Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

Google Drive un progetto sperimentale di Cloud Storage Forensic

Relatore:
Chiar.mo Prof.
Danilo Montesi

Correlatore:
Ph.D.
Flavio Bertini

Presentata da:
Ossama Gana

III Sessione
2017/2018

Abstract

La Cloud Storage Forensic è una parte della Digital Forensics, è un approccio ibrido di Computer Forensics e Network Forensics. L'utilizzo dei Cloud Storage per immagazzinare dati aumenta di giorno in giorno perché gli utenti possono accedere ai dati da qualsiasi luogo in modo sicuro. Però, il cloud diventa appetibile per coloro che intendono utilizzarlo per scopi più o meno illeciti, data la possibilità di *"disperdere"* i dati in una vasta infrastruttura. Gli investigatori forensi si trovano in grande difficoltà ad acquisire gli artefatti digitali dal cloud, visto la complessa architettura che sta dietro al Cloud Storage. In questa tesi viene proposto un processo di acquisizione dei dati dal cloud in modo automatizzato. L'attuale legislazione non prende in considerazione l'acquisizione degli artefatti di questa tipologia, la legge è molto vaga sull'acquisizione forense del Cloud Storage, è più indicata all'acquisizione su dispositivi fisici. In questa dissertazione di tesi mostro come la procedura che ho sviluppato rispetti la maggior parte dei requisiti richiesti dalla legge.

Elenco delle figure

Elenco delle tabelle

Indice

Capitolo 1

Introduzione

La Digital Forensics, conosciuta anche come scienza digitale forense, è un ramo della disciplina forense che si occupa del trattamento dei dati digitali allo scopo di rilevare prove informatiche utili all'attività investigativa. Il suo scopo è quello di esaminare dispositivi digitali seguendo processi di analisi forense al fine di identificare, preservare, recuperare, analizzare e presentare fatti o opinioni riguardanti le informazioni raccolte. Nata agli inizi degli anni '80, quando i personal computer iniziano ad essere più accessibili ai consumatori, accrescendo però il loro utilizzo in attività criminali. In parallelo sono aumentati il numero di crimini riconosciuti come crimini informatici. Prima degli anni '80, queste attività criminali venivano trattate tramite le sole leggi esistenti.
L'informatica forense, nonostante fornisca svariati metodi per l'estrazione di artefatti digitali sempre crescenti, perde la sua flessibilità quando incontra l'applicazione della legge, rigida e carente di flessibilità. In Italia la legge di riferimento che definisce come utilizzare a livello giuridico i risultati dell'attività di analisi forense in tribunale, è stata promulgata nel 2008[1], come ratifica alla Convenzione del Consiglio d'Europa sulla criminalità informatica, stipulata nel 2001.

[1]Legge n. 48 del 18 marzo 2008, "Ratifica ed esecuzione della Convenzione del Consiglio d'Europa sulla criminalità informatica, fatta a Budapest il 23 novembre 2001, e norme di adeguamento dell'ordinamento interno".

1.1 Digital Forensics

Durante le procedure di investigazione gli investigatori forensi di solito seguono una serie standard di procedure e mantengono la catena di custodia[2]. Dopo aver fisicamente isolato il dispositivo in questione per assicurarsi che non possa essere accidentalmente contaminato, gli investigatori creano un'immagine digitale del supporto di memorizzazione del dispositivo. Una volta che il supporto originale è stato preso come immagine, il valore di hash viene calcolato dalle immagini utilizzando alcuni algoritmi come MD5, SHA-1, ecc.. per autenticare la prova rinvenuta. Tutte le indagini sono fatte sulla prova digitale. Gli investigatori utilizzano una varietà di tecniche e strumenti forensi di software proprietario per esaminare la copia, cercare cartelle nascoste e spazio su disco non allocato per copie di file cancellati, crittografati o danneggiati. Qualsiasi prova trovata sulla copia digitale è accuratamente documentata in un *"rapporto di ricerca"* e verificata con l'originale in preparazione di procedimenti legali che implichino scoperte, deposizioni o contenziosi effettivi. Un'indagine digitale forense, si divide in 3 fasi: acquisizione, analisi e rapporto.

Acquisizione Per l'acquisizione delle prove da un sistema informatico il modo migliore sarebbe quello di poter accedere al sistema con il ruolo di amministratore. Un altro aspetto molto importante riguarda la gestione degli elementi di prova acquisiti, il loro trasporto e archiviazione per evitare che le prove vengano alterate o comunque possa essere in discussione la loro integrità. Per una corretta documentazione del processo di acquisizione delle prove non si esclude la possibilità di filmare addirittura tutta la fase di acquisizione in modo da poter giustificare con chiarezza ogni singola operazione eseguita. Per quanto riguarda l'autenticazione delle prove va dimostrato che essa è stata eseguita senza modificare o in qualche modo turbare il sistema, le prove stesse vanno autenticate e verificate temporalmente con opportuni programmi, in modo da poter facilmente dimostrare in sede di giudizio che le operazioni di riproduzione delle prove sono state eseguite nei modi e nei tempi indicati. Nella fase di acquisizione si possono distinguere almeno tre livelli di copia:

[2]Il termine catena di custodia si riferisce alla documentazione cronologica o alla traccia cartacea che mostra il sequestro, la custodia, il controllo, il trasferimento, l'analisi, e la disposizione di elementi di prova, fisica o elettronica.

- *Copia di livello fisico (bit-stream copy)*: il contenuto di un'unità viene letto sequenzialmente caricando la minima quantità di memoria per poi registrarla nella stessa sequenza di un file binario ottenendo l'immagine fisica.

- *Copia a basso livello (cluster-copy)*: il contenuto di una partizione logica viene letto sequenzialmente caricando la minima quantità di memoria che il filesystem consente di indirizzare per poi registrarla su un file binario.

- *Copia del filesystem*: in cui parte o tutto il contenuto di alto livello di una partizione logica viene sottoposto a backup su di un file di un particolare formato che dipende dallo strumento utilizzato.

Analisi Per l'analisi delle prove appena acquisite occorre rispettare due principi: i dati oggetto dell'analisi non devono venire alterati e dovrà essere eseguita un'analisi dettagliata non solo all'interno dei file, ma anche nei settori del dispositivo di archiviazione lasciati liberi, perché potrebbero contenere dati registrati e cancellati in precedenza. In questa fase bisogna seguire un insieme di regole:

- *Minimo trattamento dei dati di partenza*: È necessario svolgere tutte quelle operazioni minime indispensabili a scopo forense che garantiscano la minima alterazione possibile del sistema in studio.

- *Logging delle attività*: Tutte le attività svolte durante l'analisi forense devono essere accuratamente registrate in un report ricco di particolari che consentano di evidenziare se siano state effettuate alterazioni dei dati originali.

- *Ammissibilità legale dei risultati*: Le prove digitali che si ottengono devono provenire da una procedura che rispetta a pieno le leggi locali.

Rapporto Una valida e completa relazione tecnica di un'attività forense dovrebbe contenere: la sintesi dei principi scientifici accademicamente riconosciuti su cui l'analisi ed il repertamento si basano, la catena di custodia dei reperti e la loro accurata descrizione. Il rapporto deve essere scritto pensando al lettore finale. Perché in molti casi il lettore non sarà tecnico e dovrebbe essere usata la terminologia appropriata per il lettore, infatti il rapporto finale deve avere le seguenti caratteristiche:

- *Sintetico*: Non necessita di riportare eccessivi particolari tecnici dell'analisi ma, solo ciò che interessa dal punto di vista giuridico.
- *Semplificato*: Colui che valuta l'esito, di solito, non è un esperto informatico e quindi bisogna eliminare terminologie non consuete cercando di spiegare ad un livello elementare.
- *Impersonale*: Non deve contenere giudizi personali dell'analista né tanto meno valutazioni legali sulle informazioni rilevate, a meno che non vengano esplicitamente richieste.

1.2 Cloud Storage Forensic

L'acquisizione solitamente viene effettuata su dispositivi fisici, ma al giorno d'oggi la memorizzazione dei dati digitali si sta orientando verso il Cloud Storage. I consumatori non immagazzinano i loro file solamente sui dispositivi fisici personali, infatti è sempre in grande crescita l'utilizzo dei cloud, ovvero spazi logici forniti da un provider per salvare tutti i propri file ed averli reperibili online su più dispositivi. Il cloud offre molteplici vantaggi, tra i quali grandi capacità di memorizzazione, in alcuni casi gratuite, e sempre disponibili poiché è sufficiente una connessione Internet per accedere ai dati salvati sulla *"nuvola"*. In alcuni casi però il cloud diventa appetibile per gli utenti malintenzionati che intendono utilizzarlo per scopi illeciti, data la possibilità di *"sparpagliare"* i dati in una infrastruttura di molteplici server, spesso dislocati in varie parti del mondo e in molti casi inaccessibili, sia per ragioni giuridiche sia tecniche. La Cloud Storage Forensic è quindi una nuova disciplina che studia il relativo crescente utilizzo di reti, computer e dispositivi di memorizzazione digitale impiegati in infrastrutture cloud, sfruttati per attività criminali hi-tech e tradizionali. L'acquisizione di dati dai cloud è più complessa, ma anche meno completa rispetto a quella dei dispositivi fisici e sicuramente con meno legislazione a riguardo.

1.3 Obiettivo

L'obiettivo della mia tesi è stato quello di sviluppare uno strumento software [4] che permetta di automatizzare il processo di acquisizione degli artefatti digitali dal cloud

storage, focalizzandomi in particolare su Google Drive. L'automatizzazione, oltre che far risparmiare tempo agli investigatori che si potrebbero concentrare più sulla fase dell'analisi, permette la futura standardizzazione di una procedura, ovvero l'utilizzo di un modello definito da un insieme di operazioni. Questo aspetto non è da sottovalutare visto che l'informatica forense fonda le sue basi su best practice radicate in alcuni standard ISO ed RFC che rappresentano le linee guida per chi esegue perizie informatiche.
L'architettura del cloud storage è più complessa rispetto a quella di un semplice dispositivo fisico, quindi prima di iniziare a sviluppare la mia tesi ho dovuto studiarla per capirne pregi e difetti. Le metodologie per acquisire gli artefatti digitali dal cloud sono molteplici ed è stato necessario capire quali fossero le più utili per il mio progetto di tesi. Inoltre, ho analizzato l'attuale legislazione in vigore, per capire come ottenere un'acquisizione valida dal punto di vista legale.

Nei prossimi capitoli della mia dissertazione di tesi analizzerò l'architettura sottostante al cloud storage, ovvero quella del Cloud Computing. Successivamente discuterò dell'attuale stato dell'arte della Cloud Storage Forensic, per effettuare un confronto tra il mio progetto di tesi ed altri progetti svolti in questo ambito descrivendone i pregi e difetti. Oltre che discutere l'aspetto teorico del mio progetto di tesi, descriverò la fase di progettazione e di sviluppo analizzando le scelte implementative che ho deciso di intraprendere.

Capitolo 2

Cloud Computing

In riferimento alla definizione di cloud computing fornita dal National Institute of Standards and Technology (NIST) si afferma che il cloud computing è un modello per abilitare, tramite la rete, l'accesso diffuso, agevole e a richiesta, ad un insieme condiviso e configurabile di risorse di elaborazione (ad esempio reti, server, memoria, applicazioni e servizi) che possono essere acquisite e rilasciate rapidamente, con minimo sforzo di gestione o di interazione con il fornitore di servizi. Questo modello cloud è composto da cinque caratteristiche essenziali, tre modalità di servizio e quattro modelli di distribuzione [5]. La definizione in sé può sembrare molto vaga, infatti viene usata in diversi contesti con significati differenti tra loro.

2.1 Caratteristiche

Le cinque caratteristiche essenziali del cloud computing sono [5]:

- *Self-service su richiesta*: Un consumatore può unilateralmente sfruttare le capacità di fornitura di calcolo, il tempo di utilizzo del server e dello storage di rete, in base alle proprie necessità, in maniera automatica e senza interazione umana con il provider.
- *Ampio accesso in rete*: Le funzionalità sono disponibile in rete e accessibili tramite piattaforme eterogenee (smartphone, tablet, computer, ecc)
- *Condivisione delle risorse*: Le risorse di calcolo del provider sono raggruppate per servire più consumatori insieme. Diverse risorse fisiche e virtuali vengono assegnate

dinamicamente, infatti il cliente non ha alcun controllo o conoscenza sulla posizione esatta delle risorse assegnate.

- *Elasticità rapida*: Le risorse possono essere elasticamente acquisite e rilasciate, in alcuni casi, automaticamente. Al consumatore, le risorse disponibili spesso appaiono illimitate e disponibili in qualsiasi quantità, in qualsiasi momento.

- *Servizio misurato*: L'utilizzo delle risorse può essere monitorato, controllato e segnalato, fornendo trasparenza sia lato fornitore che consumatore del servizio utilizzato. I sistemi cloud controllano e ottimizzano automaticamente l'uso di queste risorse.

2.2 Modalità di servizio

I servizi di cloud computing possono essere di tre tipologie:

- *Software as a Service (SaaS)*: Consiste nell'utilizzo di programmi installati in un server remoto.

- *Platform as a Service (PaaS)*: Piattaforma software che può essere costituita da diversi servizi, programmi e librerie.

- *Infrastructure as a Service(Iaas)*: Oltre alle risorse virtuali in remoto, vengono messe a disposizione anche risorse hardware, quali server, capacità di rete, sistemi di memoria, archivio e backup.

2.3 Modelli di distribuzione

Il cloud computing ha 4 livelli di distribuzione [5]:

- *Cloud privato*: L'infrastruttura cloud è fornita ad uso esclusivo da parte di un'organizzazione.

- *Cloud comunitario*: L'infrastruttura cloud è fornita per uso esclusivo da parte di una comunità di organizzazioni con interessi comuni.

- *Cloud pubblico*: L'infrastruttura cloud è fornita per un uso aperto a qualsiasi consumatore.
- *Cloud ibrido*: L'infrastruttura è una composizione di due o più infrastrutture di quelle appena elencate, che rimangono entità distinte, ma unite attraverso le tecnologie che abilitano la portabilità di dati e applicazioni.

Questa differenziazione è utile in ambito forense per definire le responsabilità di un cloud storage.

2.4 Livelli cloud computing

È utile scomporre gli ambienti del cloud computing in una pila di livelli, come mostrato nella Figura 2.1, dal più basso:

- *Hardware*: Corrisponde alla macchina ospitante, è definito da tutti i suoi componenti e dalla sua architettura.
- *Virtualizzazione*: È un livello intermedio in mezzo ai sistemi fisici e logici, "inganna" il sistema operativo soprastante, facendogli credere di avere accesso diretto a delle risorse hardware in realtà virtuali.
- *Sistema Operativo*: È il livello del sistema operativo della macchina ospite, come anticipato nel livello precedente, quest'ultimo lavora come se fosse installato su una macchina fisica.
- *Middlware*: Parte che permette l'interfacciamento tra i diversi applicativi con l'hardware.
- *Runtime*: Gestisce l'esecuzione dei programmi
- *Dati*: Corrisponde ai dati contenuti sulla macchina virtuale.
- *Applicazione*: È il livello delle applicazioni installate sulla macchina virtuale.

I diversi livelli potrebbero essere gestiti da parti diverse. In una distribuzione privata, come ad esempio in ownCloud, l'intero stack è ospitato dal proprietario e l'immagine forense

in questo caso è molto simile all'indagine su un dispositivo fisico. In una distribuzione pubblica invece la classificazione Saas/PaaS/IaaS diventa importante perché definisce quali sono le responsabilità, la proprietà, la gestione dei dati e dei servizi.

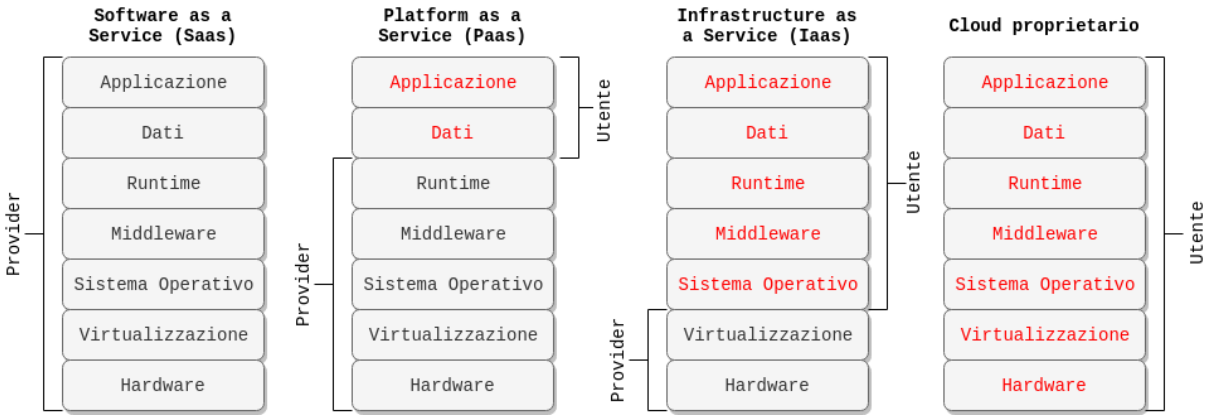


Figura 2.1: Livelli *Cloud Computing*

Figura 2.1 dimostra come gli obiettivi possano essere diversi in contesti molto eterogenei. L'approccio più produttivo allo sviluppo di soluzioni pratiche è quello di iniziare dai casi specifici, ma più comuni. Per questo motivo per il mio lavoro di tesi, mi sono focalizzato sulla ricerca forense del servizio di cloud storage offerto da Google, a partire dall'acquisizione dei file contenuti in quest'ultimo.

Capitolo 3

Stato dell'Arte Cloud Forensics

Nel corso degli anni, sono stati proposti un buon numero di standard forensi digitali, ovvero un insieme di tecniche e strategie per recuperare ed analizzare dati digitali. Tuttavia, questi metodi esistenti potrebbero non essere adatti allo scopo nell'ambiente cloud [6]. Vi è la necessità di definire un quadro per l'analisi forense che riesca a guidare le indagini in maniera più flessibile, per essere in grado di utilizzarlo con futuri fornitori che offriranno nuovi servizi. Negli standard usati dagli esaminatori forensi digitali il National Institute of Justice (NIJ) ha identificato 4 fasi che avvengono durante l'investigazione informatica forense: identificazione, conservazione, analisi e presentazione delle prove digitali. Nell'investigazione sui servizi cloud, ci sarà un ciclo di identificazione e conservazione che dovrebbe derivare dall'analisi di prove già sequestrate, ove ce ne siano. Questo perché un esaminatore non dovrebbe interrompere l'analisi delle prove già sequestrate, solo per aspettare che i dati identificati nel cloud siano conservati e forniti per l'indagine. L'esaminatore dovrebbe effettuare un'analisi continuativa sulle prove disponibili e quando vengono forniti nuovi dati dal cloud dovrebbe includerli nell'analisi. Quindi questo tipo di approccio può essere visto come un processo ciclico. La struttura sottostante, Figura 3.1, contiene i passaggi di: inizio (ambito), preparazione e risposta, identificazione e raccolta, conservazione (copia forense), analisi, presentazione, feedback e completamento. Questo è il terzo esperimento per convalidare il framework per garantire che sia forense ed è abbastanza flessibile da funzionare con diversi servizi di cloud storage [1].

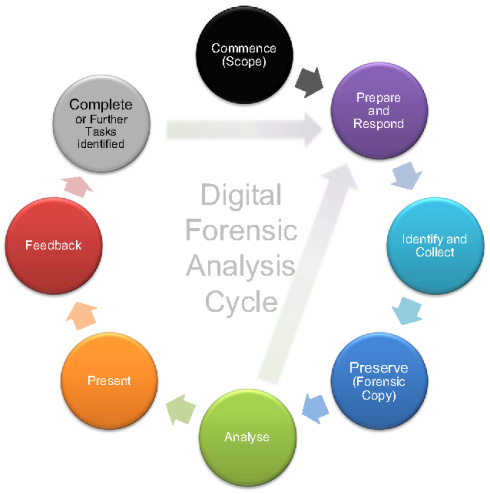


Figura 3.1: Ciclo analisi *Cloud Forensics* [1]

Un'altro caso di studi interessante, è stato svolto da Martini e Choo [7] su ownCloud, un cloud proprietario. Per questo caso di studi sono stati in grado di recuperare gli artefatti inclusi i metadati di sincronizzazione e gestione dei file, inoltre sono riusciti a recuperare i file memorizzati nella cache che descrivono i file salvati dall'utente sul dispositivo client e caricati nell'ambiente cloud o viceversa. Essendo che il proprietario ha in gestione tutti i livelli del cloud computing è possibile utilizzare un approccio più vicino a quello standard della Digital Forensics.
Durante un'indagine l'analista dovrebbe interpretare lo stato del sistema sulla base di alcuni punti cardine:

- *Completezza*: Questo aspetto non può essere sempre rispettato perché c'è una dipendenza dai dati lato client, il dispositivo di memorizzazione potrebbe non avere tutti i dati memorizzati localmente. Visto che i nuovi cloud storage offrono grandi spazi di memoria, che superano quella disponibile sul disco rigido.

- *Correttezza e riproducibilità*: Le applicazioni sul client vengono aggiornate frequentemente con nuove funzionalità, quindi risulterebbe difficile rimodulare la struttura dell'analisi forense in base ai continui aggiornamenti.

- *Costo e scalabilità*: Come anticipato nel punto precedente, l'analisi manuale risulterebbe molto costosa in termini di tempo, perché non si adatta alla rapida crescita dei servizi sul client offerti dal provider.

Capitolo 4

Acquisizione basata sulle API

Nel Capitolo 3, abbiamo visto come le attuali metodologie forensi sui cloud storage si sono concentrate sull'adattamento del tradizionale approccio forense per trovare artefatti digitali nell'applicativo client di un cloud storage. In questa sezione discuteremo delle limitazioni dell'analisi investigativa eseguita sul lato client e il perché è preferibile l'utilizzo delle API per l'acquisizione dei dati dal cloud.

4.1 Problemi acquisizione lato client

Il problema dell'acquisizione dei dati sul lato client, per i dati cloud, è che si tratta di un processo di acquisizione by-proxy. Ovvero, anche se assomiglia all'acquisizione tradizionale di un supporto fisico, questo metodo non punta alla fonte dei dati nel cloud. Come illustrato nella Figura 4.1, il contenuto del client viene visualizzato come una copia cache dei dati contenuti nel cloud storage. Questo fatto ha importanti ripercussioni sulla validità dell'acquisizione forense.

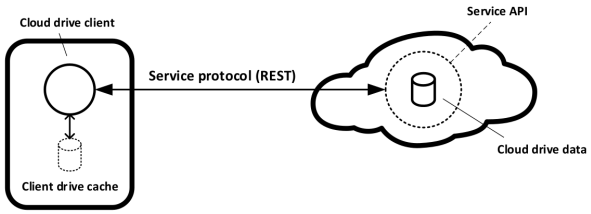


Figura 4.1: Diagramma architetturale del cloud storage per applicativi client [2]

Alcune delle problematiche più importanti dell'applicativo client sono:

- *Replica Parziale*: Non possiamo avere la garanzia che il client disponga di una copia completa dei dati contenuti nel cloud. Molti provider offrono ai propri utenti grandi spazi di archiviazione.

- *Revisioni*: La maggior parte dei servizi di cloud storage fornisce una sorta di cronologia delle revisioni per ogni file. Questa potrebbe risultare essere una preziosa fonte di informazione, che ha pochi analoghi nell'approccio tradizionale forense, ma gli analisti non sono abituati a cercarli. Le revisioni però risiedono nel cloud storage, il client contiene solamente la versione più aggiornata di ogni file.

- *File cloud-native*: Sono artefatti digitali che non hanno una rappresentazione serializzata nel filesystem locale. Un esempio noto è quello di Google Docs, i documenti vengono archiviati nel client come collegamento al documento, che potrà essere modificato solamente dalla Web App di Google. Acquisire un collegamento risulta inutile per una copia forense valida, perché l'obiettivo è il contenuto dell'artefatto digitale e non le informazioni del file.

In breve, abbiamo visto che l'approccio client-drive per l'acquisizione delle unità ha grossi difetti concettuali; abbiamo bisogno di un metodo diverso che ottenga i dati direttamente dal servizio cloud.

4.2 Google Drive REST API

Per risolvere in parte le problematiche appena discusse, per il mio lavoro ho utilizzato un approccio diverso rispetto alle metodologie usate negli articoli citati nel Capitolo 3. Per l'acquisizione dei dati, ho usato l'API ufficiale di Google Drive fornita dal provider stesso. Google ha implementato, come per la maggior parte dei suoi servizi, un'API per Google Drive che consente agli sviluppatori di terze parti la creazione di applicativi per gestire il proprio spazio sul cloud storage. Previa autorizzazione dell'utente, l'applicazione può gestire tutti i file presenti su Drive: si possono creare file, rimuovere quelli già esistenti, si posso gestire i permessi su ogni singolo file ed altro ancora. Il vantaggio immediato utilizzando un simile approccio, come descritto da Roussev, Barreto e Ahmed [2], ha

la particolarità di eliminare tutto il lavoro dell'ingegneria inversa[1] e presenta i seguenti vantaggi:

- L'API sono interfacce ufficiali ben documentate attraverso le quali le applicazioni cloud sul client comunicano con il servizio di cloud storage; tendono a cambiare lentamente e i cambiamenti sono chiaramente marcati, solo le nuove funzionalità devono essere incorporate in modo incrementale nello strumento di acquisizione.
- È facile dimostrare completezza e riproducibilità utilizzando le specifiche API.
- L'API Web tendono a seguire schemi che consentono di adattare il codice esistente a un nuovo servizio (simile) con uno sforzo modesto. È spesso pratico scrivere uno strumento di acquisizione per un servizio completamente nuovo da zero, perché lo si riesce a fare in poco tempo.

In termini generali, un cloud storage fornisce un servizio di archiviazione simile a quello di un filesystem locale, consentendo la creazione e l'organizzazione dei file. Pertanto, l'API offerta da Google assomiglia vagamente a quella del filesystem fornita dal sistema operativo locale.
Questo tipo di approccio comporta l'acquisizione di prove logiche, non fisiche. Tradizionalmente è considerata una best practice ottenere i dati al livello più basso possibile di astrazione perché equivale ad avere una prova più attendibile. La motivazione principale è che l'acquisizione logica dei dati potrebbe non essere completa dal punto di vista forense, perché non tiene conto dei dati eliminati. Inoltre qualche utente esperto sarebbe in grado di nascondere i dati dalla visualizzazione logica.
Ma come affermano Roussev, Barreto e Ahmed [2], tuttavia, è importante esaminare e tenere conto dei nuovi sviluppi tecnologici. Infatti, noteremmo che le unità dichiarate solide (SSD) e anche gli hard disk (HDD) ad alta capacità di nuova generazione somigliano molto più a computer di archiviazione autonomi rispetto alle periferiche limitate che si utilizzavano 10 anni fa, perché alcuni di essi contengono processori ARM ed eseguono complessi algoritmi di bilanciamento del carico e di usura, che includono il trasferimento dei dati in background. Sebbene supportino, ad esempio, l'accesso a livello di blocco, i risultati non si collegano direttamente a un layout fisico dei dati; questo rende l'immagine

[1] l'ingegneria inversa è il processo di analisi di un sistema software esistente, eseguito al fine di crearne una rappresentazione ad alto livello di astrazione.

acquisita logica, non fisica. Tutto ciò potrebbe portare ad una accettazione più ampia dell'acquisizione logica dal punto di vista forense.
Inoltre come da Figura 4.1, l'applicativo client del cloud storage utilizza la stessa identica interfaccia per eseguire le operazioni. Quindi l'API risulta essere il livello di astrazione più basso disponibile, pertanto, ci risulta appropriato per l'elaborazione forense. Per di più, i metadati dei singoli file spesso includono hash crittografici del contenuto, che consentono una forte garanzia di integrità durante l'acquisizione.

Capitolo 5

Progettazione

Il progetto del mio lavoro di tesi è diviso in due parti: una parte front end con il quale l'utente si interfaccia ed una parte back end che acquisisce in automatico tutti gli artefatti digitali presenti in un cloud storage.
L'obiettivo del mio progetto è quello di automatizzare i processi di acquisizione forensi dei dati. Questo farebbe risparmiare tempo e lavoro agli esaminatori, che si potrebbero concentrare più sulla parte dell'analisi. Oltre che essere efficiente, l'automatizzazione permette la creazione di una procedura standard, ovvero l'utilizzo di un modello definito da un insieme di operazioni. Come già discusso, questo aspetto è importante visto che l'informatica forense si basa su alcuni standard ISO ed RFC che delineano le linee guida per chi esegue perizie informatiche. Quindi questi standard guidano l'informatico forense nelle fasi della perizia informatica, dall'acquisizione delle prove a fini legali per utilizzo in tribunale fino alla presentazione della relazione tecnica forense e alla valutazione dei risultati.
La Figura 5.1, descrive brevemente il lavoro che ho prodotto. L'utente da un computer, dotato delle librerie necessarie per il funzionamento del mio progetto, esegue uno script che importa ed avvia l'immagine di una macchina virtuale, quindi viene data istruzione di catturare il traffico di rete della macchina virtuale.
All'avvio la macchina virtuale fa partire un programma che richiede le autorizzazioni all'utente per gestire i file di Google Drive, successivamente scarica tutti i file contenuti sul drive: file di proprietà non condivisi, file di proprietà condivisi, file non di proprietà condivisi con l'utente, file nel cestino e le revisioni dei file.

Oltre ai file il programma ricrea l'albero dei metadati, questa scelta implementativa è stata fatta per fornire all'esaminatore importanti informazioni riguardanti ogni file, come ad esempio: la data di creazione e di ultima modifica, i permessi dell'utente su ogni singolo file e ove sia presente l'hash del file. Inoltre, ricrea due file *".csv"*, uno per i file ed uno per le revisioni, per visionare le informazioni dei metadati da un unico file. Quando la macchina virtuale ha finito di ottenere gli artefatti digitali, si spegne in automatico. Successivamente viene creata una copia forense del disco virtuale e viene salvato il traffico di rete.

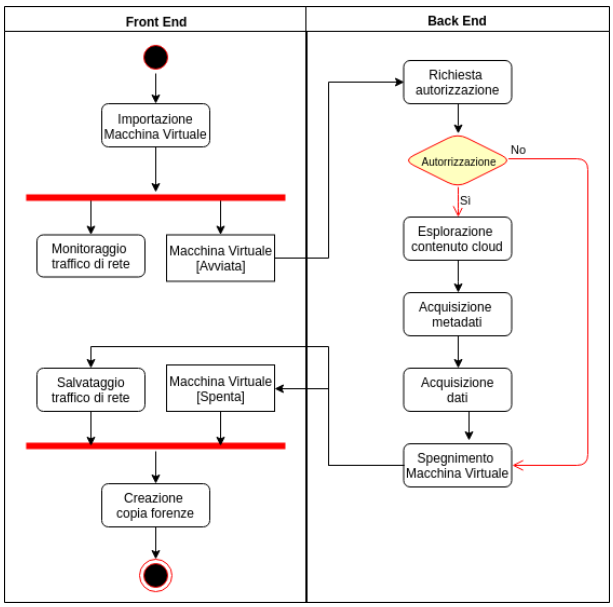


Figura 5.1: Diagramma architetturale attività

Come mostrato nella parte back end della Figura 5.1, concettualmente l'acquisizione degli artefatti dal cloud storage si dividono in tre fasi:

- esplorazione del contenuto
- acquisizione dei metadati
- acquisizione dei dati

Durante l'esplorazione del cloud storage lo strumento che ho sviluppato interroga il provider ed ottiene un elenco di dati sulle risorse (metadati) grazie ai quali successivamente otterrà le risorse stesse. Durante il processo di esplorazione, viene creato un elenco degli obiettivi

che vengono passati allo strumento di acquisizione. In base alla tipologia della risorsa utilizzo vari metodi per acquisire singolarmente ogni file.
In un'implementazione più avanzata si potrebbe sfruttare al meglio le funzionalità di ricerca concesse dal provider (API) per finalizzare l'esplorazione del contenuto su una specifica tipologia di target. Per il nostro scopo, risulta più efficace utilizzare un approccio base: ovvero ottenere tutte le risorse contenute nel cloud storage.
Secondo Roussev, Barreto e Ahmed [2] l'approccio base non è sostenibile per gli obiettivi del cloud: la quantità totale di dati può essere enorme e la larghezza di banda disponibile potrebbe essere limitata. Infatti per il loro caso di studi, hanno implementato uno strumento minimalista per la ricerca su target specifici, ovvero su una ristretta tipologia di file. Questo approccio non è adeguato per il mio intento, ovvero quello di ottenere un'istantanea completa dei dati contenuti nel cloud storage. Il loro strumento è pensato più per avere un'acquisizione dei dati contenuti solamente sullo spazio personale, senza includere i file nel cestino o i file non di proprietà che vengono condivisi con l'utente.

5.1 Acquisizione con validità legale

L'informatica forense, sebbene fornisca svariate procedure per l'estrazione degli artefatti digitali sempre crescenti perde la sua flessibilità quando incontra l'applicazione della legge, rigida e carente di flessibilità. Durante l'acquisizione dei dati nel mio progetto di tesi si possono notare due fasi critiche, che hanno bisogno di essere certificate dal punto di vista legale.
La prima riguarda l'acquisizione dei dati ottenuti dal cloud storage, questi dati viaggiano su rete e non sul filesystem. Per validare il download del file mi avvalgo di due pratiche:

- Marcatura temporale dei file scaricati.
- Cattura dei pacchetti di rete durante il download dei file.

La marcatura temporale avviene attraverso il salvataggio del timestamp nei metadati di un file dopo l'effettivo download. Il traffico di rete invece viene catturato attraverso alcuni strumenti per poi essere salvato in un file adeguato: i pacchetti che entrano ed escono dall'ambiente virtuale passano per la *Virtual Network Interface Controller (VNIC)*[1].

[1]Una VNIC è una scheda di rete virtuale basata sulla scheda di rete fisica.

L'altra fase critica è la copia forense del disco virtuale in cui vengono salvati i dati. La copia forense è il risultato dell'acquisizione di una evidenza digitale che ha *"duplicato"* il contenuto della prova generandone sostanzialmente un *"clone"* identico all'originale destinato a diventare una prova in ambito giudiziario. Il concetto di copia forense differisce dalla semplice copia di file o di un disco rigido, perché implica il soddisfacimento di alcuni requisiti richiesti dalla legge 48 del 2008 [8]:

- Deve avere impatto minimo, se non nullo, sulla fonte originale dei dati, le attività di acquisizione forense dei dati devono essere svolte *"adottando misure tecniche dirette ad assicurare la conservazione dei dati originali e ad impedirne l'alterazione"*.

- Deve essere identica all'originale, identità se possibile dimostrabile tramite metodologie scientifiche. Le copie forensi devono essere eseguite *"con una procedura che assicuri la conformità dei dati acquisiti a quelli originali e la loro immodificabilità"*.

- Deve essere il più completa possibile e riprodurre il dato originale acquisendone non soltanto i contenuti ma anche eventuali metadati o informazioni di contorno, come possono essere i dati del filesystem (nome file, attributi, data di creazione, modifica e accesso ai file) o le aree non allocate del sistema (dalle quali è poi possibile recuperare file cancellati ma non ancora sovrascritti).

- Deve essere *"statica"*, cioè una volta generata non deve essere immodificabile ovvero qualunque modifica deve poter essere rilevata e identificata, proprietà questa ribadita dalla legge e realizzata in genere tramite il calcolo di funzioni matematiche chiamate *"hash"* sul dato originale e sul dato acquisito, con verbalizzazione delle attività e confronto dei valori hash calcolati sui due supporti.

- La copia bit-stream prodotta tramite acquisizione forense deve essere verbalizzata in modo da poter reggere eventuali opposizioni su metodi, strumenti o tecniche utilizzate, riportando dati relativi alla catena di conservazione dei reperti e ai vari calcoli di valori hash che garantiscono l'integrità dei dati nei vari passaggi.

Questi requisiti fanno riferimento alle metodologie standard dell'informatica forense, ovvero alla tecniche di acquisizione effettuate su dispositivi fisici, ma non tengono conto dell'innovamento tecnologico che è avvenuto in questo ultimo decennio.
Per l'acquisizione da un cloud storage i suddetti requisiti non possono essere soddisfatti

completamente perché in primis l'acquisizione non avviene su un obiettivo fisico, ma su uno spazio logico offerto all'utente dal provider, quindi non è possibile copiare anche le parti non allocate dal sistema: infatti in questo caso viene effettuata un'acquisizione logica dei dati e non fisica. Inoltre non si possono acquisire i metadati forniti dal filesystem, ma il servizio di cloud ci propone un'interessate alternativa, fornendoci i metadati attraverso il servizio API.
L'immagine che viene creata nel mio progetto di tesi è statica, fornisco anche l'hash sulla copia prodotta, ma per la natura dell'origine dei file non è stato possibile fornire l'hash del dispositivo contenente i dati originali, essendo quest'ultimi salvati su un disco virtuale dinamico: l'hash prodotto per la copia forense è stato effettuato sull'intero contenuto del disco virtuale, quindi anche sui bit a zero, e non solo sull'effettiva grandezza di quest'ultimo, anche se l'output dello strumento che ho utilizzato per la copia forense è grande quanto il contenuto effettivo del disco virtuale. Quindi, giustamente, i valori prodotti dalla funzione hash, per l'originale e per la copia, sono differenti.
Lo strumento utilizzato per la creazione della copia forense, oltre che essere utilizzato largamente in ambito legale, fornisce molte funzionalità che permettono di attribuire maggiore validità all'acquisizione: la copia forense creata viene arricchita con alcuni metadati importanti, come l'hash della prova, con il quale lo strumento successivamente potrà verificare la validità dell'immagine. Quindi, anche se in qualche modo si riesca a modificare la prova, lo strumento non riuscirà a ritenere valida quest'ultima.

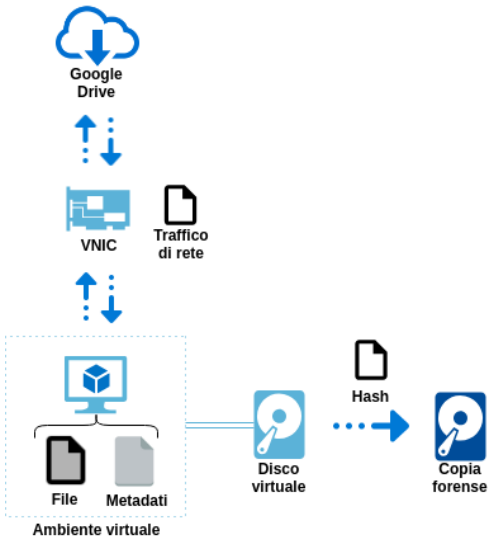


Figura 5.2: Schema acquisizione con validità legale

Nella Figura 5.2 mostro le fasi che si susseguono durante l'acquisizione degli artefatti dal cloud. Si possono notare il file del traffico di rete della scheda di rete virtuale a validare l'acquisizione dei dati dal cloud storage e l'hash calcolato durante la creazione della copia forense per garantire l'autenticità della prova.
Durante l'acquisizione fornisco dei report contenenti tutte le informazioni sull'acquisizione: timestamp inizio e fine procedura, identificativo del caso (nome della macchina virtuale), importazione macchina virtuale, inizio del monitoraggio del traffico di rete, avvio e spegnimento della macchina virtuale, salvataggio del traffico di rete, creazione della copia forense e il valore di hash del file del traffico di rete. Inoltre lo strumento che ho utilizzato per la creazione della copia forense fornisce un file di registro contenente tutte le informazioni sulla procedura.

5.2 Strumenti

Per lo sviluppo del mio progetto di tesi ho avuto bisogno di alcuni componenti software:

- Per la virtualizzazione della macchina ho utilizzato *Oracle VM VirtualBox* [9], che permette l'importazione di macchine virtuali e il salvataggio del traffico di rete dell'ambiente virtuale in una file PCAP.

- Per creare un output valido, dal punto di vista forense, ho utilizzato *FTK Imager* [10]. Questo strumento mi è servito per creare una copia forense del disco virtuale.

I software appena elencati sono a licenza gratuita, proprio perché vorrei che questo progetto possa raggiungere la più grande cerchia di utenti possibile. Per la virtualizzazione avrei potuto utilizzare VMware, un software per la creazione di macchine virtuali largamente utilizzato nelle analisi forensi, anche esso offre una versione non commerciale a licenza gratuita ma per i miei scopi avrei avuto bisogno della versione commerciale: per la gestione della macchina virtuale durante il processo di acquisizione utilizzo la funzionalità a riga di comando (CLI), VMware non permette l'utilizzo di questa funzionalità nella versione non commerciale.

5.2.1 Virtualizzazione

Oracle VM VirtualBox è un software gratuito e open source per l'esecuzione di macchine virtuali per architettura x86 e 64bit che supporta Windows, GNU/Linux e macOS come sistemi operativi host, ed è in grado di eseguire molti sistemi operativi [11]. Le tecniche e le funzionalità fornite da *Oracle VM VirtualBox* mi sono state utili per i seguenti motivi:

- *Esecuzione simultanea di più sistemi operativi*: Consente di eseguire più di un sistema operativo alla volta. In questo modo si possono effettuare più acquisizioni, su diversi utenti simultaneamente.

- *Importazioni semplici*: Invece di fornire solamente il disco virtuale all'utente, risulta più pratico fornire l'intera immagine di una macchina virtuale.

- *Test e ripristino di emergenza*: Una volta installata una macchina virtuale, i relativi dischi rigidi virtuali possono essere considerati un contenitore che può essere arbitrariamente congelato, riattivato, copiato, sottoposto a backup e trasportato tra gli host.

- *Consolidamento dell'infrastruttura*: La virtualizzazione può ridurre in modo significativo i costi dell'hardware e dell'elettricità. La maggior parte delle volte, oggigiorno, i computer usano solo una parte della loro potenza potenziale e funzionano con carichi di sistema medi ridotti. Si sprecano così molte risorse hardware e l'elettricità. Quindi,

invece di eseguire una macchina virtuale su ogni computer, si possono eseguire più macchine virtuali su un potente computer.

Open Virtualizzation Format L'Open Virtualization Format (OVF) è uno standard aperto per la creazione e la distribuzione di applicazioni virtuali o più comunemente di software che possa essere eseguito su macchine virtuali.
Lo standard descrive un "formato aperto, sicuro, portabile, efficiente ed estensibile per la pacchettizzazione e distribuzione di software che possa essere fatto eseguire su macchine virtuali" [12]. L'unità di distribuzione è l'OVF Package (Pacchetto OVF) che può contenere uno o più sistemi virtuali, ognuno dei quali può essere eseguito su di una macchina virtuale. I pacchetti OVF sono spesso descritti come file OVF, in realtà sono una raccolta di più file all'interno di una directory . Uno di quest'ultimi è il file *".ovf"*, un descrittore XML che delinea i metadati del pacchetto come ad esempio: nome, requisiti hardware e riferimenti ad altri file all'interno del pacchetto. Anche le immagini del disco e i file dei certificati sono contenuti in un pacchetto OVF.

Durante il mio progetto di tesi, ho provato vari approcci per importare un ambiente virtuale. Inizialmente avevo provato ad utilizzare direttamente un disco virtuale VMDK, questo formato non contiene informazioni sull'hardware virtuale di una macchina, come la CPU, la memoria, il disco e le informazioni di rete. Quindi, un amministratore che desideri distribuire un disco virtuale deve configurare tutte queste informazioni sulla nuova macchina, spesso manualmente.
Il formato OVF, invece, fornisce una specifica completa della macchina virtuale. Questo include l'elenco completo dei dischi virtuali richiesti e la configurazione dell'hardware virtuale richiesta, tra cui: CPU, memoria, rete e dispositivi di archiviazione. Inoltre, OVF è un formato portatile basato su standard che consente all'utente di importare una macchina virtuale in qualsiasi software di virtualizzazione che supporti OVF.
Per il mio lavoro di tesi, ho preferito usare un file di importazione ancora più semplice, ovvero l'Open Format Appliance (OVA). Questo formato è un singolo file che archivia tutti i file che costituiscono un OVF. L'OVA non è altro che un file ".tar", una cartella compressa, di tutti i file che compongono un pacchetto OVF in un singolo file ".ova". Il vantaggio principale nell'utilizzo di questo formato è che non si devono specificare i riferimenti alle immagini dei dischi virtuali e i file dei certificati, perché sono compressi nel

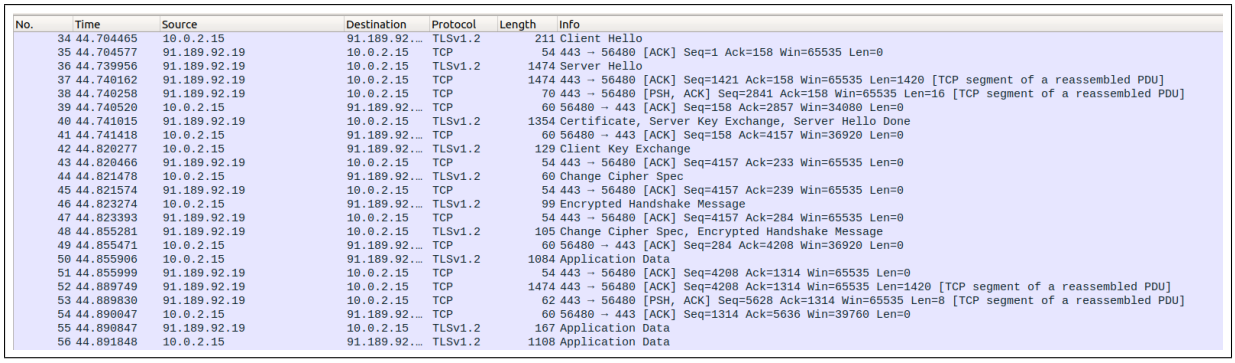
file del formato OVA.

Deft Per il mio progetto di tesi ho utilizzato un disco virtuale, contenente una sistema operativo della famiglia Linux. Esistono diverse distribuzioni pensate per essere utilizzate con scopi affini all'informatica forense. Ed alcune di esse, proprio per soddisfare al meglio le esigenze incluse in questa disciplina, sono praticamente uniche, al punto da essere considerate veri e proprio riferimenti dalla IISFA (International Information Systems Forensics Association).
Per il mio lavoro ho utilizzato Deft, una distribuzione implementata dalla Tesla Consulting. Questo sistema operativo include tutta una serie di strumenti fondamentali per l'analisi forense, quali Digital Forensics Framework, Autopsy, Guymager, Bulk extractor e molti altri. Di questa distribuzione esistono varie tipologie e versioni: la più importante è sicuramente Deft Z una versione live utilizzata per effettuare un'analisi forense post-mortem, ovvero l'attività con cui si cerca di recuperare ed analizzare i dati su una macchina spenta. Nel mio caso invece ho preferito usare Deft X, una distribuzione che viene rilasciata attraverso un disco virtuale, dove quest'ultima è installata.

5.2.2 Traffico di Rete

Tradizionalmente, in questi anni, una parte della Digital Forensics si è concentrata sul recupero di file e sull'analisi del filesystem eseguita su sistemi o dispositivi di archiviazione. Tuttavia il disco rigido è solo una piccola parte dei dispositivi che vengono attraversati da dati digitali.
Un altro sottoinsieme del Digital Forensics, è il Network Forensics. Quest'ultima, consiste nell'attività investigativa effettuata sull'intero contenuto di e-mail, navigazione Web e sul trasferimento dei file; queste informazioni possono essere recuperate dai dispositivi di rete e ricostruite per rilevare la transazione originale del traffico di rete.
In breve, le fasi che si effettuano durante l'analisi forense del traffico sono: lo sniffing, la registrazione e l'analisi del traffico di rete. Per effettuare un'analisi valida dal punto di vista legale, il traffico deve essere intenzionalmente registrato, su un preciso obiettivo.
Gli strumenti forensi utilizzati per registrare il traffico di rete, in realtà, sono quelli che si utilizzano per monitorare normalmente la rete. Il più famoso ed utilizzato è sicuramente *Wireshark*: questo strumento permette di catturare tutti i pacchetti che passano da uno

specifico dispositivo di rete; il suo funzionamento è semplice: gli si indica da quali dispositivi si vuole catturare il traffico dei pacchetti, memorizzabile in un secondo momento in un file PCAP.
Per il mio lavoro di tesi, ho preferito non utilizzare quest'ultimo strumento. Perché l'ambiente virtuale di VirtualBox non rende la scheda di rete reperibile dalla macchina host. Per questo ho preferito utilizzare *nictrace* [13], una funzionalità di VirtualBox, che permette di salvare l'intero traffico di rete della macchina virtuale, dall'avvio allo spegnimento, in un file PCAP. Nella Figura 5.3, viene mostrato il file del traffico di rete della macchina virtuale, prodotto in output dal mio progetto di tesi.



No.	Time	Source	Destination	Protocol	Length	Info
34	44.704465	10.0.2.15	91.189.92.…	TLSv1.2	211	Client Hello
35	44.704577	91.189.92.19	10.0.2.15	TCP	54	443 → 56480 [ACK] Seq=1 Ack=158 Win=65535 Len=0
36	44.739956	91.189.92.19	10.0.2.15	TLSv1.2	1474	Server Hello
37	44.740162	91.189.92.19	10.0.2.15	TCP	1474	443 → 56480 [ACK] Seq=1421 Ack=158 Win=65535 Len=1420 [TCP segment of a reassembled PDU]
38	44.740258	91.189.92.19	10.0.2.15	TCP	70	443 → 56480 [PSH, ACK] Seq=2841 Ack=158 Win=65535 Len=16 [TCP segment of a reassembled PDU]
39	44.740520	10.0.2.15	91.189.92.…	TCP	60	56480 → 443 [ACK] Seq=158 Ack=2857 Win=34080 Len=0
40	44.741015	91.189.92.19	10.0.2.15	TLSv1.2	1354	Certificate, Server Key Exchange, Server Hello Done
41	44.741418	10.0.2.15	91.189.92.…	TCP	60	56480 → 443 [ACK] Seq=158 Ack=4157 Win=36920 Len=0
42	44.820277	10.0.2.15	91.189.92.…	TLSv1.2	129	Client Key Exchange
43	44.820466	91.189.92.19	10.0.2.15	TCP	54	443 → 56480 [ACK] Seq=4157 Ack=233 Win=65535 Len=0
44	44.821478	10.0.2.15	91.189.92.…	TLSv1.2	60	Change Cipher Spec
45	44.821574	91.189.92.19	10.0.2.15	TCP	54	443 → 56480 [ACK] Seq=4157 Ack=239 Win=65535 Len=0
46	44.823274	10.0.2.15	91.189.92.…	TLSv1.2	99	Encrypted Handshake Message
47	44.823393	91.189.92.19	10.0.2.15	TCP	54	443 → 56480 [ACK] Seq=4157 Ack=284 Win=65535 Len=0
48	44.855281	91.189.92.19	10.0.2.15	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
49	44.855471	10.0.2.15	91.189.92.…	TCP	60	56480 → 443 [ACK] Seq=284 Ack=4208 Win=36920 Len=0
50	44.855906	10.0.2.15	91.189.92.…	TLSv1.2	1084	Application Data
51	44.855999	91.189.92.19	10.0.2.15	TCP	54	443 → 56480 [ACK] Seq=4208 Ack=1314 Win=65535 Len=0
52	44.889749	91.189.92.19	10.0.2.15	TCP	1474	443 → 56480 [ACK] Seq=4208 Ack=1314 Win=65535 Len=1420 [TCP segment of a reassembled PDU]
53	44.889830	91.189.92.19	10.0.2.15	TCP	62	443 → 56480 [PSH, ACK] Seq=5628 Ack=1314 Win=65535 Len=8 [TCP segment of a reassembled PDU]
54	44.890047	10.0.2.15	91.189.92.…	TCP	60	56480 → 443 [ACK] Seq=1314 Ack=5636 Win=39760 Len=0
55	44.890847	91.189.92.19	10.0.2.15	TLSv1.2	167	Application Data
56	44.891848	10.0.2.15	91.189.92.…	TLSv1.2	1108	Application Data

Figura 5.3: File del traffico di rete della macchina virtuale

5.2.3 Copia forense

La regola d'oro dell'attività forense è : *"Non toccare, modificare o alterare nulla finché non è stato documentato, identificato, misurato e fotografato"*.
Questa regola si applica anche alla Digital Forensics, quando si vuole investigare su un sistema bisogna documentare tutto ciò che si può su di esso. Nell'informatica forense abbiamo alcuni strumenti che ci permettono di creare una copia identica delle prove. Per avere un valore legale, tutto ciò non basta, bisogna effettuare una serie di rigorose procedure per garantire che venga prodotta una copia fedelmente corretta delle prove.
Nel nostro caso abbiamo bisogno di creare una copia fedele di un disco, per riuscire in questo ci sono varie modalità e formati. Non esiste un formato buono o cattivo, dipende principalmente dalle preferenze personali, dagli obiettivi e dal software che viene utilizzato. Le opzioni più comuni che ho analizzato, per il mio progetto di tesi, sono i formati: RAW e E01.

Formato RAW (DD) L'immagine nel formato RAW, è una copia bit-a-bit dei dati RAW del disco memorizzata in uno o più file. Non ci sono metadati memorizzati nei file immagine.
Il vantaggio principale del formato di immagine RAW è il fatto che i file contengono solo i dati sorgente non modificati, ciò significa che questo formato è supportato da molti strumenti, anche non forensi.
Lo svantaggio principale di questo formato, invece, è la mancanza di metadati. Inoltre manca la comprensione del file, rendendo le immagini grandi quanto l'unità sorgente. Le immagini RAW sono talvolta denominate anche immagini DD dal momento che il formato dell'immagine RAW ha le sue origini nello strumento DD.

Formato E01 In ambito forense, *EnCase* rappresenta l'insieme di strumenti più utilizzati, uno dei quali è *FTK Imager*. Quest'ultimo permette la creazione dell'immagine del dischi nel formato E01, il suo formato principale.
Il formato E01 contiene una copia bit-stream memorizzata in uno o più file arricchiti da metadati, questi metadati includono le informazioni sui casi: nome dell'esaminatore, note, checksum e un hash MD5.
Il vantaggio principale di questo formato di file è la compressione, la protezione della password e il checksum dei file.
Lo svantaggio di questo formato, invece, è il fatto che si tratti di un formato chiuso non documentato. La maggior parte degli strumenti forensi supportano questo formato di file, non è supportato invece dagli altri strumenti.

Documentazione Uno dei passi più importanti per rendere forense la copia è la documentazione. Come affermato prima, è la regola d'oro dell'attività investigativa: non si deve toccare, modificare o cambiare nulla finché non è stato documentato. La stesura della documentazione dipende dal tipo di lavoro o indagine, ma ci sarebbero alcune basi che dovrebbero essere rispettate.
Nel caso della copia delle immagini dei dischi, dobbiamo specificare:

- Software o hardware utilizzati
- Nome del sorgente
- Nome della destinazione

- Data e ora di inizio
- Data e ora di fine
- Valori Hash

Questi elementi sono il minimo indispensabile di cosa deve essere documentato. Fortunatamente la maggior parte dei software, tra cui *FTK Imager*, crea in automatico un file di registro contenente le informazioni appena elencate.
L'elemento più importante della documentazione è sicuramente il valore hash. Quest'ultimo è considerato un'impronta digitale dei dati, il contenuto delle prove viene elaborato attraverso un algoritmo crittografico e viene prodotto un valore numerico univoco. Se la copia viene anche solo minimamente modificata, il valore hash della copia forense non corrisponderà a quello della prova.
La recente legislazione, infatti, impone la creazione di una catena di custodia che permetta di preservare i reperti informatici da eventuali modifiche successive all'acquisizione: tramite i codici hash è possibile in ogni momento verificare che quanto repertato sia rimasto immutato nel tempo. Se i codici hash corrispondono, entrambe le parti in un procedimento giudiziario hanno la certezza di poter lavorare sulla stessa versione dei reperti, garantendo quindi una uniformità di analisi e, in genere, di risultati [14].
Per l'implementazione della copia forense nel mio progetto di tesi ho preferito utilizzare l'immagine nel formato E01 rispetto al formato RAW per i seguenti motivi:

- Arricchimento immagine con i metadati
- Creazione automatica del file di registro contenente i metadati
- Comprensione delle copie create

Il primo punto ha un'importanza pratica dal punto di vista legale, come già discusso, i metadati e il valore hash dell'immagine sono degli elementi importanti per validare le prove. La comprensione invece è utile per non creare output troppo grandi. Tradizionalmente *FTK Imager* viene utilizzato per i dischi fisici, ma supporta anche i dischi virtuali.

Capitolo 6

Sviluppo

In questo Capitolo descriverò più precisamente le due parti del mio progetto di tesi, discutendo di alcuni ostacoli che ho incontrato e delle scelte implementative.

6.1 Front end

Per il mio lavoro di tesi ho implementato la parte front end con uno script Python. Lo schema della chiamata è descritto nella Figura 6.1.

```
python3 cloud_downloader.py -w [WORKING_DIRECTORY] -o [OUTPUT_DIRECTORY]
```

Figura 6.1: Schema chiamata applicativo Python

Lo script, come appena mostrato, richiede due input obbligatori per il funzionamento del programma:

- *WORKING DIRECTORY*: È la cartella dove VirtualBox lavora, ovvero dove importa, salve e crea le macchine virtuali. La cartella di default è in *Documenti/Virtual Box*, ma spesso viene cambiata ad esempio per carenza di memoria.
- *OUTPUT DIRECTORY*: È la cartella dove vengono salvati i dati di output, ovvero la copia del disco virtuale nel formato E01, il file di registro prodotto per il formato E01 ed il file del traffico di rete nel formato PCAP.

Le best practices della Digital Forensics impongono di utilizzare un dispositivo di memoria diverso per le due cartelle. La prima può essere una cartella contenuta nel disco rigido di un computer o in un disco rigido esterno, che può contenere anche altri dati. Il dispositivo di memoria utilizzato per l'output, invece deve essere diverso dal primo. Inoltre per una maggiore validità dal punto di vista forense quest'ultimo deve essere formattato completamente.

Lo script, una volta partito, controlla che ci sia abbastanza spazio libero per lavorare: il disco virtuale è allocato dinamicamente fino ad al massimo di *1 TeraByte* di memoria. Se lo spazio libero è di almeno *1 TeraByte* di memoria allora lo script inizia l'importazione della macchina virtuale; per gestire al meglio quest'ultima quando viene importata indico un nome composto in questo modo: *Deft-[stringa UUID]*.
Il vantaggio di utilizzare un nome con questo formato è di non avere macchine virtuali diverse con lo stesso nome. In VirtualBox le macchine virtuali sono distinte con un UUID unico attribuito alla creazione, quindi permette l'esistenza di più macchine virtuali con lo stesso nome; l'UUID della macchina purtroppo viene attribuito solamente alla creazione della macchina virtuale, quindi durante l'importazione dell'immagine virtuale non sono riuscito a reperire tale identificativo. Per aggirare questo problema prima di creare la macchina virtuale con il pacchetto di importazione, genero un nome con all'interno un codice UUID creato sul momento e durante l'importazione del pacchetto indico quale deve essere il nome della macchia virtuale. Nella Figura 6.2 vengono mostrati i log che avvengono durante l'importazione del pacchetto virtuale, nelle sezione *1* di *Virtual system 0*, viene mostrato nel campo *–vmname* il nome dato in input della macchina virtuale.

```
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Interpreting /media/ossama/Data/TesiWorkStation/DeftX.ova...
OK.
Disks:
  vmdisk2   1099511627776   -1  http://www.vmware.com/interfaces/specifications/vmdk.html#streamOptimized   DeftX-disk001.vmdk  -1  -1

Virtual system 0:
 0: Suggested OS type: "Ubuntu_64"
    (change with "--vsys 0 --ostype <type>"; use "list ostypes" to list all possible values)
 1: VM name specified with --vmname: "Deft-672813b3-12ad-4152-b1d1-2cb422b8aa93"
 2: Suggested VM group "/"
    (change with "--vsys 0 --group <group>")
 3: Suggested VM settings file name "/media/ossama/Data/VirtualBox_VMs/Deft/Deft.vbox"
    (change with "--vsys 0 --settingsfile <filename>")
 4: Suggested VM base folder "/media/ossama/Data/VirtualBox_VMs"
    (change with "--vsys 0 --basefolder <path>")
 5: Number of CPUs: 1
    (change with "--vsys 0 --cpus <n>")
 6: Guest memory: 1024 MB
    (change with "--vsys 0 --memory <MB>")
 7: Sound card (appliance expects "", can change on import)
    (disable with "--vsys 0 --unit 7 --ignore")
 8: USB controller
    (disable with "--vsys 0 --unit 8 --ignore")
 9: Network adapter: orig NAT, config 3, extra slot=0;type=NAT
10: CD-ROM
    (disable with "--vsys 0 --unit 10 --ignore")
11: IDE controller, type PIIX4
    (disable with "--vsys 0 --unit 11 --ignore")
12: IDE controller, type PIIX4
    (disable with "--vsys 0 --unit 12 --ignore")
13: SATA controller, type AHCI
    (disable with "--vsys 0 --unit 13 --ignore")
14: Hard disk image: source image=DeftX-disk001.vmdk, target path=DeftX-disk001.vmdk, controller=11;channel=0
    (change target path with "--vsys 0 --unit 14 --disk path";
    disable with "--vsys 0 --unit 14 --ignore")
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Successfully imported the appliance.
```

Figura 6.2: Log fase importazione

Ad importazione avventa, consento a VirtualBox di iniziare a tracciare il traffico di rete della macchina virtuale appena importata attraverso lo strumento *nictrace*: indico l'interfaccia di rete della quale voglio catturare i pacchetti e il path del file PCAP dove salvare il traffico. Quest'ultima funzionalità mi è stato possibile applicarla perché ho utilizzato un pacchetto di importazione per la macchina virtuale, se avessi aggiunto manualmente il disco non avrei avuto tutte le configurazioni della macchina virtuale già definite.
Dopo aver avviato la macchina virtuale, effettuo periodicamente un controllo su quest'ultima, verificando se è ancora in esecuzione. In caso negativo, ovvero quando la macchina viene spenta dallo script della parte back end, effettuo la copia del disco virtuale nel formato E01 utilizzando *FTK Imager*. Il disco virtuale è contenuto in una sotto cartella della cartella di lavoro di VirtualBox, per questo è indispensabile specificare la *Working Station*. Nella Figura 6.3, possiamo vedere il file di registro prodotto, nel mio progetto di tesi, durante la creazione della copia forense.

```
Case Information:
Acquired using: ADI3
Case Number:
Evidence Number:
Unique Description:
Examiner:
Notes:

------------------------------------------------------------

Information for /media/ossama/Element/Deft-98c70d5a-6be5-4922-8ed7-537b7561bab1:

Physical Evidentiary Item (Source) Information:
[Device Info]
 Source Type: Physical
[Drive Geometry]
 Cylinders: 2130440
 Heads: 16
 Sectors per Track: 63
 Bytes per Sector: 512
 Sector Count: 2147483648
[Physical Drive Information]
 Drive Interface Type: ide
[Image]
 Image Type: VMWare Virtual Disk
 Source data size: 1048576 MB
 Sector count:    -2147483648
[Computed Hashes]
 MD5 checksum:    e216ea95fc6afad4d61e6a6a406ba6be
 SHA1 checksum:   b812293fa235878181ffb8d78be839afb4b996a2

Image Information:
 Acquisition started:   Wed Feb 13 01:20:22 2019
 Acquisition finished:  Wed Feb 13 02:55:30 2019
 Segment list:
  /media/ossama/Element/Deft-98c70d5a-6be5-4922-8ed7-537b7561bab1.E01
```

Figura 6.3: File di registro prodotto da *FTK Imager*

6.2 Back end

La parte back end è costituita dalla macchina virtuale importata e dallo script scritto con il linguaggio Python, che acquisisce tutti i file da Google Drive. La macchina importata costituisce un ambiente virtuale di lavoro "pulito", perché ogni volta che si vuole effettuare l'acquisizione dei dati viene importata e creata sempre una nuova macchina. Non possiamo avere un ambiente virtuale "sporco", ovvero con dati non pertinenti all'analisi forense su un obiettivo, e ritenerla una prova valida.

Systemd La macchina all'accensione, dopo il caricamento degli elementi grafici di Ubuntu e dopo essersi connessa alla rete, avvia il programma per l'acquisizione dei dati

dal cloud storage. Per implementare questa funzionalità ho utilizzato systemd, un gestore di sistemi e servizi. Quest'ultimo fornisce un manager di sistema che è eseguito con PID 1, quindi è il primo processo avviato dal kernel Linux. Le attività di systemd sono organizzate come unità, le più comuni sono:

- *Servizi (.service)*
- *Mount point (.mount)*
- *Dispositivi (.device)*
- *Socket (.socket)*
- *Timer (.timer)*

Per il mio progetto di tesi, ho utilizzato un service ovvero un'applicazione (o una serie di applicazioni) che viene eseguita in background in attesa di essere utilizzata o che svolge attività essenziali. Per sapere il momento in cui un programma deve partire, nel service, si possono definire alcuni target che devono essere completati prima dell'avvio del servizio. I target sono gruppi di unità, per il mio lavoro ne ho utilizzati due:

- *graphical.target*
- *network-online.target*

Il target bersaglia le unità di chiamata per mettere insieme il sistema richiesto. Ad esempio, nel mio caso, *graphical.target* chiama tutte le unità necessarie per avviare una workstation con interfaccia utente grafica. *Network-online.target*, invece, è un target che attende attivamente fino a quando il network è *"up"*, dove la definizione di *"up"* è definita dal software di gestione della rete. Solitamente indica un indirizzo IP configurabile e instradabile. Il suo scopo principale è di ritardare attivamente l'attivazione dei servizi fino a quando la rete non viene configurata. Si tratta di un obiettivo attivo, vale a dire che può essere richiamato dai servizi che richiedono la rete, ma non viene attivato dal servizio di gestione della rete stesso. I bersagli possono basarsi su altri target o dipendere da altri obiettivi.

6.2.1 Autorizzazione

Lo script per l'acquisizione dei dati usa il protocollo OAuth2 per autenticare l'utente e per richiedere i permessi per accedere all'account.

OAuth OAuth è un protocollo aperto nato nel 2006, permette l'autorizzazione di API di sicurezza con un metodo standard e semplice sia per applicazioni portatili, sia per applicazioni per desktop e web. Per gli sviluppatori di applicazioni è un metodo per pubblicare e interagire con dati protetti. OAuth garantisce ai service provider l'accesso da parte di terzi ai dati degli utenti proteggendo contemporaneamente le loro credenziali. Per esempio permette all'utente di dare ad un sito, chiamato consumer, l'accesso alle sue informazioni presenti su un altro sito, detto service provider, senza condividere la sua identità [15]. Le motivazioni che hanno spinto alla nascita di questo protocollo, è quella di autorizzare terze parti a gestire documenti privati senza condividere la password. La condivisione della password non risulta una buona pratica a livello di sicurezza, rende accessibile l'intero account e il pannello di amministrazione. L'unico modo per revocare questo accesso incondizionato e indesiderato è cambiare la password dell'intero account. OAuth è nato quindi con il presupposto di garantire l'accesso delegato ad uno uno specifico client per delle determinate risorse, soprattutto per un lasso di tempo limitato e con la possibilità di revoca. In OAuth possiamo riconoscere tre attori principali:

- *Server*: Spazio in cui sono contenute le risorse protette.
- *Client*: Parte che richiede l'autorizzazione per accedere alle risorse protette.
- *Utente*: Persona fisica che vuole garantire l'accesso alle risorse sul server.

In breve, l'utente è la persona fisica che vuole garantire l'accesso alle risorse presenti sul server. Il client è la parte che richiede l'accesso alle risorse protette. Per far questo, il client interagisce con il server ottenendo delle proprie credenziali temporanee, un token di accesso. Una volta ottenuto può accedere e interagire con le risorse stabilite per un periodo limitato.
La prima versione di questo protocollo, presentava alcune limitazioni a livello di sicurezza. OAuth1 non garantisce confidenzialità né sulle richieste né sui contenuti. Sebbene questo protocollo garantisca che il client non acceda ad informazioni indesiderate, non garantisce che l'uso delle risorse autorizzate rimanga nell'ambito specificato. Per questo motivo

OAuth suggerisce al server di proteggere le risorse tramite il protocollo TLS [15]. Un ulteriore problema noto è quello del phishing: il client potrebbe indirizzare l'utente a una pagina di accesso falsa del server per richiedere l'autenticazione e ottenere le credenziali dell'utente.
L'evoluzione di OAuth1 è OAuth2. Principalmente il funzionamento è il medesimo, ma è stato migliorato nella parte riguardante il servizio. Gli attori sono i medesimi con l'aggiunta di un server mediatore. Quest'ultimo ha il principale compito di gestire i token di accesso tra client e server. Il server che funge da mediatore può essere lo stesso che ospita le risorse alle quali accedere e può gestire token di accesso provenienti da più di un solo server. Un altro vantaggio rispetto alla precedente versione è dato dalla possibilità di prolungare il tempo di utilizzo del token di accesso.

Appena lo script della parte back end viene eseguito, avvia in automatico la procedura di autorizzazione che richiede all'utente di autenticarsi con le credenziali appropriate (nome utente e password). Lo strumento fornisce all'utente un URL che viene aperto in un browser, in cui l'interfaccia di autenticazione standard per il servizio richiederà il nome utente e la password rilevanti. Nelle figure 6.4 e 6.5 , viene mostrato il processo di autenticazione.

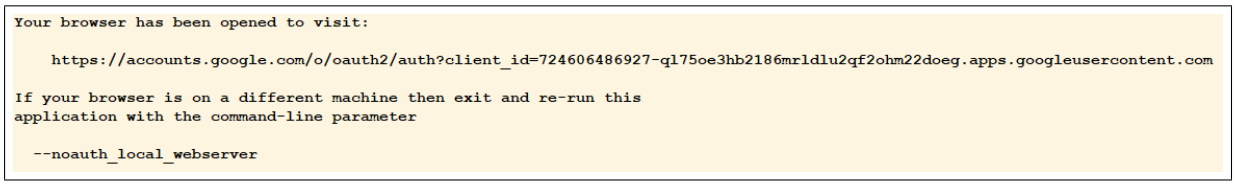


Figura 6.4: URL aperto automaticamente per l'autenticazione

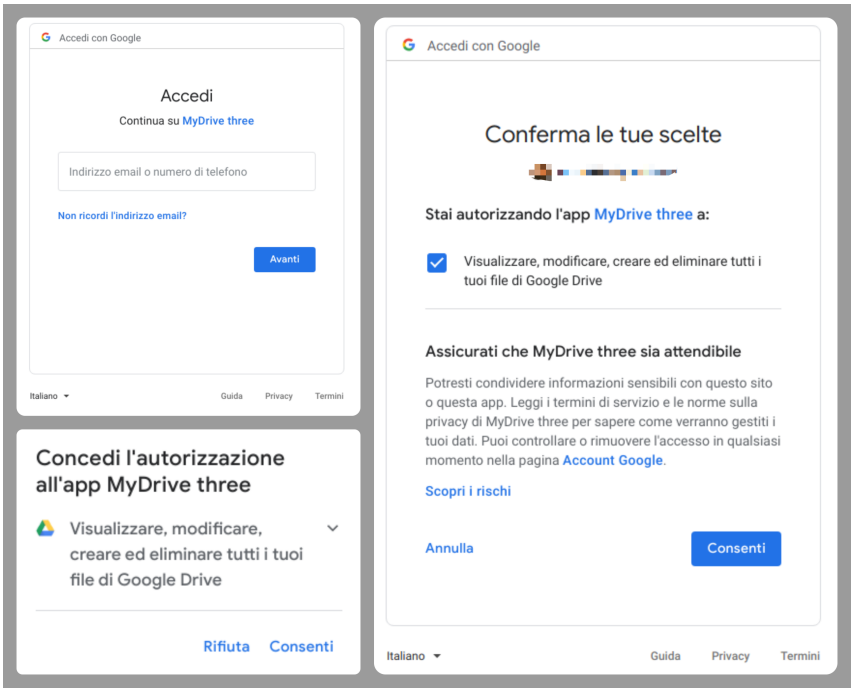


Figura 6.5: Pagina Web di Google per l'autenticazione e la gestione dei permessi

Dopo aver fornito le credenziali corrette e autorizzato l'app. Se l'autenticazione ha esito positivo, il servizio fornisce un token di accesso che viene memorizzato in modo persistente nel file *google.dat* salvato nella cartella */config* .

6.2.2 Acquisizione dati

Come discusso nel Capitolo 4 per l'acquisizione dei dati, nel mio progetto di tesi, ho utilizzato un approccio basato sull'API. La versione più recente dell'API offerta da Google per il suo servizio di cloud è la versione 3. Questa versione, oltre che essere più efficiente della versione precedente, offre molti metodi che semplificano il lavoro per la gestione dei file contenuti su Google Drive.
Quando si sviluppa un progetto, è buona pratica utilizzare le distribuzioni più recenti di tutte le librerie. Questa metodologia non ho potuto applicarla completamente al mio progetto di tesi perché sebbene l'ultima versione dell'API risulta essere più performante delle precedenti, ho riscontrato dei limiti nell'implementazione del mio lavoro. Queste restrizioni non mi avrebbero permesso di ottenere un'acquisizione completa dei file e metadati salvati sul cloud storage, quindi non sarei stato in grado di ottenere una prova

valida dal punto di vista forense.
Il primo di questi limiti è che non si possono scaricare le varie versioni dei file di GSuite, ovvero i file di Google: Documenti, Fogli, Presentazioni, Moduli e Disegni. Un secondo limite che ho riscontrato è che durante la conversione dei file di GSuite il servizio pone un limite alla grandezza dei file, quindi i file di grosse dimensioni non riuscivano ad essere esportati in altri formati: l'esportazione risulta indispensabile perché i file di GSuite non hanno un'estensione supportata dal filesystem. Per superare questi limiti, nei casi appena citati ho utilizzato la seconda versione. Quindi possiamo affermare che per il mio progetto ho utilizzato un approccio ibrido tra le versioni 2 e 3 dell'API di Google Drive, perdendo in efficienza ma acquistando in completezza.
Il download dei file utilizzando la seconda versione dell'API, avviene attraverso una richiesta http ad un URL specificato in input, nel caso dei file era un URL per il download mentre nel caso dei file di GSuite era un URL per esportare il file in un formato desiderato. Durante la richieste per il download di più revisioni dello stesso file, ho notato che il servizio di Google Drive mi ritornava un errore "429:Rate Limit Exceeded", questa irregolarità era dovuta al fatto che lo script effettuava troppe richieste in un lasso di tempo troppo breve. Per risolvere questo problema ho utilizzato la tecnica di "backoff esponenziale".

Backoff esponenziale Il backoff esponenziale è una strategia standard di gestione degli errori per le applicazioni di rete in cui il client ripete periodicamente una richiesta non riuscita per un periodo di tempo crescente. Se un volume elevato di richieste o un intenso traffico di rete fanno sì che il server restituisca errori, il backoff esponenziale potrebbe essere una buona strategia per gestire tali errori. Al contrario, non è una buona strategia per risolvere altre tipologie di errore. Il tempo che deve passare tra una richiesta e l'altra, nel caso di errore, è descritto da questa formula :

$$2^n + \frac{m}{1000} \quad \forall n \leq maxbackoff, \quad 0 \leq m \leq 1000$$

Dove n è il numero di tentativi che vengono effettuati, durante la richiesta del download di uno specifico file e/o revisione, il numero massimo di tentativi è limitato dalla costante maxbackoff solitamente settata a 8. Mentre m invece è un numero casuale di millisecondi. Utilizzato correttamente, il backoff esponenziale aumenta l'efficienza dell'utilizzo della larghezza di banda, riduce il numero di download richiesti per ottenere una risposta

positiva e massimizza il throughput delle richieste in ambienti concorrenti.

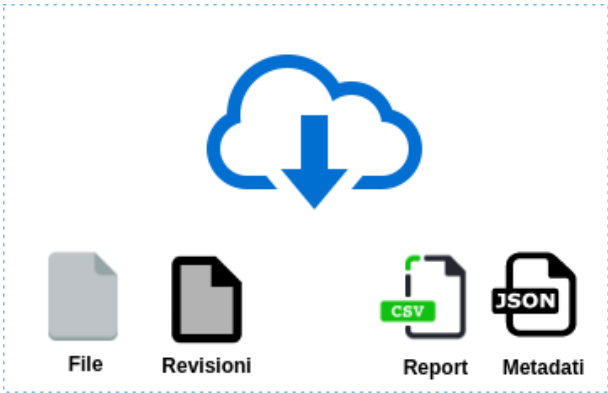


Figura 6.6: Dati acquisiti nella macchina virtuale dal cloud storage

Nella Figura 6.6, vengono mostrati le tipologie dei dati acquisiti con il mio progetto di tesi dal cloud storage: i file, le revisioni dei file e i metadati dei file. I metadati dei file sono necessari ad aumentare la validità delle prove, perché ad esempio i file per il filesystem sono stati creati durante la fase di acquisizione e non si riferiscono alla data di creazione sul cloud storage per questo motivo oltre che ricreare l'albero dei file ricreo l'albero dei metadati. Per aiutare l'esaminatore durante l'analisi forense inoltre ricreo un report per i file scaricati attraverso due file *".csv"*, uno per i file ed uno per le revisioni, in cui vengono raccolte tutte le informazioni dei metadati e per ogni file viene indicato il path remoto dove sono contenuti i metadati.

File Tutto il contenuto di Google Drive è rappresentato come file. Quest'ultimi sono composti da metadati e contenuti, possono essere: inseriti, aggiornati, copiati ed eliminati. Ogni file è identificato da un ID univoco. Gli identificativi dei file sono stabili per tutta la vita del file, vengono attribuiti alla creazione per poi venir eliminati alla rimozione di un file, questo vuol dire che tutte le modifiche al nome o al contenuto del file non hanno effetti sull'ID. Nella Figura 6.7, vengono mostrati i metadati più significativi di un file.

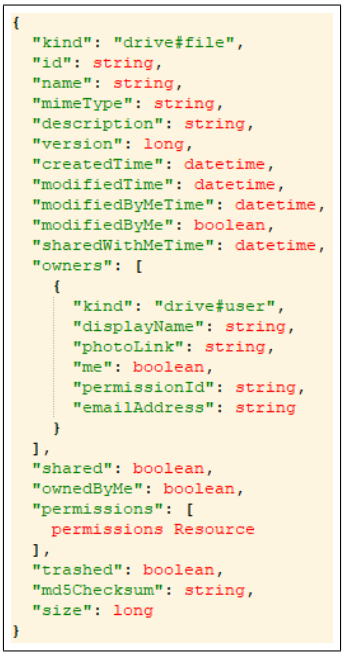


Figura 6.7: Metadati file di Google Drive [3]

Tutti i file sono di proprietà degli utenti, non del provider. Gli utenti controllano chi altro può accedere ai propri file tramite gli elenchi di controllo degli accessi, rappresentati come un elenco di *Permissions* a ciascun file. I file contengono anche una cronologia delle revisioni dei loro contenuti associati. Le revisioni non vengono archiviate indefinitamente e possono essere eliminate se necessario. Le revisioni importanti possono essere bloccate per impedire l'eliminazione.

Permessi L'accesso a file e cartelle è determinato da un elenco di controllo di accesso (ACL). Un ACL è un elenco di autorizzazioni che determina se gli utenti possono o meno eseguire azioni su un file come leggere o scrivere [16]. Gli elenchi delle autorizzazioni sono disponibili per ogni file e cartella in Google Drive. Ogni autorizzazione specifica un tipo, un ruolo e un indirizzo email, consentendo un livello di accesso a un file o una cartella. Questi valori lavorano insieme per limitare l'accesso in modo appropriato:

- Il tipo limita l'accesso a un insieme di utenti.
- L'indirizzo email e i campi del dominio specificano quali utenti possono avere accesso.

- Il ruolo dà a questi utenti la possibilità di fare qualcosa nel file, come leggerlo.

Se combinate, queste proprietà definiscono un'autorizzazione completa.

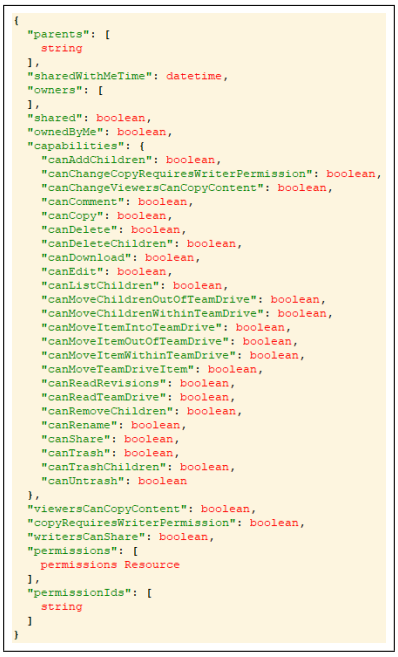
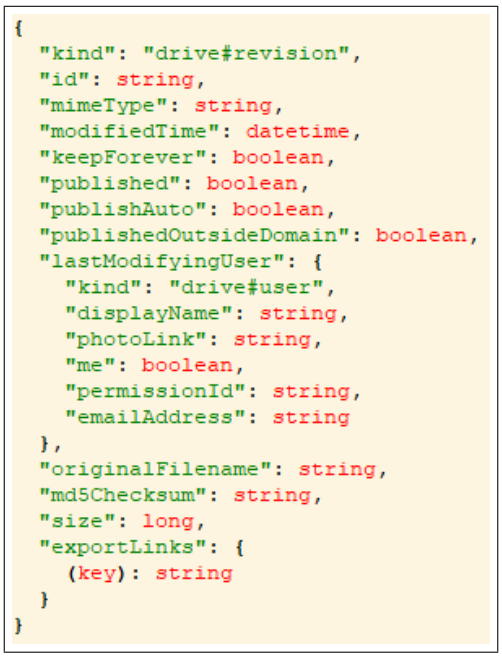


Figura 6.8: Metadati permessi di Google Drive [3]

Nella Figura 6.8, vengono mostrati i metadati dei permessi più importanti. Gli ACL impostati sulle cartelle si propagano verso il basso a tutti gli elementi contenuti, quindi sui file e sulle sottocartelle. La propagazione avviene ogni volta che le autorizzazioni o la gerarchia vengono cambiate e vengono eseguite in modo ricorsivo attraverso tutte le cartelle nidificate. Le autorizzazioni ereditate non possono essere rimosse da un elemento, ma possono essere regolate sul genitore diretto o indiretto da cui sono state ereditate. Le autorizzazioni ereditate possono essere rimosse dagli elementi in "I miei file" o "Condivisi con me".

Revisioni I file contengono anche una cronologia delle revisioni dei loro contenuti associati. Le revisioni non vengono archiviate indefinitamente e possono essere eliminate se necessario, mentre quelle più importanti possono essere bloccate per impedirne la cancellazione. Prima di scaricare un file, controllo se esistono delle revisioni su quello

specifico file, in caso positivo scarico tutte le revisioni. Le informazioni sulle revisioni come quelli sull'ultima modifica, sono descritti nella Figura 6.9.



```
{
  "kind": "drive#revision",
  "id": string,
  "mimeType": string,
  "modifiedTime": datetime,
  "keepForever": boolean,
  "published": boolean,
  "publishAuto": boolean,
  "publishedOutsideDomain": boolean,
  "lastModifyingUser": {
    "kind": "drive#user",
    "displayName": string,
    "photoLink": string,
    "me": boolean,
    "permissionId": string,
    "emailAddress": string
  },
  "originalFilename": string,
  "md5Checksum": string,
  "size": long,
  "exportLinks": {
    (key): string
  }
}
```

Figura 6.9: Metadati sulle revisioni di Google Drive [3]

Le revisioni sono un'importante fonte di informazioni, nell'ambito di un'indagine forense. Negli approcci tradizionali della Digital Forensics, questa tipologia di informazioni non vengono cercate o è difficile acquisirle.

6.2.3 Esplorazione e acquisizione dei file

L'esplorazione dei file da acquisire parte da 3 sezioni:

- *Il mio drive*: Lo spazio del cloud personale contiene file di proprietà e può contenere file condivisi con l'utente.
- *Condivisi con me*: Include tutti i file condivisi con l'utente
- *Cestino*: Contiene i file spostati nel cestino dall'utente.

Queste sezioni le possiamo vedere come delle cartelle, al loro interno infatti troviamo dei file e delle sotto cartelle. Per ottenere la lista degli elementi utilizzo la seguente funzione

della libreria di Google:

$$drive\_service\_v3.files().list(parametri).execute()$$

I parametri inseriti sono l'id della cartella padre, che inizialmente sono le sezioni sopracitate, ed eventualmente il *pagetoken*. La funzione che restituisce la lista degli elementi può essere costituita da al più 20 elementi, ognuno dei quali è composto da tutti i metadati di un file. Quando effettuo la richiesta, controllo nel JSON restituito la presenza del parametro *nextPageToken*, in caso positivo ri-effettuo una chiamata alla stessa funzione con lo stesso id della cartella del padre ed indico come *pagetoken* il *nextPageToken*. Quest'ultimo campo si ottiene quando la richiesta che effettuiamo ha molti valori da restituire.
Dopo aver effettuato le richieste degli elementi, "scorrendo" i *pagetoken*, inizio a scaricare file per poi successivamente ri-effettuare una ricerca ricorsiva degli altri elementi sulle cartelle.

File Per l'acquisizione dei file utilizzo, come anticipato nella sezione precedente, la terza versione dell'API offerta da Google Drive. I file, come già discusso, si dividono in due tipologie:

- File ordinari.
- File di GSuite.

Per ottenere i primi ho utilizzato la seguente funzione della libreria di Google:

$$drive\_service\_v3.files().get\_media(fileId = id)$$

l'unico parametro dato in input è l'id del file. Il download del file richiede che l'utente abbia almeno l'accesso in lettura. Inoltre, l'app deve essere autorizzata con un ambito che consenta la lettura del contenuto del file. Ad esempio, un'app che utilizza lo scope *drive.readonly.metadata* non sarebbe autorizzata a scaricare il contenuto del file, ma solo ad ottenere i metadati di quest'ultimi. Per il mio progetto di tesi richiedo all'utente di concedermi i permessi completi per scaricare e gestire i file.
Per i file di GSuite, la metodologia e la funzione utilizzata per ottenerli sono diverse, quest'ultimi non si possono scaricare perché non hanno un formato riconoscibile dal

filesystem, per questo devono essere esportati in formati noti. La funziona che ho utilizzato per esportare questa tipologia di file è la seguente:

$$drive\_service\_v3.files().export\_media(fileId = id, mimeType = mime\_type)$$

La funzione prende in input l'id del file ed il formato in cui si vuole esportarlo. Nei metadati dei file è possibile reperire il formato della tipologia di file di GSuite ed in base a quello si può esportare il file nel formato più consono. Nella tabella sottostante vengono descritti i formati che ho scelto per l'esportazione.

Applicazione Google	Formato Google	Formato Esportazione
Documenti	google-apps.document	opendocument.text
Presentazione	google-apps.presentation	pdf
Fogli	google-apps.spreadsheet	oasis.opendocument.spreadsheet
Disegni	google-apps.drawing	jpeg

Tabella 6.1: Tabella dei formati utilizzati per esportare i file di Google

Revisioni Prima di scaricare i file effettuo un controllo per verificare la presenza di revisioni per uno specifico file. Per ottenere una lista delle revisioni ho utilizzato la seguente funzione:

$$drive\_service\_v2.revisions().list(fileId = id).execute()$$

Come si può notare, per gestirle ho utilizzato la seconda versione dell'API offerta da Google, per venire in contro alle limitazioni della terza versione. La lista ritorna solo gli id delle revisioni di un file, il passaggio successivo è quello di utilizzare un'altra funzione per ottenere i metadati della singola revisione. Nella Figura 6.9, abbiamo mostrato le informazioni che ritorna la seguente funzione:

$$drive\_service\_v2.revisions().get(fileId = file\_id, revisionId = revision\_id).execute()$$

Tra le informazioni utili, come nel caso dei file, c'è la tipologia del formato, necessaria per esportare la revisione in un formato consono. Per scaricare le revisioni però, non utilizzo le stesse modalità utilizzate nella terza versione, ma utilizzo l'URL che mi viene fornito nei metadati. Quindi il download avviene attraverso una normale chiamata http. La funzione

utilizzata è la seguente:

$$drive\_service\_v2.\_http.request(downloadUrl)$$

se la richiesta ha esito positivo salvo il file e i relativi metadati, se invece restituisce il codice 429, metto la funzione in attesa utilizzando la tecnica di backoff descritta nella sezione 6.2.2. L'URL è ricavato dai metadati della revisione: nel caso dei formati normali l'URL è contenuto nel campo *downloadUrl*, mentre per i formati delle applicazioni di Google l'URL è contenuto in un dizionario di *exportLinks*, ogni formato di esportazione ha un link per ottenere la revisione nel formato desiderato. I formati di esportazione che ho utilizzato sono gli stessi dei file e possiamo vederli nella tabella 6.1.
Visto che il nome del file è lo stesso delle revisioni, per differenziare quest'ultime, compongo il nome utilizzando il timestamp dell'ultima modifica della revisione a cui aggiungo il nome del file.

$$(2015-04-21 \quad 12:02:19.209915)NameFile$$

Metadati L'acquisizione dei metadati avviene prima dello scaricamento dei file, come abbiamo visto nelle sezioni precedenti. Durante la fase di acquisizione arricchisco i metadati di Google aggiungendone altri:

- *remotePath*: Inserisco il path remoto del file, ovvero il path che ha alla radice una delle 3 sezioni descritte ad inizio sezione.
- *downloadTime*: Inserisco il timestamp nel momento dell'effettivo download del file/revisione.

Successivamente alla scaricamento del file/revisione, salvo i metadati nella cartella *metadata* ricreando l'albero dei file: per effettuare questo mi è bastato concatenare la cartella dei metadati con il *remotePath*. Tutti i metadati vengono salvati singolarmente in un JSON, utilizzando lo stesso nome del file o della revisione.

downloadTime	remotePath	kind	id	name	mimeType
2019-02-26 14:41:41.074288	Shared With Me/condiviso_R	drive#file	P5pgVYKzGn-Bn	condiviso_R	application/vnd.google-apps.document
2019-02-26 14:41:51.037667	Shared With Me/condiviso_RW	drive#file	VgFOSx-NL8rV	condiviso_RW	application/vnd.google-apps.document
2019-02-26 14:42:00.723110	Shared With Me/Condivisa/File 1	drive#file	9s3IWnbMJRgr5	File 1	application/vnd.google-apps.document
2019-02-26 14:42:13.854412	Shared With Me/Condivisa/condivisa_dentro/File 2	drive#file	j5feT6jFlX1ZiZ-s	File 2	application/vnd.google-apps.document

Figura 6.10: Report creato attraverso un file *".csv""*

Oltre che salvare i singoli file dei metadati, aggiungo quest'ultimi in uno dei file *".csv""*, in corrispondenza della tipologia di metadati: file o revisioni. Nella Figura 6.10 , viene mostrato un esempio del report creato con un file csv.

Capitolo 7

Conclusioni

Abbiamo visto come la Cloud Storage Forensic non abbia degli standard specifici da usare per un'investigazione forense anche perché l'attuale legislazione non prende in considerazione appieno l'acquisizione dei dati dal cloud, quindi non ci sono ancora delle linee guida ben definite che l'esaminatore forense possa seguire. Nel mio progetto ho implementato una serie di operazioni che tengono conto dei requisiti di legge, anche se in alcuni casi non ho potuto soddisfarli per problemi architetturali dello storage contenente il dato originale: effettuo un'acquisizione logica dei file dal cloud, a differenza dell'acquisizione fisica, che viene effettuata nelle metodologie standard della Digital Forensics. L'insieme delle operazioni vengono eseguite in modo automatico per permettere l'acquisizione in un ambiente isolato, quello virtuale, e fornire un resoconto documentato durante ogni fase: dalla creazione dell'ambiente virtuale fino alla creazione della copia forense.
Per l'acquisizione dei dati ho utilizzato l'API di Google, questo approccio mi ha consentito di ottenere molte informazioni aggiuntive dei file: in particolare una grande quantità di metadati rispetto a quelli che si ottengono dal filesystem e le revisioni dei file, metadati che d'altro canto non sarebbero reperibili utilizzando gli approcci standard.
L'acquisizione dei dati viene effettuata in un disco virtuale dinamico con una dimensione massima di *1 TeraByte*, questo probabilmente è uno dei limiti del mio lavoro di tesi, ovvero poter acquisire dati dal cloud storage fino ad un dato limite di memoria. Questo limite è imposto dal fatto che non sono riuscito a effettuare una stima dei file contenuti su Google Drive: i file ordinari nei metadati contenevano la grandezza del file, i file nativi di GSuite, invece, non hanno un riferimento alla grandezza perché non occupano memoria

nello spazio cloud personale dell'utente: ovvero il peso del file non grava sulla memoria concessa all'utente dal provider. Non sapendo la dimensione che potrebbe raggiungere il disco virtuale, il mio progetto di tesi richiede che ci siano almeno *1 TeraByte* liberi sia sul dispositivo contentente la cartella di *Workstation* sia sul dispositivo contentente la cartella di *Output*.
Un altro limite che ho incontrato è sicuramente la mancanza dell'hash prodotto sul *"dispositivo"* contenente i file originali, questo perché tale dispositivo è il cloud storage, del quale l'utente ha una visione ad alto livello. L'ultimo limite che ho riscontrato è l'utilizzo di un algoritmo di hash molto vecchio, rispetto a quelli comunemente utilizzati: questa limitazione è imposta dall'utilizzo di *FTK Imager Lite* che non viene aggiornato da tempo, a differenza della versione grafica di *FTK Imager* che include la funzionalità di utilizzare l'algoritmo di hash SHA-256. Ho dovuto utilizzare *FTK Imager Lite*, perché mi consente la creazione della copia forense utilizzando le CLI.
La mia tesi è un progetto sperimentale che attualmente non può essere utilizzato in ambito forense, anche perché come abbiamo visto ha alcune limitazioni, ma potrebbe diventare un punto di partenza per futuri standard della Cloud Storage Forensic.

Bibliografia

[1] Darren Quick and Kim-Kwang Raymond Choo. Google drive: Forensic analysis of data remnants. *J. Network and Computer Applications*, 40:179–193, 2014.

[2] Vassil Roussev, Andres Barreto, and Irfan Ahmed. Forensic acquisition of cloud drives. *arXiv preprint arXiv:1603.06542*, 2016.

[3] Google Developers. Resource files rappresentations. `https://developers.google.com/drive/api/v3/reference/files`.

[4] Ossama Gana. Github repository. `https://github.com/ossamagana/Tesi-progetto`.

[5] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[6] Dominik Birk and Christoph Wegener. Technical issues of forensic investigations in cloud computing environments. In *Systematic Approaches to Digital Forensic Engineering (SADFE), 2011 IEEE Sixth International Workshop on*, pages 1–10. IEEE, 2011.

[7] Ben Martini and Kim-Kwang Raymond Choo. Cloud storage forensics: owncloud as a case study. *Digital Investigation*, 10(4):287–299, 2013.

[8] Parlamento. Legge 48 ratifica ed esecuzione della convenzione del consiglio d'europa sulla criminalità informatica, 2008. `http://www.parlamento.it/parlam/leggi/080481.htm`.

[9] Oracle Tech Network. Virtualbox version 6. `https://www.virtualbox.org`.

[10] Access Data. Ftk imager lite version 3.1.1. https://accessdata.com/product-download/ftk-imager-lite-version-3.1.1.

[11] Wikipedia. Virtualbox — wikipedia, l'enciclopedia libera, 2019. http://it.wikipedia.org/w/index.php?title=VirtualBox&oldid=102608374.

[12] Wikipedia. Open virtualization format — wikipedia, l'enciclopedia libera, 2018. http://it.wikipedia.org/w/index.php?title=Open_Virtualization_Format&oldid=100810526.

[13] Oracle Tech Network. Virtualbox documentation. https://www.virtualbox.org/manual/ch08.html.

[14] Wikipedia. Hash — wikipedia, l'enciclopedia libera, 2019. http://it.wikipedia.org/w/index.php?title=Hash&oldid=101843551.

[15] Wikipedia. Oauth — wikipedia, l'enciclopedia libera, 2019. http://it.wikipedia.org/w/index.php?title=OAuth&oldid=102175018.

[16] Google Developers. About permissions. https://developers.google.com/drive/api/v3/about-permissions.

Ringraziamenti

Ringrazio il mio relatore Danilo Montesi per avermi dato la possibilità di lavorare ad un progetto importante e stimolante come questo.

Ringrazio Flavio per la sua immensa disponibilità e pazienza nel avermi fornito supporto tecnico.

Ringrazio i miei genitori che mi hanno sempre supportato in ogni mia scelta.

Ringrazio Drus, Moros e Cala per le infinite risate, le spumeggianti sessioni di studio e per i carri che mi hanno dato in questo triennio.

Ringrazio gli amici delle superiori per continuare a portare avanti un'amicizia importante e non scontata.