU) que determina el alcance de aplicación.

1.3.2 Modelo de procesamiento: LSDOU y precedencia

- **Orden de procesamiento (LSDOU):** las GPOs se procesan en el orden: **Local → Site → Domain → Organizational Unit(s)** (donde para OUs anidadas la aplicación va desde OU padre hacia OU hijo, de fuera hacia dentro). Para políticas de equipo vs usuario existen matices (por ejemplo: las políticas de equipo se aplican durante el arranque; las de usuario al logon).
- **Precedencia y link order:** cuando hay varias GPOs en el mismo contenedor, se aplica la GPO según su *link order*; las GPOs con menor número de *link order* tienen mayor prioridad (se aplican después y por

tanto pueden sobreescribir valores previos). Es fundamental enseñar el concepto de precedencia para evitar conflictos de política.

- **Herencia y bloqueo:** GPOs son heredadas; es posible bloquear la herencia o forzar la aplicación con *Enforced* (antes *No Override*) y también excluir enlaces según diseño. Estas herramientas permiten modelar excepción/jerarquía.

1.3.3 Filtrado de seguridad (ACE / permisos) y WMI filtering

- **Security filtering (ACE-based):** además del contenedor al que está enlazado el GPO, la aplicación efectiva depende de los permisos ACL del GPO. Para que un principal (usuario/ordenador/grupo) aplique un GPO necesita permisos **Read** y **Apply group policy** sobre el objeto GPO; por defecto `Authenticated Users` tiene estos permisos, pero se puede restringir para aplicar solo a grupos concretos (security filtering). El motor de aplicación verifica estas ACEs en tiempo de aplicación.
- **WMI Filtering:** permite condicionar aplicación del GPO según consultas WMI (ej.: versión OS, membership de hardware, características). Si un GPO tiene un WMI filter, se evalúa en el destino y solo se aplica si la consulta retorna true. Útil para segmentación por atributos de máquina.

Consecuencia de seguridad: mal uso de security filtering (p.ej. eliminar `Authenticated Users` sin añadir grupos correctos) puede provocar que GPOs no se apliquen; por otro lado, dejar `Authenticated Users` por defecto puede llevar a aplicar GPOs inesperadas. Revisión de ACEs y delegación debe incluirse en auditorías de seguridad.

1.3.4 SYSVOL y replicación de GPOs (FRS vs DFS-R)

- **SYSVOL:** carpeta compartida en cada DC que contiene el contenido necesario para GPOs (plantillas, scripts, las versiones de los GPOs) y scripts de inicio/cierre; su integridad y replicación es crítica para que los GPOs se apliquen coherentemente en todos los DCs/hosts del dominio.
- **FRS (File Replication Service) vs DFS-R (Distributed File System Replication):** históricamente SYSVOL se replicaba con FRS; desde Windows Server 2008 Microsoft recomendó migrar a **DFS-R** por mejoras en eficiencia, robustez y escalabilidad. La migración es irreversible y conlleva pasos operativos; entornos con FRS obsoleto deben planificar migración.
- **Consistencia distribuida y desafíos:** la replicación de SYSVOL implica sincronizar archivos y metadatos; condiciones de red, conflictos de archivos o replicación fallida llevan a desincronización de GPOs entre DCs, lo cual genera problemas de aplicación de políticas en clientes (mismatched policy versions). Por ello, las operaciones sobre GPOs (crear/editar) deben supervisarse para garantizar replicación hacia todos los DCs relevantes.

1.3.5 Resolución de conflictos y versionado de GPO

Cada GPO tiene una versión y dentro de SYSVOL hay metainformación que permite comparar versiones; cuando hay diferencias (por ejemplo, tras edición en DC distinto) el sistema de replicación y el propio motor de política determinan qué versión se aplica según precedencia y el orden de enlace. La falta de sincronización puede dar lugar a "split brain" en configuraciones y settings contradictores.

Riesgos críticos derivados de los puntos anteriores

1. **Cambios de esquema / FSMO comprometidos:** cambios en schema o dominio (FSMO) tienen impacto forest-wide — controlar acceso al Schema Master y auditorizar.
2. **Trusts mal gestionadas:** trusts transitive y habilitación indiscriminada de SIDHistory pueden ampliar privilegios inesperadamente; habilitar SID filtering por defecto salvo necesidad justificada.
3. **Replicación y restauraciones:** malas restauraciones (o USN rollbacks) pueden romper replicación y generar inconsistencias de autorización; seguir procedimientos para restore/metadata en DCs.
4. **GPO / SYSVOL desincronizado:** afecta la capacidad de imponer configuración de seguridad y remediación; migrar FRS→DFSR y monitorear replicación.

Recomendaciones conceptuales para diseño/seguridad

- **Principio de menor privilegio administrativo** para roles FSMO y permisos de esquema.
- **Minimizar y auditar trusts**; activar SID filtering salvo migraciones controladas; documentar la razón y el alcance de cada trust.
- **Monitoreo de replicación y metadatos:** recolectar msDS-Repl* metadata y logs de Directory Service para identificar cambios sospechosos y origen de cambios (detectar modificación de grupos sensibles, etc.).
- **Asegurar SYSVOL y migrar a DFS-R** cuando proceda; auditar cambios GPO y el ACL de los objetos GPO para evitar aplicaciones accidentales.

2. Teoría avanzada de enumeración LDAP

2.1 Fundamentos del Protocolo LDAP

2.1.1 Modelo de información LDAP — Entradas, atributos y esquema

- **Entrada (Entry):** unidad básica del directorio; cada entrada representa un objeto (usuario, grupo, equipo, OU, etc.) y se compone de *atributos* (cada atributo tiene un tipo y uno o varios valores). El identificador único de una entrada es su *Distinguished Name (DN)*.
- **Atributos y multivaloridad:** muchos atributos son multivaluados (por ejemplo, `member` en un grupo puede contener múltiples DNs). Los atributos tienen sintaxis y restricciones definidas por el esquema (por ejemplo: string, integer, DN, SID). El esquema impone validación (qué atributos son válidos para cada clase de objeto, cardinalidad, matching rules).
- **Subschemas y enforcement:** el servidor LDAP expone un *subschema subentry* (metadatos) que indica qué reglas y tipos están disponibles; los servidores normalmente validan las operaciones frente al esquema, rechazando modificaciones que no cumplan la definición. En Active Directory el esquema es forest-wide y su modificación requiere privilegios especiales (Schema Master). Esto significa que la estructura y el tipo de atributos en AD están fuertemente regulados.
- **Entradas operacionales vs atributos de usuario:** existen atributos regulares (almacenados como parte de la entrada) y atributos *operacionales* o *calculados* proporcionados por el servidor (por ejemplo, `tokenGroups` es un atributo calculado en AD). El cliente debe distinguir entre ambos porque los operacionales pueden requerir consultas específicas o provocar carga adicional en el servidor.

2.1.2 Referrals y Chaining (redirección entre servidores LDAP)

- **Referrals:** cuando un servidor LDAP no es autoritativo para una porción del DIT (por ejemplo, otra partición o dominio), puede devolver una *referral* con la URL de otro servidor que sí lo es. El cliente decide si *chase* (seguir) la referral o no. RFC/implementaciones especifican el comportamiento y códigos de resultado.
- **Chaining (servidor actúa como intermediario):** algunos servidores pueden realizar chaining internamente (el servidor A contacta al servidor B y devuelve resultados al cliente), mientras que otros devuelven sólo la referral para que el cliente consulte directamente. En Active Directory, la manera de manejar referrals y si el cliente sigue o no las mismas tiene impacto directo en la visibilidad del árbol y en la efectividad de una enumeración cross-domain.

Implicación práctica (teórica): las referrals introducen una dimensión de *distribución* en el grafo: la DIT efectiva puede abarcar múltiples servidores con distintos permisos, latencias y políticas de paginación; por tanto, el algoritmo de enumeración debe contemplar la posibilidad de "nodos remotos" con coste extra.

2.1.3 Operaciones LDAP relevantes y sus propiedades

BIND (autenticación):

Mecanismos: *Simple Bind* (nombre + contraseña; puede enviarse sobre TLS para proteger credenciales) y *SASL Bind* (framework que soporta mecanismos como GSSAPI/Kerberos, DIGEST-MD5, EXTERNAL, etc.). En entornos AD, el uso de SASL GSSAPI (Kerberos) es habitual para integridad/autenticación segura del enlace.

SEARCH (búsqueda):

Parámetros clave: *baseDN* (punto de partida), *scope* (Base / OneLevel / Subtree), *filter* (expresión LDAP), *attributes* (lista de atributos a devolver), *controls* (p. ej. paged results). El SEARCH es la operación que permite descubrir estructura y atributos y por ende es la más utilizada en enumeración.

Alcances (scopes):

- `base` — sólo la entrada indicada.
- `oneLevel` — sólo los hijos directos del base DN (no incluye el propio base ni sub-hijos más profundos).
- `subtree` — el propio base y todos sus descendientes (búsqueda profunda).
  Estas distinciones determinan cuánta porción del DIT se examina con una sola operación.

MODIFY / ADD / DELETE (cambios en el directorio):

Operaciones que alteran el DIT; están sujetas a ACLs y esquema. En el contexto de enumeración son relevantes porque ciertos atributos (por ejemplo, msDS-AllowedToDelegateTo) suelen requerir controles para ser leídos/alterados y por tanto la capacidad de *ver* o *modificar* depende del permiso.

Controls and Extended Operations:

El protocolo permite *controls* (extensiones) que modifican comportamiento (p. ej. Simple Paged Results — RFC 2696 — para paginar resultados) y *extended operations* (por ejemplo, StartTLS). El uso de `paged results` es esencial para escalar consultas en bases de datos grandes sin agotar recursos.

2.2 Enumeración LDAP como problema de teoría de grafos

2.2.1 Modelado formal

- **Grafo dirigido/etiquetado:** modelamos el DIT (Directory Information Tree) como un grafo G=(V,E) donde cada vértice v ∈ V es una entrada LDAP (DN) y cada arista e=(u→v) representa una relación de contención jerárquica (por ejemplo: OU padre → OU hijo) o una relación asociativa (por ejemplo: `member` referencia a `user` DN). En AD hay, además, relaciones *indirectas* (grupos anidados, referrals entre DCs, trusts entre dominios) que se representan como aristas entre subgrafos o como hiper-edges.
- **Pesos y costes en aristas:** podemos anotar cada arista con un coste —latencia, necesidad de autenticación, coste computacional (por ejemplo, recuperar `tokenGroups` es más costoso que leer `cn`)— y así transformar la enumeración en un problema de *recorrido de grafo con costes*. Esto permite aplicar heurísticas (p. ej. preferir lecturas baratas para construir un mapa inicial).

2.2.2 Algoritmos de exploración (BFS, DFS, recorrido por niveles)

- **BFS (Breadth-First Search):** explora el grafo por niveles (primero todos los hijos a distancia 1, luego a distancia 2, etc.). Es útil para mapear la estructura organizacional y para detectar patrones de jerarquía (por ejemplo, descubrir todas las OUs de primer nivel antes de bajar). Complejidad temporal en un grafo con factor de ramificación promedio b y profundidad máxima d es $O(b^d)$ en el peor caso; su ventaja es que minimiza la profundidad de las consultas iniciales y produce un mapa por niveles.
- **DFS (Depth-First Search):** profundiza en una rama antes de retroceder; es más adecuado para análisis detallado de subárboles (por ejemplo: investigar la cadena de OU→subOU→objeto). Tiene la misma complejidad asintótica en peor caso, pero diferentes propiedades en cuanto a memoria (DFS usa menos memoria que BFS en grafos amplios) y patrones de IO (acceso profundo en el DIT).
- **Recorrido por niveles (level-order traversal):** variante de BFS orientada a recuperar relaciones de reporting y estructuras org-chart (útil para presentar mapas organizativos a un analista).

Elección del algoritmo en práctica teórica: la elección depende de la métrica objetivo (rapidez de cobertura global, profundidad de análisis, coste en DCs). Desde un punto de vista defensivo, entender que un adversario puede preferir BFS para obtener un "mapa" rápido y luego DFS para explorar ramas de interés es crucial para diseñar detecciones (p. ej. detectar patrones repetidos de búsquedas horizontales).

2.2.3 Teoría de complejidad aplicada a enumeración LDAP

- **Factor de ramificación b:** número promedio de hijos por contenedor (por ejemplo, número medio de objetos por OU). En entornos corporativos b puede variar mucho: OUs con cientos de cuentas versus OUs vacías. El coste de enumeración crece exponencialmente con d si b>1.

- **Profundidad d:** número de niveles de la jerarquía a explorar. Directorios muy "planos" tienen d pequeño y la enumeración es más barata; organizaciones con muchas delegaciones y OUs anidadas aumentan d y el coste.
- **Complejidad temporal:** en el peor caso $O(b^d)$. Sin embargo, en LDAP práctico hay factores que reducen la complejidad efectiva:

    - **Filtrado por ACLs**: el cliente solo "ve" entradas para las que tiene permiso; esto reduce el espacio efectivo explorado.
    - **Atributos seleccionados**: pedir sólo atributos necesarios reduce tráfico y coste de procesado.
    - **Paged results / controls**: permiten amortiguar coste por petición y controlar memoria.

- **Costes adicionales en AD:** atributos computados (ej.: `tokenGroups`) a menudo requieren operaciones agregadas/transitivas que son computacionalmente caras en el controlador de dominio y, por tanto, incrementan el "peso" de un vértice al ser consultado (ver nota sobre tokenGroups). Esto debe modelarse como coste por vértice.

2.2.4 Estrategias protocolarias para controlar el coste de enumeración

- **Paginación (RFC 2696 / Simple Paged Results):** el cliente solicita un número limitado de resultados por petición; útil para evitar respuestas enormes y para medir progresivamente el espacio. Importante en directorios con alto b o consultas `subtree`.
- **Uso de filtros selectivos y atributos proyectados:** en lugar de `objectClass=*` y `attributes=*`, concretar filtros (`(objectCategory=person)` o `(sAMAccountType=...)`) y pedir sólo `cn,sAMAccountName,memberOf` reduce transferencia y carga.
- **Scope control (usar oneLevel cuando sea suficiente):** evitar `subtree` innecesarios. `oneLevel` permite descubrir hijos directos con menos coste.
- **Chase control sobre referrals:** decidir si seguir referrals puede expandir de forma considerable el grafo explorado (si se recorre un bosque entero a través de referrals / cross-domain), con el coste y la necesidad de credenciales correspondientes.

2.3 Modelo Teórico de Atributos Críticos en AD

Nota: las operaciones siguientes describen atributos y su significado teórico — no instrucciones de explotación. Para cada atributo se detalla su semántica, tipo (almacenado vs calculado), implicaciones de replicación y coste de lectura.

2.3.1 sAMAccountType — clasificación de objetos de seguridad

- **Definición:** atributo entero que codifica el "tipo" de objeto de seguridad (ej.: `SAM_USER_OBJECT`, `SAM_GROUP_OBJECT`, `SAM_MACHINE_ACCOUNT`, etc.). Es útil para distinguir categorías a nivel de protocolo sin depender sólo de `objectClass`.
- **Usos en enumeración:** filtrar por `sAMAccountType` permite reducir el espacio de búsqueda (por ejemplo, sólo usuarios o sólo grupos), y por tanto es una herramienta fundamental para optimizar BFS/DFS en el DIT.

2.3.2 userAccountControl — máscara de bits de comportamiento

- **Definición:** atributo bitmask que reúne múltiples flags sobre la cuenta: deshabilitada, contraseña nunca expira, contraseña requierida, smartcardRequired, normal account vs trust account, etc. Cada bit tiene significado semántico y operacional.
- **Implicaciones:** al consultar `userAccountControl` se obtiene información relevante de estado de cuenta (por ejemplo, si está deshabilitada, si requiere smartcard, si es una cuenta de máquina). Esto es útil para priorizar análisis (por ejemplo, cuentas activas con contraseña que no expira son más interesantes desde el punto de vista de auditoría).
- **Coste:** lectura normal, no computada; bajo coste en DC, pero sensible desde la privacidad/privilegios (sólo usuarios con permiso pueden leerlo).

2.3.3 msDS-AllowedToDelegateTo — delegación Kerberos restringida

- **Definición y propósito:** atributo que en cuentas de servicio lista los SPNs (Service Principal Names) a los que la cuenta puede delegar (constrained delegation). Es usado para configurar a qué servicios una cuenta de máquina/servicio puede presentar tickets en nombre de usuarios.
- **Implicaciones de seguridad:** tener visibilidad de este atributo muestra qué cuentas están autorizadas para delegar y puede ayudar a modelar la superficie de exposición en escenarios Kerberos. Debido a su sensibilidad, la lectura/modificación está controlada y su enumeración puede requerir permisos elevados.
- **Consideraciones operativas:** en lecturas masivas su valor es relativamente pequeño (lista de SPNs), pero la enumeración de todas las cuentas con delegación en un dominio puede ser útil para el análisis de riesgo; por tanto, se considera un atributo "alto valor" desde la perspectiva de defensa/gestión de riesgo.

2.3.4 memberOf vs member — pertenencia directa vs pertenencia en grupos

- **`member` (en el objeto `group`):** atributo almacenado que lista los DNs de miembros directos del grupo. Es el registro "verdadero" de quién fue añadido directamente al grupo.
- **`memberOf` (en el objeto `user` o `group`):** atributo que lista los grupos de los que el objeto es miembro (lista construida por el DC). En AD `memberOf` no incluye el *primary group* (que se representa mediante `primaryGroupID`). `memberOf` suele ser mantenido como un atributo referencial, pero no es transitivo por sí mismo (no refleja automáticamente los miembros anidados a menos que el servidor compute un atributo distinto).
- **Diferencia clave y consecuencias:** `member` es la pertenencia explícita en el objeto grupo; `memberOf` es la vista desde la cuenta. Algunos valores (como nested groups) pueden requerir computación adicional para expandir la pertenencia transitiva; por eso existen atributos computados como `tokenGroups` que devuelven la lista completa de SIDs efectivos.

2.3.5 tokenGroups / tokenGroupsComputed / tokenGroupsGlobalAndUniversal — grupos efectivos (SIDs)

- **`tokenGroups` (atributo calculado):** atributo *computado* que contiene la lista de SIDs resultante de la expansión transita de la pertenencia (grupos locales/globales/universales) — es la lista de SIDs que Windows coloca en el *access token* del usuario. Es una forma rápida de saber «qué SIDs» tendrá un usuario sin realizar expansión recursiva localmente. **Nota técnica:** `tokenGroups` es una operación

costosa para el DC porque requiere calcular la expansión transitive; además no es replicado a menos que exista un Global Catalog con la capacidad de calcular memberships transitive.

- **Implicaciones de rendimiento y gobernanza:** solicitar `tokenGroups` masivamente puede sobrecargar DCs; por eso muchas APIs/clients lo usan con cuidado (BASE scope queries para un objeto a la vez). Desde el punto de vista de detección, lecturas repetidas o masivas de `tokenGroups` pueden indicar mapeo de privilegios.

2.3.6 msDS-PrincipalName / userPrincipalName (UPN)

- **Atributo UPN (`userPrincipalName`):** cadena en formato `user@domain` que sirve como nombre principal para Kerberos/SSO y suele mapear al correo electrónico del usuario; es el identificador usado para logon en muchos escenarios. El UPN es único por contenedor y su valor es visible en atributos estándar.
- **`msDS-PrincipalName`:** existen atributos y variantes introducidas por Microsoft en extensiones del esquema; en cualquier caso el concepto general es el mismo: identificar principalidades utilizadas por Kerberos/servicios. Consultar la especificación AD particular para su semántica exacta (estos atributos pueden ser operative/extension-specific).

Consideraciones finales — privacidad, rendimiento y controles de acceso

- **Privilegios y ACLs:** muchos atributos sensibles (delegation, ciertos SPNs, atributos operacionales) están sujetos a ACLs; la enumeración masiva efectiva depende tanto del modelo de permisos como de la configuración del DC.
- **Coste y throttling:** atributos computados (tokenGroups) y búsquedas `subtree` sin paginación son operaciones costosas; servidores pueden aplicar límites o causar latencia elevada. Enseñar a estimar coste (b y d) y a usar controls (paged results) para respetar la estabilidad del servicio.
- **Referrals y ámbito distribuido:** la enumeración cross-domain/forest requiere planificar chase de referrals y modelar el grafo como multi-servidor; en entornos con trusts/referrals la complejidad aumenta y el analista debe considerar latencia, permisos y replicación.

3. Teoría profunda de Kerberos y ataques avanzados

3.1 Fundamentos criptográficos y de protocolo de Kerberos

3.1.1 Panorama general del diseño

Kerberos (versión 5, especificada en RFC 4120) es un **protocolo de autenticación centralizada** basado en un **servicio de distribución de claves simétricas** (KDC). La idea esencial es permitir que dos entidades (cliente y servicio) se autentiquen mutuamente sin transmitir contraseñas en la red, usando tickets en los que el KDC cifra/autoriza información con claves derivadas del secreto del dominio.

Componentes lógicos:

- **KDC (Key Distribution Center):** entidad de confianza que incluye dos "roles lógicos" — el **AS (Authentication Service)** y el **TGS (Ticket Granting Service)** — y mantiene el secreto maestro del dominio (clave asociada a la cuenta `KRBTGT`).

3.1.2 Intercambios básicos y tipos de tickets (visión teórica)

Kerberos organiza la autenticación en tres intercambios principales (simplificado):

1. **AS-REQ / AS-REP (autenticación inicial):**
   - El cliente solicita un **TGT** al AS.
   - Si la autenticación inicial (por ejemplo, pre-autenticación) es satisfactoria, el AS devuelve un **TGT** cifrado con la clave del KDC (`K_{tgs}` — derivada del secreto de `KRBTGT`) y una **session key** (clave de sesión K_c,tgs) cifrada con una clave derivada de la contraseña del usuario (u otra forma según mecanismo).
   - El TGT contiene datos del cliente y es **consumible por el TGS** (no por el cliente): el cliente no puede leer el contenido del ticket sin la clave del KDC.
2. **TGS-REQ / TGS-REP (petición de ticket de servicio):**
   - Usando el TGT, el cliente solicita al TGS un **TGS/Ticket de servicio** para un servicio concreto (por ejemplo `HTTP/service.example.com`).
   - El TGS verifica el TGT, crea el ticket de servicio cifrado con la clave del servicio objetivo (`K_service`) y devuelve una **session key** cliente-servicio. El ticket de servicio (encabezado cifrado) viajará al servicio y será verificado por éste sin necesidad de contactar de nuevo al KDC.
3. **AP-REQ / AP-REP (established application authentication):**
   - El cliente presenta el ticket de servicio al servidor (AP-REQ) junto con un **authenticator** cifrado con la session key; el servidor valida el ticket, descifra el authenticator y puede opcionalmente responder con AP-REP para completar la autenticación mutua. El authenticator incluye timestamps y un checksum para evitar re-replay.

Consecuencia clave (seguridad del diseño): la confidencialidad e integridad de los tickets están garantizados mediante cifrado simétrico; la validez de un ticket depende de la clave secreta que usa el KDC/servicio, por eso la posesión de la clave `KRBTGT` permite **forjar TGTs** indistinguibles para los servicios.

3.1.3 Criptografía aplicable: claves, enctypes y derivación

- **Encriptación simétrica:** Kerberos v5 define el uso de enctypes simétricos (por ejemplo, RC4-HMAC en implementaciones históricas, y AES variants en implementaciones modernas). El ticket está cifrado con la clave del servicio al que está destinado (o con la clave KRBTGT para TGTs).
- **Session keys:** cada intercambio TGS-REP provee una *session key* efímera (K_c,s) que se usa para cifrar el authenticator y proteger la comunicación cliente-servicio. Estas claves son derivadas y no están basadas directamente en la contraseña del usuario una vez que el TGT fue emitido.
- **Pre-authentication y protección contra offline guessing:** Kerberos v5 incluye mecanismos de pre-autenticación (por ejemplo, `PA-ENC-TIMESTAMP`) para evitar que un actor capture AS-REP con datos que

permitan offline brute-force; la configuración y uso de enctypes modernos (AES) refuerzan la resistencia.

3.1.4 PAC (Privilege Attribute Certificate) — semántica y rol en autorización

- El **PAC** es un contenedor de datos de autorización (claims) que incluye el SID del usuario, las memberships y otros atributos necesarios para que el servicio aplique control de acceso. En la práctica Microsoft incluye el PAC dentro del ticket concesionado para que los servicios lo validen localmente. La PAC incorpora firmas / checksums que permiten a los servicios confiar en los datos del ticket (firmados por la clave de KDC o por claves derivadas).

Efecto: si un ticket contiene un PAC con grupos y privilegios, el servicio usará esos datos para construir el token del usuario; por eso forjar el contenido del PAC (o el ticket que lo contiene) es equivalente a forjar identidad/privilegios a ojos del servicio.

3.2 Teoría matemática y de protocolo de Pass-the-Hash (PtH)

3.2.1 Principio criptográfico del challenge-response en NTLM

- **NTLM** es un protocolo challenge-response: el servidor envía un *challenge* aleatorio y el cliente responde con un *response* que se calcula a partir del secreto (hash de la contraseña) y del challenge. El servidor verifica calculando el response esperado con el secreto que tiene almacenado (normalmente el NT hash). En este esquema la contraseña **no se transmite** en claro, pero sí es suficiente poseer el **hash** para computar respuestas válidas o para autenticarse con ese hash en muchos contextos.
- **Evolución LM → NTLM → NTLMv2:** LM (LanManager) usaba esquemas criptográficos débiles (DES con división de la clave) y es prácticamente roto. NTLM (NT hash) mejoró la representación (MD4 sobre UTF-16LE), y NTLMv2 añadió HMAC y mejoras para robustecer contra replay y mejorar bindings. Aun así, el diseño permite que el **NT hash** actúe de credencial equivalente en ciertos protocolos.

3.2.2 El hash como «credencial suficiente» — fundamento del PtH

- **Separación hash/password:** en NTLM la verificación se realiza con el hash (p. ej. NT hash, `NTOWF`), por lo que conocer el hash permite al adversario generar respuestas válidas al challenge o, en muchas implementaciones, presentar el hash directamente a la API de autenticación (esto es lo que se denomina Pass-the-Hash). En términos formales: si H=f(password) y el protocolo usa H para derivar claves de respuesta, entonces H es una *credencial delegable*.
- **LSASS como vector de obtención:** en Windows el proceso `LSASS` (Local Security Authority Subsystem Service) mantiene credenciales/derivados en memoria (hashes, tickets); el acceso a memoria privilegiada o a volcado del proceso es una forma práctica de extraer hashes. Desde la perspectiva teórica, la memoria es una *copia* de secrets críticos cuya exfiltración permite PtH o token impersonation. Microsoft documenta este riesgo y las mitigaciones (Credential Guard, protección de LSASS).

3.2.3 Token impersonation y propagación lateral (modelo formal)

- **Token impersonation:** en Windows, la identidad de proceso/usuario se representa por un *access token* con SIDs y privilegios. Si un actor obtiene credenciales (hashes) o tickets válidos, puede obtener un token con la identidad objetivo y ejecutar procesos con ese contexto — matemáticamente esto es una transición del sistema desde estado S (actor con credenciales X) a S′ (actor actuando como usuario Y).
- **Propagación lateral:** un atacante que puede autenticar hacia otras máquinas (por PtH o por reuso de tickets) genera un camino en el grafo de red/autenticación; la teoría de grafos se aplica: vértices = hosts / cuentas, aristas = posibilidad de autenticarse/ejecutar acciones remotas con las credenciales disponibles. La posesión de hashes expande las aristas disponibles y posibilita mover el control por el grafo de dominios.

Mitigaciones teóricas: restringir la capacidad de extraer secrets (protección de LSASS), minimizar cuentas con privilegios y segmentar el grafo (firewalls, SMB signing, políticas) para romper aristas explotables reduce la probabilidad de propagación.

3.3 Teoría del Golden Ticket y compromiso del KDC (KRBTGT)

3.3.1 Rol crítico de la cuenta KRBTGT y su clave maestra

- **KRBTGT** es la cuenta de servicio usada por el KDC para *firmar y cifrar* los TGTs emitidos por el dominio. La clave asociada a `KRBTGT` actúa como *clave maestra* que prácticamente valida cualquier TGT en ese dominio: quien posea la clave puede generar TGTs que serán aceptados por el TGS y por los servicios que confían en tickets firmados por el dominio.
- **Consecuencia lógica:** la seguridad del dominio frente a forja de Tickets depende explícitamente de la confidencialidad de la clave `KRBTGT`. Desde un punto de vista criptográfico, la forja de TGTs es trivial si se conoce la clave de cifrado/firma usada por el KDC: basta construir un ticket con los campos deseados, incluir un PAC con privilegios y firmarlo con la clave correcta.

3.3.2 Anatomía del Golden Ticket — semántica formal

- **Golden Ticket = TGT forjado** firmado con la clave del KDC que contiene claims (PAC) arbitrarios (por ejemplo, Domain Admin SIDs) y un **lifetime** controlado por el forjador. Un Golden Ticket replica la estructura de un TGT legítimo y, si está bien formado y firmado, es indistinguible para un servicio que procesa el ticket localmente (no consulta al KDC por cada uso).
- **PAC y firma:** el PAC dentro del ticket incluye las autorizaciones; su validez se verifica con las firmas que dependen de las claves de dominio (derivadas de KRBTGT). Por eso, la integridad del PAC sólo es garantizada si el servicio confía en la firma (es decir, en la clave del dominio). Si un adversario puede reproducir esa firma (posee la clave KRBTGT), la PAC puede contener cualquier privilegio.

3.3.3 Validez temporal, persistencia y el problema de remediación

- **Validez arbitraria:** al forjar el campo de validez (start/expiry) en un ticket, un atacante puede emitir tickets con longitudes de vida extensas (hasta máximos permisibles por la implementación), manteniendo acceso persistente aún tras rotación de contraseñas de cuentas administrativas — mientras

la clave KRBTGT usada para firmar el ticket no haya sido rotada y la verificación local del servicio confíe en esa firma.

- **Independencia del KDC:** puesto que los servicios validan localmente la firma del ticket, un Golden Ticket no requiere comunicación adicional con el KDC en cada uso; esto otorga independencia al atacante y dificulta la detección basada en comunicación KDC-servicio.
- **Recuperación compleja — rotación doble:** las guías de mitigación recomiendan **rotar la contraseña de la cuenta KRBTGT dos veces** con un intervalo entre rotaciones, porque hay claves anteriores en tickets ya emitidos; la rotación doble asegura que tanto la clave antigua como la inmediatamente anterior queden invalidadas para firmas existentes. Microsoft y proveedores de seguridad describen este procedimiento como parte de la remediación tras compromiso del KDC.

3.3.4 Detección teórica y señales relevantes

Aunque un Golden Ticket forjado parezca "válido", hay señales que permiten detectarlo si el monitoreo se modela correctamente:

1. **Anomalías en lifetimes y emisión de TGTs:** tickets con lifetimes fuera de la norma (por ejemplo, expiración inusualmente larga) o TGTs emitidos en horarios raros por cuentas no usuales.
2. **Uso de identidades fuera de patrón:** si un servicio recibe tickets de cuentas que no deberían iniciar sesiones en ese contexto (p. ej. un equipo de servicio recibiendo tokens de cuentas humanas con dominio admin SIDs).
3. **Eventos y logs de creación/uso de tickets:** correlación entre eventos inusuales en DC/LSASS y actividad en servicios objetivo. Herramientas EDR/IDS que recolectan telemetría Kerberos pueden detectar patrones de abuso previos a la concesión de un Golden Ticket.

Limitación: la detección requiere telemetría y correlación (no basta un log aislado), y por eso la prevención (proteger KRBTGT, limitar privilegios y segmentar) es crítica.

3.4 Modelo formal de riesgo y recomendaciones conceptuales

3.4.1 Modelo de amenazas (resumen formal)

- **Actores:** adversario con acceso privilegiado (dominio local admin) o con capacidad de extraer secrets de LSASS.
- **Recursos críticos:** hash/passwords de cuentas privilegiadas, clave KRBTGT (derivada del secreto de la cuenta), y la capacidad de crear/usar tickets.
- **Objetivo adversario:** forjar credenciales válidas (tickets o hashes) que permitan escalado y persistencia. En términos de grafo, el adversario crea nuevas aristas de confianza entre cuentas/hosts que antes no tenía.

3.4.2 Controles teóricos (alto nivel)

- **Proteger la confidencialidad de secretos en memoria** — reducir la posibilidad de extracción de LSASS (p. ej. Credential Guard, restricción de accesos privilegiados).
- **Minimizar alcance de cuentas privilegiadas** — principio de menor privilegio: reducir cuentas que pueden obtener KRBTGT-equivalentes y proteger FSMO/KDC hosts.

- **Rotación controlada de KRBTGT** — plan de rotación (doble rotación) cuando se sospecha compromiso; procedimiento que invalida tickets previos.
- **Detección basada en telemetría Kerberos** — reglas que detectan anomalías en issuance/usage patterns, duración de tickets, uso de cuentas con derechos inesperados.

4. Teoría de detección y mitigación avanzada

4.1 Modelo teórico de detección de anomalías en Active Directory

4.1.1 Enfoque: líneas base comportamentales (behavioral baselining)

Concepto: construir un modelo de "normalidad" a partir de telemetría histórica (logs de inicio de sesión, creación de objetos, solicitudes LDAP, TGS/TGT issuance, llamadas a servicios críticos, comandos administrativos) y detectar desviaciones significativas que puedan indicar actividad maliciosa. Esto es la base del *anomaly detection* aplicado al dominio.

Tipos de perfiles para modelar (niveles y atributos):

- **Perfiles de usuario:** horario típico de logons, origen de sesión (subred/host), servicios accedidos (SMB, LDAP, RDP), volumen y tipos de búsquedas LDAP, patrones de creación/edición de objetos.
- **Perfiles de servicio:** comportamiento esperado de cuentas de servicio (hosts que usan el SPN, frecuencia de ticket requests, duración de sessions).
- **Perfiles de sistema / endpoint:** procesos habituales, escalas de red saliente, llamadas a APIs sensibles; para DCs incluir eventos de replicación y SYSVOL activity.
  Modelar estos perfiles requiere selección y normalización de features (véase sección de ingeniería de características).

Supuestos y limitaciones del enfoque:

- Supone existencia de telemetría completa y representativa (si falta cobertura —p. ej. no hay logs de Kerberos— el modelo estará ciego).
- Riesgo de cambios reales en el entorno (p. ej. reestructuraciones o cambios de turnos) que invalidan la baseline si no se re-entrena.
- Generación de falsos positivos si la baseline no incorpora variabilidad legítima (p. ej. trabajos batch nocturnos).

4.1.2 Selección de features y pre-procesado (ingeniería de señales)

Categorías de features recomendadas:

- **Temporal:** hora del día, día de la semana, latencia entre eventos (p. ej. tiempo entre TGT y TGS), tasa de tickets por cuenta.
- **Espacial / red:** origen IP/subnet, salto entre hosts (topología), país/ASN de conexiones salientes.
- **Relacional:** número de grupos añadidos/eliminados por administrador X, cambios en ACLs, sucesión de búsquedas `subtree` sobre OUs sensibles.

- **Agregados:** conteos por ventana (p. ej. número de búsquedas LDAP `subtree` por minuto por cuenta), desviaciones respecto a la media histórica.
  Pre-procesado esencial: normalización (z-score / min-max), manejo de valores faltantes, codificación de categorías (embeddings o one-hot), y downsampling/upsampling para balanceo en entrenamiento supervisado.

4.1.3 Algoritmos de detección: teoría y trade-offs

1. Detección estadística de outliers (baseline + tests):

- Métodos: control charts, z-score, thresholds por percentil, modelos ARIMA para series temporales.
- Ventaja: interpretabilidad y bajo coste computacional.
- Desventaja: difícil para patrones multivariados y relaciones complejas; elevados falsos positivos si la distribución no es estacionaria.

2. Métodos de Machine Learning no supervisado (Anomaly detection):

- **Isolation Forest (iForest):** aísla puntos atípicos construyendo árboles aleatorios y midiendo la profundidad de aislamiento; eficiente y escalable para grandes datasets. Adecuado para telemetría ad hoc.
- **One-class SVM / Autoencoders (deep learning):** modelan la "normalidad" y detectan reconstrucciones/decisiones fuera de distribución. Autoencoders son útiles para captar relaciones no lineales entre features.
- **Clustering (DBSCAN, k-means + distancia):** identificar grupos de comportamiento y marcar elementos fuera de ellos.
  Trade-offs: iForest y clustering son buenos en entornos con pocos labels; deep learning requiere mucha data y tuning, y ofrece menor interpretabilidad.

3. Métodos supervisados (si hay labels):

- Usar modelos supervisados (Gradient Boosting, Random Forests, XGBoost, redes) cuando existen ejemplos de ataques/negativos. Proporcionan alto rendimiento, pero requieren datasets balanceados y actualizados; cuidado con *label leakage* y overfitting.

4. Signature-based (reglas y TTP matching):

- Detección basada en patrones conocidos (p. ej. MITRE ATT&CK TTPs). Muy útil para identificar técnicas ya observadas (Pass-the-Hash, Golden Ticket usage patterns), y de baja tasa de falsos cuando el patrón es específico. Debe combinarse con detección comportamental.

4.1.4 Diseño de sistemas híbridos y pipelines de detección

Arquitectura recomendada (conceptual):

1. **Ingesta y normalización** (logs Kerberos, Windows Event IDs, LDAP logs, Sysmon/EDR telemetry).
2. **Capa de reglas rápidas** (signature-based) para detección inmediata de IOAs/IOCs críticos.

3. **Capa de anomalía** (iForest / autoencoder) para detectar desviaciones y patrones nuevos.
4. **Correlación y enriquecimiento** (mapear alerts a MITRE ATT&CK, contabilizar relaciones entre hosts/cuentas).
5. **Prioritización y triage** (score combinado, confianza, impacto potencial).
   Justificación: un pipeline híbrido aprovecha la velocidad y precisión de firmas allí donde existen, y la capacidad de descubrimiento de ML para lo novedoso.

4.1.5 Métricas para evaluar detectores (teoría de evaluación)

- **Precision / Recall / F1:** métricas básicas; en seguridad se prioriza usualmente *high recall* para no perder incidentes, aunque esto eleva falsos positivos.
- **ROC AUC y PR AUC:** para modelos que devuelven scores; PR AUC es más informativo en datasets desequilibrados (p. ej. ataques raros).
- **Time-to-detect (TTD):** latencia media entre inicio de actividad maliciosa y alert.
- **False positive rate (FPR) y false alarm cost:** incluir coste operacional (analistas/human time) en la evaluación.
- **Robustness / adversarial tests:** evaluar resiliencia a adversarios que intentan camuflarse (p. ej. imitación de patrones legítimos).

4.1.6 Señales concretas útiles para detectar anomalías en AD (guía docente)

- Incremento súbito en número de búsquedas `subtree` por una cuenta administrativa.
- Emisión de TGTs/TGS fuera del horario habitual o a ritmos inusuales; lifetimes inusualmente largos.
- Lecturas masivas de atributos computados (ej.: tokenGroups) por cuentas no autorizadas.
- Creación/edición de GPOs y cambios en SYSVOL por cuentas que nunca lo hicieron.
- Patrones de movimiento lateral (RDP/SMB/WinRM desde hosts inusuales, múltiples inicios de sesión a diferentes hosts en ventana corta).

4.2 Teoría de hardening de Active Directory

4.2.1 Principio general: defensa en profundidad aplicada a AD

Idea clave: AD debe protegerse mediante capas: control de cuentas y privileges, endurecimiento de hosts críticos (DCs), control de red y segmentación, control de identidad/credenciales y monitoreo con respuesta. Las prácticas se alinean con Zero Trust y con benchmarks (CIS).

4.2.2 Principio de mínimo privilegio aplicado

Delegación granular y separación de funciones (SoD):

- **Delegación granular:** otorgar permisos mínimos y específicos (por OU, por atributo) en lugar de roles amplios; utilizar grupos de roles con permisos aceitados en lugar de cuentas humanas con permisos directos.
- **Separación de funciones:** dividir roles críticos (p. ej. gestión de GPOs, gestión de cuentas, backups, administración de FSMO) entre distintos operadores para evitar privilegios concentrados.

- **RBAC (Role-Based Access Control):** modelar permisos por roles y seguir políticas de cambio controlado (requests/approvals, just-in-time elevation) para minimizar acceso permanente. JIT (Just-In-Time) y JEA (Just Enough Administration) son patrones recomendados.

Argumento teórico: reducir la *cantidad de aristas* en el grafo de privilegios (menos cuentas con acceso amplio) reduce combinatoria de caminos que un atacante puede usar para escalar/propagarse.

4.2.3 Protección y aislamiento de controladores de dominio (DC hardening)

- **Aislamiento físico y lógico:** minimizar servicios en DCs, evitar que usuarios estándar se loguen en DCs, controlar el acceso administrativo y usar bastion hosts (administrative workstations) para tareas.
- **Endurecimiento OS & SYSVOL:** aplicar mejores configuraciones (CIS Benchmarks), endurecer SMB, deshabilitar servicios innecesarios y asegurar replicación (DFSR).
- **Protección de secretos en memoria:** técnicas como Credential Guard y reducción de procesos que pueden interactuar con LSASS.

Justificación: los DCs contienen piezas críticas (KRBTGT, base de cuentas); su compromiso tiene impacto forest/domain-wide, por tanto, deben ser minimizados como objetivo y protegidos con controles fuertes.

4.2.4 Teoría de segmentación de red aplicada a AD

Modelos de segmentación:

- **Aislamiento de DCs (network zones):** ubicar DCs en zonas con reglas estrictas (control de accesos a RPC/LDAP/Kerberos únicamente desde subredes autorizadas).
- **Microsegmentación:** implementar políticas de seguridad por workload (seguro a nivel de flujo) que limitan comunicación entre workloads solo a lo imprescindible; reduce la superficie para movimiento lateral. NIST Zero Trust apoya un enfoque de segmentación basada en políticas y verificación continua.

Efecto teórico: con segmentación adecuada, la probabilidad de que una cuenta comprometida salte lateralmente se reduce radicalmente porque muchas aristas de red/autenticación quedan bloqueadas o requieren controles adicionales.

4.2.5 Controles de configuración y políticas (checklist conceptual)

- Minimizar cuentas de servicio con privilegios elevados; usar Managed Service Accounts o group Managed Service Accounts (gMSA) donde proceda.
- Habilitar SMB signing, deshabilitar LLMNR/NetBIOS si no se usan, MFA para accesos administrativos, aplicar logon restrictions (workstation restriction) a cuentas privilegiadas.
- Aplicar CIS Benchmarks y mantener inventario/gestión del ciclo de vida de cuentas, GPOs y delegated roles.

4.3 Marco teórico de respuesta a incidentes en Active Directory

4.3.1 Principios generales y fases (adaptación NIST)

Basado en NIST SP 800-61 y guías especializadas, la respuesta a incidentes en AD se articula en fases: preparación, identificación/detección, contención, erradicación, recuperación y lecciones aprendidas. Estas fases se aplican con consideraciones AD-específicas (replicación, trusts, KRBTGT).

4.3.2 Teoría de contención de breaches en AD

Objetivos de contención: impedir que el atacante obtenga más privilegios o expanda su alcance en el grafo de cuentas/hosts. Estrategias conceptuales:

- **Aislamiento de cuentas comprometidas:** revocar/invalidar tokens y credenciales comprometidas (teoría: cortar aristas salientes del vértice comprometido). En la práctica esto pasa por reset de contraseñas, suspensión de cuentas y bloqueos de sesión.
- **Rotación de secretos (teoría de invalidez de credenciales):** cambiar contraseñas y secretos críticos para que credenciales extraídas pierdan validez. En AD esto incluye rotación de cuentas privilegiadas, claves de servicio y, en caso extremo, la rotación de la cuenta `KRBTGT` (procedimiento complejo). La rotación reduce la ventana temporal de validez de secretos robados.
- **Purga de sesiones / tokens:** invalidar sesiones activas y limpiar tokens de acceso (por ejemplo, forzar desconexiones, invalidar tickets Kerberos donde sea posible). Esto reduce la capacidad del atacante de continuar usando sesiones existentes.

Trade-offs conceptuales: contención rápida puede afectar disponibilidad (p. ej. reset masivo de cuentas causa impacto business). Por tanto, el plan debe priorizar activos críticos y balancear riesgo vs continuidad.

4.3.3 Modelo de recuperación post-compromiso (AD-centric)

1. Forensia de AD (alcance y metodología):

- Recolección de metadatos de replicación, logs de DCs (security event logs), SYSVOL changes, GPO edits, y análisis de msDS-Repl metadata para identificar origen y timeline de cambios. La correlación entre estos objetos permite reconstruir el recorrido del atacante en el grafo AD.

2. Reconstrucción de Trusts y validación de integridad:

- Verificar que no existan trusts maliciosos/externos añadidos, revisar SIDHistory, y comprobar que la topología de trusts se ajuste a la política de empresa. Si los trusts se ven comprometidos, su remediación debe planificarse con enclave y pruebas.

3. Hardening post-incidente:

- Implementar controles adicionales sugeridos por la post-mortem (segmentar hosts, endurecer DCs, reducir delegaciones, desplegar detección adicional) y ejecutar test de validación (red team / purple team) para confirmar eficacia.

4.3.4 Procedimientos críticos y recomendaciones conceptuales

- **Plan de comunicación y autorización:** procedimiento formal para quienes tienen autorización de cambiar contraseñas de cuentas privilegiadas o realizar rotaciones críticas (involucrar lineas de negocio).
- **Backups forenses y pruebas de restauración:** antes de cualquier cambio destructivo, realizar imágenes forenses y backups; la recuperación debe probarse offline y validarse contra criterios de integridad.
- **Rotación KRBTGT (si es necesaria):** plan controlado de doble rotación documentado (procedimiento complejo que debe realizarse siguiendo guías oficiales y con monitoreo exhaustivo).

5. Teoría de gobernanza y gestión de identidades (Identity Governance)

5.1 Modelo Teórico del Ciclo de Vida de Identidades

5.1.1 Estados formales del ciclo de vida (modelo de máquina de estados)

Podemos modelar una **identidad** (human user, service account, device identity) como una máquina de estados finita $M = (S, \Sigma, \delta, s_0, F)$ donde los estados relevantes típicos son:

- **Pre-Provision (Requested)**: solicitud creada (un flujo administrativo existe, pero aún no se ha materializado la cuenta).
- **Provisioned / Active**: cuenta creada y habilitada en el sistema de destino; credenciales provisionadas.
- **Modified (Role/Attribute Change)**: cambios de atributos/roles (promoción, cambio de OU, additions to groups).
- **Suspended / Disabled**: cuenta deshabilitada temporalmente (bloqueo por incidente, licencia, suspensión temporal).
- **Deprovisioned / Deleted**: cuenta eliminada o archivada; credenciales revocadas y accesos eliminados.
- **Archived / Retained**: estados donde los metadatos se conservan por cumplimiento/forense, pero la cuenta no está activa.

Transiciones δ son disparadas por eventos (HR event, approval, expiración, incidente) y deben registrarse con atributos de metadatos (quién, cuándo, por qué, evidencia). Este modelo permite razonar sobre ventanas de exposición y operaciones atómicas requeridas para coherencia entre sistemas.

5.1.2 Provisioning — teoría y requisitos (integridad, autorización, idempotencia)

Propósito: crear identidades con el conjunto mínimo de atributos y credenciales necesarios para cumplir funciones de negocio, garantizando trazabilidad.

Propiedades teóricas deseables:

- **Autoridad única de la verdad (Source of Truth):** identificar un sistema fuente (por lo común HRIS para humanos) que actúe como autoridad para atributos primarios (empleado activo, fecha de inicio/termino, rol). La existencia de una sola fuente evita conflictos semánticos y de sincronización.
- **Atomicidad e idempotencia de operaciones:** las operaciones de provisioning deben ser diseñadas para ser idempotentes (reintentos no producen duplicados) y atómicas respecto a la creación de la cuenta + asignación de permisos, o devolver un estado compensatorio en caso de fallo. Esto minimiza inconsistencias entre AD y sistemas target.
- **Aprobación y separación de funciones:** modelizar flujos multi-etapa (request → approver(s) → provisioning) con control de cambios y registro de evidencias (quién aprobó, tiempo de aprobación). Los flujos deben soportar políticas de expiración de approval y rollback.

Técnicas y patrones conceptuales:

- **Role-based templates / Entitlement templates:** proponer que el provisioning use plantillas basadas en rol (RBAC) en lugar de asignaciones ad-hoc para reducir la combinatoria de permisos. Esto se traduce a una función f : role→set permisos.

5.1.3 Sincronización de atributos entre sistemas (modelo de consistencia)

Problema: mantener coherencia de un conjunto de atributos AAA entre sistemas heterogéneos (HRIS, AD, SaaS apps, PAM).

Modelado: cada sistema S_i mantiene una vista parcial $V_i(A)$; la sincronización implementa funciones de reconciliación $R_{i\to j}$ que mapearán/transformarán atributos (normalización, formato, reglas de negocio). Es crítico definir:

- **Autoridad de atributo**: para cada atributo $a \in A$ definir la fuente primigenia src(a).
- **Política de resolución de conflictos**: reglas deterministas (timestamp-based last-write wins, versioning, or manual reconciliation) para resolver cambios concurrentes.

Casos prácticos de complejidad: formatos diferentes (e.g. nombre separado vs nombre completo), longitud de atributos, valores multivaluados (groups), y atributos calculados en AD (tokenGroups). La sincronización debe contemplar transformaciones seguras y validación semántica antes de aplicar cambios.

5.1.4 Deprovisioning automático y revocación (modelo de ventanas de riesgo)

Objetivo: minimizar la *ventana de exposición* WWW (tiempo entre evento de terminación/role change y la revocación efectiva de todos los accesos).

Estrategias y propiedades:

- **Event-driven deprovisioning:** vincular HR events (terminación, licencia no renovada) a triggers automáticos que deshabilitan/eliminen cuentas y revocan sesiones; así $W \to 0$ (prácticamente inmediato) en el mejor diseño. Requiere fiabilidad del pipeline y medidas compensatorias para errores.

- **Soft-delete vs hard delete:** primero deshabilitar (soft) para permitir reversión durante periodo forense; después archivar o eliminar tras retención definida por cumplimiento. Este enfoque balancea disponibilidad forense y reducción de riesgo.
- **Revoke credentials + purge sessions:** revocar claves, certificados y forzar purga de sesiones/tickets (teoría: invalidar tokens/tickets reduce aristas instantáneamente en el grafo de acceso). El diseño debe considerar dependencias (service accounts, scheduled tasks).

5.1.5 Identidades especiales: cuentas de servicio y cuentas privilegiadas

Modelado de riesgo: las cuentas no humanas (service accounts, gMSA, managed identities) tienen patrón de uso distinto y, teóricamente, mayor riesgo si no se gestionan ciclo de vida ni rotación de credenciales.

Propiedades recomendadas:

- **Credential management automatizado:** rotación periódica y automática de secrets, uso de vaults (PAM) como fuente de entrega dinámica.
- **Limitación de scope / JEA / JIT:** minimizar privilegios y aplicar elevaciones temporales.
- **Auditoría reforzada:** mayor granularidad de logging para operaciones que usan cuentas privilegiadas.

5.1.6 Métricas y KPIs (forma matemática y umbrales)

Definir métricas que permitan gobernar el proceso:

- **Time-to-provision (TTP):** tiempo promedio desde request → cuenta activa = TTP.
- **Time-to-deprovision (TTD):** ventana W comentada; objetivo: TTD ≤ SLA (p. ej. ≤ 1 hora para terminaciones).
- **Provisioning success rate (PSR):** $PSR=N_{success}/N_{total}$ en periodo TTT.
- **Orphan accounts:** número de cuentas sin owner / con lastLogon > X meses.
- **Access review completion rate:** porcentaje de recertificaciones completadas.

Estas métricas permiten modelar riesgo y priorizar remediación.

5.2 Teoría de Auditoría y Cumplimiento

5.2.1 Objetivos y alcance de la auditoría en AD

Objetivo principal: proveer evidencia verificable e íntegra que permita demostrar cumplimiento (políticas internas, regulaciones legales) y reconstruir acciones para análisis forense. La auditoría cubre eventos de autenticación, cambios críticos (GPO, ACLs, creación/edición de cuentas), replicación y administración de priv. roles.

5.2.2 Modelo de Auditoría Continua (arquitectura lógica)

Componentes:

1. **Generación de eventos (endpoints/DCs/services):** logs locales (security event log, Directory Service logs, Sysmon, application logs).
2. **Transporte seguro y reliable:** agente → colector/SIEM usando canales seguros; evitar pérdida de logs en el endpoint (buffering, ACK). NIST recomienda infraestructuras de log management robustas.
3. **Normalización/Enriquecimiento:** mapear event IDs a esquemas comunes (CEF, ECS) e integrar identidad/contexto (owner, OU, geolocation).
4. **Almacenamiento inmutable/retención:** logs preservados con controles WORM o almacenamiento inmutable, políticas de retención alineadas con cumplimiento.
5. **Correlación y análisis:** SIEM/UEBA combinando reglas (signature-based) y modelos de anomalía (behavioral baselining).
6. **Reporting y auditoría periódica:** informes automáticos, trails de recertificación y evidencias para auditores.

5.2.3 Políticas de retención y minimización (compliance trade-offs)

- **Principio:** retener lo suficiente para investigaciones y cumplimiento legal, pero respetar la minimización de datos (privacy). Políticas deben mapearse a regulaciones aplicables (GDPR, SOX, sectoriales). NIST SP 800-92 y guías de log management proporcionan criterios para diseñar retenciones y capacidades de búsqueda.

5.2.4 Cadena de custodia y preservación de evidencia digital

Definición: la cadena de custodia documenta el control, transferencia y almacenamiento de las evidencias para preservar su integridad y admisibilidad forense. En el contexto de AD/identidades, esto incluye snapshots forenses de DCs, volcado de logs y metadatos de replicación. CISA y NIST describen prácticas para documentar cada acceso/transferencia.

Elementos críticos a preservar:

- Timestamp confiable (usar NTP y firmar/hashear artefactos), registro de identidad del colector, hash de artefacto (SHA-256), almacenamiento inmutable y registros de acceso a las evidencias. Cada cambio en la evidencia debe quedar registrado.

5.2.5 Correlación de eventos y reconstrucción temporal (timeline)

Método: construir una **línea de tiempo forense** que combine:

- Eventos de autenticación (event IDs 4624/4625 /Kerberos events),
- Cambios en AD (audit directory service, SACL events),
- Logs de SYSVOL/GPO,
- Telemetría EDR/Network.

La correlación requiere normalizar timestamps (timezone/NTP), armonizar IDs de objeto (DNs, SIDs) y mapear acciones a actores (principal) y a recursos afectados. Este proceso es la base del análisis de causa raíz.

5.2.6 Análisis de causa raíz (root cause analysis) aplicado a incidentes en identidades

Fases conceptuales:

1. **Recolección y preservación de evidencia** (ver 5.2.4).
2. **Reconstrucción del camino de acceso**: modelar como grafo donde vértices = cuentas/hosts/objetos y aristas = autenticaciones/transferencias de privilegio; buscar caminos cortos que expliquen el escalado observado.
3. **Identificación de control fallido:** comparar contra políticas esperadas (provisioning rules, MFA enforcement, least privilege) y localizar la violación (por ejemplo: cuenta con permisos indebidos, ausencia de MFA).
4. **Medidas correctivas y validación:** implementar controles, rotaciones, y medir que la superficie de ataque se ha reducido (KPIs).

5.2.7 Indicadores, métricas de compliance y SLAs

- **SLA de TTD (Time to Deprovision):** objetivo máximo tolerable (ej.: ≤ 1 hora).
- **Coverage de logging:** porcentaje de DCs/endpoints que envían telemetría al SIEM.
- **Compleción de access reviews:** % de revisiones de acceso completadas en plazo.
- **Incidents per 1000 identities / detection rate:** métricas de efectividad del programa. Estas métricas alimentan dashboards de gobierno y auditoría.

5.3 Controles técnicos y organizativos recomendados (resumen)

Controles técnicos

- **Source of Truth (HRIS) integrado con provisioning automatizado (SCIM/Entra provisioning).**
- **Just-in-Time / Just-Enough admin y PAM para cuentas privilegiadas** (minimizar cuentas permanentes con wide privileges).
- **Automated deprovisioning / event-driven:** reducir TTD mediante integración HR → IDP/AD.
- **Centralized secure log collection + SIEM + retention policy** (WORM / immutable storage) y firma de evidencia.

Controles organizativos

- **Procesos de approval multi-factores** para creación de cuentas sensibles y cambios en delegaciones.
- **Access certification / recertification periódica** (programas de gobernanza) y documentación de excepciones.
- **Política de cadena de custodia y playbooks forenses** que definan roles, responsables y herramientas de preservación.

6. Conclusión teórica integradora

Este tema integra los marcos teóricos vistos en los temas previos (arquitectura AD, LDAP, Kerberos, detección, gobernanza) para presentar una **visión unificada**: cómo diseñar, defender y recuperar infraestructuras empresariales basadas en Active Directory siguiendo principios de seguridad por diseño, defensa en profundidad y gestión de riesgos basada en datos.

1. Síntesis de principios fundamentales

1.1 Seguridad por diseño (Secure by Design)

- **Definición práctica:** incorporar controles de seguridad desde la etapa de arquitectura —no como "parche"— mediando en decisiones de particionado de dominios, diseño de trusts, roles FSMO y asignación de minimales de privilegio. El principio reduce coste y riesgos a largo plazo. Esta idea está alineada con las prácticas "Secure by Design" y las guías de arquitecturas seguras.
- **Implicación para AD:** evaluar decisiones como el número de dominios/forests, placement de FSMO, uso de application partitions y el diseño de Global Catalog bajo la óptica del impacto denegable si un componente se compromete.

1.2 Defensa en profundidad (Defense-in-Depth)

- **Idea clave:** colocar múltiples capas de controles (identidad, endpoints/EDR, red/segmentación, detección y respuesta) de forma que la falla de una capa no implique la ruptura total de la defensa. La defensa en profundidad es un patrón arquitectural recomendado por agencias como CISA/NCCIC.
- **Aplicación en AD:** ejemplos de capas: (1) endurecimiento y aislamiento de DCs; (2) políticas RBAC / Just-In-Time; (3) controles de red (SMB signing, microsegmentación); (4) detección (EDR + UEBA + rules); (5) gobernanza y respuesta (PAM, rotación, playbooks).

1.3 Gestión de riesgos basada en datos (Data-driven risk management)

- **Marco teórico:** aplicar procesos de gestión de riesgos (NIST RMF y similares) con métricas cuantificables (probabilidad × impacto) y priorización basada en evidencia operacional (telemetría, scans de privilegios, exposición de servicios). El RMF de NIST establece pasos estructurados para este enfoque.
- **Consecuencia práctica:** las decisiones (p. ej. rotar KRBTGT, segmentar subredes, invertir en detección) deben fundamentarse en análisis de riesgo cuantitativo —no sólo en percepciones— usando KPIs definidos (Time-to-Detect, Time-to-Contain, cuentas privilegiadas expuestas, coverage de logs).

2. Evolución del panorama de amenazas — tendencias teóricas y consecuencias

2.1 Ataques cada vez más especializados

- **Observación:** los adversarios han profundizado conocimientos sobre AD (Kerberos, replication, trusts, GPO/SYSVOL) y explotan vectores compuestos (PtH, Golden Ticket, abuse of trusts). La literatura y guidance de agencias han documentado técnicas avanzadas dirigidas específicamente a AD.

- **Implicación:** defender AD requiere controles específicos (protección de KDC/LSASS, hardening DCs, auditar cambios en schema/FSMO, detección telemétrica Kerberos).

2.2 Automatización ofensiva y uso de IA/ML por atacantes

- **Tendencia:** los atacantes incorporan automatización y, en algunos casos, técnicas de ML para optimizar reconocimiento, generación de payloads y evasión. El avance de IA acelera la capacidad de los atacantes para explorar grandes entornos y adaptar técnicas.
- **Contramedida:** los defensores deben usar también automatización y ML (UEBA, detección adaptativa) pero con controles para evitar sesgos y adversarial ML. Evaluar resiliencia del detector ante adaptación adversarial es esencial.

2.3 Detección adaptativa: necesidad de sistemas que aprenden

- **Concepto:** detección estática ya no basta; los sistemas deben ser capaces de aprender (baseline, modelos anómalos), correlar eventos y priorizar alertas. UEBA e iForest/autoencoders son ejemplos de aproximaciones técnicas.
- **Riesgo:** dependencia excesiva en ML sin gobernanza genera falsos positivos y explicabilidad limitada; diseño híbrido (reglas + ML + analista) suele ser más robusto.

3. Marco integrador: cómo traducir teoría a arquitectura y operaciones

3.1 Arquitectura de referencia (visión integrada)

Un diseño resiliente para infra con AD debería incluir, al menos:

1. **Secure-by-design / control plane:** decisiones de forest/domain diseño, FSMO placement y esquema gestionadas con control de cambios.
2. **Identity hardening:** PAM, JIT/JEA, gMSA, rotación automát., restrict logon-to workstations.
3. **DC protection & network segmentation:** DCs en zonas protegidas, microsegmentación y reglas estrictas de RPC/Kerberos.
4. **Observability:** telemetría completa (Kerberos events, LDAP queries, EDR syscall/memory), centralizada y en almacenamiento inmutable.
5. **Detection pipeline:** reglas signature-based + anomaly detection + triage/correlation + playbooks de respuesta.
6. **Governance & recovery:** políticas de provisión, deprovisioning, cadena de custodia, planes de rotación KRBTGT y recuperación.

Cada bloque tiene métricas y SLAs asociados (por ejemplo: TTD ≤ X horas, coverage logs ≥ 95%, time-to-detect objetivo, % cuentas privileged bajo PAM).

3.2 Modelo de riesgo y priorización (formalización)

- **Riesgo** R_i para un activo/servicio i puede modelarse como:

$$R_i = \sum_j P_{ij} \times I_{ij}$$

  donde P_{ij} es probabilidad de amenaza j sobre i (estimada a partir de telemetría, exposiciones y vulnerabilidades) e I_{ij} el impacto (negocio). Adoptar NIST RMF para documentar controles y monitoreo reduce incertidumbre en P_{ij}.

- **Priorizar inversiones:** seleccionar controles que reduzcan producto P×I más eficientemente por coste (cost-benefit). Por ejemplo, rotación de cuentas privilegiadas y PAM reduce significativamente P para caminos de escalado, con coste relativamente bajo comparado a una re-ingeniería de la red.

4. Detección adaptativa y evaluación continua

4.1 Pipeline de detección y aprendizaje continuo

- **Loop de mejora:** ingest → features → rules & models → alerts → analyst feedback → retrain. Este bucle permite que las detecciones se ajusten a cambios legítimos en la organización y a nuevas tácticas adversarias.

4.2 Evaluación y métricas operacionales recomendadas

- **Time to Detect (TTD)**, **Time to Contain (TTC)**, **False Positive Rate (FPR)** ponderado por coste humano, **Coverage of Telemetry** (% DCs/endpoints reporting), **% privileged accounts under PAM**, **Mean time to remediate (MTTR)**. Estas métricas permiten medir la capacidad defensiva y priorizar mejoras.

5. Recuperación y resiliencia — principios para planificación post-compromiso

5.1 Preparación antes del incidente

- **Backups forenses de DCs y SYSVOL**, procedimientos de rotación KRBTGT testados en laboratorio (doble rotación), playbooks de comunicación y autorización. Estas acciones deben ser practicadas y validadas periódicamente.

5.2 Contención y remediación (principios)

- **Cortar aristas del grafo:** revocar credenciales comprometidas, purgar sessions/tickets, aislar hosts comprometidos. Balancear contención vs continuidad del negocio.

- **Reparación estructural:** después de contención, revisar trusts, ACLs, GPOs y FSMO roles; aplicar hardening adicional y reevaluar riesgos.

Defender Active Directory en una organización moderna no es una sola tecnología: es **arquitectura (secure by design) + operaciones (obs./detection) + gobernanza (policies/rotations)**. La mejora continua —apoyada por métricas, pruebas y simulacros— es imprescindible porque el panorama de amenazas evoluciona rápidamente y los atacantes usan automatización y técnicas adaptativas. Este enfoque multidisciplinar (redes, criptografía, detección estadística/ML, gestión de identidades) constituye la base para diseñar infraestructuras robustas y recuperables.