SICUREZZA INFORMATICA

SPUNTI ED APPROFONDIMENTI

Andrea Pasquinucci, PhD CISA CISSP

ICT Security MAGAZINE

www.ictsecuritymagazine.com

AUTORE

Andrea Pasquinucci
(PhD CISA CISSP)

Consulente freelance in sicurezza informatica: si occupa prevalentemente di consulenza al top management in Cyber Security e di progetti, governance, risk management, compliance, audit e formazione in sicurezza IT.

ICT Security Magazine

ICT Security, prima pubblicazione italiana dedicata in forma esclusiva alla sicurezza informatica e alla business continuity, si pone l'obiettivo di coinvolgere i più importanti attori del settore, aziende e istituzioni pubbliche, per la diffusione degli elementi conoscitivi legati a tutti gli aspetti della Cyber Security.

Con oltre 20 anni di esperienza, il magazine affronta le ultime novità di settore con approfondimenti mirati e fornisce al lettore nuovi spunti e linee guida per migliorare ed ampliare le proprie conoscenze.

Indice

Indice

INTRODUZIONE

In questo libro sono stati raccolti e parzialmente rielaborati otto articoli pubblicati su **ICT Security Magazine** nell'arco degli ultimi cinque anni.

Partendo dagli albori dell'universo IT – nascita dell'hardware, avvento di Internet, prime tipologie di attacchi, modelli embrionali di sicurezza informatica – l'autore ne approfondisce storia e "miti" fondativi per ripercorrere la vertiginosa evoluzione che arriva fino ai giorni nostri e oltre, interrogandosi anche su rischi e potenzialità di strumenti attuali (come le tecniche di riconoscimento biometrico) ovvero futuribili (quali elaboratori quantistici o algoritmi Post-Quantum).

Mantenendo la sicurezza come lente d'osservazione e filo conduttore, la raccolta è organizzata nelle seguenti aree tematiche, cia-

scuna delle quali comprende uno o più argomenti specifici:

1. Sicurezza IT, Hardware e Confidential Computing
2. Sicurezza in Internet
3. Privacy e Sicurezza
4. Autenticazione e Sicurezza
5. L'evoluzione della Crittografia.

Al fine di facilitare la consultazione, ogni capitolo è corredato dalla propria bibliografia di riferimento.

Pur differendo rispetto all'ambito applicativo e agli aspetti tecnico-procedurali tutti i contenuti contribuiscono, arricchendolo di spunti mai banali, al dibattito sulla gestione della sicurezza informatica. Riflessioni ed analisi tanto necessarie quanto stimolanti, poiché riguardano applicazioni e sistemi a cui affidiamo quotidianamente dati personali e verso cui deleghiamo funzioni che rappresentano - in modo ormai irreversibile – parte essenziale della nostra vita personale e professionale.

A cura della redazione di **ICT Security Magazine**

01

SICUREZZA IT, HARDWARE E CONFIDENTIAL COMPUTING

1.1 L'Hardware e la sicurezza IT

1.1.1 UN PÒ DI STORIA

Il 3 gennaio 2018 sono state divulgate informazioni riguardo due vulnerabilità di sicurezza informatica chiamate "Spectre" e "Meltdown" [Rif. 1] la cui origine è in alcune funzionalità dei microprocessori. "Spectre" e "Meltdown" sono pertanto vulnerabilità presenti nell'Hardware e risolvibili definitivamente solo con la sostituzione o modifica dell'Hardware stesso, anche se ovviamente sono possibili delle mitigazioni software.

Per capire come si è arrivati a "Spectre" e "Meltdown", perché sono così importanti e pericolosi, e apprezzare i principi su cui si basano, è utile partire dalle origini e riassumere velocemente un po' di storia dell'informatica.

1.1.1.1 LA NASCITA DELL'HARDWARE MODERNO

E' negli anni '60 che vedono la luce i primi elaboratori la cui struttura di base è la progenitrice degli elaboratori attuali. Sino ad allora gli elaboratori, oltre ad essere estremamente costosi e di enormi dimensioni, avevano quasi esclusivamente tre grandi clienti: i militari, le grandi aziende e le università. Questi elaboratori, che ora chiamiamo genericamente Mainframe, funzionavano secondo l'approccio chiamato "Batch": i programmatori preparavano un "Job", ovvero l'esecuzione di un programma od un gruppo di programmi, e caricavano nell'elaboratore[1] il codice da eseguire e tutti i dati necessari all'esecuzione. Nell'elaboratore risiedeva permanentemente un programma, il Sistema Operativo, con i compiti di caricare i Job, lanciarne l'esecuzione e riprendere il controllo dell'elaboratore al termine dell'esecuzione di un Job per avviare il Job successivo.

E' importante notare che solo un Job è caricato ed eseguito nell'elaboratore in ogni istante: durante l'esecuzione un Job ha a disposizione e controlla tutto l'hardware dell'elaboratore, non vi è condivisione alcuna delle risorse. Non essendoci condivisione di risorse ne connessione ad altri elaboratori, non vi sono problematiche o possibili minacce di sicurezza. Gli unici possibili rischi sono dovuti a errori interni ai programmi stessi che possono portare a risultati errati o ad errori tali da far abortire l'esecuzione.

1) I programmi e i dati erano tipicamente caricati per mezzo di schede cartacee perforate lette da opportune unità di input dell'elaboratore.

Ma questo approccio all'esecuzione dei programmi non è efficiente: tra un Job e l'altro la CPU, ai tempi la risorsa più costosa, poteva rimanere inutilizzata per molto tempo in attesa del caricamento e scaricamento di dati e istruzioni. Per ovviare a questo si introdussero, in Hardware, "pipeline" di esecuzione e CPU SuperScalari. Le "pipeline" (Fig. 1.1.1) sono componenti Hardware con lo scopo di caricare, preparare per l'esecuzione, e scaricare dati e istruzioni, mentre le CPU SuperScalari hanno molteplici unità di esecuzione, ad esempio una per eseguire calcoli su numeri interi, una su numeri a virgola mobile (floating point), una su numeri booleani ecc., in modo da poter eseguire in parallelo calcoli diversi.

Per gestire questi sistemi è necessario che il ruolo del Sistema Operativo aumenti: da puro sistema di carico/scarico di dati e istruzioni diventa il gestore delle code e dell'elaborazione, anche se sempre di un solo Job alla volta.

Si noti anche una caratteristica che diventerà importante per alcuni tipi di vulnerabilità "Spectre" e "Meltdown": nelle CPU SuperScalari le istruzioni non sono sempre eseguite nell'ordine previsto dal programma, spesso anzi alcuni tipi di istruzioni, ad esempio calcoli su numeri a virgola mobile, sono eseguite in anticipo non appena disponibili i dati necessari. Questo è fatto perché se poi il calcolo risulta necessario, allora si è guadagnato del tempo perché il risultato è già pronto, altrimenti se il calcolo non è utilizzato dal programma, viene eliminato. La gestione dell'esecuzione delle istruzioni fuori dall'ordine del programma è demandata principalmente ai circuiti Hardware che fanno si che per il Sistema Operativo ed i programmi l'esecuzione risulti sempre nell'ordine dato.

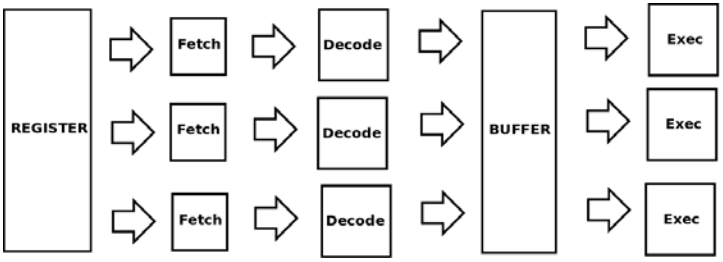

Figura 1.1.1 - Modello di Pipeline

La logica adottata in Hardware è che è meglio tenere impegnate le CPU anche con calcoli che nel caso non saranno utili, piuttosto che tenerle ferme: alcune volte ci sarà un grande guadagno di tempo, altre volte il guadagno sarà minimo, ma in ogni caso non si perderà mai tempo.

1.1.1.2 MODELLI DI SICUREZZA IT

Prima di proseguire con l'evoluzione degli elaboratori, è utile rammentare quale fosse ai tempi l'approccio alla sicurezza informatica.

Visto che gli elaboratori erano per lo più mono-processo, mono-utente e non erano connessi tra di loro, le principali esigenze di sicurezza provenivano dai militari. Il principale modello di sicurezza militare è quello formalizzato e dimostrato da Bell - La Padula nel 1973, un modello di sicurezza a multi-livello: rispecchiando la gerarchia militare, il modello richiede che ognuno possa leggere solo i documenti al proprio livello ed ai livelli inferiori, mentre

possa inviare nuovi documenti solo al proprio livello ed ai livelli superiori (write-up, read-down).

Bell e La Padula dimostrarono che questo modello garantisce la riservatezza delle informazioni. Biba nel 1977 dimostrò che il modello opposto (write-down, read-up) garantisce l'integrità delle informazioni.

Vi sono altri modelli per la sicurezza delle informazioni: ad esempio il controllo di accesso discrezionale (DAC) prevede che ogni utente possa decidere chi può accedere e come alle proprie informazioni, usualmente tramite la configurazione di liste di controllo accesso (ACL). Questo approccio è stato adottato ad esempio dai Sistemi Operativi di tipo Unix.

Più in generale si è giunti a formulare il concetto di sistemi "di fiducia" ("Trusted") che prevedono che vi sia un insieme di regole di sicurezza ("Trusted Computing Base", TCB) gestite dal responsabile della sicurezza, e che all'interno del Sistema Operativo vi sia un processo sicuro ("Reference Monitor") che verifica che l'esecuzione di ogni istruzione soddisfi la TCB.

Questi concetti sono anche formulati nei requisiti e certificazioni di sicurezza quali gli Orange Book e i Common Criteria [Rif. 2, 3].

Più avanti sarà necessario approfondire alcuni aspetti dell'approccio ai sistemi Trusted, TCB e Reference Monitor. Per ora però è importante ricordare l'approccio alla sicurezza come descritto dai sistemi a multi-livello.

1.1.1.3 MULTIPROCESSO E MULTIUTENTE

Ritornando alla nostra breve storia, il passo successivo nell'evoluzione degli elaboratori è l'esecuzione contemporanea di più programmi, ovvero di più Job ognuno di un utente diverso. Questo è il passo cruciale per la nascita degli elaboratori moderni ed il principale attore di questo passo è il Sistema Operativo. L'evoluzione del Sistema Operativo lo porta ad adempiere principalmente a due compiti:

1. **Macchina Estesa** (o macchina virtuale): la scrittura di programmi direttamente in linguaggio macchina o in linguaggi appena più evoluti richiede comunque la conoscenza dell'Hardware e del suo funzionamento, questo rende molto complessa la scrittura dei programmi; invece il Sistema Operativo può interporre una interfaccia applicativa, che semplifica e nasconde le complessità dell'Hardware rendendo più semplice la scrittura dei programmi; non solo, permette anche di eseguire lo stesso programma su Hardware diversi;
2. **Gestore delle Risorse**: se si vogliono eseguire più programmi allo stesso tempo, è necessario che il Sistema Operativo, che ha il compito di caricarli e scaricarli dalla CPU, li gestisca durante tutto il tempo della loro esecuzione, non solo all'inizio ed alla fine, ed in maniera molto precisa e dettagliata.

E' necessario approfondire come il Sistema Operativo svolge il compito di Gestore delle Risorse. Prima di tutto, all'interno di una CPU (e per ora consideriamo solo elaboratori con una CPU ciascuno) viene eseguito un solo programma alla volta. Vengono

però caricati in memoria più programmi e la CPU ne alterna l'esecuzione.

Il Sistema Operativo decide quale programma deve essere eseguito, per quanto tempo, e verifica che durante l'esecuzione un programma non tocchi i dati degli altri programmi nè utilizzi periferiche Hardware in modo inappropriato.

L'alternanza di esecuzione dei programmi permette di rendere molto più efficiente l'utilizzo delle risorse, ad esempio mentre un programma legge o scrive dei dati, un altro viene eseguito nella CPU.

Il Sistema Operativo quindi gestisce l'accesso a tutte le risorse Hardware da parte dei programmi in modo che questo sia efficiente e che al contempo l'integrità, riservatezza e disponibilità dei dati siano garantite. Questo deve essere fatto dal Sistema Operativo in modo tale che durante l'esecuzione un programma possa comunque disporre di tutte le risorse Hardware, come se fosse eseguito da solo, in assenza di altri programmi.

1.1.1.4 MULTICS

Nel 1965 MIT, AT&T, IBM e GE decisero di sviluppare in comune un sistema operativo 'sicuro' chiamato Multics.

Questo sistema operativo doveva essere:

1. Progettato top-down

2. Capace di supportare almeno 1000 utenti contemporaneamente

3. *'Reliable'*

4. *'With sufficient control of access to allow selective sharing of information'*

ed avere molte altre caratteristiche di sicurezza militare. Il progetto non ebbe successo commerciale, uno dei motivi fu che le richieste di sicurezza intrinseca al sistema, garantite da uno sviluppo controllato top-down, portarono ad una eccessiva complicazione del software rispetto alle piattaforme hardware disponibili in quegli anni. Così nel 1969 AT&T convinse i partner a chiudere lo sviluppo di Multics, anche se Multics venne sviluppato in maniera indipendente ancora per una decina di anni.

Multics fu però un passaggio molto importante nella storia dei Sistemi Operativi, anche per essere il padre di Unix (e la somiglianza dei nomi non è casuale).

Ma come Multics avrebbe potuto soddisfare questi requisiti?

Come poteva un Sistema Operativo gestire le risorse, garantire l'isolamento tra programmi ed al contempo permettere che un programma utilizzasse tutte le risorse del sistema?

La soluzione che si adottò allora anche per Multics, e che tuttora è alla base dell'Hardware IT, è simile all'approccio alla sicurezza multi-livello citato precedentemente. In particolare Multics fu progettato con 8 livelli, chiamati Anelli di Protezione (*"Protection*

Rings") ed implementati in Hardware (vedi Fig. 1.1.2). L'anello più interno, convenzionalmente numerato con 0, è l'anello con maggiori privilegi, il livello massimo di sicurezza, mentre l'anello più esterno, il 7, ha il livello minore di privilegi. Ogni programma viene caricato ed eseguito solo ed esclusivamente in un livello, ma può chiamare altri programmi che sono eseguiti in altri livelli (tipicamente superiori). E' l'Hardware che isola i livelli, e non permette ad un programma eseguito ad un certo livello di accedere direttamente alle risorse di un altro livello. In ogni istante nell'Hardware vi è un indicatore che segna a quale livello sono eseguite le istruzioni. Per accedere a risorse gestite da un altro livello, un programma effettua una richiesta, ovvero una chiamata ad una funzione speciale (*System Call*) che attiva una chiamata Hardware dedicata (*Call Gate*). Queste chiamate Hardware permettono di accedere a specifiche routine ad un determinato livello: se i parametri passati con la richiesta sono corretti, allora la richiesta è accettata e viene eseguito il comando al nuovo livello. Ad esempio per leggere e scrivere su disco, un programma in esecuzione ad un livello esterno esegue una *System Call* al Sistema Operativo a livello 0 chiedendo di leggere/scrivere i dati. E' quindi il Sistema Operativo stesso che esegue la lettura/scrittura e poi ritorna i dati e l'esecuzione al programma al procedente livello.

In questo modo l'Hardware garantisce che il Sistema Operativo, a livello 0, gestisca direttamente tutte le risorse, sia in grado di fornire una interfaccia applicativa che virtualizza e semplifica l'accesso alle risorse, garantendo allo stesso tempo l'integrità, riservatezza e disponibilità dei dati. Senza la separazione tra Sistema Operativo e altri programmi fornita dall'Hardware, non sarebbe

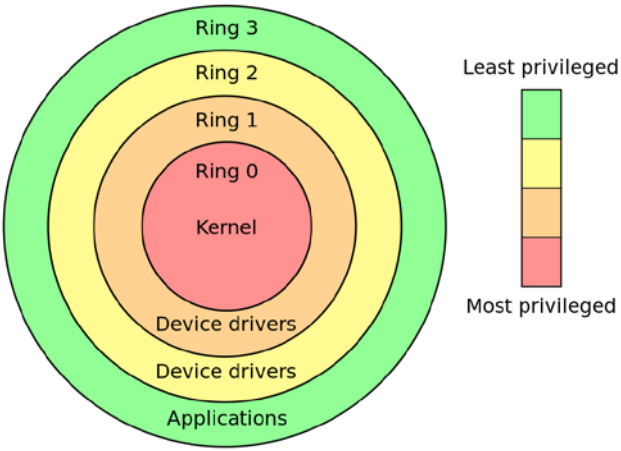

Figura 1.1.2: Anelli di protezione per le CPU Intel x86 [fonte Wikipedia]

possibile garantire la sicurezza dell'esecuzione contemporanea di programmi né la separazione tra programmi e dati di utenti diversi.

La maggior parte delle moderne CPU adotta una struttura ad Anelli di Protezione, a partire ad esempio dalle CPU Intel x86 che hanno 4 anelli[2]. In generale però gli attuali Sistemi Operativi non sfruttano tutti gli Anelli, ma ne usano solo due, tipicamente quello più privilegiato e quello meno privilegiato. Si dice che il Sistema Operativo opera in "kernel mode" (ring 0) mentre tutti gli altri programmi sono eseguiti in "user mode". Ci sono alcuni motivi per cui in pratica si usano solo due anelli tra cui la semplicità di gestione

2) Le più recenti CPU hanno 5 livelli, un nuovo livello è dedicato esclusivamente al supporto per gli Hypervisor di macchine virtuali ed in questo caso sono utilizzati 3 livelli: "Kernel mode", "User mode" e un livello solo per gli Hypervisor.

del codice, la portabilità del codice su diversi modelli di CPU, la maggiore facilità nel gestire molti privilegi in software piuttosto che in Hardware.

La ricerca di maggiori prestazioni insieme all'estensione delle capacità di calcolo delle CPU, all'esplosione del numero di periferiche e di utilizzi degli elaboratori, ha portato ad una crescita quasi incontrollata della dimensione dei Sistemi Operativi eseguiti in Kernel Mode. Ovviamente le dimensioni e la complessità sono tra i primi nemici della sicurezza, e questo si riscontra quotidianamente nelle vulnerabilità di sicurezza che si scoprono nei Sistemi Operativi.

1.1.1.5 LA GESTIONE DELLA MEMORIA DINAMICA

La presenza di Gate e System Call non è però sufficiente per raggiungere gli obiettivi del progetto Multics, è anche necessario che durante l'esecuzione ogni programma possa accedere, anche solo virtualmente, a tutto l'Hardware, in particolare a tutta la memoria dinamica. La gestione della memoria (principalmente la RAM, *Random Access Memory, e le Cache*) è un argomento molto complesso ma per i nostri scopi è necessario farne almeno un accenno di cui avremo assolutamente bisogno per capire il funzionamento di "Spectre" e "Meltdown".

L'idea di partenza è quella di dare ad ogni programma accesso a tutta la memoria in maniera virtuale. Ogni programma può quindi allocare qualunque indirizzo (virtuale) della memoria, questi indirizzi però non sono reali, vengono tradotti in tempo reale

negli indirizzi di memoria realmente utilizzati da una componente Hardware chiamata Unità di Gestione della Memoria (MMU) con l'ausilio di tabelle Hardware di conversione (alcune di queste sono chiamate TLB, LTD, GDT ecc.). In questo modo un programma pensa di utilizzare tutta la memoria, ma in realtà solo l'Hardware ed il Sistema Operativo sanno veramente quali locazioni di memoria fisica sono utilizzate dal programma, ed in quale memoria: RAM, Cache (e di quale livello) od anche disco nel caso di *swap*.

L'approccio di Multics ed anche di molti dei processori moderni quali ad esempio quelli della famiglia Pentium Intel, è quello di utilizzare un processo di gestione della memoria chiamato Segmentazione con Paginazione (*"Segmentation with Paging"*) che implementa sia il mapping tra indirizzi virtuali e fisici che la gestione della memoria fisica, con lo spostamento delle pagine di memoria da una memoria all'altra a seconda della necessità di uso. Infatti le memorie fisiche sono organizzate in maniera gerarchica: dallo swap su disco, memoria più lontana dalla CPU e più lenta, alla RAM, ai vari livelli di Cache di cui L1 è quello più vicino alla CPU, più veloce ma tipicamente piccolo (vedi Fig. 1.1.3).

Il Sistema Operativo fa si che vengano caricati nella Cache L1 le istruzioni ed i dati che servono all'esecuzione del programma. Si definisce "Cache Hit" quando il processore trova nella Cache L1 i dati o le istruzioni di cui ha bisogno. Quando invece il processore non trova nella Cache L1 i dati o le istruzioni di cui ha bisogno per proseguire l'elaborazione, si dice che si verifica un "Cache Miss".

Nel caso di Cache Miss, il Sistema Operativo deve dapprima liberare sufficiente spazio nella Cache per poter caricare i nuovi

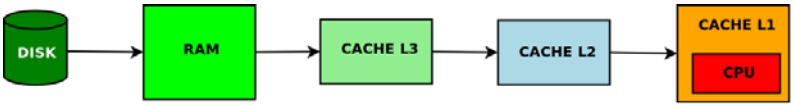

Figura 1.1.3: Gerarchia delle memorie

dati, e questo richiede trovare spazio nelle Cache inferiori, nella RAM o su disco, ove scrivere i dati. Poi trovare nelle Cache inferiori, nella RAM o su disco i dati da caricare e copiarli nella Cache L1. Dopodiché può essere ridato il controllo al programma che può procedere nell'esecuzione. E' chiaro che nel caso di Cache Miss i tempi di esecuzione di una istruzione sono molto più lunghi, anche 100 volte, che nel caso di Cache Hit.

Tutto ciò è ulteriormente complicato dal fatto che, come indicato inizialmente, le "pipeline" di esecuzione del codice sono molteplici e non è detto che le istruzioni vengano eseguite nell'ordine previsto dal programma, e questo è fatto per utilizzare al massimo le CPU.

Per "Spectre" e "Meltdown" è necessaria qualche ulteriore informazione tecnica su come è gestita la memoria di un elaboratore. Come già indicato, la memoria è tipicamente organizzata in Pagine e Segmenti.

Una Pagina è una unità di memoria sia fisica che virtuale, spesso di 4KB, che viene gestita dall'Hardware come un blocco unico con un unico indirizzo. Questo semplifica l'indirizzamento della memoria e le attività di copia dei dati tra le diverse Cache e memorie.

I Segmenti invece sono spazi di memoria virtuali, gestiti dall'Har-

dware, che permettono ad un programma di avere a disposizione più di uno spazio di memoria con indirizzi consecutivi che partono da 0. I Segmenti possono avere dimensioni (virtuali) differenti a seconda delle necessità del programma.

Un tipico esempio di uso di un Segmento è quello delle librerie condivise ("Shared Libraries"): alcune librerie di sistema, ad esempio quelle grafiche, sono spesso utilizzate da molti programmi contemporaneamente. In assenza di Segmenti, ogni programma deve caricare nel proprio spazio virtuale una copia della libreria, mentre se l'Hardware supporta i Segmenti, il Sistema Operativo può caricare una libreria in un Segmento (read-only) ed includere il Segmento nello spazio virtuale di memoria di tutti i programmi che lo utilizzano, velocizzando le operazioni e riducendo l'utilizzo di memoria. Visto che un Segmento può essere molto grande, in CPU Intel x86 i segmenti sono di 4GB - dimensione superiore a quella della Cache L1, è necessario suddividere in Pagine i Segmenti in modo da gestire il caricamento dei dati nelle diverse memorie.

Un'ultima osservazione: nelle attuali CPU a 64 bit, l'utilizzo dei Segmenti non è più veramente necessario poiché lo spazio di memoria a 64 bit indirizzabile è veramente enorme e i dati possono essere distribuiti direttamente in blocchi di memoria con indirizzo di base differente. I Segmenti sono comunque utilizzati per la protezione della memoria, ad esempio indicano se un'area di memoria può essere acceduta solo in "kernel mode" od anche in "user mode".

1.1.1.6 SICUREZZA E HARDWARE

L'Hardware ha quindi un ruolo fondamentale ed indispensabile nel garantire la sicurezza, intesa come riservatezza, integrità e disponibilità dei dati e delle risorse. In particolare l'Hardware:

- isola il Sistema Operativo da tutti gli altri programmi;
- permette solo al Sistema Operativo di accedere all'Hardware ed a tutte le risorse del sistema;
- obbliga tutti gli altri programmi ad accedere alle risorse tramite il Sistema Operativo che può quindi sia gestire le risorse fisiche (integrità e disponibilità) sia verificare i permessi di accesso alle risorse assumendo il ruolo di Reference Monitor (riservatezza);
- gestisce, insieme al Sistema Operativo, la concomitanza di esecuzione di più programmi contemporaneamente ed in particolare della memoria dinamica.

Cosa succederebbe se un programma potesse anche solo leggere i dati di un altro programma in esecuzione, senza passare attraverso il Sistema Operativo? Dovremmo sicuramente definire questa una vulnerabilità Hardware.

1.1.2 ATTACCHI SIDE CHANNEL: ROW HAMMER ED ATTACCHI ALLA CACHE

La sicurezza dei sistemi IT è principalmente conosciuta in relazione ad attacchi esterni od interni che sfruttano vulnerabilità del codice od abuso di privilegi. Abbiamo tutti sentito parlare di virus, worm e malware di vario tipo che sfruttano vulnerabilità dirette di sistemi e applicazioni. Un'altra grande categoria di attacchi noti a tutti sono quelli che genericamente possiamo indicare con il termine "Phishing", ovvero truffe e inganni che convincono l'utente di un sistema IT a fornire all'attaccante informazioni riservate od a sottrarre danaro. Inoltre una categoria di attacchi realizzabili con i sistemi IT sono quelli svolti dall'interno ("Insider attack") ovvero da persone che abusano dei propri privilegi sui sistemi IT per danneggiare l'organizzazione per cui lavorano a proprio favore od a favore di terzi.

Nell'ambito dei sistemi IT vi è però un'altra importante categoria di minacce e attacchi anch'essa di grande rilevanza e lunga storia, sono gli attacchi "Side Channel" (ovvero condotti tramite un canale laterale) e di inferenza. Invece di sfruttare una vulnerabilità esplicita e diretta di un sistema, un attacco Side Channel sfrutta delle caratteristiche indirette del sistema per giungere comunque alla sua compromissione. Gli esempi sono molti, ma è comunque utile ricordare alcuni dei principali.

Una delle principali famiglie di attacchi Side Channel va ora sotto il nome di TEMPEST e riguarda metodi per spiare, ovvero estrarre informazioni da un sistema IT, sfruttando l'emissione dal sistema

IT di segnali radio, elettrici, termici, di vibrazioni ecc. [Rif. 4, 5]. TEMPEST è un nome in codice della NSA (National Security Agency, USA) e si riferisce ad un insieme di misure da implementare, di verifiche da effettuare su strumenti informatici, e relative certificazioni, per prevenire alcune classi di attacchi Side Channel. I primi studi delle minacce di tipo TEMPEST risalgono ai primi anni del 1900 quando si incominciarono a sviluppare metodi alternativi per intercettare messaggi trasmessi via telefono o radio. E' però principalmente negli anni '60 e '70 che TEMPEST fu ideato e sviluppato[3].

Uno dei principali canali di attacchi Side Channel è quello delle radiazioni elettromagnetiche. Ad esempio è stato mostrato che è possibile ricostruire quanto mostrato su di un monitor non schermato anche a distanza di metri e attraverso muri [Rif. 7]. Analogamente sfruttando l'emanazione di radiazioni elettromagnetiche è stato possibile intercettare quanto in transito attraverso modem non schermati, ma anche attraverso cavi di connessione. Inoltre smart-card, schede FPGA, CPU, chip crittografici ecc. emettono radiazioni elettromagnetiche che, se non schermate, possono permettere l'intercettazione delle informazioni.

E' possibile intercettare informazioni anche monitorando l'utilizzo di corrente elettrica da parte dell'Hardware ed in alcuni casi monitorando l'emissione di onde sonore o di emissioni termiche.

Più recentemente è stato mostrato [Rif. 8] come la ricostruzione

3) Oggi il termine corretto sarebbe EMSEC, ovvero "Emissions Security", anche se TEMPEST è ancora utilizzato come termine generico.

di quanto presente su di un monitor, anche di uno smartphone, è possibile analizzando i rumori acustici di fondo emessi dagli schermi e registrati da comuni microfoni presenti nei nostri dispositivi quotidiani.

Le principali misure di sicurezza TEMPEST sono:

1. la schermatura dei sistemi

2. la modifica dei processi software o dei componenti hardware in modo da non emettere radiazioni che possano portare o far dedurre informazioni.

Un altro esempio recente [Rif. 10] riguarda lo studio dei chip dedicati alle comunicazioni Wireless, come WiFi e Bluetooth. Questi chip includono sullo stesso substrato sia una componente ricetrasmittente analogica che una componente digitale utilizzata anche per la cifratura del traffico. E' stato notato che la presenza sullo stesso substrato di silicone di entrambe le componenti porta in molti chip alla presenza di interferenze fra le due componenti, generate dalla componente digitale e amplificate dalla componente analogica. Ne risulta quindi che studiando il "rumore" della trasmissione radio è possibile identificare i segnali emessi dalla componente digitale del chip durante la cifratura / decifrazione del traffico. In alcuni casi questo fenomeno può portare alla identificazione della chiave segreta di cifratura anche a qualche metro di distanza.

Vi sono anche attacchi Side Channel non direttamente fisici come i precedenti, ma implementati tramite monitoraggio delle appli-

cazioni IT o dell'attività dell'Hardware stesso. Una classe importante di questi attacchi è quella che utilizza la misura dei tempi di elaborazione in particolare quando applicati alla crittografia. Sin a partire da [Rif. 9] è stato mostrato come conoscendo l'algoritmo di cifratura e misurando i tempi di esecuzione dell'algoritmo crittografico, sia possibile in alcuni casi particolari da parte di una terza parte presente sul sistema, risalire alle chiavi di cifratura. Questo può essere evitato se l'algoritmo crittografico esegue le elaborazioni in tempo costante, ovvero in modo indipendente dai dati in ingresso (incluse le chiavi), od in alcuni casi con una modifica dell'algoritmo chiamata "Bliding" che introduce dei dati casuali all'interno dell'elaborazione, poi rimossi, in modo da rendere inintelligibile il risultato dell'attacco tramite misura dei tempi di elaborazione.

1.1.2.1 ROW HAMMER

A partire dal 2015 con lo studio di Kim et al. [Rif. 11] sulla vulnerabilità poi chiamata Row Hammer, cresce grandemente l'interesse sia accademico che dei professionisti di sicurezza informatica, a partire dai componenti del Project Zero di Google, all'interazione tra funzionalità hardware in particolare delle CPU e sue componenti, e le violazioni del modello di sicurezza ad anelli descritto nella prima parte di questo articolo.

L'osservazione alla base dell'effetto Row Hammer è che la miniaturizzazione delle componenti hardware, in particolare delle DRAM, può portare ad errori dovuti a disturbi provenienti da loca-

zioni vicine.

Semplificando il più possibile la descrizione del processo fisico, si può dire che in una cella DRAM il valore 0 o 1 è rappresentato dalla presenza o meno di carica elettrica in un condensatore. Ogni cella che rappresenta 1 bit è composta da un condensatore e da un transistor, e le celle sono organizzate in righe. Le celle di una riga vengono lette / scritte insieme in un'unica operazione (si veda la Figura 1.1.4). Le operazioni di lettura in realtà rimuovono fisicamente le cariche dai condensatori che poi devono essere ricaricati, ovvero riscritti, per mantenere il dato. Ma i condensatori, soprattutto quelli così piccoli, si scaricano velocemente, per cui è necessario rinfrescare i dati, ovvero rileggerli/riscriverli, molto frequentemente. Ad esempio DDR3 DRAM hanno un periodo di refresh di 64 millisecondi, mentre sono necessari circa 50 nanosecondi per leggere / scrivere una riga di memoria.

Per ingrandire la capacità delle memorie, e ridurne il costo, è necessario ridurne le dimensioni, ma questo vuol dire avvicinare tra loro le celle. I produttori di DRAM si sono sempre preoccupati[4] di minimizzare i fenomeni in cui effetti elettromagnetici di una cella influiscono sulle celle limitrofe, chiamati anche Errori di Disturbo ("Disturbance Errors").

Questi errori sono stati ridotti sia con misure fisiche di isolamento delle celle, sia con il processo di rinfrescamento dei dati, sia con l'introduzione di memorie con ECC (Error Correction Code), in gra-

4) Sin dagli anni '70, ad esempio già con la DRAM Intel 1130..

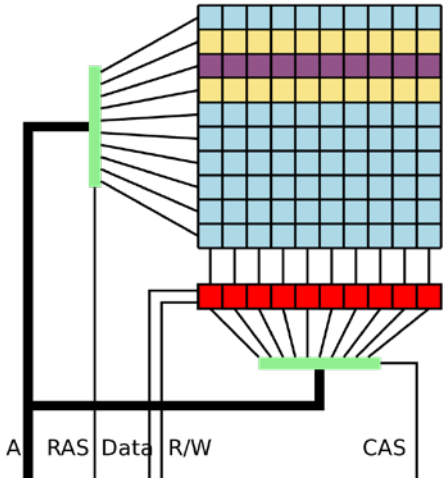

Figura 1.1.4: Esempio di attacco Row Hammer: la lettura rapida delle righe gialle può indurre il cambio di valore di alcune celle viola, la riga rossa indica il "row buffer" [fonte Wikipedia]

do di correggere 1 o 2 errori per riga / parola. Le ECC DRAM sono però più costose e un po' più lente delle memorie non-ECC e sono utilizzate tipicamente solo in server di alto livello[5].

I produttori di DRAM hanno sempre considerato errori prodotti casualmente o da fenomeni fisici esterni, ma non hanno però considerato che qualcuno potesse causare volontariamente degli errori di disturbo per violare la sicurezza del sistema. Invece questo è proprio quello che Kim et al. si sono proposti di fare nello studio [Rif. 11].

5) ECC DRAM furono introdotte principalmente per correggere errori indotti da radiazioni elettromagnetiche esterne e radiazioni di background quali particelle alpha, neutroni ecc. ad esempio emessi dal sole, che possono interferire con una cella e cambiarne il valore.

Quello che Kim et al. hanno notato è che leggendo un bit[6] alcune celle nelle righe limitrofe perdono carica elettrica ad una velocità leggermente più veloce di quanto succede normalmente. Se il bit viene letto alcune volte nell'arco di un periodo di rinfrescamento, la perdita di carica elettrica nelle righe limitrofe è minima e non produce alcun effetto. Ma se invece si ripete la lettura dello stesso bit o della stessa riga, centinaia di migliaia di volte nell'arco di un periodo di rinfrescamento, allora alcune celle possono scaricarsi completamente, cambiando quindi il loro valore.

Per riuscire a leggere migliaia di volte lo stesso bit, od un bit nella stessa riga, nell'arco di un periodo di rinfrescamento è necessario però aggirare le misure di miglioramento delle prestazioni presenti nell'hardware. In particolare all'interno di ogni DRAM sono presenti dei "row buffer" (anche chiamati "sense-amplifier") che mantengono copia di una riga in modo che letture immediatamente successive siano servite da quest'ultimo e non direttamente dalla riga.

Inoltre, come descritto nella prima parte di questo articolo, una volta letto un dato dalla RAM, questo viene caricato nelle varie Cache presenti nella CPU, in modo che letture immediatamente successive dello stesso dato non accedano per nulla alla RAM ma ad una Cache e siano molto più veloci.

E' possibile, anche se non facile, scrivere programmi che aggirino le Cache, i "row buffer" ecc. In [Rif. 12] sono riportati alcuni

6) Come indicato, la lettura di un bit induce la lettura / annullamento / riscrittura di una intera riga di memoria.

riferimenti ad articoli che descrivono tecniche per implementare attacchi Row Hammer. Sono stati ideati anche attacchi in javascript all'interno di browser, via network, che utilizzano le GPU, su smartphone ecc.

La difficoltà di implementazione degli attacchi non ha però evitato di trovare modi di renderli molto pericolosi. L'idea principale è quella di modificare un bit in memoria che garantisce la sicurezza del sistema: ad esempio se si riesce a modificare un bit che indica che un settore di memoria è privilegiato, allora qualunque processo può accedere a quei dati riservati. Modificando quindi opportuni bit in memoria è possibile sia accedere a dati riservati al Kernel o di altri utenti e processi, ed anche accedere direttamente all'anello 0 (Kernel mode) senza passare da System Call e Gate, e così via. Potenzialmente quindi gli attacchi Row Hammer possono completamente aggirare qualunque misura di sicurezza e permettere all'attaccante qualunque azione sul sistema. Le possibili conseguenze di un attacco Row Hammer includono: l'accesso amministrativo al sistema operativo, l'accesso a chiavi crittografiche e dati riservati, l'accesso da una macchina virtuale alle applicazioni e dati di un'altra macchina virtuale sullo stesso Host ecc.

Per implementare un attacco Row Hammer il problema principale quindi è quello di identificare esattamente i bit in memoria fisica[7] DRAM che si vogliono modificare, ed ideare una procedura in grado di modificare solo quelli. La difficoltà di esecuzione di questi

7) Si ricordi la differenza tra memoria fisica e virtuale, pagine, segmenti ecc. descritta precedentemente.

attacchi li rende quindi generalmente poco efficaci, ma come è stato dimostrato, sicuramente possibili.

1.1.2.2 CONTROMISURE PER ROW HAMMER

Bisogna innanzitutto notare che non tutte le memorie DRAM sono vulnerabili ad un attacco Row Hammer, dipende non solo dal modello di memoria ma in alcuni casi anche dalla singola memoria.

In generale la soluzione definitiva per eliminare la vulnerabilità è quella di produrre nuove memorie DRAM che per costruzione non siano suscettibili a Row Hammer. Questo però non è semplice in quanto le richieste del mercato sono di maggiore capacità, minore dimensione e minor costo delle memorie, e non è facile sviluppare nuove memorie che siano al contempo anche non vulnerabili a Row Hammer.

Inoltre le memorie ECC, come già indicato, garantiscono di correggere errori ad 1 o 2 bit per parola / riga, mentre gli attacchi Row Hammer tipicamente includono più bit e quindi possono aver successo anche su queste memorie.

Come mitigazione Hardware, in alcune nuove CPU e memorie sono state introdotte istruzioni che permettono di monitorare la frequenza di accesso alle righe di memoria e, se questa supera certe soglie, di forzare l'esecuzione di un rinfrescamento, il che impedisce l'effetto Row Hammer.

Sono anche state proposte ed implementate molteplici altre mi-

sure di protezione. Tra le più semplici misure di protezione vi sono le seguenti:

- il dimezzamento del tempo di refresh, ad esempio ogni 32 millisecondi, riduce l'efficacia di molti tipi di attacchi Row Hammer, potenzialmente a costo di rallentare l'accesso alla memoria;

- l'introduzione di procedure per verificare se le memorie sono vulnerabili ad attacchi Row Hammer, in modo da implementare misure di protezione solo su queste;

- visto che l'attacco Row Hammer esegue delle attività molto particolari, l'introduzione di misure di monitoraggio degli accessi alla RAM o dell'utilizzo delle Cache (che devono essere svuotate o aggirate dall'attaccante) o di Buffer di memoria, per identificare e bloccare i processi che eseguono un attacco Row Hammer;

- analogamente i sistemi operativi possono limitare l'accesso o l'uso di istruzioni utili ad implementare Row Hammer, come le istruzioni clflush e kmalloc heap in linux;

- infine è possibile che il sistema operativo separi fisicamente nelle memorie i dati con privilegi diversi o di diversi utenti (ad esempio per macchine virtuali) in modo che un attaccante non possa leggere dati in una riga confinante ad una con altri privilegi, anche se questo può introdurre molte complicazioni alla già non semplice gestione della memoria.

1.1.2.3 ATTACCHI ALLA CACHE

Come abbiamo visto, i dati sono gestiti non solo nella RAM ma anche in una gerarchia di Cache, tipicamente 3, ciascuna più piccola della precedente ma più veloce. L'ultima Cache, usualmente la terza "L3" anche detta Last Level Cache "LLC", quella più lontana dalla CPU e più vicina alla RAM, è quella di maggior interesse in quanto tipicamente contiene tutti i dati presenti anche nelle Cache inferiori e per tutti i Core presenti sul sistema. Sono stati ideati alcuni tipi di attacchi che sfruttano la caratteristica principale della Cache LLC, ovvero quella di essere più veloce della RAM. I più noti tra questi attacchi sono "Prime + Probe", "Flush + Reload" e "Flush + Flush" [Rif. 13]. L'idea di base di questi attacchi è di misurare il tempo di lettura di un dato, se il dato è già presente in Cache il tempo di lettura è molto inferiore al caso in cui il dato non è presente in Cache ed il sistema operativo deve leggerlo nella RAM e copiarlo nella Cache.

Sono state ideate due classi di attacchi, entrambe violano la riservatezza dei dati ma non permettono la modifica degli stessi e l'aumento di privilegi di un processo. Gli attacchi alla Cache non sono così distruttivi come Row Hammer, ma possono comunque avere conseguenze molto gravi.

Il primo tipo di attacco permette di creare un "Covert Channel", ovvero un canale di comunicazione non previsto, non facilmente rilevabile e che può violare le politiche di sicurezza del sistema. Un esempio eclatante è la realizzazione di una connessione Secure Shell (SSH) tra due macchine virtuali eseguite sullo stesso Host,

macchine virtuali tra le quali non dovrebbe esserci alcuna comunicazione [Rif. 14]. Questo canale di comunicazione è stabilito tramite un attacco alla Cache dell'Host che ospita le due macchine virtuali.

L'idea di base di questo tipo di attacco assomiglia un poco all'utilizzo del codice Morse, ovvero alla codifica dell'informazione in segnali brevi e lunghi, rispettivamente presenza o assenza di un dato nella Cache LLC.

Si può descrivere brevemente l'attacco come segue: l'attaccante è in grado di installare su entrambe le macchine virtuali il proprio codice (eventualmente diffuso anche tramite Malware). L'attaccante deve identificare un'area fisica della Cache da utilizzare per l'attacco. Questo è forse il punto più complesso dell'attività in quanto, come descritto, l'accesso alla Cache è tramite la memoria virtuale, mentre la Cache è organizzata secondo la memoria fisica. Inoltre i due programmi sono eseguiti su due macchine virtuali distinte per cui il mapping tra memoria virtuale e fisica effettuato sia dal sistema operativo che dall'Hypervisor usualmente differisce[8].

Nondimeno da un'analisi dell'hardware e di come sistemi operativi e Hypervisor gestiscono la memoria e le Cache, è possibile in alcuni casi trovare procedure tecniche per identificare le stesse

8) L'identificazione di una stessa area della Cache può essere semplice nel caso in cui l'Hypervisor utilizzi la "deduplication" (mappa alla stessa memoria fisica di blocchi di memoria uguali), misura utile a ridurre l'utilizzo della memoria.

aree della Cache LLC in entrambe le macchine virtuali. Una volta trovata una procedura per scrivere, leggere o rimuovere dati da un'area fisica della LLC, i due programmi eseguono con la stessa periodicità ma in maniera necessariamente asincrona le seguenti azioni:

- se il programma che invia il messaggio deve inviare un 1, legge/scrive dei dati (casuali) nella LLC un numero concordato di volte in rapida successione
- altrimenti se il programma che invia il messaggio deve inviare uno 0, non scrive nulla nella LLC per lo stesso periodo di tempo
- il programma che riceve il messaggio richiede, con la stessa periodicità, la lettura di uno stesso set di dati che ha caricato in memoria (RAM) e misura il tempo di ogni lettura.

Quando il processo che invia il messaggio invia un 1, cancella dalla LLC i dati del processo che riceve il messaggio, quindi quando il processo che riceve il messaggio legge i suoi dati deve attendere che questi siano ri-caricati dalla RAM nelle Cache: il tempo di lettura sarà quindi alto. Invece quanto il processo che invia il messaggio invia uno 0, l'area della LLC non viene sovrascritta, quindi il processo ricevente trova direttamente in Cache i propri dati e la lettura è molto più veloce. I tempi di lettura del processo ricevente sono quindi alti, per un 1, e bassi, per uno 0.

Ovviamente vi sono molte complicazioni nell'esecuzione di questa semplice procedura, ad esempio i programmi sono interrotti nell'esecuzione dal sistema operativo per l'esecuzione di Interrupt,

od altri programmi possono utilizzare la stessa area della LLC[9]. Possono quindi venire introdotti molti errori nel canale di trasmissione. La presenza di rumore ed errori in canali di comunicazione è però un argomento ben noto delle reti di comunicazione e si possono adottare protocolli di correzione degli errori che permettono di rimuovere quasi completamente gli errori e realizzare un canale di comunicazione funzionale, a costo di rallentare la velocità di invio dei dati.

Il secondo tipo di attacco permette di leggere dati riservati utilizzati da un altro programma in esecuzione sullo stesso sistema fisico. Tipicamente si considera come attacco esemplare la lettura della chiave segreta di de/cifratura utilizzata da un altro programma in esecuzione anche su di un'altra macchina virtuale, ma sempre sullo stesso Host.

L'esempio più semplice di questo tipo di attacco è quello di due utenti sulla stessa macchina (virtuale o fisica), uno che utilizza una chiave segreta per de/cifrare dei dati ed il secondo utente che intercetta la chiave segreta del primo. L'attacco si basa sull'utilizzo di librerie condivise, ovvero entrambi gli utenti utilizzano la stessa libreria crittografica per de/cifrare i dati. Essendo una libreria condivisa, il sistema operativo mantiene in memoria fisica RAM una sola copia della libreria, e mappa la stessa copia fisica nelle due distinte memorie virtuali dei due processi, quello che esegue la de/cifrazione e quello che esegue l'attacco. Supponiamo inoltre che la

9) La gestione della LLC è comunque in carico al Sistema Operativo, in questo caso Host, ed i processi non possono utilizzare aree della LLC in maniera esclusiva.

libreria implementi l'algoritmo RSA utilizzando la procedura matematica chiamata "Square-and-multiply" (questa procedura non è più utilizzata dalle librerie crittografiche attuali)[10]. La caratteristica principale di nostro interesse di questa procedura matematica è che è iterativa sui bit della chiave segreta, ripetuta per ogni blocco di dati da de/cifrare, e che esegue due operazioni distinte nel caso in cui un bit della chiave segreta sia 1 o 0. Utilizzando la stessa libreria crittografica, l'attaccante può agire come segue:

- per prima cosa l'attaccante rimuove dalla Cache le istruzioni della libreria crittografica, utilizzando l'istruzione clflush o caricandovi altri dati

- poi l'attaccante attende un tempo sufficiente a che il processo vittima esegua un ciclo della procedura "Square-and-multiply"

- infine l'attaccante misura il tempo di lettura (o esecuzione) di una delle due operazioni della libreria.

Se il processo vittima ha utilizzato la stessa operazione dell'attaccante, il tempo misurato è breve perché il processo vittima ha già caricato l'operazione in LLC, altrimenti il tempo misurato è notevolmente maggiore perché l'operazione deve essere caricata dal processo attaccante dalla RAM.

10) Un simile attacco è possibile quando l'algoritmo AES è implementato tramite "T-tables", anche questa implementazione non è più utilizzata dalle librerie crittografiche attuali.

Visto che l'utilizzo delle due operazioni è legato alla presenza di un 1 o di uno 0 nella chiave segreta, è possibile, ripetendo molte volte le misure ed a meno di rumori ed errori come nel caso precedente, individuare i bit della chiave segreta anche con precisioni superiori al 70%.

1.1.2.4 CONTROMISURE PER ATTACCHI ALLA CACHE

Le contromisure contro gli attacchi alla Cache sono in parte simili a quelle adottate per limitare gli attacchi Row Hammer. Con le tecnologie di oggi è difficile immaginare architetture fisiche diverse dall'attuale con la gerarchia di Cache tra la RAM e i Core e quindi risulta molto difficile eliminare il fenomeno fisico alla base di questi attacchi, ovvero la differenza di tempi di accesso alla Cache rispetto alla RAM.

Quello che invece è possibile fare è limitare o controllare l'accesso dei processi alle istruzioni che misurano con altissima precisione i tempi di esecuzione di blocchi di codice, e l'accesso ad istruzioni, come clflush, che permettono di gestire le Cache od indirizzarne i contenuti.

Queste misure possono essere implementate sia in hardware che nei Sistemi Operativi, ad esempio limitando l'esecuzione di alcune istruzioni solo in Kernel mode.

Altre misure includono l'eliminazione o forte diminuzione dell'utilizzo della deduplication, e una diversa gestione delle librerie comuni (shared libraries) e delle aree comuni delle memorie.

Questo impedirebbe ad esempio a diverse macchine virtuali di utilizzare le stesse aree fisiche della Cache, evitando l'attacco Covert Channel qui sopra descritto, ma al contempo rendendo più complesso e meno efficiente l'utilizzo stesso della Cache.

Infine per quanto riguarda gli algoritmi crittografici, questi devono comunque garantire elevati livelli di sicurezza e devono essere scritti in modo da eseguire il codice in tempo costante indipendentemente dagli ingressi, sia dei dati che delle chiavi. Le librerie crittografiche più recenti soddisfano questi requisiti, almeno per quanto riguarda gli attacchi noti.

1.1.3 MELTDOWN E SPECTRE

Nelle sezioni precedenti sono stati riassunti brevemente i principali eventi ed alcune informazioni utili per poter svolgere una breve analisi delle vulnerabilità di sicurezza informatica dovute all'Hardware e chiamate "Meltdown" e "Spectre" [Rif. 1], divulgate al pubblico il 3 gennaio 2018.

Ma prima di addentrarci in alcuni dettagli di queste vulnerabilità è necessario chiarire un poco il motivo della loro rilevanza e della loro fama. A prima vista infatti Row Hammer risulta essere una vulnerabilità di gravità maggiore in quanto permette di modificare dei dati (anche privilegiati) ed in alcuni casi di prendere il controllo totale del sistema. Invece Meltdown e Spectre permettono di accedere in sola lettura ad informazioni riservate presenti sul sistema. Row Hammer è però mitigato da alcuni fattori:

- si applica solamente alla memoria RAM e non tutte le memorie RAM sono vulnerabili (in linea di principio sarebbe possibile sostituire RAM vulnerabili con RAM non vulnerabili nei sistemi esistenti);

- gli attacchi Row Hammer sono molto evidenti in quanto vengono eseguite delle azioni non usuali;

- sono possibili e disponibili varie contromisure sia software che hardware che possono mitigare o impedire gli attacchi.

Meltdown e Spectre sono invece vulnerabilità hardware del disegno delle CPU. Meltdown si applica praticamente a tutti i processori Intel disegnati e prodotti dal 1995 (ad eccezione dei processori Intel Itanium e Intel Atom prima del 2013) ed in parte anche ai processori ARM. Spectre si applica a tutti i recenti processori.

E' da notare anche che sia di Meltdown che di Spectre esistono molte varianti che differiscono tipicamente per dettagli tecnici di esecuzione ma che ovviamente possono rendere del tutto inefficace una particolare mitigazione. Infatti sono state prodotte delle patch software sia per i sistemi operativi sia con microcodice di aggiornamento delle CPU, patch che mitigano alcune, ma non tutte, delle molte varianti di Meltdown e Spectre. Tra le varianti di Meltdown e Spectre vi sono anche:

- Netspectre [Rif. 16] che implementa un attacco Spectre via rete in Javascript;

- Foreshadow [Rif. 17] che implementa un attacco al Software

Guard eXtension (SGX), estensione delle CPU Intel per gestire dati riservati (Trusted Execution Environment), e a qualunque informazione presente nella Cache L1, anche del Kernel del Sistema Operativo, di un Hypervisor o del System Management Mode (SMM) della CPU stessa.

Come vedremo, Meltdown e Spectre si basano sul disegno di funzionalità di base delle CPU moderne. Per questo soluzioni definitive richiedono il ridisegno, la produzione e la sostituzione di tutte le CPU esistenti, nei server, nelle postazioni di lavoro, negli smartphone e anche di quelle integrate nei molti piccoli dispositivi IoT presenti nella nostra vita quotidiana. Se da una parte la gravità di queste vulnerabilità non è potenzialmente grave quanto quella di Row Hammer, l'origine e la diffusione fanno di Meltdown e Spectre due vulnerabilità molto più significative e la cui risoluzione richiederà ancora parecchi anni [Rif. 18].

1.1.3.1 MELTDOWN

Delle due vulnerabilità, Meltdown è la più semplice da implementare ed al contempo quella per cui, come vedremo, esistono alcune contromisure software.

La caratteristica principale di Meltdown è di violare uno dei principali assunti di sicurezza del disegno delle CPU moderne. Infatti, come abbiamo descritto nella sezione §1.1.1.4, uno degli assunti del disegno delle CPU moderne è la presenza di anelli di privilegi, implementati in hardware, tali per cui i processi non possono acce-

dere a dati ed istruzioni ad un livello di privilegio maggiore rispetto a quello a cui sono eseguiti. In particolare, accede ed è eseguito a livello zero (il più alto) solo il Kernel del Sistema Operativo, che controlla e gestisce tutto l'Hardware, i dati e l'esecuzione di tutti i processi. La vulnerabilità Meltdown permette di accedere in lettura a tutti i dati gestiti dal Kernel del Sistema Operativo da parte di un processo con privilegi bassi, evitando i controlli Hardware presenti nella CPU.

L'osservazione principale su cui si basa Meltdown è che le micro-istruzioni non sono sempre eseguite dalle CPU nell'ordine previsto dal programma. Come è stato descritto in §1.1.1.1 , caratteristiche comuni delle CPU moderne, sin dagli anni '60 e '70, sono le pipeline di esecuzione e l'esecuzione di istruzioni non in ordine ("out-of-order"). Nel corso degli anni, queste caratteristiche sono state sviluppate ampiamente dai progettisti e le CPU attuali sfruttano ampiamente sia l'esecuzione parallela di istruzioni di tipo diverso, sia l'esecuzione out-of-order che l'esecuzione speculativa (si veda Fig. 1.1.5). Il caso più semplice di esecuzione out-of-order è l'esecuzione speculativa in presenza di un Branch nel programma: ad esempio l'esito di un test su di un valore di una variabile decide se eseguire alcune istruzioni o delle altre (il tipico IF- THEN-ELSE). In attesa che vengano calcolati i valori che permettono di valutare la condizione (IF), se sono già presenti i valori per calcolare il THEN (o l'ELSE), allora la CPU procede all'esecuzione delle istruzioni nel Branch. Quando poi sarà valutata la condizione (IF) la CPU decide se mantenere il Branch già calcolato nel caso di scelta corretta, oppure abbandonarlo e calcolare un altro Branch. La presenza di un Branch non è però l'unica situazione in cui una CPU esegue

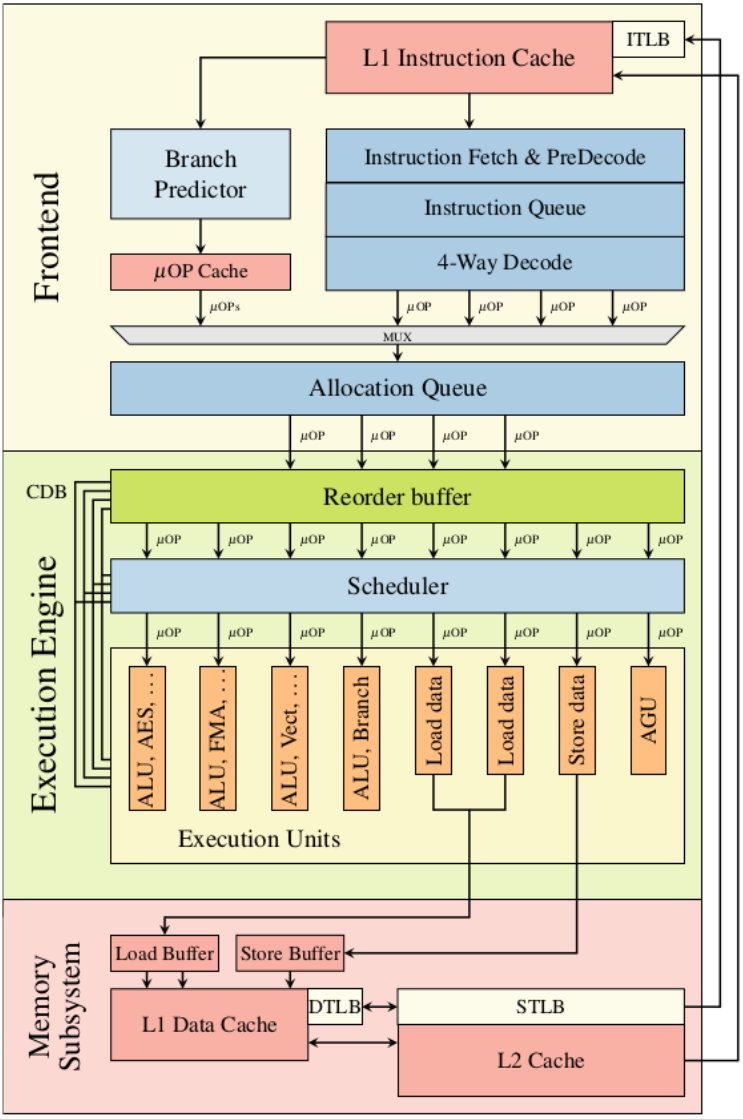

Figura 1.1.5: Descrizione schematica della micro-architettura di una CPU Intel Skylake con le principali componenti per l'esecuzione out-of-order delle micro-istruzioni [fonte Rif. 15]

istruzioni out-of-order, ve ne sono molte altre individuate dai progettisti delle CPU e che permettono delle ottimizzazioni, alle volte anche notevoli, delle prestazioni.

L'attacco Meltdown si può riassumere come segue:

1. l'attaccante è un processo utente senza alcun privilegio particolare, per cui non dovrebbe avere accesso diretto in lettura alla memoria del Kernel

2. per prima cosa viene individuata una linea di esecuzione che garantisce l'esecuzione out-of-order delle istruzioni successive

3. tra le istruzioni da eseguire out-of-order ed in anticipo rispetto al flusso delle altre operazioni, viene inserita, come descritto nei passi seguenti, la lettura di una locazione di memoria del Kernel, in linea di principio operazione vietata dall'Hardware

4. la CPU carica in un proprio registro il contenuto della locazione di memoria del Kernel per prepararsi all'istruzione di lettura

5. la lettura del registro viene eseguita nel Branch out-of-order e sulla base del valore del registro viene scritto un dato nella Cache

6. in parallelo la CPU verifica se il processo in esecuzione nella Branch out-of-order ha privilegi sufficienti per accedere ai dati

7. quando la verifica dei privilegi è completata senza succes-

so, la CPU esegue una eccezione che blocca l'esecuzione del programma.

E' chiaro che tra l'esecuzione del punto 5 e quella del punto 7 c'è una "race condition" (corsa critica), ma è possibile scrivere un flusso di istruzioni tale che 5 venga tipicamente eseguito prima di 7.

L'attacco non è però finito, perché l'informazione è presente solo nella Cache della CPU, non è memorizzata dal processo attaccante e non è direttamente leggibile in quanto privilegiata. Per estrarre l'informazione dalla Cache prima che la CPU la sovrascriva o la cancelli, si possono sfruttare gli attacchi alla Cache descritti in §1.1.2.3. In particolare la prima versione di Meltdown utilizza Flush+Reload come Covert Channel per inferire i dati presenti in Cache.

Ripetendo la procedura di attacco per tutte le locazioni di memoria del Kernel, il processo attaccante può leggere tutti i dati nella memoria del Kernel della macchina, violando l'isolamento che dovrebbe essere garantito dalla struttura Hardware in anelli[11].

Nell'esecuzione di Meltdown c'è un passaggio cruciale: l'attaccante deve già conoscere gli indirizzi in memoria (virtuale) dei dati del Kernel. In realtà, nei principali Sistemi Operativi attuali lo spazio di memoria del Kernel (kernel space) è mappato anche

11) Questo può avere conseguenze particolarmente critiche per sistemi di virtualizzazione a Container o para-virtualizzazione, ove il kernel del Sistema Operativo è comune a tutte le istanze virtuali.

nello spazio di memoria virtuale di ogni processo (user space). Questo rende molto più veloce e semplice la chiamata al Kernel (via System Call) per eseguire azioni anche comuni ma privilegiate e controllate dal Sistema Operativo, quali ad esempio la lettura e scrittura di dati su disco e qualunque altra operazione che interagisca con l'Hardware. L'assunto era che non vi fossero rischi di sicurezza in quanto l'Hardware avrebbe protetto l'accesso allo spazio di memoria virtuale riservato al Kernel.

Visto che Meltdown viola questo assunto, la più semplice ed efficace contromisura per impedire Meltdown è di non mappare lo spazio di memoria del Kernel nello spazio di memoria virtuale di ogni processo. Questa modifica del Sistema Operativo era già stata proposta[12] ed è stata rapidamente adottata dai produttori di Sistemi Operativi. E' da notare che questa non è la soluzione definitiva per Meltdown in quanto, ad esempio, nello spazio virtuale di ogni processo devono comunque essere presenti degli indirizzi di memoria privilegiati necessari per eseguire System Call ecc. Un effetto negativo di questa contromisura è che riduce le prestazioni dei sistemi: a seconda del tipo di elaborazione e di come la contromisura è stata implementata, si possono avere rallentamenti minimi o con incrementi sino al 30% del tempo di esecuzione dei programmi.

12) Ad esempio in Linux è chiamata Kernel Page Table Isolation (KPTI) o Kernel Address Isolation to have Side-channels Efficiently Removed (KAISER), ed era stata proposta per sopperire ad alcune limitazioni del Kernel Address Space Layout Randomization (KASLR), introdotto a sua volta per impedire lo sfruttamento di alcuni tipi di vulnerabilità del Kernel.

D'altronde altre soluzioni richiedono modifiche all'Hardware, anche tramite microcodice di aggiornamento delle CPU, ad esempio per modificare il flusso di esecuzione delle istruzioni out-of-order o per separare più efficacemente l'area di memoria utilizzata dal Kernel da quella utilizzata dagli altri processi.

1.1.3.2 SPECTRE

Se Meltdown permette di accedere a tutta la memoria del Sistema Operativo, è relativamente semplice da implementare ma al contempo sono presenti delle misure di mitigazioni efficaci, al contrario Spectre è molto complesso da implementare, non dà accesso a dati privilegiati del Sistema Operativo ma di altri processi e utenti sul sistema. Inoltre Spectre è implementabile su praticamente tutte le CPU moderne ed è difficile introdurre contromisure sia Software che Hardware senza modificare significativamente l'architettura delle moderne CPU.

Come Meltdown, Spectre si basa sull'abuso dell'esecuzione out-of-order delle istruzioni, ma in questo caso si basa specificatamente sul processo di esecuzione speculativa di alcune istruzioni. Ad alto livello, possiamo descrivere un attacco Spectre come segue:

1. l'attaccante e la vittima sono due utenti dello stesso sistema in grado di eseguire programmi sulla stessa CPU

2. l'attaccante conosce un programma od una libreria utilizzata dalla vittima, ad esempio una libreria che contiene delle pro-

cedure crittografiche, e vuole ottenere le chiavi crittografiche segrete utilizzate dalla vittima

3. l'attaccante individua nel codice utilizzato dalla vittima un Branch che la CPU esegue tipicamente in maniera speculativa, ovvero in anticipo rispetto alla disponibilità dei valori per la valutazione della condizione di scelta; tipicamente la CPU sceglie di eseguire il ramo di codice che di recente è stato eseguito più frequentemente

4. l'attaccante scrive un breve programma il cui codice esegue frequentemente le stesse istruzioni macchina del Branch della libreria sotto attacco, ma con dati diversi in modo da indurre la CPU a ritenere più probabile l'esecuzione da parte della vittima sul Branch errato; contemporaneamente l'attaccante svuota alcune aree della Cache in modo che l'esecuzione del Branch errato porti al caricamento in Cache solo dei dati segreti

5. quando la CPU esegue il programma della vittima, basandosi sull'esperienza dell'esecuzione del codice malevolo, segue il Branch errato e carica in Cache i dati segreti, quali ad esempio delle chiavi crittografiche

6. a questo punto l'attaccante utilizza un attacco alla Cache quale Flush+Reload o Evict+Reload come Covert Channel per leggere i dati segreti dalla Cache.

L'implementazione di un attacco Spectre è molto complessa e difficile, anche se vi sono esempi persino in Javascript eseguiti

all'interno di un browser Web da remoto [Rif. 16].

Non di meno è difficile trovare delle contromisure a questo attacco: l'esecuzione di un Branch errato di un processo non dovrebbe portare ad alcuna conseguenza: infatti non vi è alcuna conseguenza per l'esecuzione del processo vittima, i dati calcolati nel Branch errato sono scartati nell'esecuzione finale, ma inevitabilmente i dati del calcolo errato compaiono in Cache per un breve periodo di tempo. Il problema è come evitare che un attaccante forzi una CPU a seguire un Branch errato in modo da poi poter accedere, anche indirettamente, a dati presenti in Cache. A differenza di Meltdown, in un attacco Spectre non sono coinvolte aree di memoria privilegiate, per cui la CPU e il Sistema Operativo non hanno informazioni Hardware da poter eventualmente sfruttare per impedire l'attacco.

Quindi la soluzione definitiva di Spectre richiede un ridisegno delle logiche Hardware di esecuzione out-of-order e speculative nelle CPU di prossime generazioni, anche se per alcune varianti di Spectre sono state identificate delle contromisure sia a livello di microcodice di aggiornamento delle CPU che per i Sistemi Operativi.

1.1.3.3 L'HARDWARE E LA SICUREZZA IT

Come descritto inizialmente, il disegno architetturale alla base delle moderne CPU risale agli anni '60 e negli ultimi anni sono state individuate delle vulnerabilità insite nel disegno stesso che

minano alcuni dei principali assunti per la sicurezza nella gestione dell'elaborazione. Come spesso accade in questi casi, non è la singola funzionalità od il singolo blocco del disegno ad essere problematico, ma una combinazione non prevista di attività possono portare alla violazione dei requisiti di sicurezza assunti.

Ad oggi le conseguenze di Row Hammer, degli attacchi alla Cache, di Meltdown e Spectre sono più teoriche che pratiche, in quanto non sono stati realmente rilevati gravi incidenti ove queste vulnerabilità fossero la causa principale della violazione della sicurezza. Ancora oggi lo sfruttamento delle vulnerabilità dei Sistemi Operativi, delle applicazioni e delle configurazioni dei sistemi, è molto più semplice e produttivo per un attaccante.

D'altra parte non possiamo minimizzare il fatto che alcuni degli assunti principali per la sicurezza dei sistemi IT, assunti basati sul disegno e le funzionalità di base dell'Hardware, non possono più essere considerati completamente veritieri. Visto anche che la risoluzione di una vulnerabilità di disegno dell'Hardware richiede tempi lunghi e alti costi (dovendo ridisegnare l'Hardware, produrlo, adattare i Sistemi Operativi al nuovo Hardware e sostituire l'Hardware esistente) è necessario iniziare sin d'ora lo studio di nuovi disegni architetturali per l'Hardware IT o delle modifiche necessarie all'attuale, e di pianificare nell'evoluzione dei sistemi IT non solo gli aspetti di prestazioni, qualità e affidabilità ma anche nuove e più estese misure di sicurezza.

1.1.4 RIFERIMENTI BIBLIOGRAFICI

Rif. 1: Alcuni riferimenti su "Spectre and Meltdown":

- sito ufficiale: https://meltdownattack.com/
- Wikipedia: https://it.wikipedia.org/wiki/Spectre_(vulnerabilità_di_sicurezza) https://it.wikipedia.org/wiki/Meltdown_(vulnerabilità_di_sicurezza)
- Schneier on Security: https://www.schneier.com/blog/archives/2018/01/spectre_and_mel_1.html

Rif. 2: The Orange Book, DoDD 5200.28-STD, pubblicato nel 1983 e aggiornato 1985 dal National Computer Security Center (NCSC), parte della National Security Agency (NSA), http://csrc.nist.gov/publications/secpubs/rainbow/std001.txt

Rif. 3: "Common Criteria for Information Technology Security Evaluation" (anche noto come "Common Criteria" o CC), ISO/IEC 15408, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50341

Rif. 4: A.S. Tanenbaum, "Modern Operating Systems", Prentice Hall

Rif. 5: National Security Agency (NSA), "TEMPEST: A Signal Problem", 1972, declassificato 2007-09-27, https://www.nsa.gov/news-features/declassified-documents/cryptologic-spectrum/assets/files/tempest.pdf

Rif. 6: si veda ad esempio J.M. Atkinson, "Tempest 101", Granite Island Group, http://www.tscm.com/TSCM101tempest.html

Rif. 7: si veda ad esempio M.G. Kuhn, "Electromagnetic eavesdropping risks of flat-panel displays", International Workshop on Privacy Enhancing Technologies, 88-107, 2004, https://www.cl.cam.ac.uk/~mgk25/pet2004-fpd-slides.pdf

Rif. 8: D. Genkin, M. Pattani, R. Schuster, E. Tromer, "Synesthesia: Detecting Screen Content via Remote Acoustic Side Channels", preprint 2018/08/21, https://www.cs.tau.ac.il/~tromer/synesthesia/

Rif. 9: P.C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". CRYPTO 1996

Rif. 10: G. Camurati et al., "Screaming Channels: when electromagnetic side channels meet radio transceivers", preprint 2018, http://s3.eurecom.fr/tools/screaming_channels/

Rif. 11: Y. Kim, R. Daly, J. Kim, C. Fallin, J.H. Lee, D. Lee, C. Wilkerson, K. Lai, O. Mutlu (June 24, 2014). "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors" 2014/06/24, IEEE, https://users.ece.cmu.edu/~yoonguk/papers/kim-isca14.pdf

Rif. 12:

- Rowhammer.js: https://arxiv.org/abs/1507.06955
- Nethammer: https://arxiv.org/abs/1805.04956
- Another Flip: https://arxiv.org/abs/1710.00551
- Flip Feng Shui: https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_razavi.pdf

- Glitch: https://www.vusec.net/projects/glitch/
- RAMpage: https://vvdveen.com/publications/dimva2018.pdf

Rif. 13:

- Prime+Probe: http://palms.ee.princeton.edu/system/files/SP_vfinal.pdf
- Flush+Reload: https://eprint.iacr.org/2013/448.pdf
- Flush+Flush: https://gruss.cc/files/flushflush.pdf
- Cache attack in Javascript: https://arxiv.org/pdf/1502.07373.pdf

Rif. 14: C. Maurice et al., "Hello from the Other Side: SSH over. Robust Cache Covert Channels in the Cloud", https://gruss.cc/files/hello.pdf

Rif. 15: M. Lipp et al., "Meltdown", https://arxiv.org/abs/1801.01207v1

Rif. 16: M. Schwarz et al., "Netspectre: Read Arbitrary Memory over Network", https://gruss.cc/files/netspectre.pdf

Rif. 17: "Foreshadow: Breaking the Virtual Memory Abstraction with Transient Out-of-Order Execution", https://foreshadowattack.eu/

Rif. 18: A distanza di 4 anni continuano ad essere scoperte nuove vulnerabilità Hardware, si vedano solo come esempio

- "PACMAN Attacking ARM Pointer Authentication with Speculative Execution", giugno 2022, https://pacmanattack.com/
- "Hertzbleed Attack", giugno 2022, https://www.hertzbleed.com/

1.2 Sicurezza Hardware e Confidential Computing

Negli ultimi anni si è posta molta attenzione agli aspetti di sicurezza informatica basati su componenti Hardware: progetti sono in corso, consorzi sono stati creati (si veda ad esempio [Rif. 1]), annunci e nuovi prodotti sono stati presentati. Proviamo quindi a fare una rapida panoramica di alcuni aspetti della sicurezza sicurezza informatica strettamente basati su componenti Hardware che potrebbero portare ad interessanti sviluppi nel prossimo futuro. Per fare questo però, è utile partire ricordando alcuni dei principali concetti del ruolo dell'Hardware nella sicurezza informatica.

1.2.1 SICUREZZA E FIDUCIA (O TRUST)

Dobbiamo sempre ricordarci che la sicurezza informatica si basa comunque sempre su una catena di fiducia (o all'Inglese, Chain of Trust) che ci coinvolge tutti in vari modi.

Quando utilizziamo un software, dai Sistemi Operativi ad applicazioni grandi e piccole, è ovvio, o dovrebbe esserlo, che esplicitamente o implicitamente ognuno di noi si fida del produttore del software, anche se non sempre allo stesso modo. Per questo anche se ci fidiamo, utilizziamo anti-virus, verifichiamo (o dovremmo verificare) le notizie pubbliche su eventuali incidenti di sicurezza delle applicazioni, facciamo eseguire (o dovremmo fa eseguire) Vulnerability Assessment, Penetration Test, Source Code Analysis,

Black Box Test ecc. per verificare l'assenza di lacune di sicurezza dai prodotti. Nessuno di noi dovrebbe procedere ad installare un prodotto software per il quale non si ha nessuna fiducia sulla sicurezza, sia basata su evidenze tecniche sia sulla storia ed il buon nome del produttore.

In genere ci fidiamo di più dell'Hardware, ed a ragione, tipicamente perché vi sono meno incidenti di sicurezza informatica dovuti a problemi Hardware, ma anche perché spesso siamo meno consapevoli del suo ruolo cruciale nella sicurezza informatica.

Ma dove comincia la catena della fiducia? Il primo punto di vista è che ci dobbiamo fidare di tutti i produttori, sia Software che Hardware. Il secondo punto di vista è di guardare a come la sicurezza è implementata nei sistemi informatici, ed in questo caso dovrebbe essere ovvio che si parte dall'Hardware e dal suo Firmware.

1.2.2 PROTEZIONE HARDWARE DI BASE E CATENA DELLA FIDUCIA

Che l'Hardware sia il punto di partenza (o Root of Trust) della catena della fiducia della sicurezza operativa è fatto ben noto e risale agli albori dell'informatica. Come descritto in §1.1.1, l'approccio attuale alla sicurezza Hardware ebbe le prime implementazioni negli anni '60 e storicamente si fa riferimento a Multics (1964), il papà di Unix. All'interno delle CPU ogni programma è eseguito a un livello di protezione (Protection Ring) controllato dall'Hardware e che ha un set predefinito di privilegi, ovvero nel quale i program-

mi possono eseguire solo un certo insieme di istruzioni macchina. Tipicamente i programmi sono eseguiti nel livello meno privilegiato nel quale non hanno accesso diretto ad alcuna risorsa, dai file alle periferiche ecc. Nel livello più privilegiato è eseguito invece il Kernel del Sistema Operativo. Ad esempio, quando un programma deve leggere o scrivere dei dati, prepara i dati da leggere o scrivere in apposite locazioni di memoria e poi attiva un particolare bit Hardware (detto Hardware Gate) tramite una chiamata detta System Call. La CPU a questo punto interrompe l'esecuzione del programma ed attiva il Kernel del Sistema Operativo che legge la richiesta del programma, verifica che sia lecita, ovvero ad esempio che non voglia leggere o scrivere dati di altri utenti o su dispositivi ove l'utente o il programma non ha il permesso di accesso, e procede a leggere o scrivere i dati per conto del programma.

Qualsiasi funzionalità di sicurezza informatica nell'esecuzione degli eseguibili è basata su questa architettura Hardware che fa si che solo il Kernel del Sistema Operativo abbia diretto accesso alle periferiche e all'Hardware in generale. Il Kernel del Sistema Operativo ha quindi il compito di implementare le politiche di sicurezza operative.

Il Kernel di un Sistema Operativo è caricato nella CPU all'avvio del sistema e tipicamente non dovrebbe essere modificato durante l'esecuzione[13]. Ma come possiamo essere sicuri che il Sistema

13) Per limitarne le dimensioni, i Kernel sono tipicamente modulari ed i moduli sono caricati ed eseguiti a run-time solo al momento del reale utilizzo.

Operativo non sia stato manomesso al momento dell'avvio del sistema o durante la sua esecuzione?

Entrambi gli scenari sono possibili: il primo ad esempio tramite un aggiornamento fraudolento del Sistema Operativo anche tramite un accesso fisico alla macchina, il secondo ad esempio sfruttando una vulnerabilità del Sistema Operativo stesso durante la sua esecuzione.

Per garantire l'integrità del Sistema Operativo è necessario risalire al momento dell'avvio del sistema quando viene eseguito il Firmware presente in Hardware e poi i programmi BIOS/UEFI necessari all'avvio del Sistema Operativo.

La catena della fiducia deve quindi risalire all'Hardware al momento dell'avvio del sistema: ci fidiamo del produttore dell'Hardware che a sua volta si fida del Firmware, del BIOS/UEFI, del Sistema Operativo e così via. Ad ogni passo è necessario verificare che il componente successivo sia autentico, ovvero di un produttore di cui ci si fida, e integro, ovvero non modificato. E le verifiche devono essere fatte periodicamente per individuare eventuali modifiche non autorizzate a run-time. Il processo è implementato utilizzando tecniche crittografiche calcolando il Hash degli eseguibili (integrità) e apponendo firme digitali (autenticità) tramite moduli crittografici Hardware dedicati o inclusi nelle CPU.

Questo processo è tipicamente chiamato Secure Boot ed è ormai implementato nella maggior parte dei dispositivi informatici e supportato da quasi tutti i Sistemi Operativi.

1.2.3 INTEGRITÀ, AUTENTICITÀ E CONFIDENZIALITÀ

Prima di procedere è necessario distinguere due problematiche che nella pratica sono interconnesse ma hanno scopi e finalità ben diversi, come riassunto in Fig. 1.2.1.

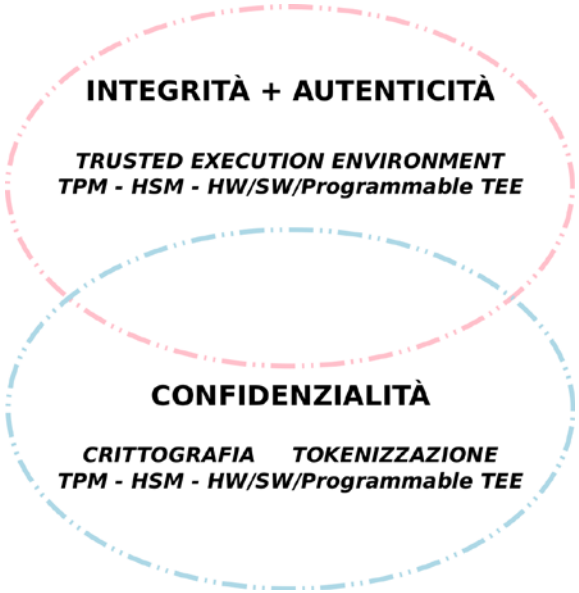

Figura 1.2.1: Approcci e alcuni esempi di tecnologie di sicurezza delle elaborazioni

La prima, quella appena descritta, ha come scopo di garantire che gli strumenti Hardware e Software che utilizziamo per le elaborazioni informatiche siano quelli in cui riponiamo la nostra fiducia e che vogliamo utilizzare. Inoltre vogliamo essere immediatamente informati nel caso questi siano modificati in maniera non autorizzata ad esempio da qualche malware. In generale le

soluzioni a questi requisiti vanno sotto il nome di Trusted Execution Environment (TEE). Ovviamente in questo caso le proprietà che ci interessano sono l'Integrità e l'Autenticità di Hardware e Software.

La seconda, che di recente sta assumendo sempre maggior interesse, ha come scopo di garantire la Confidenzialità delle elaborazioni anche nei confronti degli amministratori del sistema e del Sistema Operativo stesso e non solo degli altri utenti non amministrativi. Questa esigenza nasce anche dall'utilizzo da parte di più utenti, completamente estranei, dello stesso sistema Hardware/Software in particolare negli ambienti Cloud.

Questo aspetto è usualmente chiamato Confidential Computing (Rif. [1-5]). Prima però di discutere le relazioni tra Confidential Computing, Secure Boot e Trusted Execution Environment, è necessario approfondire le principali tematiche del Confidential Computing.

1.2.4 CONFIDENTIAL COMPUTING

Consideriamo subito il caso più complicato ma sicuramente di maggior interesse: l'utilizzo di servizi Cloud erogati in modalità SaaS, PaaS o IaaS. Lo scenario ben noto è quello di una piattaforma Hardware e Software gestita dal fornitore ed utilizzata in maniera condivisa e self-service dai clienti. I clienti condividono le risorse Hardware e Software, inclusi Sistemi Operativi, Middleware e applicazioni. D'altra parte ogni cliente richiede la confidenzialità dei propri dati, sicuramente rispetto agli altri clienti e in molti casi

anche rispetto al fornitore Cloud stesso.

La confidenzialità può essere ottenuta principalmente utilizzando due approcci:

1. sostituendo i dati sensibili con dati non significativi tramite processi quali la Tokenizzazione, il Data Masking o la pseudo-anonimizzazione;

2. cifrando tutti i dati o soli i dati sensibili.

In entrambi i casi, se solo parte dei dati sono protetti, si pone il problema di garantire che non sia possibile estrarre informazioni sensibili, via inferenza, dai dati non protetti. Il problema dell'inferenza e dei canali paralleli (side channel) è un problema reale e spesso di difficile soluzione, se non quella di ridurre al massimo la quantità di dati non protetti e rendere assolutamente inintelligibili tutti gli altri.

I processi di mascheramento o sostituzione dei dati sensibili sono particolarmente esposti ai problemi d'inferenza e analisi. Inoltre sono di norma completamente in carico all'utente, che deve preparare correttamente i dati prima d'inviarli al fornitore Cloud. Infine non è possibile adottare queste soluzioni ad esempio quando le elaborazioni devono essere fatte proprio e direttamente sui dati sensibili.

Sembrerebbe quindi che la cifratura di tutti i dati possa essere l'approccio vincente e più semplice per garantire la loro confidenzialità, ma vanno considerati i tre possibili stati dei dati (si veda anche la Tabella 1):

1. la cifratura dei dati in transito, ovvero quando trasmessi sulle reti di telecomunicazione
2. la cifratura dei dati a riposo, ovvero quando archiviati su un Filesystem, Storage, Database ecc.
3. La cifratura dei dati in uso, ovvero durante l'elaborazione.

In transito	Cifratura comunicazioni		
A riposo	Cifratura disco / FileSystem per furto HW o accesso fisico	Cifratura dati (es. Tabella / Colonna in DB) con chiave dell'utente, Confidential Computing	Tokenizzazione, Data Masking, Pseudonimizzazione
In uso	Confidential Computing	Crittografia omomorfa	Tokenizzazione, Data Masking, Pseudonimizzazione

Tabella 1: alcune tecniche per la confidenzialità nei tre possibili stati dei dati

La cifratura di tutti i dati in transito è ormai la norma e non dovrebbe costituire un problema per nessuno.

La cifratura dei dati a riposo è ormai abbastanza comune ma è necessario fare alcune precisazioni. La cifratura di interi dischi o FileSystem garantisce la confidenzialità dei dati sicuramente rispetto a scenari quali l'accesso fisico od il furto del supporto Hardware, ma non è utile per garantire la confidenzialità rispetto gli accessi degli amministratori di sistema o quando l'applicazione

è condivisa da più utenti e utilizza lo stesso Storage per tutti gli utenti. Al giorno d'oggi questa però è la situazione più comune.

Per garantire la confidenzialità dei dati a riposo anche rispetto agli amministratori dei sistemi Cloud ed agli altri utenti della stessa applicazione, è necessario che i dati siano cifrati con chiavi crittografiche gestite e note solo all'utente. Un caso ormai comune è fornito dai servizi di Storage Web in Cloud, ovvero i servizi di tipo "Cloud Drive" accessibili tipicamente tramite Browser Web e App dedicata. In questi servizi, i dati sono cifrati dal Browser dell'utente sul proprio dispositivo informatico prima dell'invio al servizio di Storage in maniera trasparente all'utente. Lo stesso avviene per servizi Cloud di Backup e di puro Storage, ove i dati sono cifrati / decifrati da una applicazione sui sistemi interni dell'utente. Questo da un lato garantisce la confidenzialità dei dati ma dall'altro rende impossibile qualunque elaborazione dei dati sui sistemi in Cloud visto che la chiave di cifratura è loro ignota.

Ma come poter garantire al contempo la confidenzialità dei dati a riposo, ovvero sullo Storage dei servizi Cloud, e la possibilità per le applicazioni Cloud di elaborare i dati?

Questo è il principale problema che il Confidential Computing si propone di risolvere. La soluzione è basata sull'utilizzo di moduli Hardware crittografici da affiancare od integrare in tutti i processori (si veda ad esempio [Rif. 6]), che permettano solo all'utente padrone dei dati di cifrarli e decifrarli sui server Cloud. Questi moduli sono chiamati Hardware-Based Trusted Execution Environment (o HW-TEE). Proveremo più avanti a dare una descrizione ad alto livello di come questo possa funzionare, il punto principale

è che le chiavi segrete e le elaborazioni sensibili, quali la de-cifratura, sono svolte esclusivamente in Hardware dedicato e non accessibile, in modo da impedire a chiunque non autorizzato, inclusi gli amministratori dei sistemi, di accedervi. Da questo punto di vista gli HW-TEE sono da considerare come un'evoluzione degli Hardware Security Module (HSM) e dei Trusted Platform Module (TPM): i primi sono sistemi HW per gestire in sicurezza le chiavi e le operazioni crittografiche, i secondi sono micro-chip presenti ormai su quasi tutti i computer anche personali, che gestiscono in sicurezza chiavi crittografiche (si veda anche la Fig. 1.2.2).

	HW TEE	Homomorphic Encryption	Secure element e.g., TPM
Data integrity	Y	Y (subject to code integrity)	Keys only
Data confidentiality	Y	Y	Keys only
Code integrity	Y	No	Y
Code confidentiality	Y (may require work)	No	Y
Authenticated Launch	Varies	No	No
Programmability	Y	Partial ("circuits")	No
Attestability	Y	No	Y
Recoverability	Y	No	Y

Figura 1.2.2: Caratteristiche dei tre principali approcci crittografici per la confidenzialità [Fonte Rif. 4]

Un esempio semplice in ambito Cloud IaaS può essere utile a chiarire questo scenario. Si supponga di creare in un ambiente Cloud una macchina virtuale con un suo storage virtuale dedicato. Si cifra però lo storage con una chiave di cifratura nota solo all'utente che utilizza la macchina virtuale. La maniera tradizionale e poco funzionale di fare ciò, è di inserire manualmente la chiave di cifratura dello storage al momento in cui il relativo FileSystem viene montato sulla macchina virtuale. Questo ovviamente richiede una gestione manuale delle chiavi, e la presenza

di una persona che deve digitarle al momento dell'avvio (e riavvio) della macchina virtuale. L'approccio corrente è invece totalmente automatizzato: alla macchina virtuale sono associate delle chiavi crittografiche che permettono di identificarla esattamente e, tramite queste chiavi, solo la macchina virtuale può accedere alla chiave di cifratura del FileSystem. Le chiavi crittografiche possono essere gestite in sistemi Hardware (TPM, HSM, HW-TEE) ma più comunemente sono gestite tramite applicazioni dedicate di Key Management, sistemi sui cui torneremo più avanti quando descriveremo brevemente un altro approccio detto comunemente Bring Your Own Key (BYOK). Il risultato finale è che solo gli utenti della macchina virtuale possono accedere ai dati presenti sullo Storage virtuale ad essa collegato, gli utenti delle altre macchine virtuali e gli amministratori del servizio Cloud non hanno alcun accesso in chiaro ai dati sullo storage.

Riassumendo, rispetto alle due situazioni descritte inizialmente, è possibile creare una configurazione intermedia in cui i dati non sono cifrati dallo Storage stesso né dall'utente sui propri sistemi remoti, ma a livello utente sui sistemi o applicazioni in Cloud con chiavi legate all'utente o all'istanza dell'applicazione (o macchina virtuale). In questo modo la cifratura dei dati garantisce la confidenzialità degli stessi a riposo a livello utente, ma permette di decifrarli e poterli elaborare anche sui sistemi remoti.

Questo approccio può essere ripetuto (anche se non ancora in tutti i casi) per la protezione dei dati a riposo in database o direttamente in istanze di applicazioni.

1.2.5 CONFIDENZIALITÀ DEI DATI "IN USO" – CIFRATURA RAM

Purtroppo la cifratura dei dati a riposo è il problema semplice e la cui soluzione sta diventando sempre più standard anche nei servizi Cloud. Il problema difficile e non ancora risolto è quello di proteggere i dati "in uso".

Prima di tutto dobbiamo chiarire cosa si intende per dati "in uso": secondo l'accezione generale sono i dati quando non in trasferimento su reti dati e non archiviati in memorie permanenti. I due principali punti di "uso" dei dati sono ovviamente la CPU, che li elabora, e la RAM, che li memorizza temporaneamente. I dati "in uso" sono presenti poi in tutti i registri, cache, bus eccetera che compongono l'architettura hardware degli elaboratori, ed in particolare della scheda madre.

Nella CPU e nei circuiti Hardware che ne formano l'architettura, i dati devono essere in chiaro, ovvero non cifrati, altrimenti nessun tipo di elaborazione è possibile a meno di utilizzare la Crittografia Omomorfica che sarà brevemente considerata più avanti. Inoltre, a meno di attacchi di inferenza ("side channel") quali ad esempio Spectre e Meltdown o vulnerabilità Hardware, la CPU ed i circuiti Hardware sono i punti ove i dati sono a minor pericolo per la confidenzialità in quanto protetti dall'Hardware stesso. Analogo discorso non vale per la RAM in quanto un utente o programma con privilegi di amministratore di sistema (o amministratore di Host per i sistemi virtualizzati), può leggere tutti i dati in essa contenuti.

AMD propose nel 2016 [Rif. 7] la realizzazione di moduli RAM nei

quali i dati sono cifrati. Questa tecnologia è ormai disponibile da parte di AMD con il nome di "AMD Memory Guard" (inizialmente si chiamava "Secure Memory Encryption") e Intel ha annunciato una simile soluzione con il nome "Total Memory Encryption" (TME) [Rif. 8]. L'implementazione della cifratura dei dati in RAM è, da un punto di vista logico, abbastanza semplice, come riassunto in Figura 1.2.3.

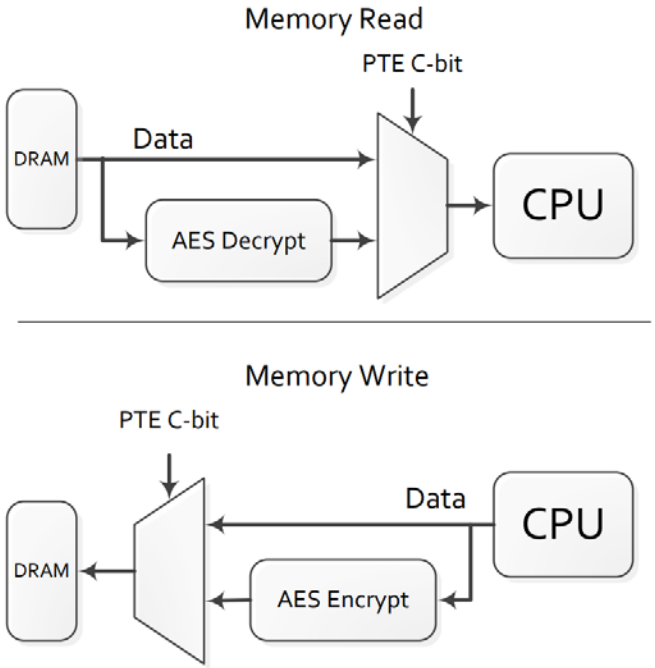

Figura 1.2.3: Schema logico di AMD "Secure Memory Encryption" [Fonte Rif. 7]

L'idea di base è di inserire dei circuiti dedicati tra la CPU e la RAM che cifrano i dati con AES (128bit) quando questi sono scritti in memoria dalla CPU, e li decifrano quando sono trasferiti dalla memoria alla CPU. In questo modo i dati sono sempre cifrati nella RAM e sempre in chiaro nella CPU.

Nella tabella "Page Table Entry" (PTE) che indirizza i dati nella memoria, un bit ("C-bit") è dedicato a indicare se nella locazione di memoria i dati sono cifrati o in chiaro. Quindi a seconda del valore del "C-bit" nel trasferimento tra CPU e RAM i dati vengono (de-) cifrati o meno. Le chiavi di cifratura sono generate da un generatore di numeri pseudo-casuali Hardware e mantenute in registri Hardware dedicati, temporanei e non accessibili se non al Chip di cifratura.

Le chiavi di cifratura sono tipicamente create ad ogni avvio del sistema. AMD-SME permette di cifrare con una singola chiave segreta tutta la RAM o solo i dati gestiti da un particolare processo, ad esempio l'Hypervisor e quindi tutte le macchine virtuali. In questo secondo caso, l'amministratore di sistema dell'Host di virtualizzazione non ha accesso ai dati delle macchine virtuali in RAM.

Inoltre in modalità "Secure Encrypted Virtualization" (AMD-SEV) viene creata una chiave di cifratura dedicata per ogni macchina virtuale. Questo permette di proteggere i dati in RAM di una macchina virtuale non solo rispetto all'Host ma anche rispetto a tutte le altre macchine virtuali.

1.2.6 CONFIDENZIALITÀ DEI DATI "IN USO" - ENCLAVI SICURE

Cifrare i dati in RAM mette in sicurezza un solo scenario di attacco (lettura dei dati in RAM da parte di un utente privilegiato) e non risolve la problematica generale di eseguire elaborazioni di dati sensibili in sicurezza anche su sistemi terzi come gli ambienti Cloud.

Il passo successivo, che ci porta al Hardware-Based Trusted Execution Environment (HW-TEE), è quello della realizzazione all'interno della CPU di una "Enclave Sicura", di cui l'esempio più noto sono le Software Guard Extensions (SGX) di Intel [Rif. 10] [14].

Piuttosto che cercare di descrivere una tecnologia specifica o l'approccio tecnologico che è in costante e rapida evoluzione, cercheremo di descrivere ad alto livello le finalità di queste ricerche e sviluppi, e quello che ci possiamo aspettare diventi realtà nel prossimo futuro.

L'idea di base, e la nostra descrizione è per forza approssimativa, è di creare all'interno della CPU una zona Hardware, appunto una "Enclave Sicura", nella quale possano essere elaborati in maniera sicura dati sensibili anche su sistemi di terzi.

Alla base di tutto è il fatto che questa elaborazione sicura è protetta dall'Hardware, ovvero utilizza specifiche caratteristiche dell'Hardware per proteggere i dati; per la nostra discussione non

14) Alcune implementazioni di enclavi sono riportate in [Rif. 14].

è importante che quanto sia protetto sia una macchina virtuale, un container o direttamente un'applicazione, né come la protezione sia implementata in Hardware.

Indicativamente una Enclave Sicura dovrebbe offrire:

- la creazione e la gestione di chiavi crittografiche
- la cifratura/decifrazione di dati con le chiavi prodotte internamente o ricevute dall'esterno
- l'esecuzione di programmi caricati nell'enclave, anche se non tutte le istruzioni macchina della CPU possono essere disponibili
- che nessun altro programma esterno all'enclave, incluso il sistema operativo, possa accedere ai dati o alle istruzioni eseguite all'interno dell'enclave stessa
- istruzioni macchina che permettono la creazione / utilizzo di un'enclave e lo scambio di dati e istruzioni con l'enclave da parte di un processo utilizzatore dell'enclave stessa.

Una modalità semplificata, e vedremo che è già ben complessa, di utilizzare questo scenario potrebbe essere la seguente. Si assuma di avere a disposizione un sistema presso un fornitore Cloud che mette a disposizione una Enclave Sicura di cui ovviamente ci si fida.

Sul sistema in Cloud sono caricati dal sistema on-premises alcuni dati cifrati, vedremo più avanti come, ad esempio in un Database.

Sul sistema in Cloud è anche presente un'applicazione che è in grado di elaborare normalmente i dati non cifrati presenti nel database, ma non è in grado di elaborare i dati cifrati.

Quando è necessario elaborare dei dati cifrati, l'applicazione è in grado di inviarli all'Enclave Sicura e di caricare nell'Enclave Sicura anche una piccola componente applicativa scritta appositamente per essere eseguita nell'Enclave Sicura e che può elaborare i dati una volta decifrati.

Una descrizione come sempre approssimata ed ad alto livello di questo processo, è la seguente:

- i sistemi del cliente e l'Enclave Sicura sul sistema Cloud si scambiano le relative chiavi crittografiche pubbliche o dei certificati digitali che le contengono; in questo modo i sistemi del cliente possono cifrare dei dati che possono essere decifrati solo all'interno dell'Enclave Sicura, e viceversa l'Enclave può cifrare dei dati che possono essere decifrati solo sui sistemi del cliente;
- l'applicazione in esecuzione sul sistema Cloud è in grado di comunicare con l'Enclave Sicura sia scambiando dati cifrati, sia inviando in esecuzione del codice con una firma digitale riconosciuta dall'enclave sicura;
- quando l'applicazione in esecuzione sul sistema Cloud deve elaborare dei dati sensibili e cifrati, invia all'Enclave Sicura il codice (firmato) necessario per l'elaborazione ed i dati cifrati;
- l'Enclave Sicura esegue l'elaborazione come segue:

- verifica la firma sul codice da eseguire;
- decifra i dati con la propria chiave privata[15];
- esegue l'elaborazione;
- cifra i dati risultato dell'elaborazione con la chiave pubblica del cliente;
- invia i dati cifrati all'applicazione in esecuzione sul sistema Cloud.

Sin da questa prima descrizione, è chiaro che questo non è un processo molto semplice, in primo luogo perché richiede che le applicazioni siano adattate a poter eseguire proprie componenti (che devono essere scritte appositamente) all'interno di una Enclave Sicura quando devono agire su dati sensibili e cifrati per l'Enclave stessa. In secondo luogo è necessario gestire le chiavi crittografiche e la cifratura dei dati tra i sistemi del cliente e l'ambiente Cloud.

Tutto questo però ci riporta a considerare il concetto di "fiducia".

15) Tipicamente con la chiave asimmetrica (componente pubblica), viene cifrata una chiave simmetrica (ad esempio per AES) utilizzata per cifrare i dati; le chiavi asimmetriche non vengono normalmente utilizzate per cifrare i dati direttamente.

1.2.7 LA CATENA DELLA FIDUCIA DISTRIBUITA

Come appena descritto, oltre al necessario sviluppo applicativo che permette alle applicazioni in esecuzione su sistemi con Enclave Sicura di comunicare ed eseguire elaborazione in essi, è necessario gestire le chiavi crittografiche con cui identificare sia i sistemi del cliente che l'Enclave Sicura e cifrare i dati. Questo risulta, di nuovo, abbastanza complesso.

Consideriamo il provider Cloud: sinora abbiamo ragionato come se l'applicazione in Cloud fosse eseguita sempre e solo su di un specifico server Hardware la cui CPU ha una Enclave Sicura, ma questo al giorno d'oggi sicuramente non è vero. La situazione più comune è che l'applicazione è eseguita in una o più macchine virtuali che possono migrare tra macchine fisiche anche in sale macchine e località diverse.

Come facciamo a gestire le relative chiavi crittografiche o anche solo ad essere sicuri che l'applicazione sia in esecuzione su una CPU con una Enclave Sicura reale e non una versione virtuale che permette ad un attaccante di intercettare tutti i dati del cliente?

Per rispondere a questa domanda dobbiamo ripartire dalla catena della fiducia: per prima cosa il cliente si deve fidare del fornitore Cloud che deve assicurare di avere un gruppo di server fisici ognuno con la propria Enclave Sicura. Nel caso di una Enclave Sicura SGX è richiesto che [Rif. 15]:

1. L'Enclave è in esecuzione su un vero processore Intel;
2. La piattaforma è aggiornata al livello di sicurezza più recente;

3. L'identità dell'Enclave è quella asserita;

4. L'Enclave non è stata manomessa.

Per soddisfare queste richieste è necessario che il Secure Boot (si veda §1.2.2) garantisca l'autenticità ed integrità del sistema Hardware e Software, e fornisca anche le relative versioni di Hardware e Software.

Per gestire tutto ciò è necessario che il provider Cloud abbia un servizio di Remote Attestation [Rif. 2, 3] in grado di raccogliere le attestazioni di Secure Boot di ogni server (firmate digitalmente) contenenti anche l'indicazione delle versioni di Firmware, sistema operativo ecc.

Tramite il servizio di Remote Attestation del provider Cloud, un cliente può quindi identificare i server e le CPU le cui caratteristiche Hardware e Software soddisfano i requisiti di sicurezza di cui necessita.

A questo punto è necessario fare una osservazione che riguarda le macchine virtuali: la configurazione standard dei sistemi di virtualizzazione rende disponibile alle macchine virtuali solo Hardware virtuale e non permette l'accesso all'Hardware reale. Diversamente i Container accedono all'Hardware reale (non virtuale) anche se il sistema operativo limita fortemente le loro possibilità di accesso. Quindi se si usano macchine virtuali è necessario accertarsi che queste possano accedere a Enclavi Sicure fisiche e non virtualizzate (almeno per quanto è qui discusso), mentre questo problema non si pone per i Container.

Ritorniamo ora al problema dell'Attestazione delle Enclavi Sicure considerando come esempio il caso di Container. L'Orchestratore che gestisce i Container (ad esempio Kubernetes o OpenStack) deve pertanto essere a conoscenza dell'Attestazione di ogni server e deve poter gestire una classificazione di ogni Container secondo i requisiti di sicurezza richiesti dal cliente, che permette di selezionare i server e le CPU sui quali il Container può essere eseguito in sicurezza ed in presenza di un Enclave Sicura "attestata".

Ovviamente il provider Cloud può raggruppare i propri server in classi di sicurezza che tengono conto di vari parametri, dall'Attestazione Hardware e Software alla località o regione in cui il server è situato.

Questo sistema permette quindi al cliente di selezionare i server con Enclave Sicura sui quali può essere eseguito il proprio applicativo capace di utilizzarla.

Il successivo passo è quello di scambiare le chiavi crittografiche necessarie a cifrare e decifrare i dati. Ovviamente anche questa procedura deve essere automatizzata, ma deve avere anch'essa una propria radice della catena della fiducia (o Root of Trust) sia presso il provider Cloud che presso il cliente.

Per fare questo dobbiamo ricordarci che ogni Enclave Sicura ha le proprie chiavi crittografiche, e che è necessario gestire anche le chiavi crittografiche pubbliche (o certificati digitali) del cliente. Per questo è in pratica necessario utilizzare un Key Manager, ovvero un'applicazione dedicata alla gestione delle chiavi crittografiche e che può quindi raccogliere e rendere disponibili le chiavi

crittografiche necessarie.

Un Key Manager di norma utilizza un Hardware Security Module (HSM) come Root of Trust, ovvero la chiave privata che identifica il Key Manager è creata e gestita unicamente all'interno dell'HSM, protetta in Hardware. Se anche il cliente utilizza un Key Manager, basta che ogni Key Manager riconosca la chiave crittografica Root of Trust dell'altro per stabilire la fiducia relativa tra i due ambienti.

Il processo è simile idealmente a quello delle Certification Authorities (CA) per la navigazione Web: una volta riconosciuta ed accettata la CA superiore (in questo caso la chiave crittografica Root of Trust dell'altro Key Manager), tutti i certificati digitali e le chiavi crittografiche firmate da questa, anche in cascata, sono accettati.

Infine la comunicazione ed il trasferimento di chiavi crittografiche tra Key Manager, HSM e server è possibile ad esempio utilizzando il "Key Management Interoperability Protocol" (KMIP) [Rif. 16, 17], protocollo sviluppato apposta da OASIS a questo scopo.[16]

In Figura 1.2.4 abbiamo provato a riassumere i principali componenti di questa architettura.

16) Oltre a KMIP vi sono altri protocolli / Programming Interface / API (alcuni proprietari) quali MS-CAPI, MS-EKM, PKCS#11 ecc., che permettono il trasferimento di chiavi crittografiche.

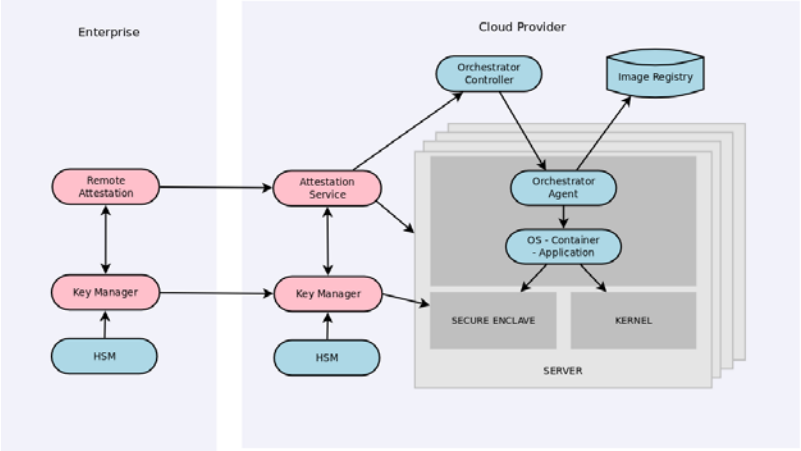

Figura 1.2.4: Principali componenti di una architettura di Confidential Computing

E' opportuno notare che il processo appena descritto di gestione delle chiavi crittografiche del cliente da parte di un provider Cloud, si sta diffondendo e molte aziende lo hanno o stanno adottando sotto il nome di Bring Your Own Key (BYOK). L'idea è che una applicazione o servizio del provider Cloud o presso il provider Cloud, sia essa uno Storage, un Database o dei dati cifrati a livello applicativo, per cifrare e decifrare i dati utilizza delle chiavi crittografiche non create, gestite e archiviate nei sistemi del provider Cloud stesso, ma nei sistemi del cliente. Le chiavi crittografiche sono fornite all'applicazione o servizio solo quando sono necessarie per la cifratura/decifrazione dei dati e direttamente dai sistemi del cliente tramite i Key Manager e protocolli quali KMIP. Le chiavi crittografiche non sono mai archiviate o salvate sui sistemi del provider Cloud ma mantenute in memoria sicura solo per il tempo

strettamente necessario alla cifratura/decifrazione dei dati.

Tornando al Confidential Computing, dovrebbe ormai essere chiaro che la sua implementazione è abbastanza complessa sia a livello tecnico che di gestione. Per raggiungere e garantire la confidenzialità (e integrità) completa della elaborazione di informazioni su di una piattaforma di terzi, dobbiamo comunque partire dalla fiducia di base nell'Hardware e Firmware stesso dei sistemi utilizzati dalla terza parte e dobbiamo dotarci di applicazioni e servizi che permettano di:

- identificare e verificare l'identità e integrità dei sistemi sui cui eseguire le elaborazioni in sicurezza (Attestazione Remota);
- gestire le chiavi di cifratura e la cifratura dei dati in modo che i dati siano disponibili in chiaro solo per le elaborazioni nelle Enclavi Sicure sui sistemi Hardware scelti;
- adattare le applicazioni in esecuzione sulla piattaforma del Provider in modo che possano comunicare ed eseguire le componenti di codice necessarie nelle Enclavi Sicure.

1.2.8 SICUREZZA E HW-TEE

L'architettura che abbiamo appena descritta molto velocemente ed ad alto livello, dovrebbe permettere l'elaborazione di dati sensibili su sistemi di terze parti garantendo la massima confidenzialità ed integrità dei dati. Infatti non vi dovrebbe essere alcun modo in cui la terza parte sui cui sistemi avvengono le elaborazioni, ab-

bia la possibilità di accedere ai dati in chiaro.

La domanda che sorge spontanea è quanto sia sicura in se stessa questa architettura. Vi sono vari aspetti da considerare:

- la complessità dell'architettura stessa ed il coinvolgimento di diversi protocolli può esporre a vulnerabilità nei protocolli stessi o dovute all'interazione tra i protocolli;
- come sempre, è possibile che vi siano vulnerabilità di sicurezza nel codice di Firmware, sistemi operativi ed applicazioni, ad esempio anche quelle utilizzate per la gestione delle chiavi crittografiche;
- similmente è possibile che l'implementazione in Software degli algoritmi crittografici abbia una vulnerabilità mentre, a meno dell'arrivo di un elaboratore quantistico, gli algoritmi crittografici in uso sono oggi considerati sicuri;
- infine non va sottovalutato l'aspetto di possibili vulnerabilità dell'Hardware stesso.

E' opportuno in particolare approfondire l'ultimo punto. Tutta l'architettura che abbiamo descritto si basa sulla presenza di una Enclave Sicura in Hardware. Come ben sappiamo anche solo dall'esperienza con le vulnerabilità Hardware delle CPU quali Meltdown e Spectre, la reale soluzione di vulnerabilità Hardware richiede la sostituzione dell'Hardware stesso, anche se in alcuni casi è possibile mitigare i rischi introducendo dei specifici controlli nel Firmware. Quindi una vulnerabilità nell'Hardware di una Enclave Sicura minerebbe alla base la sicurezza delle informa-

zioni gestite con il Confidential Computing e HW-TEE. Purtroppo le prime generazioni di HW-TEE non sono immuni da vulnerabilità di sicurezza in Hardware (si veda [Rif. 9] per un elenco esemplificativo) anche se alcune di queste sono state mitigate con appositi controlli Firmware.

E' importante notare che le Enclavi Sicure Hardware, quali Intel SGX, disponibili al momento sul mercato, sono state progettate senza considerare come scenario di sicurezza gli attacchi di inferenza "Side Channel", quali gli appena citati di Meltdown e Spectre. Pertanto sono stati dimostrati, in condizioni di laboratorio, alcuni attacchi Side Channel in grado di estrarre dati sensibili da Enclavi Sicure Hardware. Ovviamente future generazioni di Enclavi Sicure risolveranno queste vulnerabilità, ma la soluzione sarà più costosa e complessa rispetto ad un aggiornamento software.

Questo rende ancora più importante il processo di Attestazione Remota dei sistemi, in questo caso per l'Hardware, per cui un cliente potrebbe individuare i singoli modelli di HW-TEE del fornitore su cui eseguire le proprie elaborazioni basandosi sulle vulnerabilità Hardware note ed i possibili rischi alle proprie informazioni che ne potrebbero derivare.

Infine va citato un aspetto, per certi versi controverso, degli HW-TEE, ovvero la possibilità di un loro abuso od uso per fini non leciti. I dati elaborati in un HW-TEE non sono accessibili a nessuno se non il legittimo proprietario, per cui ad esempio anche applicazioni quali anti-virus/malware ecc. non possono verificare la presenza di codice maligno in esecuzione in un HW-TEE (ma potrebbero essere identificati gli effetti sui sistemi esterni all'HW-TEE), e non è

possibile verificare che non vengano eseguite transazioni all'interno di HW-TEE non lecite per motivi legali o contrattuali. Comunque al momento non è del tutto chiaro quanto questi possano essere dei rischi reali.

1.2.9 CRITTOGRAFIA OMOMORFICA

Un approccio complementare o alternativo al Confidential Computing è quello di utilizzare la Crittografia Omomorfica [Rif. 11]. Sin dal 1978 nel contesto dello studio degli algoritmi crittografici asimmetrici quali RSA, Rivest, Adleman e Dertouzos proposero il concetto di Crittografia Omomorfica, ovvero di algoritmi crittografici in qualche modo commutativi rispetto alle operazioni matematiche. Si può descrivere questo concetto con il seguente esempio: dati due numeri M e N ed una operazione O[,] tra di loro (somma, prodotto ecc.), un algoritmo crittografico omomorfico C( ) permette di cifrare i dati in C(M) e C(N) ed ha una operazione P[,] sui dati cifrati tale che

$$P[\, C(M), C(N)\,] = C(\, O[M,N]\,)$$

Ad esempio supponendo che O sia la somma e che P sia il prodotto:[17]

$$C(M) * C(N) = C(\, M + N\,)$$

17) Esistono algoritmi omomorfici che implementano esattamente questa proprietà; invece RSA ha la caratteristica di essere commutativo rispetto alla moltiplicazione, ovvero nel nostro esempio sia O che P sono la moltiplicazione.

In altre parole, se si cifrano i due numeri, si moltiplicano i due numeri cifrati e infine di decifra il risultato del prodotto, si ottiene la somma dei due numeri originali:

$$M + N = C^{-1}(C(M) * C(N))$$

Assumiamo quindi che il nostro fornitore Cloud offra sistemi che supportano il calcolo omomorfico di questo tipo, possiamo quindi cifrare on-premises tutti i dati numerici prima di inviarli all'applicazione Cloud, far eseguire il prodotto dei numeri cifrati dall'applicazione Cloud, scaricare dall'applicazione Cloud il risultato del prodotto dei numeri e decifrarlo on-premises ottenendo la somma dei numeri originali (si veda Fig. 1.2.5). In tutto ciò il sistema Cloud gestisce solo dati cifrati e nessuna chiave di cifratura/decifrazione, e quindi non ha nessuna possibilità di accedere ai dati in chiaro.

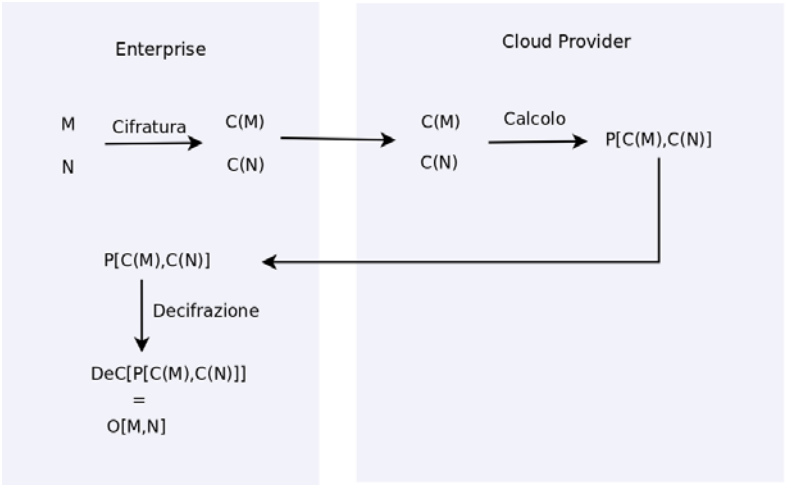

Figura 1.2.5: Flusso logico di un calcolo omomorfico

La crittografia omomorfica è da sempre considerata come la soluzione definitiva per la confidenzialità dei dati in quanto permette al proprietario dei dati di mantenere la confidenzialità degli stessi (solo il proprietario può decifrarli) ed al contempo permette ad altri di elaborare i dati per conto del proprietario senza però venire a conoscenza delle informazioni rappresentate dai dati stessi.

Purtroppo, come al solito, non tutto è così semplice come si desidererebbe (altrimenti utilizzeremmo già da decenni la crittografia omomorfica). Prima di tutto, dall'enunciazione dell'idea di crittografia omomorfica (1978) sino al primo modello teorico reale sono passati ben 31 anni, ovvero sino alla tesi di dottorato del 2009 di Craig Gentry [Rif. 12]. Da allora sono stati identificati ben 4 tipi di "homomorphic encryption scheme" (HE o FHE)[18] , ed i lavori dei crittografi continuano. Non solo, nel frattempo si è anche aggiunta la richiesta di sicurezza di produrre algoritmi resistenti ad eventuali

18) Le quattro generazioni di "Homomorphic Encryption Scheme" sono:

1. Partially Homomorphic Encryption: schemi di cifratura che permettono l'esecuzione di un solo tipo di operazione, ad esempio solo la somma o il prodotto
2. Somewhat Homomorphic Encryption: schemi di cifratura che permettono l'esecuzione di due tipi di operazioni, ad esempio somma e prodotto, ma solo per un numero limitato di composizioni tra le operazioni (detto anche "sottoinsieme di circuiti")
3. Leveled Fully Homomorphic: schemi di cifratura che permettono l'esecuzione di molti tipi di operazioni, ad esempio somma, prodotto ecc., ma solo per un numero limitato di iterazioni (detto anche "circuiti a profondità limitati") senza limitazioni sul numero di composizioni
4. Fully Homomorphic: schemi di cifratura che permettono l'esecuzione di molti tipi di operazioni senza limitazioni sul numero di iterazioni né di composizioni.

futuri arrivi dei Computer Quantistici [Rif. 13] che richiede ulteriori valutazioni e, in parte, lo sviluppo di nuovi algoritmi.

Ad oggi sono già disponibili librerie sperimentali che permettono di utilizzare la crittografia omomorfica ed il consorzio in [Rif. 11] ha proprio lo scopo di definire degli standard in modo che soprattutto i fornitori di servizi Cloud la possano offrire ai propri clienti.

I problemi principali sono dovuti alla lentezza e dimensione degli algoritmi e dei dati, ed alle reali garanzie di sicurezza. Benché gli ultimi algoritmi siano ordini di grandezza più veloci e leggeri dei primi proposti, sono ancora molto lenti rispetto al calcolo svolto sui dati in chiaro e la procedura di cifratura rende i dati molto più grandi, occupando più spazio a riposo, in transito ed in uso.

Gli aspetti di sicurezza riguardano da una parte la sicurezza intrinseca degli algoritmi e dei protocolli stessi, che è ovviamente in corso di valutazione, e dall'altra le modalità di uso e applicative anche per evitare attacchi di inferenza e "side channel".

Uno dei principali problemi degli attuali algoritmi omomorfici è che di norma permettono di effettuare solo alcune operazioni aritmetiche, ad esempio solo la somma e la moltiplicazione. Questo rende difficile, se non impossibile, utilizzarli con applicazioni complesse che gestiscono molti dati, come un DataWareHouse od un modello di Machine Learning. Ovviamente si spera che queste siano solo difficoltà tecniche che in futuro possano essere superate.

Inoltre, finché ci si limita a effettuare somme e prodotti di numeri è difficile immaginare attacchi di inferenza sui dati, ovvero de-

durre delle informazioni direttamente dai dati cifrati. Supponiamo invece come semplice esempio, di avere una tabella con gli stipendi dei dipendenti cifrata con crittografia omomorfica e che sia possibile effettuare un test logico su due dati cifrati che ha come risultato di sapere se il primo dato reale è maggiore del secondo (questo è lo stesso che introdurre un meccanismo per ordinare i dati reali). Visto che i dati sono cifrati, guardando questa operazione da sola (atomicamente) non ne otteniamo quasi alcuna informazione sui dati reali. Ma se prendiamo un gruppo di dati, possiamo incominciare ad estrarne delle informazioni: per prima cosa il loro ordinamento reale; inoltre se due dati (cifrati) hanno lo stesso numero di dati più grandi di loro, vuol dire che i dati reali sono uguali. Quindi la possibilità di ordinare i dati cifrati secondo l'ordinamento numerico dei dati reali (ignoti a chi esegue l'ordinamento) permette di ottenere alcune informazioni sui dati reali stessi.

Questo è ovviamente un esempio banale, ma ci deve far riflettere se e come la crittografia omomorfica sarà da sola in grado di garantire la confidenzialità delle informazioni anche durante l'elaborazione in un ambiente potenzialmente ostile, o se più probabilmente sarà un altro strumento a nostra disposizione per la protezione delle nostre informazioni da utilizzare insieme agli altri, a partire dal Confidential Computing.

1.2.10 RIFERIMENTI BIBLIOGRAFICI

Rif. 1: Confidential Computing Consortium, https://confidentialcomputing.io/

Rif. 2: NIST "Hardware-Enabled Security for Server Platforms: Enabling a Layered Approach to Platform Security for Cloud and Edge Computing Use Cases", Draft 28 aprile 2020, https://doi.org/10.6028/NIST.CSWP.04282020-draft

Rif. 3: NIST "Hardware-Enabled Security: Container Platform Security Prototype", Draft dicembre 2020, https://doi.org/10.6028/NIST.IR.8320A-draft

Rif. 4: Confidential Computing Consortium "Confidential Computing Deep Dive v1.0", ottobre 2020, https://confidentialcomputing.io/wp-content/uploads/sites/85/2020/10/Confidential-Computing-Deep-Dive-white-paper.pdf

Rif. 5: Confidential Computing Consortium "Hardware-Based Trusted Execution for Applications and Data", luglio 2020, https://confidentialcomputing.io/wp-content/uploads/sites/85/2021/01/confidentialcomputing_outreach_whitepaper-8-5x11-1.pdf

Rif. 6: si vedano ad esempio i seguenti articoli: The Register "Microsoft brings Trusted Platform Module functionality directly to CPUs under securo-silicon architecture Pluton", https://www.theregister.com/2020/11/17/microsoft_pluton_cpu_hardware_security/ ; The Hacker News "Intel Adds Hardware-Enabled Ransomware Detection to 11th Gen vPro Chips", https://thehackernews.com/2021/01/intel-adds-hardware-enabled-

ransomware.html

Rif. 7: "AMD Memory Encryption white paper", aprile 2016, https://developer.amd.com/wordpress/media/2013/12/AMD_Memory_Encryption_Whitepaper_v7-Public.pdf ; AMD Memory Guard white paper", marzo 2020, https://www.amd.com/system/files/documents/amd-memory-guard-white-paper.pdf

Rif. 8: ArsTecnica "Intel promises Full Memory Encryption in upcoming CPUs", https://arstechnica.com/gadgets/2020/02/intel-promises-full-memory-encryption-in-upcoming-cpus/

Rif. 9: si vedano ad esempio i seguenti recenti articoli:

- ArsTechnica "Hackers can use just-fixed Intel bugs to install malicious firmware on PCs", https://arstechnica.com/information-technology/2020/11/intel-patches-high-severity-bugs-protecting-lost-stolen-or-confiscated-pcs/ ;
- The Register "Intel's SGX cloud-server security defeated by $30 chip, electrical shenanigans", https://www.theregister.com/2020/11/14/intel_sgx_physical_security/ ;
- A. Nilsson, P.S. Bideh, J. Brorsson "A Survey of Published Attacks on Intel SGX", giugno 2020, https://arxiv.org/abs/2006.13598 ;
- G. Steel "How Ledger Hacked an HSM", giugno 2019, https://cryptosense.com/blog/how-ledger-hacked-an-hsm ;
- Wikipedia "Software Guard Extensions", https://en.wikipedia.org/wiki/Software_Guard_Extensions ;

- JP Aumasson, Luis Merino, “SGX Secure Enclaves in Practice: Security and Crypto Review”, BlackHat 2016, https://www.blackhat.com/docs/us-16/materials/us-16-Aumasson-SGX-Secure-Enclaves-In-Practice-Security-And-Crypto-Review.pdf

Rif. 10: per Intel SGX si veda ad esempio https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard-extensions.html

Rif. 11: per riferimenti si veda ad esempio consorzio “Homomorphic Encryption Standardization”, https://homomorphicencryption.org/

Rif. 12: Craig Gentry “Fully Homomorphic Encryption Using Ideal Lattices”. In the 41st ACM Symposium on Theory of Computing (STOC), 2009, http://portal.acm.org/citation.cfm?id=1536414.1536440

Rif. 13: si veda ad esempio lo standard NIST in sviluppo “Post-Quantum Cryptography”, https://csrc.nist.gov/Projects/Post-Quantum-Cryptography

Rif. 14:

- Microsoft “Azure confidential computing virtual machines (VMs) overview”, https://docs.microsoft.com/en-us/azure/confidential-computing/confidential-computing-enclaves ;
- Microsoft “Always Encrypted with secure enclaves”, https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/always-encrypted-enclaves?view=sql-

server-ver15 ;

- Amazon “AWS Nitro Enclaves”, https://aws.amazon.com/it/ec2/nitro/nitro-enclaves/ ;
- Google “Confidential Computing”, https://cloud.google.com/confidential-computing

Rif. 15: Intel Corporation “Strengthen Enclave Trust with Attestation”, 2020, https://software.intel.com/en-us/sgx/attestation-services

Rif. 16: “OASIS Key Management Interoperability Protocol (KMIP)”, https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=kmip

Rif. 17: NIST Special Publication 800-57 1.rev5 “Recommendation for Key Management”, https://doi.org/10.6028/NIST.SP.800-57pt1r5

02

SICUREZZA IN INTERNET

2.1 Aspetti di sicurezza di BGP e Routing in Internet

Lo straordinario successo di Internet, ovvero della rete globale di comunicazione che ormai viene usata dalla maggioranza degli abitanti del nostro pianeta Terra, è dovuto ed è possibile grazie fondamentalmente a due protocolli che permettono all'infrastruttura di telecomunicazioni di funzionare egregiamente: il "Domain Name System (DNS)" ed il "Border Gateway Protocol (BGP)" ovvero il protocollo di Routing dei pacchetti IP.

Entrambi protocolli risalgono agli anni '80 [Rif. 1], sono evoluti nel tempo ed hanno dimostrato una incredibile resilienza, scalabilità e capacità di supportare la disponibilità del servizio. D'altra parte funzionalità specifiche per garantire l'integrità e l'autenticità dei dati[19] non sono state previste inizialmente e col tempo è risulta-

19) Essendo tutti dati pubblici, non si pongono problemi riguardo la riservatezza dei dati se non per quanto riguarda eventuali aspetti di Privacy, che però non sono qui considerati.

to difficile includerle. Se per DNS è in corso l'implementazione di DNSSEC[20], in questo capitolo si considerano gli aspetti di sicurezza del Routing in Internet, ovvero del protocollo BGP-4 [Rif. 3] attualmente in uso.

2.1.1 ROUTING IN INTERNET

Prima di descrivere minacce, vulnerabilità e rischi, è necessario un breve ripasso di cosa sia Routing IP in Internet. Il protocollo IP (Internet Protocol [Rif. 2]) assegna ad ogni dispositivo connesso alla rete Internet un indirizzo numerico IPv4,[21] di 32 bit tipicamente scritto con 4 ottetti come 192.168.222.1, o IPv6, di 128 bit. Per semplicità si considerano solo gli indirizzi IPv4 ma quanto discusso si applica in generale anche agli indirizzi IPv6, seppur nel caso con qualche modifica. Scopo primario del protocollo IP è di permettere ad un pacchetto di dati inviato da un elaboratore di raggiungere un altro elaboratore di cui sia noto l'indirizzo IP. Visto che la rete IP si estende su tutta la superficie terreste,[22] il problema che il Routing, in questo caso BGP-4, risolve è quello di individuare il percorso nelle reti di trasporto di dati digitali per raggiungere l'indirizzo di

20) Si veda il seguente capitolo §3.1

21) Si noti che tra il 2011 ed il 2019 si è esaurita l'assegnazione di indirizzi IPv4, oggi si possono richiedere nuovi indirizzi IPv6 o attendere che vengano resi liberi degli indirizzi IPv4 in uso.

22) In realtà dal 2010 anche gli astronauti sulla Stazione Spaziale possono accedere ad Internet, ma indirettamente tramite un elaboratore a terra.

destinazione. Figuratamente la situazione è simile a quella di una rete autostradale (i cavi di trasporto dati) sulla quale transitano vetture (i pacchetti dati) e gli svincoli tra due o più autostrade (i router). Come agli svincoli autostradali sono presenti i cartelli di segnalazione per indicare su quale autostrada proseguire per raggiungere una destinazione, nei router è presente una "tabella di Routing" che indica su quale cavo/rete inviare un pacchetto perché giunga a destinazione. Il protocollo BGP-4 ha come compito quello di gestire la "tabella di Routing" di ogni router IP in Internet.[23] Si incontrano immediatamente due difficoltà pratiche:

- il numero di Prefissi, ovvero di "cartelli stradali", che ogni ruoter IP che ha una tabella di Routing completa deve mantenere, oggi supera 800.000;
- Le tabelle di Routing sono molto dinamiche, gli aggiornamenti sono continui e molto frequenti, tipicamente nell'ordine dei minuti.

Questo si comprende facilmente visto che una tabella completa di Routing contiene le indicazioni su come raggiungere ogni indirizzo IP pubblico, in qualunque parte della Terra. Le modifiche della tabella sono dovute non solo all'attivazione o rimozione di indirizzi IP, ma anche al cambio di percorso per l'attivazione o spegnimento di percorsi dati, ad esempio cavi sottomarini, collegamenti satellitari o anche solo collegamenti tra due Internet Provider.

23) Questo è tecnicamente indicato con E-BGP, External BGP, mentre I-BGP, Internal BGP, descrive l'applicazione del protocollo a reti interne ad una organizzazione, o più precisamente ad una AS, come viene descritto di seguito.

2.1.2 INDIRIZZI IP E ROUTING

E' necessario approfondire un poco la gestione degli indirizzi IP ed il funzionamento del Routing BGP-4 anche per quanto sarà discusso più avanti.

Gli indirizzi IP sono gestiti a livello globale dalla IANA (Internet Assigned Numbers Authority) che assegna blocchi di indirizzi IP alle cinque RIR (Regional Internet Registries): APNIC, LACNIC, ARIN, AFRINIC, RIPE. Ognuna delle cinque RIR ha competenza solo su di una zona geografica [Rif. 4], ad esempio il RIPE su Europa e Asia del Nord e Ovest. Le RIR a loro volta assegnano blocchi di indirizzi IP ai LIR (Local Internet Register) che tipicamente sono Internet Provider, Carrier ma anche grandi organizzazioni e università. A loro volta le LIR allocano blocchi di indirizzi IP ai propri clienti finali. Un'organizzazione che vuole connettersi a Internet con indirizzi IP esplicitamente a lei assegnati, deve per prima cosa ottenere dalla propria LIR uno (o più per organizzazioni molto grandi o distribuite a livello mondiale) numero di AS (Autonomous System). Gli indirizzi IP possono essere assegnati solo ad un AS che è stato assegnato a sua volta ad un'organizzazione.

Gli indirizzi IP sono assegnati in blocchi ed ogni blocco è descritto dal proprio Prefisso. Il Prefisso sono le cifre più significative (a sinistra) comuni a tutti gli indirizzi IP del blocco. Ogni Prefisso ha una lunghezza indicata con /nn ove nn è la lunghezza in bit del Prefisso. Ad esempio il blocco di indirizzi IP con inizio [24] 10.10.0.0 e

24) Gli indirizzi IP mostrati in questo articolo sono del tutto casuali e tipicamente non validi in Internet ma solo come indirizzi privati locali.

fine 10.10.255.255 ha Prefisso 10.10.0.0/16.

Visto che gli indirizzi IP sono assegnati solo a AS e solo in blocchi identificati da Prefissi, è logico che una tabella BGP contenga Prefissi e AS.

BGP-4 prevede la presenza, almeno logica, di tre tipi di tabelle: per comodità in questo articolo è chiamata "tabella BGP" (o "Adj-RIB-In") una tabella logica in cui un router raccoglie tutti i dati di Routing BGP che ha ricevuto e che possono essere utili; con "tabella di Routing" (o Loc-RIB, Local-Routing-Information-Base) è indicata la tabella con solo le informazioni utilizzate dal router stesso per il Routing dei pachetti IP in ogni istante; e infine con tabella "Update BGP" (o "Adj-RIB-Out") una tabella logica con le informazioni di Routing da inviare ai propri Peer. La procedura di compilazione di queste tabelle è in linea di principio non troppo difficile anche se richiede alcuni passaggi.

1. Accordi tra organizzazioni

Potrà sembrare strano, ma il primo passo è un accordo, tipicamente commerciale, tra organizzazioni che intendono scambiarsi traffico dati su Internet tramite il protocollo IP. Questo accordo disciplina eventuali costi, tipo e quantità di traffico e anche quali router BGP dell'AS di una organizzazione scambiano "annunci", ovvero i dati necessari a compilare le tabelle BGP, di Routing e di Update, con quali router BGP dell'altra organizzazione.

2. "Peering" BGP

Una volta definito l'accordo tra le organizzazioni, è possibile confi-

gurare un collegamento dati detto “Peering” tra i router BPG delle due AS allo scopo di scambiare “annunci” BGP ed i loro aggiornamenti (modifiche, cancellazioni ecc.). Un annuncio BGP contiene molte informazioni, ma per i nostri scopi ci limitiamo solo a quelle principali descritte in Fig. 2.1.1.

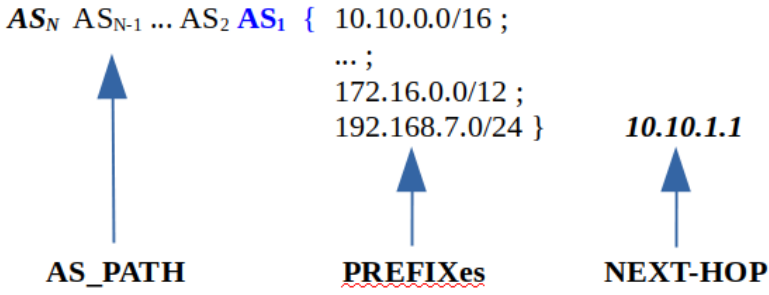

Figura 2.1.1: Principali informazioni presenti in un annuncio BGP

Quando un AS (AS1) invia i propri indirizzi IP ad un’altra AS con cui ha un peering BGP, invia i Prefissi (vedi Fig. 2.1.1), prepende (a destra in AS_PATH e anche più di una volta) il proprio numero di AS (AS1) e postpone l’indirizzo IP (Next-Hop) verso il quale il Peer deve inviare il traffico per raggiungere i Prefissi. Il router BGP dell’AS ricevente deve validare il numero di AS da cui ha ricevuto i dati e la raggiungibilità del Netx-Hop. In generale ogni volta che un router BGP riceve un annuncio da un’altra AS, verifica che all’annuncio sia stato preposto il numero dell’AS da cui lo ha ricevuto. In questo modo, mentre l’annuncio transita da AS ad AS, si forma la AS_PATH, ovvero il cammino tra le AS che l’annuncio ha fatto per arrivare al router BGP in cui è presente. Si noti che i Prefissi appar-

tengono alla prima AS in AS_PATH (AS1 nel caso di Fig. 2.1.1).

L'indirizzo IP del Next-Hop, ovvero del primo router a cui inviare i dati per raggiungere quei Prefissi, è invece un'informazione locale, può essere l'indirizzo del router BGP che ha inviato l'annuncio od un altro indirizzo IP dell'AS (ASN) da cui è arrivato l'annuncio e concordato tra le due AS in Peering.

Tutti gli annunci ricevuti da un router BGP dai propri Peer vengono inseriti e tenuti aggiornati nella "tabella BGP". Si noti come i dati contenuti nella "tabella BGP" sono sempre una vista locale condivisa con i propri Peer.

3. La "tabella di Routing"

E' importante notare che nella "tabella BGP" lo stesso Prefisso può essere presente più volte, annunciato da AS diverse con AS_PATH diverse ma sempre terminanti in AS1 (ma nel caso di grandi organizzazioni con più AS, lo stesso prefisso può anche essere annunciato inizialmente da AS diverse, come sarà discusso più avanti). Un router BGP deve quindi decidere a quale dei propri Peer inviare il traffico per ognuno dei Prefissi conosciuti. I criteri di scelta sono molti, a partire dai requisiti stipulati negli accordi con i propri Peer: alcune rotte devono essere escluse (filtrate), altre privilegiate ecc. In generale ogni AS deve decidere la propria "Politica di Routing" ed in base a questa vengono configurati i router BGP in modo da selezionare le rotte appropriate. La rotta scelta, viene quindi inserita nella "tabella di Routing" (o Loc-RIB, Local-Routing-Information-Base) ove le uniche informazioni necessarie sono il Prefisso di destinazione del pacchetto e l'indirizzo IP a cui inviarlo, ovvero

il Next-Hop.

La "tabella di Routing" è però molto dinamica, si consideri il seguente esempio: il router BGP di una AS in mezzo all'AS_PATH scelta si guasta ed un aggiornamento dell'annuncio pertanto cancella la rotta dalle tabelle. Il router BGP che utilizzava quella rotta, deve scegliere una rotta alternativa per raggiungere quel Prefisso. Alla riattivazione del router BGP guasto, viene re-inserita la rotta precedente, e così via.

4. La "tabella di Update"

Infine il router BGP invia ai propri Peer gli annunci BGP che contengono solo rotte (Prefissi) presenti nella "tabella di Routing" (altrimenti i Prefissi non sarebbero raggiungibili tramite il router BGP che invia l'annuncio) con l'AS_PATH e gli altri dati BGP presenti nella "tabella di Update". Si noti che un router BGP non è obbligato ad inviare la propria intera "tabella di Routing" ma solo quei Prefissi concordati con il Peer negli accordi formali.

2.1.3 MINACCE, RISCHI E ATTACCHI AL ROUTING BGP

La principale minaccia al Routing BGP è usualmente chiamata "BGP Hijacking" e consiste nella manipolazione (modifica, cancellazione, inserimento ecc.) delle informazioni (annunci) utilizzate per costruire le tabelle BGP e di Routing, o la modifica diretta delle tabelle.

Le possibili conseguenze di un attacco di BGP Hijacking possono

essere raggruppate in due classi:

Denial of Service: se un Prefisso è rimosso dalla tabella di Routing od il traffico è inviato su di una rotta che non è in grado di consegnarlo al destinatario (fenomeno anche chiamato "blackholing"), effettivamente viene rimosso da Internet qualunque sistema e servizio che utilizza gli indirizzi IP in quel Prefisso. Questo è spesso accaduto per errore, ma vi sono stati alcuni casi di attacchi volontari. Tra gli episodi più famosi di questo tipo si possono citare:

- 24 dicembre 2004: l'Internet Provider TTNet in Turchia per errore fa convergere tutto il traffico Internet verso la propria rete;
- 7 maggio 2005: per un errore di configurazione di un Internet Provider, Google scompare da Internet per poco meno di 1 ora;
- 24 febbraio 2008: nel tentativo di bloccare l'accesso a YouTube, il Pakistan lo rimuove completamente da Internet;
- 5 gennaio 2017: nel tentativo di bloccare l'accesso ad alcuni siti solo per adulti, l'Iran blocca l'accesso a questi siti in tutto Internet;
- 12 novembre 2018: per un errore di configurazione di un Internet Provider, Google scompare da Internet per poco meno di 1 ora.

Ovviamente per un sito web commerciale di grande traffico lo scomparire da Internet anche per poco tempo può provocare dei danni economici ingenti.

Man in the Middle: in questo caso la manipolazione delle tabelle di Routing ha lo scopo di deviare il traffico e farlo passare attraverso reti anche nemiche. Il traffico può essere solamente letto oppure indirizzato verso siti o servizi civetta che imitano il servizio originale. Molti di questi attacchi sono stati perpetrati allo scopo di rubare monete digitali (ad esempio Bitcoin) o di attaccare siti di banche e istituti finanziari, tra questi si possono citare:

- Febbraio 2013: il traffico diretto ad alcuni siti di banche e grandi aziende di diverse nazioni è stato instradato per piccoli periodi di tempo, una vittima alla volta, attraverso una rete in Bielorussa;
- Agosto 2013: il traffico verso alcuni Prefissi principalmente americani è diretto attraverso un Internet Provider Islandese;
- 24 aprile 2018: diversione del traffico diretto ad alcune reti Amazon allo scopo di rubare monete digitali da alcuni servizi su AWS;
- 30 luglio 2018: il traffico mondiale dell'App Telegram viene fatto transitare attraverso l'Iran;
- 2016-2017: forse il più famoso caso di ridirezione del traffico Internet è quello relativo al dirottamento del traffico attraverso la Cina, per lunghi periodi di tempo, anche mesi, tra gli Stati Uniti ed alcuni Prefissi in altre nazioni tra cui Italia, Canada, Corea, Giappone, Scandinavia ecc. [Rif. 5].

Se un attaccante riesce a dirottare il traffico di suo interesse attraverso una rete che controlla, oltre a leggere il contenuto dei

dati scambiati (se non cifrati), può modificarli o impersonare il destinatario.

Un altro interessante esempio di BGP Hijacking è stata la frode "3ve" [Rif. 6] che nel 2017 arrivò a controllare 1,5 milioni di indirizzi IP e defraudò alcune aziende di pubblicità online di circa 29 milioni di US dollari. Fornendo spesso documenti falsi, questa organizzazione criminale riuscì a far assegnare a proprie società di comodo, alcuni AS in realtà assegnati ad aziende chiuse o fallite, e Prefissi assegnati ad altre organizzazioni ma non utilizzati, ovvero non "annunciati". Sfruttando la mancanza di controlli e di interesse dei veri assegnatari delle AS e dei Prefissi, "3ve" utilizzò questi indirizzi IP per creare traffico realistico ma falso[25] su propri siti web nei quali erano presenti le pubblicità delle aziende da frodare. Nell'arco di circa un anno, vi fu un enorme numero di visualizzazioni delle pubblicità su questi siti web che da una parte fruttò all'organizzazione criminale lauti guadagni, ma al contempo suscitò l'interesse delle aziende di pubblicità online che avviarono una investigazione che portò all'individuazione ed all'arresto di alcuni dei criminali.

2.1.4 ROUTE FLAPPING E BGP ROUTE FLAP DAMPING

Uno dei problemi che può creare gravi difficoltà alla gestione delle rotte BGP, è quello dell'instabilità negli annunci. Vi possono es-

25) Gli indirizzi IP erano assegnati a server sui quali erano eseguiti dei Bot che visitavano i siti web e cliccavano sulle pubblicità.

sere molti motivi, dalla non convergenza dell'algoritmo di scelta del miglior annuncio per un certo Prefisso, al troppo traffico sul canale di comunicazione con un Peer (si ricordi che il traffico BGP è inline, non utilizza connessioni dedicate), per i quali un annuncio può essere inserito nella Tabella di Routing e poco dopo rimosso. L'alternarsi, o "flapping", dell'annuncio di alcune rotte può rallentare il router BGP impiegando troppe risorse nel calcolo del miglior annuncio, nella modifica delle rotte, e nell'invio delle modifiche ai Peer. E la distribuzione di un "flapping" può causare nei Peer ulteriori "flapping" in una reazione a catena.

La soluzione prevista da BGP-4 è quella del "route flap damping" [Rif. 7]: nel caso in cui un annuncio vari troppo frequentemente, il router deve sospenderlo (quindi non inserirlo nella propria tabella di Routing ne inviarlo ai Peer) per un certo periodo di tempo in attesa che l'instabilità si risolva. Più è instabile la rotta e più il periodo di sospensione è lungo, sino ad un massimo di tempo dopo il quale il router prova comunque a ri-annunciare la rotta. Ma l'utilizzo stesso del "route flap damping" può in alcune situazioni causare più danni di quanti ne risolva (si vedano ad esempio le due posizioni del RIPE riportate in [Rif. 8]) e va quindi adottato con cautela.

2.1.5 MODALITÀ DI ATTACCO A BGP-4

Il modo più semplice per effettuare un attacco al Routing BGP-4 o "BGP Hijacking", è quello di accedere ad un router BGP di un Internet Provider o Carrier, attraverso credenziali rubate o vulnerabilità

di sistemi. Una volta avuto accesso ad un router BGP, si possono modificare gli annunci BGP inviati ai Peer in modo da bloccare o dirottare il traffico di interesse. Il protocollo BGP-4 non prevede meccanismi espliciti di controllo o sicurezza: in linea di principio ogni router ripone completa fiducia negli annunci che riceve dai propri Peer. Quindi se un router BGP invia annunci errati o malevoli, questi possono essere accettati senza alcun controllo dagli altri router BGP.

Un altro modo per modificare le tabelle di Routing è quello di intercettare e modificare il traffico dati tra due router BGP e quindi gli annunci che i due router si scambiano. Di base il traffico tra due router BGP è costituito da sessioni TCP non autenticate e non protette da cifratura o altre misure di sicurezza, sono verificati solo gli indirizzi IP dei due router. In alcune situazioni è pertanto possibile, anche se non è del tutto facile, inserire o modificare i dati scambiati tra due router BGP.

E' anche possibile inserire volontariamente (o per errore) nei propri router BGP degli "annunci" allo scopo di deviare o bloccare il traffico verso alcuni Prefissi. Gli effetti di questi annunci malevoli sono tipicamente locali, ma in alcuni casi particolari possono estendersi a livello globale (come già visto per alcuni incidenti descritti precedentemente).

Infine, come indicato ad esempio dalla frode "3ve" descritta precedentemente, è possibile attaccare i processi formali e burocratici di assegnazione di AS e Prefissi alle organizzazioni per ottenerne un uso formalmente valido. Ad esempio, "3ve" utilizzò, tra gli altri, gli indirizzi IP del Prefisso 192.73.128.0/18 assegnato ma

non utilizzato dall'Aeronautica Militare Americana (United States Air Force); questo annuncio BGP fu accettato globalmente sino a quando non fu mostrato che l'Aeronautica Militare Americana non aveva autorizzato ad utilizzare i propri indirizzi IP una sconosciuta azienda Ucraina sospettata di gestire reti e servizi criminali.

2.1.6 PRIME MISURE DI SICUREZZA

La prima misura di sicurezza pratica applicabile agli annunci BGP consiste nell'applicare dei filtri [Rif. 9]. Il primo tipo di filtri è quello che va genericamente sotto il nome di "Bogon Filtering"[26] ovvero filtri di rotte invalide. La comunità degli operatori mantiene delle liste basate principalmente sui documenti ufficiali di IANA e delle RIR di AS e Prefissi non allocati e che quindi non devono comparire in alcun annuncio BGP [Rif. 10]. Sta però a chi configura e mantiene i router BGP applicare filtri agli annunci sia ricevuti che inviati ai Peer per impedire che questi AS e Prefissi compaiano nella tavola di Routing dei propri router e che vengano inviati ai Peer. Configurando in questo modo i propri router BGP, un AS impedisce che annunci e traffico invalido possano transitare sulle proprie reti.

Il secondo tipo di filtri su AS e Prefissi è invece specifico per ogni Peer. Infatti a seconda del tipo di accordo contrattuale in essere con un Peer, è norma di sicurezza, anche per il rispetto delle clausole contrattuali, configurare dei filtri che accettino e inviino

26) Anche chiamato "Martian Filtering" ovvero filtri di rotte al più valide su Marte, denominazione valida sinché l'uomo non arriverà su Marte.

solo gli annunci previsti dagli accordi. Ad esempio si consideri il caso di un'organizzazione con proprio AS che ha come Peer alcuni Internet Provider e Carrier che le forniscono accesso ad Internet. L'organizzazione configura sui propri router BGP dei filtri sugli annunci inviati ai fornitori che permettono l'invio solo del proprio AS e dei propri Prefissi. In questo modo l'organizzazione impedisce di diventare un nodo di transito del traffico tra i fornitori. L'organizzazione può poi mettere dei filtri sugli annunci che riceve in modo ad esempio di utilizzare un fornitore solo per il traffico in uscita verso una particolare regione geografica, e comunque sempre rispettando gli accordi contrattuali. Specularmente i Peer dell'organizzazione configurano dei filtri BGP in modo da accettare solo gli annunci dell'AS e dei Prefissi dell'organizzazione. Questo impedisce ad esempio che anche solo per errore, l'organizzazione invii annunci di AS e Prefissi non propri e che diventi un nodo di transito di traffico verso altre AS. Infine i Peer possono configurare filtri sugli annunci che inviano all'organizzazione basati sugli accordi contrattuali in essere. [27]

Un altro tipo di filtro da applicare non agli annunci BGP ma al traffico stesso, è quello sugli indirizzi IP sorgenti di ogni pacchetto IP. Il routing BGP considera solo gli indirizzi di destinazione ed il problema di consegnare al corretto destinatario i pacchetti IP. Molti attacchi in rete, in particolare di Denial of Service, sfruttano però la possibilità di inviare un pacchetto IP indicando un indirizzo sor-

27) Tipicamente un fornitore di traffico Internet, in assenza di specifici accordi contrattuali, invia tutti gli annunci BGP al cliente finale in modo che questi possa bilanciare il proprio traffico tra i vari fornitori.

gente diverso da quello reale. L'effetto è che chi riceve il pacchetto IP non sa chi è il vero mittente e nel caso invia il pacchetto IP di risposta ad un altro sistema. Questo meccanismo è stato spesso utilizzato per attacchi di Distributed (Reflected and Amplified) Denial of Service, in cui moltissimi sistemi attaccanti inviano contemporaneamente richieste a servizi quali DNS o NTP indicando come sorgente lo stesso indirizzo IP, che viene quindi subissato dalle risposte. Per contrastare questo attacco, si può applicare sui router il "Ingress Filtering" [Rif. 11] ma solo quando si conoscono esattamente i Prefissi IP da cui proviene il traffico. Tipicamente questo è il caso descritto precedentemente di un'organizzazione che si connette a Internet tramite alcuni Peer: visto che i Peer ricevono da una certa connessione solo il traffico originato dall'organizzazione, possono inserire filtri sugli indirizzi IP sorgenti per limitarli a quelli dei Prefissi annunciati dall'organizzazione. In generale però, quando un AS od un Peering è di transito, ovvero scambia qualunque traffico IP, non è possibile filtrare gli indirizzi IP sorgenti, se non eliminando i Prefissi Bogon, che comunque non sono raggiungibili. E' importante notare che nel routing BGP il percorso di andata di un pacchetto può essere diverso dal percorso di ritorno, pertanto anche i filtri su "reverse-path", ovvero che il pacchetto di ritorno segua la stessa strada di quello d'andata, vanno applicati con cura e solo ove possibile.

I filtri appena descritti, se implementati da tutti ed in maniera completa, avrebbero impedito molti degli incidenti descritti precedentemente.

E' anche possibile mettere in sicurezza la comunicazione BGP tra

due Peer. Benché il protocollo BGP originale non lo prevedesse, un'estensione di BGP-4 permette di garantire l'integrità dei pacchetti scambiati tra i due Peer aggiungendo ad ognuno di essi un HMAC ("Hash-based Message Authentication Code"), ovvero un codice d'impronta del pacchetto BGP basato su di una chiave segreta condivisa tra i due router [Rif. 12].

Un'altra soluzione è quella di stabilire un tunnel IPSEC tra i due router BGP ed instradare il traffico BGP attraverso questo tunnel. Visto che le informazioni scambiate sono comunque pubbliche, non è tipicamente necessario che siano cifrate a protezione della confidenzialità, quindi possono essere utilizzati tunnel IPSEC AH invece di IPSEC ESP. Questa soluzione è ormai disponibile nella maggior parte dei router in quanto IPSEC vi è in buona parte implementato in Hardware con circuiti dedicati. Al contempo però va valutato se la presenza di un tunnel IPSEC può comportare maggiori rischi di instabilità degli annunci. Infatti in caso di caduta del tunnel IPSEC tutti gli annunci ricevuti dal Peer vengono cancellati ed il traffico con il Peer bloccato, anche se il problema sia da addebitare al tunnel IPSEC e non alla linea tra i due router.

2.1.7 ULTERIORI PROBLEMI DI SICUREZZA DI BGP-4

Le misure di sicurezza appena descritte, aiutano ma non risolvono i problemi di sicurezza di BGP-4. Sempre facendo riferimento alla Fig. 2.1.1, è possibile identificare le seguenti due principali problematiche di sicurezza:

1. garantire e dare la possibilità di verifica che AS1 sia autorizzato ad annunciare i Prefissi indicati;

2. garantire e dare la possibilità di verifica che la AS_PATH non sia stata manipolata.

Il primo è principalmente un problema di autenticazione chiamato anche Origin Authentication ("OA"), mentre il secondo si può accostare ad un problema di integrità, anche se non completamente.

Sin dagli anni '90 sono state proposte e studiate molte possibili soluzioni per queste problematiche di sicurezza di BGP-4. Queste proposte possono essere organizzate in tre categorie:

1. Nuovi protocolli di Routing che sostituiscono completamente BGP-4;

2. Nuove versioni di BGP che estendono il protocollo includendo anche le necessarie misure di sicurezza;

3. Protocolli paralleli ma indipendenti da BGP-4 e che possono essere facilmente integrati con questo;

per una rassegna, si veda ad esempio [Rif. 13], in particolare la tabella II.

La quasi totalità delle soluzioni proposte prevede l'utilizzo di algoritmi crittografici principalmente per firmare gli annunci e/o i dati relativi agli annunci BGP. Ben poche però di queste proposte hanno avuto anche un minimo successo. Negli ultimi anni, due proposte sono state sviluppate ed ora sono in fase di valutazione

e possibile adozione: BGPSec [Rif. 14] e RPKI [Rif. 15].

2.1.8 BGPSEC E RPKI

BGPSec è una nuova versione di BGP-4 che si propone di affrontare il problema dell'integrità di AS_PATH. L'idea di base del protocollo è quella di firmare digitalmente ogni "annuncio" sostituendo la AS_PATH con una BGPSEC_PATH che contiene sia i dati dell'AS_PATH ma anche il numero di AS a cui l'annuncio è inviato e la firma digitale dell'annuncio. In questo modo non solo l'annuncio è firmato, ma ogni annuncio è specifico tra due Peer. Un router BGP che riceve un annuncio firmato deve recuperare usando RPKI, discusso di seguito, le chiavi pubbliche di tutti i router delle AS presenti in un BGPSEC_PATH e verificare tutte le firme presenti. Allo stesso tempo, un router BGP che invia annunci BGPSec deve firmare singolarmente ogni annuncio inviato ad ogni Peer. E' chiaro che BGPSec richiede molte più risorse di BGP-4 ma non è chiaro quanto questo sia compatibile con l'hardware attuale dei router BGP. Inoltre l'utilizzo di BGPSec è opzionale rispetto a BGP-4, per cui se un router BGP nella rotta non supporta BGPSec, da quel punto in avanti non saranno più presenti le firme digitali ma solo la AS_PATH tradizionale.

Si può porre la situazione di ricevere da due Peer distinti per lo stesso Prefisso una AS_PATH BGP-4 e una BGPSEC_PATH, come deve procedere il router BGP nella scelta dell'annuncio? Deve scegliere BGPSEC_PATH solo perché firmato digitalmente anche se la rotta non sarebbe conveniente rispetto a tutti gli altri parametri di

scelta ed alla propria "Politica di Routing"? Va anche considerato che la firma digitale degli annunci, anche se fosse implementata da tutti i router BGP, non coprirebbe tutti gli scenari di attacco discussi: ad esempio annunci dovuti ad errori involontari di configurazione o ad intrusioni nei router BGP e perfino ad azioni volontarie, sarebbero comunque firmati digitalmente. Non è chiaro quindi se e come il fatto di firmare digitalmente a catena gli annunci BGP possa fornire il livello di integrità e "assurance" in grado di contrastare efficacemente le problematiche di sicurezza relative alla gestione delle AS_PATH. Vi sono parecchie voci discordi sull'adozione di BGPSec, e nel prossimo futuro vedremo se avrà fortuna o farà parte delle tante soluzioni proposte ma non adottate.

Nel frattempo in assenza di BGPSec, si può monitorare, anche attraverso servizi pubblici quali BGP Route Servers e Looking Glass Servers, lo stato degli "annunci" e delle AS_PATH ed implementare tecniche di baselining, analytics e visualizzazione per identificare modifiche e possibili anomalie sulle quali poi nel caso intervenire anche manualmente.

Se BGPSec è una proposta formalmente del 2017 anche se basata su idee già avanzate anche 20 anni prima, RPKI, ovvero "Resource Public Key Infrastructure" ha una storia ancora più lunga e culminata nella sua formalizzazione nel 2012/2013.

Prima di RPKI è necessario partire da quello che comunemente è chiamato WHOIS. La necessità di creare una base dati di assegnatari di AS e Prefissi, nomi ed indirizzi, risale ai primi anni '70 in ARPANET, prima ancora di Internet, ed una delle prime formalizzazioni è in [Rif. 16]. Sono stati creati quindi degli archivi che permet-

tono di raccogliere in modo più o meno strutturato, tra l'altro, le informazioni sull'assegnazione di AS, Prefissi, nomi a dominio ecc. Queste basi dati sono oggi chiamate "Internet Routing Registries" (IRR), sono gestite da varie organizzazioni e le principali sono in carico alle RIR (e LIR), mentre una famosa base dati che risale ai primi anni '90 è Merit RADb (https://www.radb.net/). E' procedura comune di molti Internet Provider e Carrier di permettere un Peering BGP solo se i corrispondenti dati di AS e Prefissi sono presenti nelle basi dati IRR / Whois. Ma oltre ai problemi di Privacy e conformità al GDPR, che non sono qui discussi, molte di queste basi dati contengono informazioni non aggiornate ed in alcuni casi del tutto errate. In generale queste basi dati sono di difficile manutenzione in quanto spesso la responsabilità di gestione è destrutturata e assegnata diversamente a seconda della nazione in cui opera chi mantiene la base dati. E' opinione comune che queste basi dati siano sì utili ma solo a livello informativo, più o meno come un elenco telefonico non troppo aggiornato di cui ci si può fidare sino ad un certo punto. Ad esempio, nel 2014 il "Expert Working Group on gTLD Directory Services" di ICANN [Rif. 17] ha caldamente suggerito di rimuovere del tutto i servizi IRR / Whois perché ritenuti inaffidabili, e di sostituirli completamente con nuovi servizi di registrazione. Malgrado ciò, ancora oggi le informazioni presenti nelle basi dati IRR / Whois sono utilizzate quotidianamente anche per validare gli annunci BGP-4.

Nel frattempo però, a partire dal 2013 è stata introdotta la "Resource Public Key Infrastructure" (RPKI). Questa è una forma di PKI simile a quella a cui siamo ben abituati dalle Certification Authorities (CA) e dai certificati della navigazione Web. Invece di CA, la

RPKI è basata su Trust Anchors (TA) che sono state individuate nelle cinque RIR. Ogni RIR genera i propri certificati X.509 "Trust" (ovvero "root") con i quali firma altri certificati, inclusi in caso quelli delle LIR, e le "Route Origination Authorizations" (ROAs). Un ROA è un documento firmato digitalmente che attesta che un AS è autorizzato ad annunciare un Prefisso con una massima lunghezza in bit: ad esempio AS1 può annunciare 10.1.0.0/16 con lunghezza massima /22. Quindi AS1 può annunciare 10.1.0.0/16 o 10.1.64.0/18 ma non 10.1.9.0/24. Ogni certificato ed ogni ROA ha una definita validità temporale, entro il quale deve essere nel caso rinnovato.

Le RIR hanno anche implementato delle basi dati pubbliche che contengono tutti i certificati pubblici necessari per verificare le firme, e tutti i ROA di competenza di quella RIR. Per verificare che gli annunci dei Prefissi siano corretti ed autorizzati, un operatore BGP tipicamente allestisce dei server RPKI che interrogano le basi dati dei RIR, scaricano i certificati ed i ROA e verificano le firme svolgendo così la parte più onerosa dei calcoli. Le versioni più recenti dei router BGP di tutti i vendor permettono di interrogare server RPKI per verificare l'autorizzazione di un AS ad annunciare un prefisso. Il protocollo RPKI prevede solo tre possibili risposte alla domanda "AS1 è autorizzato ad annunciare il Prefisso X ?":

1. Valido: è stato trovato un ROA con firma valida che autorizza AS1 all'annuncio del Prefisso;

2. Non Valido: è stato trovato un ROA per il Prefisso con firma valida ma per un altro AS, oppure l'AS è corretto ma la lunghezza del Prefisso supera il massimo previsto nel ROA;

3. Sconosciuto: tutti gli altri casi, in particolare quando per il Prefisso non è stato trovato alcun ROA valido.

Si noti che vi è almeno un caso di difficile gestione: ritornando all'esempio precedente si supponga che AS1 annunci 10.1.0.0/16, con risultato Valido, e AS2 annunci 10.1.9.0/24, con risultato Sconosciuto in quanto non c'è un ROA valido per 10.1.9.0/24. Il router BGP deve accettare l'annuncio 10.1.9.0/24? In realtà no, in quanto il ROA di 10.1.0.0/16 ha lunghezza massima /22 ed assumendo che non vi siano altri ROA per 10.1.0.0/16 con altre AS, questo annuncio non dovrebbe essere accettato. Questo però è lasciato alla "Politica di Routing" decisa dall'operatore ed alle relative configurazioni dei router BGP.

Sinora abbiamo silentemente assunto che un Prefisso possa essere annunciato solo da un AS. In realtà questo non è vero in quanto più AS possono annunciare lo stesso prefisso e questo capita in particolare per grandi fornitori di connettività e servizi Internet (si veda ad esempio [Rif. 18] per alcune statistiche sui Prefissi annunciati da più AS). L'esempio precedente, anche se come caso limite, può essere pertanto riformulato in maniera più generale: come bisogna comportarsi se un Prefisso ed un suo sotto-Prefisso sono annunciati da due AS uno con RPKI Valido e uno con RPKI Sconosciuto? Come prima, la risposta è lasciata alla "Politica di Routing" decisa dall'operatore.

Fig. 2.1.2 mostra il livello di adozione di RPKI, quasi il 40% di Prefissi è autorizzato tramite un ROA mentre l'60% circa dei Prefissi non è descritto da alcun ROA.

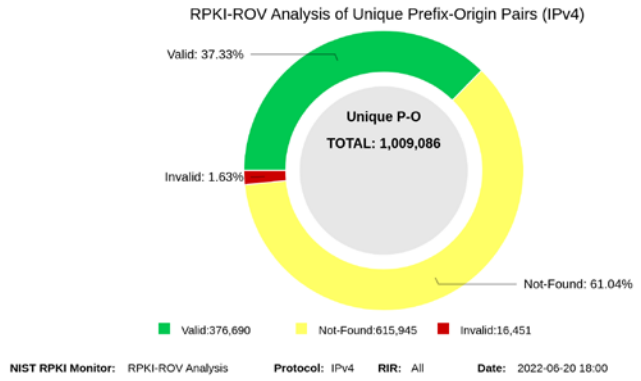

Figura 2.1.2: Statistica sull'adozione di RPKI IPv4 al 20 maggio 2022 [Fonte NIST https://rpki-monitor.antd.nist.gov/]

L'adozione di RPKI comunque prosegue come è mostrato in Fig. 2.1.3 ove la linea gialla indica l'andamento del numero dei Prefissi non descritti da alcun ROA mentre la somma delle linee verde e rossa indica l'andamento del numero dei Prefissi descritti da un ROA.

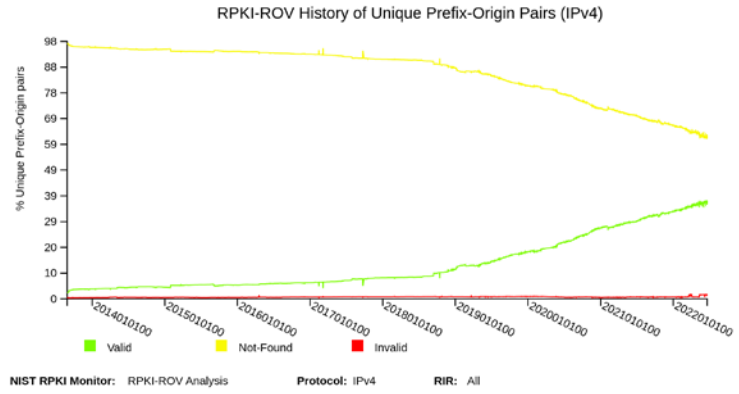

Figura 2.1.3: Adozione nel tempo di RPKI IPv4 [Fonte NIST https://rpki-monitor.antd.nist.gov/]

Se le recenti statistiche saranno confermate, sembra quindi che RPKI sarà adottato nel prossimo futuro da un numero sufficiente di operatori BGP da poter coprire la grande maggioranza di Prefissi. Questo porterà sicuramente a migliorare l'autenticità degli annunci dei Prefissi, riducendo di molto le possibilità di attacchi di BGP Hijacking.

2.1.9 RIFERIMENTI BIBLIOGRAFICI

Rif. 1: I protocolli originali sono specificati in RFC-882 e RFC-883 per DNS e RFC-1105 per BGP.

Rif. 2: RFC-791 "Internet Protocol".

Rif. 3: RFC-4271 "A Border Gateway Protocol 4 (BGP-4)"

Rif. 4: si veda ad esempio https://en.wikipedia.org/wiki/Regional_Internet_registry per cartine e maggiori dettagli.

Rif. 5: C.C. Demchak, Y. Shavitt "China's Maxim – Leave No Access Point Unexploited: The Hidden Story of China Telecom's BGP Hijacking", Military Cyber Affairs: Vol. 3, Iss. 1, Article 7, https://scholarcommons.usf.edu/mca/vol3/iss1/7/

Rif 6: Google and White Ops "The Hunt for 3ve", https://services.google.com/fh/files/blogs/3ve_google_whiteops_whitepaper_final_nov_2018.pdf , si veda anche A. Pasquinucci "Like a Movie Plot: the 3ve Defrauding Scheme", https://blog.ucci.it/2019/01/02/like-a-movie-plot-the-3ve-defrauding-scheme/ , e ArsTechnica "How 3ve's BGP hijackers eluded the Internet—and made $29M",

https://arstechnica.com/information-technology/2018/12/how-3ves-bgp-hijackers-eluded-the-internet-and-made-29m/

Rif. 7: RFC-2439 "BGP Route Flap Damping"

Rif. 8: "RIPE Routing Working Group Recommendations On Route-flap Damping" https://www.ripe.net/publications/docs/ripe-378 e https://www.ripe.net/publications/docs/ripe-580

Rif. 9: RFC-7454 "BGP Operations and Security"

Rif. 10: si veda ad esempio https://www.team-cymru.com/bogon-reference.html

Rif. 11: RFC-2827 "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing"

Rif. 12: RFC-5925 "The TCP Authentication Option"

Rif. 13: K.Butler, T.Farley, P.McDaniel, and J.Rexford "A Survey of BGP Security Issues and Solutions", 2008, https://www.cs.princeton.edu/~jrex/papers/bgp-security08.pdf

Rif. 14: RFC-8205 "BGPsec Protocol Specification", 2017/09

Rif. 15: RFC-6480 "An Infrastructure to Support Secure Internet Routing" 2012/02

Rif. 16: RFC-920 "NAME/FINGER"

Rif. 17: ICANN "Final Report from the Expert Working Group on gTLD Directory Services: A Next-Generation Registration Directory Service (RDS)", https://www.icann.org/en/system/files/files/final-

report-06jun14-en.pdf , 6 June 2014

Rif. 18: Hurricane Electrics “Multi Origin Route Report” https://bgp.he.net/report/multi-origin-routes

2.2 Sicurezza Web e Web Application Firewall

Lo straordinario successo di Internet è supportato e dovuto ai miliardi di siti e servizi Web accessibili a chiunque. Ovviamente tutti questi siti e servizi Web sono primari obiettivi di attacco e punti cruciali per la sicurezza in Internet.

Sin dalla fine degli anni ‘90 con l’avvio dei primi siti di commercio elettronico fu evidente l’esigenza di misure dedicate di sicurezza a loro protezione. Nacquero quindi i primi Web Application Firewall (ad esempio il progetto Open Source ModSecurity iniziò nel 2002 [Rif. 1]) il cui scopo è introdurre misure di protezione a livello applicativo, non di rete, a partire dal protocollo HTTP sino alle applicazioni ed alle transazioni svolte dagli utenti.

Ancora oggi però, più di vent’anni dopo la loro nascita, i Web Application Firewall, o WAF, non sono diffusi come ci si potrebbe aspettare e la loro efficacia è spesso messa in dubbio. Cercheremo qui di capire i punti di forza e di debolezza dei WAF, partendo da una breve descrizione di cosa sono e come vengono utilizzati.

2.2.1 UTILIZZO DI UN WAF

E' utile partire da una descrizione di come utilizzare un WAF. Per fare questo consideriamo un esempio teorico utile a identificare i punti di interesse. Ovviamente si parte dall'avere un servizio che espone un'interfaccia pubblica in Internet, ovvero un'interfaccia che può essere raggiunta da chiunque da Internet.

Questo però non vuol dire che chiunque possa accedere a tutti i servizi e dati dell'applicazione in quanto può essere richiesta un'autenticazione per fare ciò. In questa discussione però non è necessario arrivare a considerare queste logiche applicative che sono spesso specifiche per ogni applicazione.

Per le valutazioni che verranno fatte non è inoltre necessario considerare se il servizio esposto è un classico sito web o di commercio elettronico, il backend di una Mobile App o sono esposte delle Web API. Il punto importante è che il trasferimento di dati in Internet è effettuato utilizzando il protocollo HTTP e codice in HTML, XML, JSON, Javascript ecc.

Considerando una struttura comune di servizio Web, ovvero basato su protocollo HTTP/HTTPS, esposto in Internet, la prima cosa da considerare è dove collocare il WAF nella catena di servizi che compongono l'applicazione.

On-premises, un'architettura logica abbastanza comune per un servizio Web con WAF è descritta in Fig. 2.2.1.

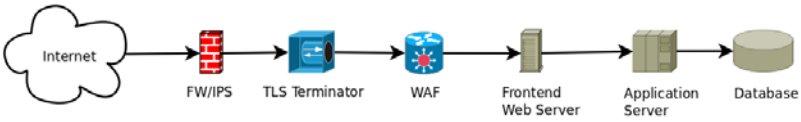

Figura 2.2.1: Architettura logica di un servizio Web con WAF

L'architettura descritta in Fig. 2.2.1 è puramente logica in quanto alcuni elementi possono in realtà sia essere realizzati dallo stesso sistema sia da molteplici sistemi. Inoltre le comunicazioni tra i vari componenti sono tipicamente cifrate e sono presenti ulteriori Firewall, appliance di sicurezza dedicate (e router, bilanciatori ecc.) che separano e proteggono i diversi livelli applicativi. Scopo del diagramma è quello di descrivere il principale percorso di un accesso applicativo proveniente da Internet al servizio Web:

- esposto dal Frontend Web Server
- gestito dall'Application Server
- con dati archiviati nel Database.

L'accesso applicativo da Internet utilizza di norma il protocollo HTTPS/TLS e quindi tutti i dati sono cifrati quando attraversano i servizi di sicurezza perimetrali quali il Firewall/IPS presente in figura.

Per proseguire è necessario pertanto che i dati vengano decifrati in modo che possano essere acceduti dai sistemi seguenti e per questo motivo il primo servizio presente è il terminatore TLS che appunto termina la sessione TLS stabilita con l'end-point dell'u-

tente. Questo però non vuol dire che da questo punto in avanti i dati sono trasferiti in chiaro, ma che i canali sono cifrati punto-punto ("hop-by-hop") da un sistema al successivo.

Visto che ogni sistema successivo al TLS Terminator può accedere ai dati in chiaro, il primo elemento presente è proprio il Web Application Firewall, che esamina i dati ricevuti (sia in ingresso dall'utente sia in uscita verso l'utente) e ne garantisce la sicurezza. Se il WAF non identifica problemi di sicurezza, i dati sono inoltrati al Frontend Web Server che gestisce le logiche del protocollo HTTP, all'Application Server che implementa le logiche applicative, e al Database che archivia i dati.

E' necessario specificare come il WAF riceve e gestisce i dati della connessione. Come dovrebbe essere chiaro dalla posizione del WAF, questo riceve i dati utilizzando il protocollo HTTP che però viene terminato definitivamente solo sul Frontend Web Server. Quindi la configurazione più semplice per un WAF è quella di Reverse HTTP Proxy[28] in cui il WAF termina la connessione HTTP[29] dell'utente, esamina i dati, e stabilisce una nuova connessione HTTP con il Frontend Web Server. Questa modalità introduce però degli ulteriori carichi e dei ritardi, oltre a mascherare le vere sorgenti degli accessi al Frontend Web Server, per cui di solito si preferisce con-

28) Per convenzione un HTTP Proxy è posizionato di fronte ai client degli utenti, mentre un Reverse HTTP Proxy di fronte ai server HTTP / applicativi.

29) Più precisamente una connessione TCP con protocollo HTTP; per semplicità non vengono qui considerati HTTP/2 e QUIC+HTTP/3.

figurare il WAF come Transparent Reverse HTTP Proxy, modalità in cui il WAF esamina i dati ma non termina le connessioni HTTP, o anche come Transparent Bridge in cui il WAF gestisce i dati a livello 2 Data Link nel modello della pila ISO/OSI (es. Ethernet) e quindi risulta trasparente agli altri componenti del sistema.

Come verrà discusso in seguito, un'altra possibilità è che il WAF riceva solo una copia dei dati e quindi non possa agire sui dati in linea e bloccare la connessione, ma solo inviare degli allarmi.

2.2.2 WAF TRADIZIONALE

In linea di principio, i controlli svolti da un WAF sono facilmente classificabili in due gruppi:

1. verifica dei protocolli utilizzati, a partire da HTTP;
2. verifica dell'assenza di codice nocivo nei dati trasferiti nei linguaggi adottati dal servizio Web che possono comprendere da HTML a XML, Javascript, JSON ecc., ma anche Java, Php, Python, Ruby ecc., ed infine SQL, no-SQL ecc.

E' chiaro che se il primo punto è relativamente semplice, il secondo non lo è per nulla.

D'altra parte il paragone e la similarità con un servizio Anti-Virus è d'obbligo: in entrambi i casi si tratta soprattutto di individuare codice nocivo ed evitare che venga trasmesso o che siano sfruttate delle vulnerabilità (note) dell'applicazione. Infatti il cuore dei

servizi WAF è basato, come per gli Anti-Virus, su librerie di "firme" di attacchi. Come si vedrà più avanti, questa è comunque solo la base delle funzionalità di un WAF.

Ad esempio si assuma che un'applicazione Web sia scritta in Java con un database SQL e che utilizzi pagine HTML con Javascript. Il WAF deve essere configurato con le librerie di firme per attacchi relativi a Java e SQL (ad esempio per attacchi di SQL-Injection), HTML e Javascript. Come per gli Anti-Virus, queste librerie devono essere aggiornate frequentemente in modo da avere disponibili al più presto le firme degli attacchi in corso e delle vulnerabilità appena scoperte per tutti i linguaggi e componenti che compongono il servizio. Tra le librerie di firme devono essere anche presenti (ed aggiornate) tutte le vulnerabilità dello specifico Frontend Web Server, Application Server e Database.

Questo primo livello di funzionalità di un WAF, da alcuni chiamato "WAF Tradizionale", permette comunque di proteggere l'applicazione Web da molti tipi di attacchi.

I WAF possono essere molto utili anche come misura compensativa di protezione in presenza di vulnerabilità di un componente del servizio in attesa dell'applicazione della relativa Patch (questa funzionalità è anche chiamata "Virtual Patching").

Ad esempio si consideri un servizio Web con un Application Server in Java per il quale viene resa nota una vulnerabilità grave e che la soluzione di questa vulnerabilità richieda sia l'aggiornamento di Java sui sistemi, sia l'applicazione di una patch per l'Application Server e la modifica di parte del codice applicativo. Ovviamente

questo processo richiede del tempo per sviluppare e aggiornare il codice, eseguire tutti i test necessari ed installare in produzione la nuova versione. Nel frattempo l'applicazione sarebbe esposta ad attacchi da Internet che sfruttano questa vulnerabilità e che potrebbe portare alla compromissione completa dell'applicazione. Per mitigare questo rischio è sufficiente che sia presente un WAF a protezione dell'applicazione con le firme di questa vulnerabilità. Ovviamente vi è sempre un intervallo di tempo in cui l'applicazione è indifesa da questa vulnerabilità tra il momento della sua scoperta e quello dell'installazione della firma (periodo indicato usualmente come "zero day"), ma è un intervallo di tempo sicuramente molto inferiore (anche ore rispetto a settimane) a quello necessario per installare il Patch completo all'applicazione.

I WAF Tradizionali non sono però solo degli strumenti che verificano ciecamente la presenza o meno delle firme degli attacchi nei dati in transito. Come già indicato, verificano anche la correttezza dell'implementazione dei protocolli e della loro sintassi e possono tipicamente applicare delle logiche basate su regole ("rule-based logic") che permettono sia di concatenare serie di dati e condizioni sia di applicare verifiche solo a particolari tipi di dati. Ad esempio un costrutto potrebbe essere valido se riferito ad un linguaggio di programmazione ma non valido o maligno se riferito ad un altro linguaggio.

Sino ad ora si è descritta la funzionalità del WAF assumendo che in caso di rilevazione di codice maligno o non valido questi blocchi la connessione. In realtà spesso i WAF sono configurati in modo non attivo ma passivo, ovvero solo come un sistema di ri-

levamento di attacchi le cui segnalazioni vengono poi integrate nei sistemi SIEM/SOC dell'azienda. Il motivo principale di questa modalità di utilizzo dei WAF è nel rischio di falsi positivi nel rilevamento di attacchi e quindi del blocco di connessioni valide. Per evitare quindi che il WAF blocchi le connessioni ed in pratica effettui un attacco di Denial of Service al servizio Web, si preferisce alle volte utilizzarlo solo in modalità passiva ed intervenire in caso di reale attacco ed intromissione tramite le procedure standard per la gestione degli incidenti di sicurezza informatica. Questo rischio verrà discusso più in dettaglio di seguito.

Ovviamente, i WAF permettono anche di essere configurati in modalità mista, ovvero sia attiva che passiva, nel qual caso l'amministratore lo configura in modo che certi tipi di rilevazioni siano bloccati mentre altri vengano solo segnalati. Ad esempio una configurazione mista potrebbe prevedere che la connessione sia bloccata in caso di rilevamento di una firma associata ad un attacco o vulnerabilità grave, mentre in caso di vulnerabilità non grave o di violazione della sintassi di un protocollo venga fatta solo una segnalazione al SIEM/SOC. Infatti capita anche che alcune rilevazioni di minore gravità da parte di un WAF siano dovute ad errori di programmazione o configurazione dei servizi applicativi o degli end-point degli utenti che possono ridurre l'efficacia del servizio ma non veramente danneggiarne la confidenzialità, integrità o disponibilità. Risulta pertanto più utile segnalare queste situazioni al gestore dell'applicazione piuttosto che bloccarle direttamente.

2.2.3 IPS E ULTERIORI FUNZIONALITÀ DEI WAF

La descrizione che è stata data finora di un WAF non si discosta molto da quella di un Intrusion Prevention System (IPS). Gli IPS sono nati come evoluzione degli Intrusion Detection System (IDS) che hanno solo capacità passive di tracciamento di pacchetti di rete nocivi, aggiungendo anche la possibilità di blocco del traffico. Gli IPS sono di base dei servizi di sicurezza di rete che esaminano tutto il traffico a partire di solito dal livello 3 Networking nel modello ISO/OSI ma arrivano ad esaminare anche i più alti livelli applicativi (es. livello 7 Application). Anche gli IPS lavorano con "firme" di pacchetti nocivi e possono verificare la sintassi dei protocolli con logiche basate su regole. Quindi se sostituissimo in Fig. 2.2.1 il WAF con un IPS dovrebbe essere possibile intercettare e bloccare buona parte degli attacchi noti ed implementare il Virtual Patching.

Lo scopo dei due strumenti è però molto diverso: un IPS è uno strumento di sicurezza principalmente a livello rete e generico, che può proteggere contemporaneamente molti servizi e applicazioni di tipo diverso. Un WAF invece è uno strumento di sicurezza a livello applicativo e specializzato che può essere personalizzato per proteggere una specifica singola applicazione. Non è detto che un IPS sia in grado di implementare regole di protezione che dipendono molto profondamente dal linguaggio di programmazione adottato nell'applicazione come invece è naturale per un WAF.

La specializzazione e personalizzazione è però un'arma a doppio taglio. Come prima cosa ogni applicazione dovrebbe avere il pro-

prio WAF dedicato, od almeno la propria istanza di WAF personalizzata. Questo da solo richiede sia molto lavoro di configurazione e gestione che l'utilizzo di risorse umane e applicative adeguate, il che include risorse di rete, di calcolo, di memoria senza dimenticarsi dei costi delle licenze delle applicazioni, middleware e sistemi operativi (o appliance fisiche o virtuali) necessarie.

Come sempre quando si progetta un sistema di sicurezza, bisogna decidere quale approccio adottare: bloccare solo il traffico che è noto essere nocivo, o permettere solo il traffico che è considerato lecito. Come ben noto le due logiche sono opposte, la prima è riassumibile in "blocca eventi specifici e permetti tutto il resto" (default-permit) la seconda in "permetti eventi specifici e blocca tutto il resto" (default-block). Tipicamente la logica default-block è utilizzata dai firewall mentre gli IPS utilizzano spesso la logica default-permit. Nell'accezione più avanzata, un WAF dovrebbe adottare la logica default-block, ma questo richiederebbe di specificare esattamente tutti i contenuti leciti che possono transitare alla e dall'applicazione.

Due semplicissimi esempi possono dare l'idea di quanto sia complesso questo problema. Si consideri un sito Web teorico con due pagine banali e super semplificate[30], la prima statica con solo testo HTML e qualche immagine, la seconda con un Form HTML, Javascript e la possibilità per l'utente di inviare tramite il Form dei dati costituiti solo da lettere, spazi e numeri. Quando l'utente ri-

30) Questa descrizione ignora molti dettagli, a partire dalla presenza di link nelle pagine HTML ecc.

chiede la prima pagina, il WAF dovrebbe verificare che la richiesta contiene solo il URL esatto senza alcun parametro o dato allegato e che la pagina sia richiesta solo con il metodo GET. Inoltre la risposta dell'applicazione dovrebbe contenere solo testo HTML e immagini, nessun altro tipo di dati quali documenti, codice Javascript o altro. La seconda pagina è più complessa, inizialmente l'utente la richiede in modalità GET come la prima pagina e con controlli simili, riceve però dall'applicazione un Form HTML e codice Javascript: il WAF deve quindi permettere il transito solo di questi tipi di dati. L'utente compila il Form e lo invia all'applicazione con il metodo POST. Il WAF deve permettere l'invio all'applicazione solo dei campi previsti dal Form e solo compilati con dati del tipo previsto: ad esempio se sono presenti caratteri di punteggiatura, la comunicazione deve essere bloccata. Per comodità dell'utente, il codice Javascript può essere utilizzato nell'interfaccia cliente sul Browser per verificare che i dati siano permessi prima dell'invio all'applicazione. La verifica deve comunque essere fatta dal WAF (e dall'applicazione) in quanto l'utente può aggirare il codice Javascript ed inviare campi con qualunque dato voglia. Infine il WAF deve verificare che l'applicazione invii all'utente solo i dati previsti come conclusione della transazione, ad esempio solo testo HTML.

Ovviamente non è pensabile poter creare e gestire manualmente una configurazione WAF con questo livello di dettaglio. Una possibile alternativa e via di mezzo tra i due approcci default-permit e default-block è per primo di bloccare i dati nocivi basandosi sulle "firme" e sulla verifica della sintassi dei protocolli, linguaggi di programmazione, applicazioni e database adottati, poi aggiungere regole abbastanza generiche e ampie che permettono il passag-

gio di dati ritenuti non nocivi per la specifica applicazione, ed infine bloccare tutto il resto. L'approccio è sempre alla default-block, ma la presenza di regole ampie che permettono il traffico, un po' come nell'approccio più tradizionale della configurazione dei Firewall, rende questa configurazione meno dettagliata e personalizzata e quindi più facilmente gestibile.

2.2.4 NEXT GENERATION WAF

Sinora abbiamo descritto i WAF Tradizionali, i loro principali punti di forza ed alcune delle loro criticità, ma come ormai comune nel linguaggio commerciale, i WAF evolvono continuamente in nuove "generazioni".

Come abbiamo indicato, la gestione di un WAF non è sempre cosa facile e per questo ormai da tempo sono sorte soluzioni Cloud, ovvero servizi WAF erogati in modalità Software-as-a-Service (SaaS). Le modifiche all'architettura descritta in Fig. 2.2.1 sono in realtà minime, come si vede in Fig. 2.2.2.

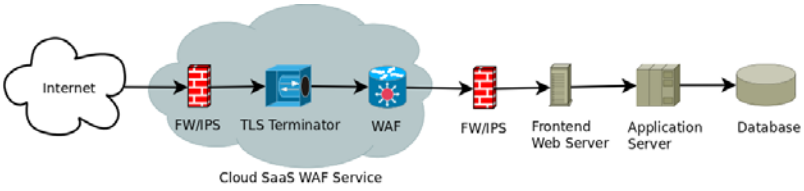

Figura 2.2.2: Architettura logica di un servizio Web con WAF in ambiente Cloud SaaS

I punti di forza di un servizio WAF SaaS sono molteplici e simili in generale a quelli di altri servizi SaaS:

- la gestione dell'infrastruttura (network, hardware, software) è in carico al fornitore
- gli aggiornamenti continui del WAF e la sua manutenzione sono in carico al fornitore che sfrutta notevoli economie di scala garantendo al contempo che l'aggiornamento delle firme e delle configurazioni sia molto più veloce e comprensivo per tutti i clienti
- tipicamente il fornitore offre modalità di scalabilità delle prestazioni non possibili quando il WAF è on-premises in appliance hardware, e non facilmente replicabili o così velocemente implementabili per appliance software
- il fornitore ha visibilità diretta del traffico di tutti i clienti e quindi ha la possibilità di valutare se anomalie di traffico sono eventi locali e veramente anomali o se invece sono nuove campagne di attacco che vengono man mano indirizzate a tutti i servizi Web
- il fornitore è tipicamente in grado anche di offrire servizi anti-Distributed-Denial-of-Service (DDoS) specifici per i servizi Web protetti dai WAF, protezione difficilmente replicabile per i singoli servizi Web con le soluzioni WAF on-premises.

Ovviamente valgono anche i comuni aspetti negativi delle soluzioni Cloud SaaS, a partire dalle a volte limitate possibilità di personalizzazione del servizio.

Una delle principali difficoltà di gran parte delle misure di sicurezza attiva è quella di fronteggiare gli zero-day, ovvero il periodo di tempo tra la scoperta di una vulnerabilità e la disponibilità di una misura per contrastarla, tipicamente aggiornando l'applicazione. Come già descritto, un WAF è utile contro gli zero-day come soluzione di Virtual Patching ma solo se è in grado di individuare gli attacchi che sfruttano la vulnerabilità. Oltre alla velocità nella creazione e distribuzione delle firme, i WAF più recenti adottano alcuni ulteriori approcci.

Il primo consiste nella classificazione del traffico che normalmente viene gestito dall'applicazione e dalla creazione di una policy o di un profilo che lo descrive[31]. In altre parole, osservando il traffico ritenuto "normale" e quindi lecito, il WAF ne crea automaticamente una descrizione che traduce in regole che ne permettono il traffico. Viene bloccato tutto il traffico che non rispetta questa descrizione o che fa scattare una regola che descrive traffico maligno (come descritto nella sezione precedente).

Questo approccio rende automatica la creazione della soluzione descritta nella sezione precedente e che dovrebbe essere ottimale: dovrebbe infatti permettere solo il traffico lecito, desunto dal traffico noto, e bloccare tutto il resto. Il problema è che quasi sempre il traffico noto è un sottoinsieme molto ridotto del traffico lecito, e questo per molteplici motivi tra cui:

- il modo in cui gli utenti interagiscono con l'applicazione può

31) Questo approccio è anche chiamato "Application Learning".

essere molto vario sia nei contenuti sia nell'ordine delle azioni e nelle modalità

- gli utenti possono connettersi all'applicazione Web con strumenti software diversi (ad esempio Mobile App, Browser Web su smartphone, tablet e PC, accessi via API o programmatici ecc.) ed anche solo una nuova versione di un Browser Web può introdurre modifiche al comportamento del Client rispetto all'applicazione
- le applicazioni sono in costante evoluzione (si pensi solo ai metodi Agile di sviluppo) ed introducono costantemente e continuamente nuove funzionalità e dati, il che può in breve tempo rendere superata la descrizione del traffico fatta dal WAF.

Conseguenza di tutto ciò è un alto rischio di blocco di traffico lecito (ovvero di falsi positivi). Per alcune, poche, applicazioni di nicchia e ad alti requisiti di sicurezza, il blocco di traffico lecito può anche essere accettato (se in limitate quantità), ma non può essere accettato per nulla nella maggioranza delle applicazioni di business, dal commercio elettronico ai siti finanziari, di informazione ecc. E' questo il principale motivo per cui i WAF spesso sono utilizzati solo in modalità passiva, ovvero di tracciamento di eventuali attacchi ma non di loro blocco.

Un altro approccio alla configurazione e gestione di un WAF parte dalla creazione di un profilo di rischio dell'applicazione: siti di commercio elettronico, finanziario o di informazione sono soggetti a minacce di tipo diverso e soprattutto a rischi diversi relativi ai

differenti dati e transazioni gestite. Questo permette di creare regole diverse a seconda del profilo di rischio, molto strette su un certo tipo di traffico rilevante per l'applicazione e meno strette per tutto il resto. Questo riduce i falsi positivi che nel caso sono maggiormente possibili per transazioni a maggior rischio per le quali il business può essere più disposto ad accettare il danno conseguente al blocco della transazione. D'altra parte questo permette maggior traffico non lecito, e quindi riduce la capacità del sistema di bloccare eventuali zero-day.

Infine, come in tutti i sistemi di sicurezza di rete, sono anche utili gli approcci che integrano nei WAF sistemi di Intelligenza Artificiale, o più precisamente Machine Learning. L'approccio è molto simile a quello appena descritto di creazione di un profilo di traffico lecito, solo che in questo caso si utilizza un modello ad esempio di Deep Learning che viene addestrato utilizzando traffico lecito e non lecito. Il modello poi è in grado di estrapolare i dati che ha analizzato individuandone le caratteristiche principali in modo da poter correttamente classificare anche dati che non gli sono mai stati sottoposti. Questi modelli tipicamente ricevono in ingresso in blocchi il traffico diretto all'applicazione e indicano la probabilità che un blocco sia nocivo. Vengono poi definite delle soglie, ad esempio il 90% di probabilità, che possono dipendere dal tipo di dato o dalla catena di regole che gli si applicano, al superamento delle quali i dati sono classificati come nocivi dal WAF. I modelli di Machine Learning, in combinazione con gli altri approcci utilizzati dai WAF, da una parte permettono sia di ridurre sia i falsi positivi dovuti a traffico lecito ma nuovo, sia i falsi negativi, ovvero dati nocivi non identificati, si potrebbe dire per analogia con i dati noti.

Comunque pur migliorando l'efficacia dei WAF, i modelli di Machine Learning non eliminano del tutto falsi positivi e falsi negativi, sia per la loro natura intrinseca probabilistica, sia perché anch'essi dipendono comunque essenzialmente dai dati leciti e nocivi utilizzati per la loro istruzione.

2.2.5 EFFICACIA REALE DEI WAF

Quanto efficaci sono i WAF realmente? Uno studio di Ponemon [Rif. 2] del 2019 indica che poco più della metà degli intervistati non è soddisfatta dell'efficacia dei propri WAF e che alcuni attacchi reali non sono stati identificati dai WAF stessi. Il costo e la complessità di gestione sono anche indicati come fattori negativi. Infine solo 1/5 degli intervistati utilizza i WAF in modalità di blocco, in quanto ritiene che il rischio di falsi positivi sia troppo alto.

D'altro canto, sarebbe interessante avere statistiche su quanti attacchi sono stati bloccati o segnalati correttamente dai WAF, e quindi quanti incidenti o intrusioni sono state evitate. Non c'è dubbio che i WAF non sono strumenti perfetti, come tutti gli strumenti di sicurezza informatica, e che sono difficili e spesso costosi da gestire. E' pertanto difficile fare un rapporto costo-benefici, o meglio tra costi e potenziali danni, per l'utilizzo dei WAF.

I WAF sono comunque uno strumento molto utile per la sicurezza delle applicazioni Web esposte in Internet, ed i continui sviluppi li stanno rendendo sempre più efficaci e più semplici da utilizzare.

2.2.6 RIFERIMENTI BIBLIOGRAFICI

Rif. 1: ModSecurity: https://www.modsecurity.org/

Rif. 2: "The State of Web Application Firewalls" https://ponemonsullivanreport.com/2019/07/the-state-of-web-application-firewalls/

2.3 QUIC: un nuovo protocollo Internet per la sicurezza IT

E' possibile che tra non molto cambi buona parte del traffico Internet, a partire da quella generata dai Browser Web. Questo è dovuto all'arrivo di un nuovo protocollo di comunicazione chiamato QUIC (inizialmente era l'acronimo di "Quick UDP Internet Connections", ma ora è solo un nome, si vedano [Rif. 1, 2]) che sarà inizialmente utilizzato dalla terza versione di HTTP, HTTP/3 [Rif. 1].

In questa sede descriveremo molto brevemente QUIC, la sua collocazione tra i protocolli di rete, i principali motivi che hanno portato alla sua creazione, le principali novità che introduce e le ripercussioni che potrebbe avere per la sicurezza delle comunicazioni.

2.3.1 HTTPS E LO STACK DI RETE

Per chiarire dove si colloca QUIC e qual è il suo scopo, è meglio partire da un breve ripasso di come è organizzata una connessione HTTPS secondo la pila ISO/OSI[32]:

Livello	Protocollo	Note
Application	HTTP	L'applicazione Web che utilizza HTTP spesso stabilisce una sessione ad esempio a seguito di un accesso e utilizzando Cookies
Presentation	TLS	Viene stabilita una sessione
Transport	TCP	Viene stabilita una sessione
Network	IP	

Tabella 2.3.1: HTTPS nella pila ISO/OSI

Una vista forse più interessante è quella che considera il processo di incapsulamento dei dati quando passano dall'applicazione sino al livello fisico di trasmissione come descritto sommariamente in Fig. 2.3.1:

32) Il livello "5. Session" non è utilizzato in questa descrizione di HTTPS, si veda anche [Rif. 4], e non sono indicati esplicitamente i livelli inferiori "1. Physical" e "2. Data Link".

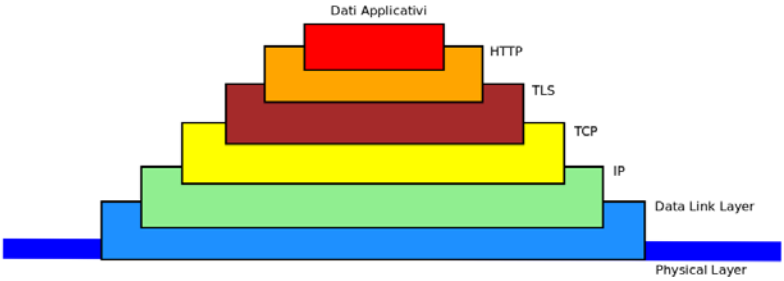

Figura 2.3.1: Incapsulamento dei dati per HTTPS.

L'incapsulamento nei vari protocolli ai differenti livelli è sicuramente il motivo della fantastica duttilità delle comunicazioni IT ove ad ogni livello, eccetto al più al livello di Network IP, può essere utilizzato un protocollo diverso a seconda dell'applicazione, del tipo di comunicazione o di rete fisica. Ad esempio, in Fig. 2.3.1 si è riportato il caso di HTTPS, ma un diagramma simile per HTTP è identico a parte l'esclusione di TLS.

2.3.2 DA HTTP/1.1 A QUIC

Non è detto però che l'incapsulamento, a cui non si vuole e non si può rinunciare, costituisca sempre la soluzione più efficace, efficiente e semplice da implementare. Questo è maggiormente possibile quando sono utilizzati dei protocolli generici, ovvero non orientati e adattati specificatamente all'applicazione che li utilizza.

Già in Tabella 2.3.1 è riportato un primo segnale di inefficienza: una

comunicazione Web verso un'applicazione tipicamente richiede la gestione di sessioni, ovvero la gestione dello stato della comunicazione, a livello Applicativo, TLS e TCP. Visto che la sessione applicativa deve essere gestita direttamente dall'applicazione, una possibile idea per semplificare la comunicazione potrebbe essere di unificare la gestione delle sessioni TLS e TCP.

In realtà il problema pratico che forse ha contribuito di più alla nascita di QUIC nel 2012 da parte Jim Roskind (Google) è quello chiamato tecnicamente come "multiplexing without head-of-line blocking", ovvero un problema di Disponibilità.

Una descrizione semplificata del problema è la seguente. Una pagina Web tipicamente contiene molti elementi, ad esempio immagini, codice Javascript, Frame HTML ecc. ognuno dei quali ha un proprio indirizzo (URL). La soluzione più semplice è di far richiedere dal Client Web e far inviare dall'applicazione i componenti in serie, uno dopo l'altro, utilizzando la stessa sessione TCP (e TLS). Questo però sicuramente non è efficiente. Si possono stabilire sessioni TCP (e TLS) parallele per scaricare i componenti della pagina, ma queste comportano un ulteriore carico sui server Web, che anzi si vorrebbero alleggerire.

In HTTP/1.1 è possibile utilizzare il metodo Pipelining, ovvero l'invio dal Client di molteplici richieste GET in sequenza senza attendere i dati di risposta di ognuna. L'uso di questo metodo dovrebbe velocizzare la trasmissione dei dati ma in pratica l'implementazione nei Browser è risultata problematica e poco efficiente per cui la quasi totalità dei Browser non lo utilizza.

HTTP/2 ha introdotto la possibilità di gestire molteplici Stream di richieste e di trasferimenti dati all'interno della stessa connessione TCP (e TLS) utilizzando Multiplexing ed altre ottimizzazioni quali la compressione degli Header con HPACK [Rif. 5][33]. HTTP/2 permette quindi che si instaurino tra server e client sullo stesso canale di comunicazione molteplici e concorrenti sessioni di comunicazione[34]. Tipicamente l'utilizzo di HTTP/2 permette di velocizzare il caricamento di una pagina Web dal 10% al 50%.

2.3.3 PERCHÉ QUIC

HTTP/2 non risolve però il problema del "head-of-line blocking" o più precisamente del "TCP head-of-line blocking". Come indicato, HTTP/2 utilizza una sola connessione TCP e nel caso si perda un pacchetto TCP durante la trasmissione, tutti gli Stream HTTP/2 sono bloccati e devono essere ritrasmessi almeno in parte. La resilienza di TCP e la capacità di recuperare una sessione interrotta, in questo caso generano un problema di efficienza.

QUIC si prefigge i seguenti cinque obiettivi principali:

1. Implementare il "multiplexing without head-of-line blocking";

33) L'implementazione di HPACK in QUIC è chiamata QPACK ed è uno degli standard RFC QUIC.

34) HTTP/2 non richiede l'utilizzo di TLS, ma la pratica comune dei Browser è di utilizzarlo solo insieme a TLS.

2. Minimizzare i tempi di creazione di una connessione e di trasporto dei dati per qualunque applicazione, prendendo HTTP come applicazione d'esempio;

3. Potersi implementare solo a livello applicativo, ovvero senza cambiare protocolli di rete e sistemi operativi;

4. Permettere la comunicazione con percorsi diversi su rete (es. il cambio dinamico di indirizzi IP del Client) e la gestione degli errori di comunicazione;

5. Garantire sempre la sicurezza delle comunicazioni, per Confidenzialità e Integrità, tramite l'adozione ed inclusione di TLS (specificatamente al momento TLS-1.3).

QUIC non è una nuova versione di HTTP, e la sua collocazione nella pila dei protocolli di rete è descritta sommariamente in Fig. 2.3.2:

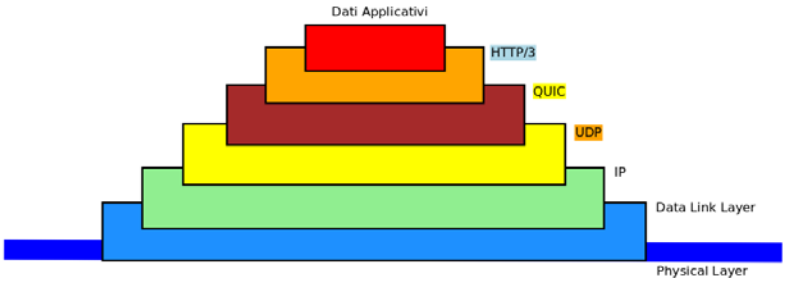

Figura 2.3.2: Posizionamento di QUIC nella pila di incapsulamento

QUIC è posizionato a livello "6. Presentation" al posto di TLS, utilizza UDP a livello "4. Transport" ed in questo esempio è richiamato da HTTP/3 a livello "7. Application". HTTP/3 è l'evoluzione di HTTP/2 in grado di utilizzare tutte le funzionalità di QUIC: in particolare alcune funzionalità di HTTP/2 sono ora implementate da QUIC. HTTP/3 sarà solo il primo, e probabilmente il più importante, protocollo ad utilizzare QUIC, ma altri protocolli o applicazioni potranno implementarlo e utilizzarlo.

Infatti, la prima osservazione è che QUIC, come dal terzo requisito, è implementabile a livello applicativo, anche tramite librerie dedicate. Anzi ci si aspetta che librerie che oggi forniscono TLS (es. Openssl) implementino anche QUIC con TLS (oggi TLS-1.3).

La seconda osservazione è che QUIC utilizza UDP e non TCP, quindi un protocollo di trasporto che non gestisce la connessione, ogni pacchetto è inviato e non viene tracciato il suo stato, ovvero se è ricevuto correttamente ed in quale ordine. La gestione della connessione è fatta da QUIC che integra la gestione di TLS e della sicurezza del trasferimento dei dati con la gestione di Stream concomitanti (come in HTTP/2).

2.3.4 LA STRUTTURA DI QUIC

La struttura di un pacchetto dati trasferito da QUIC è abbastanza complessa:

- 1 Datagramma UDP contiene 1 o più QUIC Packet;

- Ogni QUIC Packet contiene 1 o più Frame;
- Ogni Frame contiene dati di 1 Stream;
- Ogni Connessione ha 1 o più Stream.

Tutti i dati sono protetti con TLS, i dati applicativi sono sempre cifrati mentre i dati necessari per la comunicazione sono verificati solo per l'Autenticità e l'Integrità[35]. La Figura 2.3.3 cerca di descrivere questa organizzazione dei dati di QUIC:

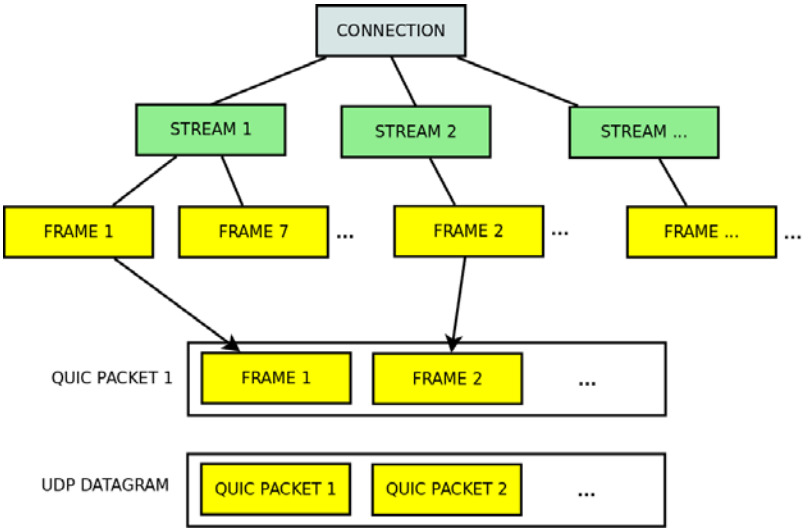

Figura 2.3.3: organizzazione dei dati in un pacchetto di rete secondo il protocollo QUIC

35) I dati nei pacchetti sono cifrati utilizzando il protocollo "Authenticated Encryption with Associated Data" (AEAD) che protegge non solo i dati trasportati ma anche gli Header dei pacchetti.

QUIC suggerisce che le applicazioni inseriscano in un Datagramma UDP solo QUIC Packet contenenti Frame che appartengono allo stesso Stream, questo per risolvere il problema del "multiplexing with head-of-line blocking". QUIC lascia però la possibilità a chi utilizza il protocollo di mischiare Frame e QUIC Packet di Stream diversi (vedi Fig. 2.3.3) nel caso in cui sia ben chiaro allo sviluppatore che se si perde il pacchetto UDP, due o più Stream saranno impattati e dovranno inviare di nuovo i dati. Vi possono essere casi, soprattutto se i Frame sono piccoli, in cui può risultare più conveniente inviare un solo Datagramma UDP con dati di diversi Stream piuttosto che molteplici piccoli Datagrammi UDP ognuno con i dati di un solo Stream.

Ovviamente la macchina a stati di QUIC mantiene lo stato di invio di ogni Frame di ogni Stream implementando un proprio meccanismo di conferma della ricevuta dei dati (ACK o acknowledgment) ed eventuale re-invio, visto che UDP a differenza di TCP non gestisce le connessioni.

La sicurezza dei dati, garantita da TLS (oggi TLS-1.3), è integrata in QUIC in modo che non interferisca con le caratteristiche di Disponibilità ed efficienza nella gestione delle connessioni. Ad esempio, ogni QUIC Packet è cifrato indipendentemente in modo da garantire che ogni Datagramma UDP possa essere decifrato indipendentemente. Questa garanzia non c'è per HTTPS/1.1 e HTTPS/2 perché TLS e TCP agiscono a livelli diversi ed indipendentemente.

Oltre a Multiplexing, per velocizzare il trasferimento di dati, QUIC, come già HTTP/2, offre anche la possibilità per l'applicazione (server) di fare Push dei dati verso il Browser (o in generale il Client).

Il caso della navigazione Web è molto semplice da descrivere. Come già indicato, un Browser richiede ad un server Web una pagina, ma questa è composta da molti elementi, ad esempio immagini e codice Javascript, ognuno dei quali ha un proprio indirizzo (URL). L'applicazione sa che per rendere la pagina, il Browser avrà bisogno non solo del codice HTML richiesto ma anche di tutti questi ulteriori elementi. Per velocizzare il trasferimento, l'applicazione può inviare al Browser tutti gli elementi necessari senza che il Browser li richieda esplicitamente, riducendo quindi il tempo necessario a trasferire tutti i dati.

2.3.5 LA COMPLESSITÀ DI QUIC

Come abbiamo brevemente descritto, QUIC deve fare molte cose contemporaneamente. Ed è proprio il fare tutte queste cose contemporaneamente che gli permetterà di rendere più veloce e, da un certo punto di vista, più semplice, trasferire i dati tra applicazione e dispositivo utente. Un breve riassunto (molto incompleto) delle attività contemporanee di QUIC è:

1. Garantire sempre la sicurezza delle comunicazioni, per Confidenzialità, Integrità e Autenticità, tramite la protezione di tutto il traffico con TLS (TLS-1.3);

2. Gestire un'unica sessione tra Client e Server di tutti gli Stream paralleli attivi (ad eccezione di eventuali sessioni applicative a seguito di autenticazione e accesso dell'utente al servizio); questo include la gestione degli errori, la ritrasmissione dei

dati persi e l'integrazione di TLS;

3. Permettere la comunicazione con percorsi diversi su rete tracciando e gestendo eventuali cambi di indirizzo IP del Client (cambio di NAT, cambio di cella telefonica per uno smartphone ecc.) e ri-autenticando automaticamente e molto velocemente il Client con il supporto di TLS[36];

4. Gestire la congestione delle comunicazioni (Congestion Control): la perdita di molti pacchetti UDP è di norma dovuta ad una linea lenta e congestionata; QUIC quindi, soprattutto lato Server, può rallentare l'invio di dati per diminuire la perdita di pacchetti;

5. Gestire tutte le componenti di trasmissione dati che possono essere impattate dall'adozione di QUIC, ad esempio Proxy Web e Load Balancer in caso di cambio dinamico di indirizzi IP del Client.

In pratica questo richiede di re-implementare parte di HTTP/2, tutto TLS e buona parte di TCP in un unico standard.

Questa complessità si riflette direttamente nel lavoro del QUIC Working Group IETF [Rif. 1] che sta preparando ben nove standard e ad esempio la versione numero 34, consiste di 510 pagine in totale.

Vista la complessità, lo sviluppo dei protocolli va di pari passo con

36) QUIC utilizza un protocollo di rinegoziazione molto veloce chiamato 0-RTT.

l'implementazione ed i test da parte di un consorzio che comprende praticamente tutti i grandi produttori di Browser e HTTP Server (o, più precisamente, delle librerie utilizzate da Browser e HTTP Server). Vengono verificati sia i protocolli sia l'interoperabilità delle implementazioni [Rif. 2]. Lo sviluppo dei protocolli è quindi incrementale con un approccio che si può definire "Agile" e che ha lo scopo di produrre alla fine un protocollo efficace, efficiente e corretto insieme alle relative implementazioni.

Il lavoro del QUIC Working Group IETF è incominciato nel 2016, dopo i primi sviluppi di Google a partire dal 2012, e la prima versione completa "QUIC version 1" è stata pubblicata a maggio 2021 come RFC 9000 insieme ad altri sei RFC: 8999, 9001, 9002, 9114, 9204, 9221. I tempi di adozione sono stati molto rapidi visto che sia i Browser che i Server Web già lo implementavano e lo utilizzavano nelle versioni di sviluppo e produzione (anche se alle volte non era abilitato per default).

2.3.6 IMPLEMENTAZIONI E PRESTAZIONI

Come già indicato, QUIC viene al momento utilizzato per le connessioni Web, ovvero HTTP. Browser e Server Web saranno però per un lungo periodo retro-compatibili con HTTP/1.1 e HTTP/2. Il motivo più semplice è che QUIC utilizza la porta UDP/443 mentre HTTP/1.1 e HTTP/2 utilizzano TCP/80 e TCP/443. Nel caso la porta UDP/443 sia chiusa, o vi sia un disallineamento tra Client e Server, o uno dei due non supporti QUIC, Browser e Server Web ritornano ad utilizzare i protocolli esistenti. Questo è anche il motivo per cui

facendo oggi delle prove, come descritto di seguito, può capitare che componenti di pagine Web sono scaricati con QUIC, altri con HTTP/1.1 o HTTP/2.

Per quanto riguarda le prestazioni, una rapida ricerca in Internet riporta risultati molto contraddittori. In alcuni casi sono state riscontrate prestazioni nettamente migliori, ad esempio [Rif. 6] riporta prestazioni migliori di circa 10% di QUIC rispetto a HTTP/2 e [Rif. 7] prestazioni migliori dal 10% al 30% di QUIC rispetto a HTTP/1.1 (mentre in questo caso HTTP/2 si comporta peggio di entrambi). Google [Rif. 8] riporta un miglioramento rispetto a HTTP/1.1 del 2% in Google Search, 9% in Youtube, 3% sui Client Desktop e 7% sui Client Mobile. In altri casi sono stati rilevate prestazioni praticamente uguali o leggermente peggiori di QUIC rispetto a HTTP/1.1 o HTTP/2.

Vi sono vari motivi per queste discrepanze, innanzitutto ci sono state molte diverse versioni di sviluppo di QUIC, sia quelle standard IETF sia quelle Google, ed essendo appunto versioni di sviluppo contengono funzionalità diverse o implementate diversamente proprio per verificarne le prestazioni ed il comportamento. Inoltre l'aspettativa è che QUIC sia molto più performante dei protocolli attuali non in tutte le situazioni ma in quelle più critiche e problematiche; che riesca come prima cosa a ridurre il carico dei Server Web e di conseguenza possa migliorare anche le prestazioni dei Browser sui Client. Va però segnalato che in condizioni di utilizzo normali l'utente non noterà praticamente alcuna differenza essendo il guadagno troppo piccolo per potersi notare. Quello che dovrebbe invece succedere è che, in situazioni non ottimali di

connessione, la navigazione sarà più fluida, con meno blocchi o rallentamenti.

2.3.7 QUIC E SICUREZZA IT

Vi sono almeno quattro aspetti da considerare dal punto di vista della sicurezza IT per QUIC:

1. La sicurezza del protocollo;
2. La sicurezza delle implementazioni;
3. La sicurezza nelle interazioni con gli altri componenti della rete IT;
4. La sicurezza fornita nelle comunicazioni IT.

Molti studi sono stati fatti anche di recente (si vedano ad esempio [Rif. 9, 10]) sulla sicurezza del protocollo e della sua implementazione di TLS, questo a garanzia delle caratteristiche di Confidenzialità, Integrità e Autenticità dei dati. Si può dire che almeno a livello teorico, il protocollo è nato ed è stato sviluppato mantenendo la sicurezza come uno dei principali requisiti. Ci si aspetta quindi che non vi saranno problemi di sicurezza dovuti al protocollo stesso.

Diverso è il problema delle implementazioni. Come abbiamo descritto il protocollo è molto ampio e complesso, comprendendo allo stesso tempo parte di HTTP/2, tutto TLS e buona parte di TCP. Tenendo conto che tutti gli aspetti di sicurezza delle comunicazio-

ni sono gestite da QUIC, quanti e quali saranno i fraintendimenti o gli errori di implementazione che avranno ricadute dirette sulla sicurezza? Speriamo nessuno, ma la storia dello stack TCP/IP, nato 47 anni fa, e le vulnerabilità che appaiono ancora, per fortuna raramente, non ci induce ad essere del tutto ottimisti (si vedano ad esempio le vulnerabilità descritte in [Rif. 11]).

D'altra parte visto che QUIC è implementato in librerie applicative o direttamente nelle applicazioni e non nei sistemi operativi, fa sperare che il processo di Patching e Update possa essere semplice e veloce. Purtroppo vi sono implementazioni diverse, il che aumenta il rischio che qualcuna possa avere delle vulnerabilità che impattano anche pochi sistemi, e che in alcuni casi il processo di Patching e Update possa essere complesso o lento o non applicato.

Come ben noto, il problema principale di Patching e Update di sicurezza non sono le applicazioni Server, i Browser Web o le applicazioni Desktop, ma gli infiniti sistemi IoT che sono difficilmente se non impossibili da aggiornare.

QUIC è un nuovo protocollo di rete, e quindi deve essere recepito in tutti i dispositivi di rete che interagiscono dal livello 4 "Transport" in su. La prima osservazione è che QUIC utilizza UDP/443 mentre HTTP ha sempre utilizzato TCP/80 e TCP/443. Quindi i Firewall devono essere configurati per permettere il traffico sulla porta UDP/443. Questo fornisce anche un modo semplice per bloccare QUIC, basta bloccare la porta UDP/443 (facendo però attenzione a che non sia utilizzata da altre applicazioni, come ad esempio OpenVPN).

In generale comunque tutti i dispositivi di rete che agiscono sul traffico dal livello 4 “Transport” in su, come Firewall, IPS, ProxyWeb, WAF, Bilanciatori ecc., dovranno essere aggiornati in modo da riconoscere il traffico QUIC e gestirlo di conseguenza. Ovviamente questa è una problematica solo transitoria e che se la porta UDP/443 è chiusa, non porterà alcun cambiamento, cioè nessun utilizzo di QUIC, sino all’aggiornamento e riconfigurazione dei sistemi di rete.

Infine l’adozione di QUIC porta anche degli impatti positivi sulla sicurezza delle comunicazioni IT rispetto alla situazione attuale.

L’utilizzo sempre di TLS garantisce che tutte le comunicazioni siano sempre protette per Confidenzialità, Integrità ed Autenticità mentre il protocollo stesso è stato disegnato per migliorare le caratteristiche di Disponibilità delle comunicazioni soprattutto in situazioni non ottimali o critiche. Inoltre QUIC può essere utilizzato da qualunque applicazione, non solamente per il traffico Web e HTTP, possibilmente ampliando il numero di applicazioni e i tipi di informazione che sono protetti.

Parafrasando una nota affermazione, possiamo dire che QUIC mira a implementare la sicurezza nelle comunicazioni IT “by design and by default”. D’altra parte va anche evidenziato che rispetto ad un attuale corretto uso di HTTP+TLS+TCP, i benefici saranno ottenuti più dai fornitori di servizi IT che dagli utenti.

2.3.8 PROVARE QUIC

Per concludere, visto che i principali fornitori di servizi Web hanno abilitato QUIC sui propri servizi, è ormai possibile utilizzare QUIC anche se non tutti i siti Web lo supportano. Si noti che anche i principali Browser Web supportano QUIC, ma non sempre è abilitato per default.

Ad esempio rifacendosi a Google, che comunque è sempre il padre del progetto, si può procedere come segue:

- accertarsi che la porta UDP/443 sia aperta dal proprio dispositivo verso Internet
- aprire il browser Chrome (o Chromium)
- aprire la pagina "chrome://flags/", cercare "QUIC" e metterlo "Enabled" se non lo è già (il "Default" dovrebbe essere "Enabled")
- dal menu aprire i "Developer Tools" (o CTRL-SHIFT-I), selezionare "Network" ed assicurarsi che la colonna "Protocol" sia presente nel frame in basso (altrimenti cliccare con il tasto destro sul nome di una delle colonne presenti ed aggiungerla)
- infine visitare una pagina di Google, come nell'esempio in Fig. 2.3.4 in cui la colonna "Protocol" riporta l'indicazione "h3" ovvero che è stato utilizzato HTTP/3 su QUIC [Rif. 8] .

Una procedura simile può essere utilizzata con un altro Browser e su altri siti abilitati a QUIC.

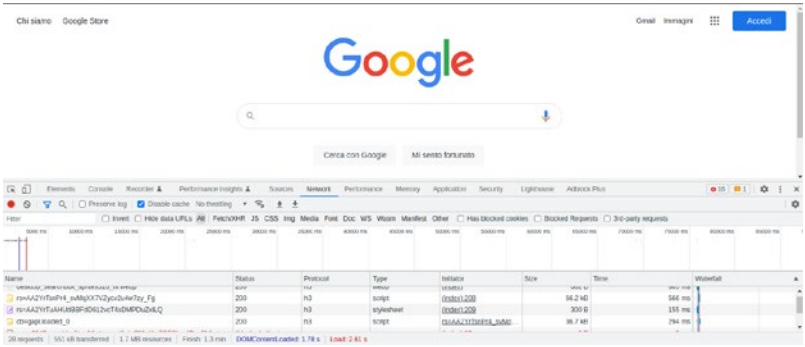

Figura 2.3.4: Esempio di navigazione Web con QUIC sulla pagina di Google Search

3.3.9 RIFERIMENTI BIBLIOGRAFICI

Rif. 1: IETF QUIC Working Group, https://quicwg.org/ e https://datatracker.ietf.org/wg/quic/about/

Rif. 2: Implementazioni di QUIC disponibili a https://github.com/quicwg/base-drafts/wiki/Implementations , l'interoperabilità è descritta in https://interop.seemann.io/

Rif. 3: HTTP/2 RFC 7540, https://tools.ietf.org/html/rfc7540 (ed anche https://en.wikipedia.org/wiki/HTTP/2)

Rif. 4: per ISO/OSI si veda anche https://en.wikipedia.org/wiki/OSI_model e https://en.wikipedia.org/wiki/List_of_network_protocols_(OSI_model)

Rif. 5: HPACK RFC 7541, https://tools.ietf.org/html/rfc7541

Rif. 6: Cloudflare "Comparing HTTP/3 vs. HTTP/2 Performance", https://blog.cloudflare.com/http-3-vs-http-2/

Rif. 7: Uber Engineering "Employing QUIC Protocol to Optimize Uber's App Performance", https://eng.uber.com/employing-quic-protocol/

Rif. 8: Chromium Blog "Chrome is deploying HTTP/3 and IETF QUIC". https://blog.chromium.org/2020/10/chrome-is-deploying-http3-and-ietf-quic.html

Rif. 9: Samuel Jero "QUIC: Performance and Security at the Transport Layer", https://www.ietfjournal.org/quic-performance-and-security-at-the-transport-layer/

Rif. 10: QUIC Privacy and Security Workshop 2020, https://www.ndss-symposium.org/ndss2020/quic-privacy-and-security-workshop/

Rif. 11: Forescout "Number Jack - Weak ISN Generation in Embedded TCP/IP Stacks", https://www.forescout.com/company/resources/numberjack-weak-isn-generation-in-embedded-tcpip-stacks/

03

PRIVACY E SICUREZZA

3.1 DNSSEC e alcuni aspetti di Privacy e Sicurezza

Il Domain Name System, d'ora in avanti DNS, è uno dei protocolli fondamentali alla base di Internet. Il suo scopo è quello di risolvere un "nome" facilmente ricordabile e gestibile da una persona, in un "indirizzo IP" associato alla risorsa informatica da raggiungere. Oltre a permettere una facile e semplice gestione da parte delle persone, la separazione tra "nome" e "indirizzo IP" permette anche di risolvere un nome in più indirizzi, ad esempio a seconda della geo-localizzazione delle risorse, e di modificare gli indirizzi in maniera trasparente agli utenti.

Tutti noi utilizziamo la risoluzione dei nomi tramite il DNS ogni qualvolta accediamo a risorse in internet, ed anche in reti private aziendali. Noi tutti ci fidiamo che la risoluzione ci fornisca gli indirizzi corretti delle risorse a cui ci vogliamo connettere, e non ad esempio di qualche sito di phishing.

Nel tempo vari aspetti dell'integrità, disponibilità e riservatezza del

DNS sono stati messi a dura prova [Rif. 1]. In questo capitolo viene introdotto velocemente il protocollo DNS per poi brevemente trattare di alcuni aspetti di sicurezza, inclusa l'introduzione tuttora in corso del DNSSEC ed alcuni approcci alternativi.

3.1.1 DNS, DOH E PRIVACY

Prima di una breve introduzione su come funziona la risoluzione DNS, vogliamo però accennare ad un argomento non tecnico ma di sicuro interesse per la Privacy dell'accesso ad Internet di tutti noi. Tra i protocolli proposti ed implementati per migliorare sia la sicurezza sia l'utilizzo del DNS, ve n'è uno chiamato "DNS queries over HTTPS" (DoH, RFC 8484) che semplicemente permette di trasportare il protocollo DNS all'interno di un canale cifrato HTTPS, come per qualunque altra navigazione sicura in Internet. E' stata proposta un'applicazione di questo protocollo direttamente all'interno dei Browser, quali Firefox, Chrome, Edge ecc., tale che il Browser richieda la risoluzione dei nomi ad un servizio fornito ad esempio dal produttore del Browser stesso. Questo permetterebbe al fornitore del Browser di impedire accessi a siti compromessi o di phishing, ed in generale a materiale pericoloso, filtrando la risoluzione dei relativi nomi o sostituendo l'indirizzo IP malevolo con uno benevolo, oltre che a potenzialmente migliorare l'efficienza e velocità delle risoluzioni sfruttando cache ottimizzate.

D'altra parte, il fornitore del Browser verrebbe a conoscenza della navigazione completa di ognuno di noi visto che per accedere a qualunque pagina web la prima operazione è quella di tradurre il

nome del sito in un indirizzo IP. Come vedremo, oggi la risoluzione è un'attività distribuita e molteplici entità gestiscono informazioni diverse rendendo più difficile raccogliere la storia della navigazione di ognuno di noi se non nel Browser stesso o nelle cache DNS locali.

La discussione su questo argomento è ancora in corso (si veda ad esempio [Rif. 2]) ma sicuramente devono essere valutati i possibili impatti sulla privacy della navigazione Web e le possibilità di censura centralizzata per l'adozione e l'utilizzo del protocollo DoH direttamente all'interno dei Browser.

3.1.2 LA RISOLUZIONE DNS

Il protocollo DNS è basato su tavole di traduzione tra i "nomi" dei servizi internet ed i loro "indirizzi IP" a cui sono associati numerosi parametri utili alla gestione del protocollo stesso.[37] Queste tavole sono gestite dai server DNS organizzati in una struttura gerarchica ad albero. All'origine (cima) dell'albero vi sono tredici "root server DNS" (Fig. 3.1.1) che forniscono tutti le stesse identiche informazioni e sono configurati in modo da garantire l'autenticità e l'integrità delle informazioni (ritorneremo più avanti su questo punto), e la disponibilità del servizio praticamente in ogni condizione e sotto qualsiasi attacco difendibile con le tecnologie odierne.

37) In questa sede non viene discussa la risoluzione DNS inversa, ovvero da indirizzo IP a nome.

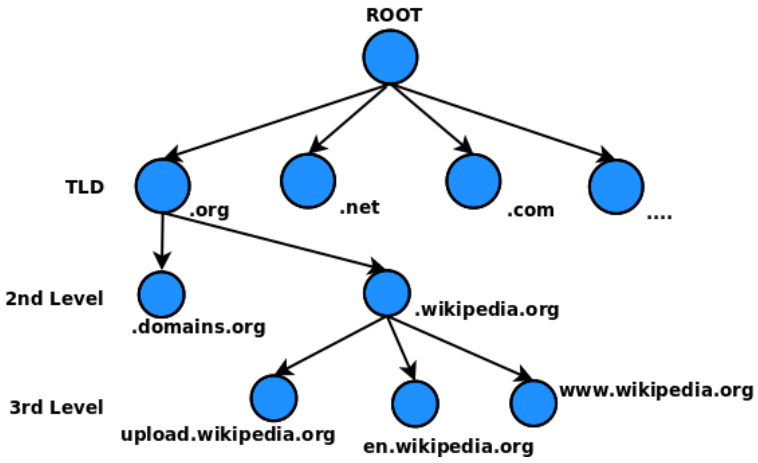

Figura 3.1.1: Gerarchia ad albero dei server e nomi DNS

Gli altri server DNS sono organizzati in maniera gerarchica ad albero sotto i 13 root server.

Un semplice esempio è la maniera più rapida per descrivere il protocollo. Supponiamo di voler trovare l'indirizzo IP del sito www.wikipedia.org.

Il punto di partenza è un PC sul quale è installata una libreria di risoluzione DNS con la tabella degli indirizzi (Fig. 3.1.2) dei 13 root server.

```
;       This file holds the information on root name servers needed to
;       initialize cache of Internet domain name servers
;       (e.g. reference this file in the "cache  .  <file>"
;       configuration file of BIND domain name servers).
;
;       This file is made available by InterNIC
;       under anonymous FTP as
;           file                /domain/named.cache
;           on server           FTP.INTERNIC.NET
;       -OR-                    RS.INTERNIC.NET
;
;       last update:     January 07, 2019
;       related version of root zone:     2019010702
;
; FORMERLY NS.INTERNIC.NET
;
.                        3600000      NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
A.ROOT-SERVERS.NET.      3600000      AAAA  2001:503:ba3e::2:30
;
; FORMERLY NS1.ISI.EDU
;
.                        3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.      3600000      A     199.9.14.201
B.ROOT-SERVERS.NET.      3600000      AAAA  2001:500:200::b
;
; FORMERLY C.PSI.NET
;
.                        3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.      3600000      A     192.33.4.12
C.ROOT-SERVERS.NET.      3600000      AAAA  2001:500:2::c
;
[...]
```

Figura 3.1.2: Indirizzi ipv4 e ipv6 dei primi tre root server DNS

La prima richiesta viene fatta ad un root server, che risponde di non conoscere l'indirizzo IP del sito, ma rimanda al server DNS del dominio "org" (primo ramo dell'albero, Top Level Domain) fornendone gli indirizzi IP. La richiesta viene quindi fatta ad un server DNS del dominio "org" che a sua volta risponde di non conoscere l'indirizzo IP del sito, ma rimanda al server DNS del dominio "wikipedia.org" (secondo ramo dell'albero, 2nd Level Domain) fornendone gli indirizzi IP. La richiesta viene infine fatta ad un server DNS del dominio "wikipedia.org" che risponde dicendo di essere l'autorità per questo dominio e di conoscere l'indirizzo IP del sito, ovvero gli indi-

rizzi ipv4 91.198.174.192 e ipv6 2620:0:862:ed1a::1. Il processo appena descritto è svolto da un servizio chiamato “iterative DNS resolver” (Fig. 3.1.3).

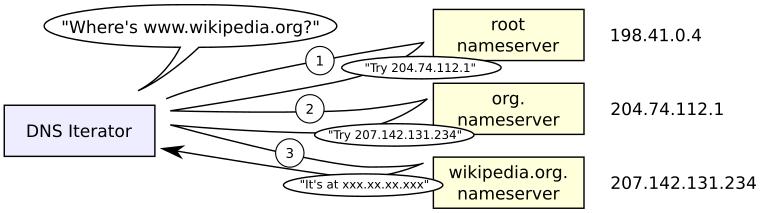

Figura 3.1.3: Processo di risoluzione iterativa da nomi agli indirizzi IP [Fonte Wikipedia]

3.1.3 LA CACHE DNS

Il processo di risoluzione DNS tramite Resolver iterativi non è efficiente in quanto richiede per ogni risoluzione di un nome di ripartire dai server Root. In pratica si utilizza un processo inverso basato sull’uso di cache e che naviga l’albero di risoluzione al contrario, dall’estremo verso l’origine. Parallelamente ai server DNS d’autorità vi è un albero di server DNS interconnesso a quello principale.

Questi sono chiamati tipicamente “server DNS ricorsivi”. Il processo di risoluzione dei nomi in indirizzi IP utilizzato in ogni istante dai nostri dispositivi informatici è, ad alto livello, il seguente (Fig. 3.1.4).

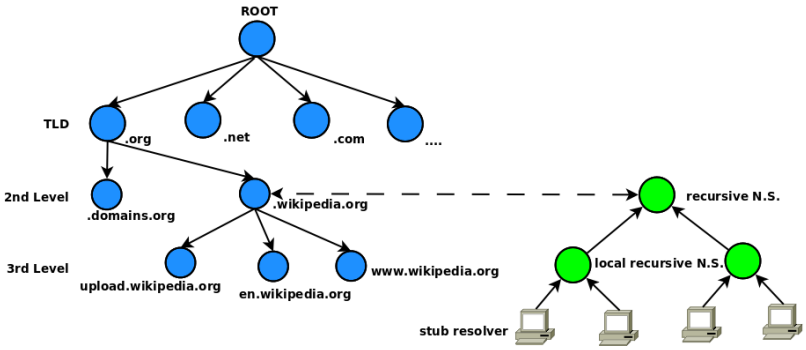

Figura 3.1.4: Risoluzione ricorsiva di www.wikipedia.org: il client (stub resolver) ed il server DNS locale non hanno in cache dati del dominio wikipedia.org, mentre il server DNS ricorsivo ha in cache gli indirizzi IP dei server DNS del dominio wikipedia.org.

Innanzitutto, quando ad un Client arriva la risoluzione di un nome nei suoi indirizzi IP, tra i parametri ricevuti vi è anche il "Time To Live" (TTL), ovvero il tempo, da alcuni secondi a qualche giorno, di validità della risoluzione. Ogni sistema operativo ha un servizio di risoluzione DNS utilizzato da tutte le applicazioni in esecuzione sul sistema, che memorizza la risoluzione del nome in una cache locale per il tempo indicato dal TTL.

Nel caso in cui la risoluzione non sia nella cache locale, il servizio DNS sul Client non chiede ai Root server DNS ma ad un "nameserver" locale, ad esempio aziendale o del proprio ISP, il cui indirizzo IP è tipicamente fornito al momento di configurazione dell'interfaccia di rete IP, ad esempio tramite il protocollo DHCP. In questa modalità, il servizio di risoluzione DNS del Client è chiamato "Stub resolver".

Il nameserver locale è tipicamente un server DNS ricorsivo, sempre con cache, a cui si rivolgono molti Client, ad esempio tutti i dispositivi di un'azienda o tutti i clienti di un ISP.

Nel caso il nameserver locale non abbia in cache la risoluzione di un nome, può rivolgersi a sua volta ad un altro server DNS ricorsivo, ad esempio del Carrier che fornisce la connettività internet nazionale ed internazionale. In questo modo si forma una catena ad albero di servizi di risoluzione DNS con server la cui cache è sempre più grande, visto che forniscono un numero maggiore di clienti.

Nel caso l'ultimo server DNS ricorsivo di questa catena non abbia la risoluzione in cache, procede con il processo iterativo descritto precedentemente partendo dai Root server o dai server DNS d'autorità di cui già conosce gli indirizzi IP.

Si noti che per bilanciare il carico di lavoro e garantire la disponibilità del servizio, i Resolver ricorsivi e iterativi utilizzati sono sempre molteplici e condivisi. Ad eccezione dei Resolver locali, nessun Resolver sa esattamente chi richiede la risoluzione di un nome sia perché può essere fornita via cache locale o distribuita, sia perché può essere un altro Resolver a chiederla.

Anche se non perfetto, questo processo garantisce un certo livello di Privacy, ed in alcuni casi anche anonimato, per l'accesso alle risorse in Internet tramite la risoluzione dei nomi in indirizzi.

3.1.4 ESTENSIONI DEL PROTOCOLLO DNS ED ALTRI DATI

Il protocollo DNS originale prevedeva lo scambio di dati via UDP con messaggi di dimensione massima di 512 byte. Ben presto però si è compreso che il DNS può essere anche utilizzato come libreria centrale di altre informazioni relative ai sistemi in rete, oltre alla necessità di estensioni del protocollo stesso principalmente per motivi di sicurezza come sarà discusso in seguito. Il protocollo DNS è stato pertanto esteso (EDNS) in maniera retro-compatibile, sia permettendo pacchetti UDP sino a 4096 byte, sia introducendo la possibilità di scambiare dati via TCP.

E' già stato indicato come un record DNS contiene il nome da tradurre, gli indirizzi IP che gli corrispondono, i nomi ed indirizzi IP dei server DNS d'autorità per il dominio (o zona), ed altri parametri quali il TTL. E' possibile anche indicare qual è il nome e l'indirizzo del server di posta elettronica che gestisce la posta per il dominio indicato (record MX), oppure creare blacklist di indirizzi IP (ad esempio: se l'indirizzo IP 1.2.3.4 è in una blacklist gestita dal servizio "example blacklist", la query DNS 4.3.2.1.blacklist.example ritorna 127.0.0.1, altrimenti ritorna 127.0.0.2, ove blacklist.example è un server DNS appositamente configurato). I record DNS possono anche essere utilizzati per distribuire informazioni di sicurezza associate a nomi e indirizzi IP quali i Certificati Crittografici (CERT records, RFC 4398), le fingerprint SSH (SSHFP, RFC 4255), le chiavi pubbliche IPSec (IPSECKEY, RFC 4025), le Trust Anchor TLS (TLSA, RFC 6698) eccetera.

Minacce e debolezze del protocollo DNS

Il protocollo DNS fu inizialmente proposto nel novembre del 1983 (RFC 882 e 883) e non prevedeva particolari misure di sicurezza se non una iniziale attenzione alla scalabilità e disponibilità del servizio. E' indubbio che il disegno gerarchico e decentrato evoluto negli anni ha sicuramente raggiunto gli obiettivi di garantire un servizio scalabile alle dimensioni di Internet di oggi, e disponibile in quanto ogni accesso a risorse in Internet richiede una preventiva risoluzione DNS. Più che la scalabilità, quello che preoccupa oggi i fornitori di servizi DNS sono gli attacchi di Distributed Denial of Service (DDoS) che possono bloccare temporaneamente l'erogazione del servizio e quindi l'accesso ad un gruppo anche esteso di risorse in Internet (si veda ad esempio [Rif. 3]).

Sino alle più recenti evoluzioni, il protocollo DNS però ha fornito ben poche funzionalità a garanzia di

- autenticità dei dati
- integrità dei dati
- confidenzialità dei dati
- Privacy, ad esempio rispetto al tracciamento della navigazione in Internet.

Le minacce all'autenticità e integrità dei dati sono quelle ritenute più critiche e si spera che le misure che saranno descritte più avanti siano in grado di mitigarle efficacemente.

Le principali minacce vanno usualmente sotto i nomi di "DNS spo-

ofing" e "cache poisoning". Questi attacchi si basano sul fatto che le relazioni nell'albero dei server DNS dipendono da un rapporto di fiducia (trust) tra i server stessi. Infatti il protocollo DNS non prevede un processo di autenticazione delle sorgenti se non il fatto di ottenere in maniera gerarchica gli indirizzi IP dei server DNS a partire dai server DNS Root. Ma le comunicazioni tra server non sono né autenticate, né protette rispetto alla confidenzialità e integrità dei dati trasmessi. In mancanza di un processo forte di autenticazione tra i server DNS, è possibile ad esempio sia introdurre dei server DNS maligni e dirottare le richieste verso questi (DNS spoofing), sia intercettare le richieste di risoluzione di un server DNS ricorsivo verso un server d'autorità od un altro server ricorsivo, e rispondere con dati modificati in modo da "avvelenarne" la cache DNS (cache poisoning). Questi attacchi sono possibili grazie al fatto che le richieste e le risposte standard delle query DNS sono singoli pacchetti UDP, che non richiedono la creazione di una sessione di comunicazione e che sono spesso facilmente modificabili in transito.[38]

Come garantire l'autenticità e l'integrità dei dati a partire dai server DNS Root sino agli stub resolver sui nostri PC (Fig. 3.1.4) anche in presenza di attacchi malevoli sulle reti pubbliche?

Prima di affrontare questo punto, è conveniente considerare gli aspetti di confidenzialità e Privacy relativi al servizio DNS. Prima di tutto, tutti i dati gestiti dal DNS sono pubblici, per cui non vi è

38) Questo richiede ovviamente di poter accedere alla rete di comunicazione ove transitano i pacchetti UDP delle risoluzioni DNS.

alcuna esigenza di proteggere la confidenzialità dei dati stessi. La confidenzialità invece è una proprietà molto utile per poter garantire la privacy degli utenti ed è solo in questo contesto che viene di solito considerata nell'ambito del DNS.

Si consideri infatti il caso di un utente che con il proprio dispositivo (smartphone, tablet, PC ecc.) naviga in Internet. Per fare questo il dispositivo si collega al nameserver locale del fornitore di accesso ad Internet. Si supponga inoltre che qualcuno possa osservare il traffico del dispositivo verso il nameserver locale. Visto che tutti i pacchetti DNS scambiati non sono di solito cifrati, questo attaccante è in grado di avere la lista di tutte le risoluzioni di nomi richieste dal Client, quindi di tutti i siti e servizi visitati dal Client con la data esatta della prima visita.[39] Se invece la connessione tra il Client ed il nameserver locale fosse cifrata, l'attaccante non potrebbe ricavare alcuna informazione sulla navigazione dell'utente dalle sue richieste di risoluzione DNS.

L'attaccante potrebbe allora osservare il traffico tra il nameserver locale e gli altri server DNS da questo contattati. In questo caso però la quantità di informazioni che riuscirebbe ad estrarre sarebbe molto limitata, soprattutto in caso di server DNS che gestiscono molti utenti, perché non è facile mettere in diretta relazione le richieste del nameserver locale con quelle dei suoi Client, e soprattutto molte richieste dei Client trovano risposta direttamente nella cache del nameserver locale senza necessità di una ulterio-

39) Le visite successive alla prima in tempi ravvicinati, e comunque entro il TTL, non sono visibili in quanto in dati sono in cache sul dispositivo utente e quindi non vengono richiesti nuovamente al nameserver locale.

re richiesta ad altro server DNS.

Si noti infine che per la Privacy degli utenti è anche importante come sia lo stub resolver sul Client stesso sia il nameserver locale traccino le richieste di risoluzione. Infatti un file di log che contiene tutti i dati di richieste di risoluzione DNS di un Client, permette di ricostruire la navigazione, almeno il primo accesso, del Client stesso. Recentemente alcuni servizi pubblici di risoluzione DNS hanno cominciato a dare informazioni sulla loro politiche di gestione dei log dei server DNS: quali dati vengono tracciati, quando vengono cancellati, chi vi può accedere e per quali scopi (anche a seguito della normativa GDPR). Prima di scegliere un servizio DNS pubblico è quindi meglio informarsi su come questo tratti i dati di log dei propri server DNS.

Chi ha particolari esigenze di Privacy può quindi valutare se procedere come segue:

- scegliere un servizio di risoluzione DNS (ovvero "nameserver locale") molto grande
- verificare le politiche di gestione dei log dei server DNS del fornitore
- connettersi al server DNS con canale cifrato
- assicurarsi che il proprio stub resolver non tracci le richieste di risoluzione DNS.

Dopo aver introdotto DNSSEC nella prossima sezione, daremo alcune indicazioni pratiche di come sia possibile implementare questo approccio.

3.1.5 DNS SECURITY EXTENSIONS (DNSSEC)

Il processo di studio, implementazione e valutazione di una nuova versione del protocollo DNS che garantisca autenticità e integrità dei dati è incominciato almeno 20 anni fa e non si è ancora concluso.

Già solo questo intervallo temporale indica la difficoltà di introdurre delle nuove misure di sicurezza nel protocollo senza stravolgerlo e garantendo l'interoperabilità con quanto in essere.

Vi sono state molte proposte diverse, ma quella che è in corso avanzato di implementazione si chiama DNSSEC. L'idea di base è molto semplice: firmare digitalmente tutti i record DNS. L'implementazione si è mostrata invece molto ardua. La seguente è una descrizione a livello molto alto del protocollo (per i dettagli, si vedano tra gli altri gli RFC 4033, 4034, 4035).

Scopo principale di DNSSEC è quello di apporre una firma digitale all'insieme di dati detto Resource Record Set (RRSet) che forniscono la risoluzione di nome nei relativi indirizzi IP con i parametri associati. Quindi nel server DNS d'autorità per un certo nome a dominio, oltre ai dati DNS, ovvero il RRSet, sono presenti anche un record RRSIG che contiene la firma digitale sul RRSet, e un record di tipo DNSKEY che contiene la chiave pubblica necessaria per la verifica della firma digitale RRSIG e corrispondente alla chiave privata utilizzata per creare la firma digitale.

Queste coppie di chiavi pubbliche-private sono chiamate Zone Signing Key pair (ZSK). Si noti che una DNSKEY che contiene la

chiave pubblica di una ZSK è identificata dal Flag 256.

Ovviamente la firma digitale è effettuata in un ambiente sicuro, tipicamente off-line e la chiave privata di firma è, ove possibile, tenuta in modulo hardware apposito.

Con la firma digitale sui RRSet viene assicurata l'integrità dei dati, ma per il momento non è risolto il problema dell'autenticità perché un attaccante potrebbe sostituire la tripletta RRSet, RRSIG(RRSet) e DNSKEY(ZSK) con i propri dati, ad esempio tramite attacchi di tipo "DNS spoofing" e "cache poisoning" descritti precedentemente.

Il problema è quindi di garantire l'autenticità delle chiavi ZSK, ovvero che queste sono proprio quelle create e gestite dal titolare del dominio DNS sul server DNS d'autorità.

Si potrebbe pensare di introdurre un archivio centrale per queste chiavi alla PGP, oppure adottare l'approccio delle Certification Authorities come per i certificati utilizzati per i siti web. Questi ed altri approcci simili però avrebbero delle notevoli conseguenze sul protocollo DNS originario e non sarebbero facilmente integrabili in esso.

DNSSEC invece segue l'approccio del protocollo DNS originario che, come descritto precedentemente, è basato sulla presenza dei 13 Root server di cui sono noti pubblicamente a priori gli indirizzi IP, e della catena di fiducia (trust) a partire da questi. Vi sono quindi due punti da risolvere: come gestire l'origine dell'albero delle firme digitali e come passare da un ramo al successivo mantenendo la fiducia. DNSSEC risolve questi punti introducendo

un nuovo tipo di chiavi[40] chiamate Key Signing Key (KSK), la cui parte pubblica è disponibile sempre in record di tipo DNSKEY. Si noti che le KSK sono chiavi "self-signed" e che una DNSKEY che contiene la chiave pubblica di una KSK è identificata dal Flag 257.

Tutto l'albero di fiducia è attestato su di una chiave, la KSK(ROOT), si veda la Fig. 3.1.5.

```
; // The root key in bind format. This can be read by most tools, including
; // named, unbound, et. For libunbound, use ub_ctx_trustedkeys() to load this
trusted-keys {
"." 257 3 8
"AwEAAaz/tAm8yTn4Mfeh5eyI96WSVexTBAvkMgJzkKTOiW1vkIbzxeF3+/4RgWOq7HrxRixH
lFlExOLAJr5emLvN7SWXgnLh4+B5xQlNVz8Og8kvArMtNROxVQuCaSnIDdD5LKyWbRd2n9
WGe2R8PzgCmr3EgVLrjyBxWezF0jLHwVN8efS3rCj/EWgvIWgb9tarpVUDK/
b58Da+sqqls3eNbuv7pr+eoZG+SrDK6nWeL3c6H5Apxz7LjVc1uTIdsIXxuOLYA4/
ilBmSVIzuDWfdRUfhHdY6+cn8HFRm+2hM8AnXGXws9555KrUB5qihylGa8subX2Nn6UwNR
1AkUTV74bU="; // key id = 20326

"." 257 3 8
"AwEAAagAIKlVZrpC6Ia7gEzahOR+9W29euxhJhVVLOyQbSEW0O8gcCjFFVQUTf6v58fLjw
Bd0YI0EzrAcQqBGCzh/
RStIoO8g0NfnfL2MTJRkxoXbfDaUeVPQuYEhg37NZWAJQ9VnMVDxP/VHL496M/QZxkjf5/
Efucp2gaDX6RS6CXpoY68LsvPVjR0ZSwzz1apAzvN9dlzEheX7ICJBBtuA6G3LQpzW5hOA2hz
CTMjJPJ8LbqF6dsV6DoBQzgul0sGIcGOYl7OyQdXfZ57relSQageu+ipAdTTJ25AsRTAoub8ON
GcLmqrAmRLKBP1dfwhYB4N7knNnulqQxA+Uk1ihz0="; // key id = 19036
};
```

Figura 3.1.5: La chiave pubblica KSK(ROOT) all'origine dell'albero di autenticità DNSSEC

In realtà in Fig. 3.1.5 sono presenti due chiavi, ma come indicato in https://data.iana.org/root-anchors/root-anchors.xml , la chiave con id=19036 è scaduta il 2019-01-11 e rimane solo per retro compatibilità.

40) Si noti che gli RFC DNSSEC non richiedono espressamente di utilizzare due chiavi diverse per KSK e ZSK.

Questo ci porta ad affrontare le modalità di gestione delle chiavi KSK(ROOT). Vista l'enorme importanza di questa chiave, la procedura di creazione e la sua gestione sono soggette a misure di sicurezza estremamente complesse ed estese. Ad esempio la procedura di creazione della chiave KSK(ROOT) include una cerimonia con la presenza fisica di autorità delegate a questo e la produzione di evidenze conservate fisicamente in luoghi separati, si veda https://www.iana.org/dnssec/files per ulteriori informazioni. La chiave KSK(ROOT) è usata raramente e la procedura di sostituzione è lunga e complessa anche perché questa chiave deve essere distribuita manualmente a tutti i server e resolver DNS, insieme agli indirizzi IP dei 13 root server. Inoltre si deve tenere conto dei TTL che indicano i tempi di permanenza dei dati nelle cache DNS in tutto il mondo. Pertanto la chiave KSK(ROOT) deve durare a lungo: la prima chiave è stata introdotta nel 2010 e, come visto, sostituita solo all'inizio del 2019 dalla seconda chiave che è stata generata nel 2016 (si veda ancora https://www.iana.org/dnssec/files).

La chiave KSK(ROOT) viene utilizzata solo per firmare le chiavi ZSK dei Root server, generate e sostituite più frequentemente e utilizzate per firmare gli RRSet presenti nei 13 (identici) Root server. Per verificare l'autenticità delle ZSK, bisogna scaricare da un Root server la ZSK e la RRSIG(ZSK), ovvero la chiave pubblica ZSK e la sua firma digitale RRSIG, e verificare la firma digitale RRSIG con la chiave pubblica KSK(ROOT) ottenuta direttamente da fonte fidata.

Ogni autorità DNS genera le proprie chiavi KSK con le quali firma unicamente le proprie chiavi ZKS. Inoltre ogni autorità DNS genera

un figerprint (o "hash") della propria chiave pubblica KSK e la invia in un record chiamato Delegation Signer (DS) alla zona genitore (ovvero il ramo dell'albero superiore) in modo che questa sia associata agli indirizzi IP ed agli altri dati dei propri server DNS, ed ovviamente sia firmata con la ZSK della zona genitore.

La procedura è sicuramente complessa e rispetto al protocollo originale DNS richiede molti più passaggi, molti più dati e molte più risorse computazionali. Ma DNSSEC è in grado di garantire sia l'integrità che l'autenticità della risoluzione DNS ad un costo ad oggi sicuramente affrontabile.

Per descrivere ad alto livello come tutto questo funzioni, utilizziamo come esempio di risoluzione il nome a dominio di prova DNSSEC sigok.verteiltesysteme.net :

1. il punto di partenza del nostro Resolver DNSSEC è la conoscenza dei 13 indirizzi IP dei Root server e della chiave pubblica KSK(ROOT)

2. la prima richiesta ad un Root server è di fornire la ZSK e la sua RRSIG

3. utilizzando KSK(ROOT) il Resolver verifica la firma RRSIG sulla ZSK; ora la ZSK è integra e autentica

4. il Resolver chiede al server Root di risolvere sigok.verteiltesysteme.net, il server Root risponde di non conoscere la risoluzione ma invia i dati del server DNS della zona .net in un RRSet firmato con la ZSK

5. il Resolver verifica la firma della ZSK sul RRSet che quindi è in-

tegro e autentico; si noti che all'interno del RRSet c'è la fingerprint della KSK della zona .net (record DS)

6. il Resolver si collega al server DNS della zona .net indicato dal server Root e richiede la chiave pubblica KSK della zona, la chiave ZSK e la firma RRSIG

7. il Resolver verifica che la fingerprint della chiave KSK sia identica a quella presente nel RRSet ricevuto dal Root server; ora la chiave KSK della zona .net è integra e autentica

8. Con la chiave KSK della zona .net, il Revolver verifica la firma RRSIG della chiave ZSK; ora la ZSK della zona .net è integra e autentica

9. il processo continua ripetendo i passaggi da 4 a 8 con i server DNS delle zone delegate scendendo nell'albero DNS sino a che un server risponde inviando un RRSet contente gli indirizzi IP e gli altri dati di sigok.verteiltesysteme.net ; a questo punto il Resolver verifica con la chiave pubblica dell'ultima ZSK la firma su questo RRSet e si conclude la risoluzione DNSSEC.

Alcuni tool online, quale ad esempio https://www.dnslookup.org/, permettono di visualizzare i principali passaggi di questa procedura ed in particolare la verifica delle firme digitali sugli RRSet e sulle DNSKEY.

Un'ultima osservazione sulle chiavi KSK e ZSK: a differenza degli altri dati delle zone la cui validità temporale è soggetta ad un TTL ovvero un intervallo di tempo relativo, queste sono valide sino ad una data assoluta. Quindi ogni server e Resolver DNS deve avere

un orologio allineato all'ora esatta. Prima di DNSSEC questo non era un requisito essenziale, ora lo diviene e apre la possibilità, anche se remota, ad attacchi basati sulla retro-datazione dell'ora nei server DNS.

3.1.6 DNSSEC E LA CACHE

Come indicato precedentemente, l'efficienza della risoluzione DNS è basata sull'uso delle cache e dei Resolver ricorsivi. Anche i dati DNSSEC possono essere mantenuti in cache secondo le logiche dei TTL. I Resolver ricorsivi DNSSEC inviano le loro richieste con il flag "DNSSEC OK" (DO) attivo, il che richiede che il server DNS risponda con tutti i dati DNSSEC, od informando il Resolver che DNSSEC non è disponibile per le informazioni richieste. Si noti che il flag "DO" fa parte delle estensioni EDNS, in ogni caso necessarie per poter gestire tutti i dati DNSSEC.

Ovviamente la cache di un Resolver ricorsivo DNSSEC è molto più grande della corrispondente cache DNS originale, in quanto devono essere gestite anche tutte le firme e le chiavi pubbliche necessarie per la loro verifica.

Si noti anche la completa interoperabilità di DNSSEC con DNS, in quanto se un server DNS risponde negativamente alla richiesta del flag "DO", la risoluzione procede esattamente come prima. Altrimenti i dati DNSSEC sono dati in più rispetto al protocollo originale.

Più complessa è la situazione degli Stub Resolver, ovvero dei Re-

solver presenti sui dispositivi Client. Uno Stub Resolver potrebbe verificare indipendentemente tutte le firme digitali e tutti i dati che riceve, ma questo imporrebbe sia un elevato carico di lavoro sui Client che un enorme carico sui server DNS, particolare sui server Root e TLD. La configurazione tipica di un Stub Resolver è invece quella di essere "non validating". In questo caso lo Stub Resolver si fida della risoluzione e delle verifiche fatte dal Resolver ricorsivo a cui si connette. Se il Resolver ricorsivo risponde con il flag Authenticated Data (AD) vuol dire che DNSSEC è valido e la risoluzione è integra ed autentica, si veda la Fig. 3.1.6.

```
$ dig +dnssec +multi sigok.verteiltesysteme.net
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 51654
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;sigok.verteiltesysteme.net. IN A

;; ANSWER SECTION:
sigok.verteiltesysteme.net. 60 IN A 134.91.78.139
sigok.verteiltesysteme.net. 60 IN RRSIG     A 5 3 60 (
                                20190730020006 20190430020006 30665 verteiltesysteme.net.
                                LrXt0utcEKK/D/sbscWNAoAT7kLYt9zUEJKxC4rFH3ZE
                                69VUjLmCTEhBv1Xoo5QQUv3NkZeVBbs8VcRaGX4wmHiN
                                NNu5PrP3SBILmICuT/pkM8n8Ex5PH3KKnNsaLwZmwo9V
                                0ziZsu6byNyjyV0Cj05hDzcXv4d1zHPmRNEcG3Y= )
```

Figura 3.1.6: Esempio di risoluzione DNSSEC valida

Se invece il processo di verifica delle firme digitali fallisce, la risoluzione non va a buon fine (l'indirizzo IP, in questo caso 134.91.78.139, non viene fornito anche se presente nei dati ricevuti dal server DNS) e viene riportato un messaggio di errore con codice SERVFAIL e senza il flag AD, come in Fig. 3.1.7.

```
$ dig +dnssec +multi sigfail.verteiltesysteme.net
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 31280
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;sigfail.verteiltesysteme.net. IN A
```

Figura 3.1.7: Esempio di risoluzione DNSSEC con errore di verifica delle firme digitali

Nel caso invece la zona DNS non sia gestita con DNSSEC, il processo di validazione DNSSEC si interrompe quando non viene trovato un record DS che permette di scendere nell'albero di risoluzione, e si prosegue con il processo DNS originario come in Fig. 3.1.8.

```
$ dig +dnssec +multi www.wikipedia.org
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28134
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;www.wikipedia.org.  IN A

;; ANSWER SECTION:
www.wikipedia.org.  106 IN A 91.198.174.192
```

Figura 3.1.8: Esempio di risoluzione DNS in zone senza DNSSEC

Infine un problema di non semplice soluzione riguarda la dimostrabilità dell'assenza di un nome a dominio. In questo caso il protocollo DNS originale fornisce una risposta vuota, ma questa è facilmente gestibile da parte di un attaccante che può rimuovere tutti i dati presenti in cache od in transito verso un Resolver. DNS-

SEC invece fornisce una risposta autentica che dimostra l'assenza di un nome a dominio utilizzando i record detti Next Secure Record (NSEC o NSEC3). NSEC3 fornisce la prova che nella lista ordinata di nomi del dominio, non vi è quello richiesto, utilizzando tecniche crittografiche di Hash in modo da non permettere la numerazione dei domini presenti. La risposta DNSSEC alla richiesta di risoluzione di un nome non esistente genera un errore NXDOMAIN e riporta i record NSEC3 firmati, come in Fig. 3.1.9.

```
$ dig +dnssec +multi wikapedikkk.org
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 63445
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 8, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;wikapedikkk.org.      IN A

;; AUTHORITY SECTION:
org.                    900 IN SOA a0.org.afilias-nst.info. noc.afilias-nst.info. (
                                2013453931 ; serial
                                1800       ; refresh (30 minutes)
                                900        ; retry (15 minutes)
                                604800     ; expire (1 week)
                                86400      ; minimum (1 day)
                                )
org.                    900 IN RRSIG SOA 7 1 900 (
                                20190523163642 20190502153642 16454 org.
                                aVTPvveCU9bd4nMmaOLe1kWIYQ+C1xJ2V8FQ2BdQFxcT
                                AX2OabGxgwlJJXKS4dR6YMkiLechbe9+nnI9D0GKNlWB
                                SjtQFaQ+psLtiu6wsj3JFNrsmhnlgzmc53jIFUuyTthW
                                VeUrrkfnv6GAxkcKLMMjys8qbzMNMFCvgzfAI7o= )
h9p7u7tr2u91d0v0ljs9l1gidnp90u3h.org. 86400 IN NSEC3 1 1 1 D399EAAB (
                                H9PARR669T6U8O1GSG9E1LMITK4DEM0T
                                NS SOA RRSIG DNSKEY NSEC3PARAM )
h9p7u7tr2u91d0v0ljs9l1gidnp90u3h.org. 86400 IN RRSIG NSEC3 7 2 86400 (
                                20190523163642 20190502153642 16454 org.
                                NQ1LLyMkJDNri2wVTPhlCxi9yLKFefF61KJmr65yU3Sk
                                AnMCBcZlVkpYuOm1KjMlLS7EvLUGD6PqjTyJZ5+qx/Mj
                                C6bjRCW7q+hn5D1Gm6jeHiz34xloznvId0VARmXIxGJb
                                FPr9xK4F7urftlaMvjn7+KcdJ2xYCehiYBOHkJY= )
[...]
```

Figura 3.1.9: Esempio di risposta NXDOMAIN NSEC3 alla richiesta di risoluzione di un dominio inesistente in DNSSEC

3.1.7 USARE DNSSEC

L'utilizzo di DNSSEC è ancora molto limitato. Solamente i livelli superiori dell'albero DNS hanno adottato consistentemente DNSSEC, mentre l'adozione per i nomi completi che usiamo tutti i giorni è ancora molto a macchia di leopardo. Ad esempio, registro.it riporta che a giugno 2022 su un totale di 1.108 Registrar accreditati per registrare nomi a dominio nella zona .it, solo 40 permettono di registrare zone con DNSSEC. Inoltre https://stats.dnssec-tools.org/ mostra che a primavera 2022 nella zona .it sono presenti solo circa 12.000 nomi a dominio DNSSEC su un totale di circa 3.450.000.

Questo vuol anche dire che molti dei Resolver DNS ricorsivi che utilizziamo ogni giorno non hanno ancora implementato DNSSEC.

In realtà non è molto complesso adottare un Resolver locale DNSSEC. I punti principali da considerare sono tre:

1. identificare un Resolver ricorsivo DNSSEC di fiducia soprattutto per quanto riguarda gli aspetti di Privacy discussi precedentemente
2. configurare una connessione sicura tra il proprio Stub Resolver ed il Resolver ricorsivo DNSSEC (questo sia per motivi di Privacy che per impedire attacchi ai dati trasmessi visto che lo Stub Resolver tipicamente è "non validating")
3. utilizzare localmente uno Stub Resolver DNSSEC.

Per la sicurezza della connessione dello Stub Resolver al Resolver ricorsivo è stato introdotto il protocollo DNS-over-TLS (RFC

7858) che permette di trasportare il protocollo DNS all'interno di un canale cifrato con TLS sulla porta TCP/853. Oppure è possibile adottare "DNS queries over HTTPS" (DoH, RFC 8484) a cui si è già accennato precedentemente, od infine DNSCrypt (https://dnscrypt.info/), una soluzione alternativa che permette di stabilire una connessione cifrata anche sulla porta TCP/443 tra uno Stub Resolver ed un Resolver ricorsivo che supporti questo protocollo.

Come esempio, su di un Client linux è possibile implementare una soluzione locale di Stub Resolver DNSSEC installando il server DNS unbound (https://www.unbound.net/) con DNS-over-TLS ed adottando una configurazione simile a quella brevemente descritta in Fig. 3.1.10.

```
# /etc/resolv.conf
nameserver 127.0.0.1

# /etc/unbound/unbound.conf
server:
  [...]
  tcp-upstream: yes
  edns-tcp-keepalive: yes
  # wget https://www.internic.net/domain/named.cache
  root-hints: "/etc/unbound/named.cache"
  # If you want to perform DNSSEC validation, run
  # unbound-anchor before you start unbound
  auto-trust-anchor-file: "/var/lib/unbound/root.key"
  trusted-keys-file: /etc/unbound/keys.d/*.key
  tls-upstream: yes
  tls-cert-bundle: "/etc/unbound/ca-bundle.crt"
  [...]
forward-zone:
  name: "."
  forward-addr: 1.1.1.1@853#cloudflare-dns.com
  forward-addr: 1.0.0.1@853#cloudflare-dns.com
  forward-tls-upstream: yes
  forward-first: no
```

Figura 3.1.10: Esempio di configurazione di unbound come stub resolver con i server Cloudflare come resolver ricorsivi DNSSEC

3.1.8 SICUREZZA DI DNS E DNSSEC

Come abbiamo visto, il protocollo DNSSEC è sicuramente complesso da adottare e malgrado sia stato proposto ormai da molti anni, ancora non è diffuso a sufficienza per poter fornire tutte le garanzie di sicurezza di cui avremmo bisogno. Inoltre DNSSEC non fornisce una sicurezza end-to-end, ovvero sino allo Stub Resolver locale, né offre misure per garantire la Privacy dell'accesso a internet.

Più in generale, la gestione gerarchica del DNS ed in particolare quella di ICANN dei principali TLD, ha da sempre sollevato critiche sia sulla possibilità di censura sia su possibili influenze commerciali a danno della circolazione delle idee e della libertà di accesso alle informazioni ed ai servizi in internet.

A questo proposito si noti che i protocolli DNS e DNSSEC permettono anche la presenza di più alberi paralleli, ovvero di utilizzare più origini fidate contemporaneamente per la risoluzione di nomi in alberi diversi. Questa possibilità è stata sfruttata ad esempio da OpenNIC, Open Root Server Network (ORSN) e New Nations. Questi servizi DNS sono compatibili con quello principale gestito da ICANN ed aggiungono anche la risoluzione di domini in ulteriori TLD quali ad esempio .ko (Kosovo) e .libre.

Un approccio alternativo alla risoluzione dei nomi in indirizzi IP è quello utilizzato per primo da Tor con la risoluzione del TLD .onion gestita all'interno dello stesso protocollo di anonimizzazione. Con l'avvento delle monete virtuali, a partire dal bitcoin, e dell'uso della blockchain, altri servizi alternativi di risoluzione dei nomi

sono stati introdotti quali Blockchain-DNS, Emercoin, Namecoin eccetera. Questi servizi si basano tipicamente su protocolli decentralizzati quali la blockchain e permettono la risoluzione di un nome in un indirizzo IP tramite un plugin da installare sul browser web che contatta direttamente questi servizi. Sia per Tor che per Blockchain-dns ecc., la risoluzione del nome in indirizzo IP avviene solo utilizzando l'apposita applicazione o plugin e non è un servizio offerto dal sistema operativo e disponibile generalmente a tutte le applicazioni presenti in un dispositivo informatico.

Vi sono opinioni contrastanti sul fatto che sia utile e salutare per Internet la presenza concomitante di servizi paralleli ed anche in concorrenza, per la risoluzione dei nomi in indirizzi IP. Da una parte vi sono logiche di libertà di mercato, di libera scelta ed anche di alta affidabilità dei servizi, dall'altra una parcellizzazione del servizio può renderlo più fragile ad esempio rispetto ad attacchi DDoS o di oscuramento di alcune zone, meno omogeneo e di più difficile accesso.

La risoluzione dei nomi in indirizzi IP è una di quelle attività poco conosciute ma senza la quale non esisterebbe Internet. Malgrado la grandissima efficienza del protocollo che è stato in grado di supportare la crescita esponenziale di Internet degli ultimi 20 anni, rimangono delle serie problematiche di sicurezza relative all'autenticità, all'integrità ed alla privacy dell'accesso a Internet. La lenta adozione di DNSSEC sicuramente risolve alcune di queste criticità, ma rimane ancora tanto da fare perché la risoluzione dei nomi in indirizzi IP sia un servizio non solo efficiente ma anche "sicuro".

3.1.9 RIFERIMENTI BIBLIOGRAFICI

Rif. 1: Si veda ad esempio l'avviso CERT "Alert (AA19-024A): DNS Infrastructure Hijacking Campaign" https://www.us-cert.gov/ncas/alerts/AA19-024A , e Ars Technica "The wave of domain hijackings besetting the Internet is worse than we thought" https://arstechnica.com/information-technology/2019/04/state-sponsored-domain-hijacking-op-targets-40-organizations-in-13-countries/

Rif. 2: L'approccio di Mozilla Firefox e una delle tante discussioni sull'implementazione di DoH direttamente nei browser è disponibile a questo indirizzo: https://mailarchive.ietf.org/arch/browse/doh/?gbt=1&index=HPTOUtziIYe_PFuawExeetkSjVg

Rif. 3: famoso è l'attacco DDOS a Dyn il 21 ottobre 2016, si veda ad esempio https://en.wikipedia.org/wiki/2016_Dyn_cyberattack

04

AUTENTICAZIONE E SICUREZZA

4.1 L'autenticazione che si evolve: dalla Password ai token U2F

E' ben chiaro a tutti che le prime misure di sicurezza di un sistema informatico, grande o piccolo che sia, sono l'identificazione e autenticazione dell'utente che si collega per utilizzarlo. Solo una volta identificato ed autenticato, è possibile autorizzare un utente ad utilizzare il sistema informatico.

Sin dai tempi dei primi sistemi informatici, si capì che l'utilizzo solo di un identificativo dell'utente, oggi chiamato "nome utenza" o "username", non fosse sufficiente. Infatti il nome utenza permette di identificare, appunto, l'utenza sul sistema ma non di garantire che chi si collega abbia diritto ad utilizzare quell'utenza. Il nome utenza non è un'informazione riservata, anche se non sempre è pubblica. Spesso il nome utenza è il nome della persona, oggi spesso è l'indirizzo email o un'informazione simile. Per autenticare una persona (ma anche un altro sistema) è necessario utilizzare

un segreto che sia noto solo a quella persona (o altro sistema). Ecco la nascita della "password", informazione segreta nota solo alla persona (o sistema) che in aggiunta al "username" permette di identificare ed autenticare chi vuole connettersi ad un sistema informatico.

4.1.1 PASSWORD: USO E SICUREZZA

La sicurezza dell'autenticazione tramite password si basa su alcuni assunti, il principale è che la password sia nota solo a chi deve connettersi e, almeno in parte (vedi di seguito), al sistema che si vuole accedere. Lo scenario ideale è quello di una persona che tiene a mente la password, non la comunica a nessuno, né la scrive su alcun supporto, sia cartaceo che informatico. Quindi una password deve essere:

- facile da ricordare
- difficile da indovinare.

Ogni sistema informatico deve archiviare delle informazioni che gli permettono di verificare la correttezza delle password. Sui sistemi le password non devono essere archiviate in chiaro (purtroppo è ancora troppo frequente che questo venga fatto per "semplicità"), ma devono essere adottate specifiche routine crittografiche (hash crittografici) uni-direzionali, ovvero non-reversibili, che trasformano le password in stringhe all'apparenza pseudo-casuali. Visto che l'algoritmo non è invertibile, se si viene a conoscenza della stringa pseudo-casuale non è possibile risa-

lire alla password applicando una procedura automatica. L'unica possibilità è di provare ad indovinare la password e riapplicare l'algoritmo per vedere se la stringa appena generata corrisponde a quella della password che si vuole scoprire.

Quindi se le password sono difficili da indovinare, è praticamente impossibile risalire dalla stringa alla password, mentre data la password in chiaro è facile verificare che la sua trasformata corrisponde alla stringa memorizzata. In questo modo, anche in caso di un Data Breach sul sistema informatico e divulgazione delle stringhe, non dovrebbero esserci problemi di sicurezza in quanto non dovrebbe essere possibile risalire dalle stringhe alle password associate.

Come purtroppo ben sappiamo, questo modello teorico in pratica non funziona per molteplici ragioni. La prima ragione è che non siamo molto bravi a creare password difficili da indovinare, anzi dalle periodiche statistiche risulta che sono sempre molto utilizzate password quali "12345", "qwerty", password=username eccetera. Questo anche perché c'è una chiara dicotomia tra "facile da ricordare" e "difficile da indovinare": normalmente ciò che è facile da ricordare è anche facile da indovinare!

A questo è necessario aggiungere il problema pratico di gestire un alto numero di credenziali: 40 anni fa chi utilizzava sistemi informatici doveva ricordarsi al più tre o quattro credenziali.

Oggi ogni persona deve ricordarsi decine di credenziali utilizzate sia in ambito lavorativo che privato. Una facile soluzione è di utilizzare la stessa password per tutte le utenze, ma questo comporta

il grave rischio che una volta carpita la password di un'utenza, un attaccante può accedere a tutte le utenze utilizzate dalla stessa persona, sia in ambito lavorativo che privato.

Per combattere la tendenza ad utilizzare la stessa password in molteplici utenze, è comune politica forzare la modifica della password periodicamente e richiederne una certa complessità, ovvero lunghezza minima e presenza di lettere, numeri, caratteri di punteggiatura ecc. (si veda ad esempio [Rif. 1]).

Allo stesso tempo i programmi di Password Cracking basati per lo più sul concetto di provare le password più facili da indovinare e le loro variazioni, diventano sempre più sofisticati, veloci e di facile uso, e gli incidenti di Data Breach con diffusione di milioni di credenziali sono ormai notizie fin troppo frequenti [Rif. 2].

La situazione è peggiorata a tal punto che per un utilizzatore qualunque la gestione delle password è praticamente l'opposto di quello che dovrebbe essere, le password spesso risultano:

- difficili da ricordare
- facili da indovinare da parte di un attaccante!

Tutto questo ha portato anche il NIST nel 2017 a rivedere le proprie linee guida sulla gestione delle credenziali rimuovendo la richiesta di cambio password periodico e aggiornando il requisito di complessità (si veda [Rif. 3] per i dettagli).

4.1.2 GESTIRE LE PASSWORD

Per migliorare la gestione delle password si è compreso ormai da tempo che bisogna procedere come minimo in due direzioni:

1. ridurre il numero di credenziali che ogni persona deve gestire
2. rendere più semplice la gestione delle password.

In teoria, in ambito aziendale la soluzione esiste, si tratta di:

a. integrare tutte le applicazioni aziendali in un unico Dominio di Autenticazione aziendale (ad esempio gestito con un LDAP server, Microsoft Active Directory ecc.)

b. estendere il Single-Sign-On (SSO) a tutte (o quasi) le applicazioni

c. aggiungere per utenze amministrative o per l'accesso ad applicazioni con requisiti particolari di sicurezza, un ulteriore fattore di autenticazione (discuteremo più avanti di Multi Factor Authentication – MFA/2FA).

In questo modo ogni utente aziendale avrebbe una sola utenza con una sola password per accedere a qualunque dispositivo e applicazione aziendale (con alcune eccezioni quali il PIN della SIM dello smartphone o della carta di credito/bancomat). Questo porterebbe tanti benefici: ovviamente una password per persona è molto più facile da ricordare, anche se complessa, ma anche per l'azienda in quanto una gestione centralizzata delle credenziali migliora l'efficienza e la sicurezza, e le applicazioni o sistemi in SSO non devono gestire direttamente le password ed il processo

di autenticazione. Il principale rischio è quello insito in ogni gestione centralizzata con la presenza di un "Single Point of Failure": l'accesso a qualunque sistema richiede la disponibilità del Dominio di Autenticazione. Ma la preoccupazione maggiore viene dalla possibilità di un accesso fraudolento: in questo caso l'attaccante avendo carpito una credenziale potrebbe aver accesso con questa identità a tutte le applicazioni e sistemi aziendali (a meno della presenza di ulteriori livelli di autenticazione, MFA ecc.). Data la numerosità di Data Breach ed il rischio di diffusione di credenziali che possano permettere l'accesso alla rete aziendale, è particolarmente importante che gli accessi da remoto o in mobilità siano soggetti a misure di sicurezza superiori all'uso della sola password, come vedremo più avanti.

La gestione delle password personali, includendo anche quelle che una persona usa per i sistemi informatici aziendali, è sicuramente più complessa. Da anni sono presenti sul mercato applicazioni per la gestione delle credenziali, i Borsellini delle Password o Password Manager, anche online. Questi strumenti sono molto utili e fondamentalmente permettono di gestire un grande numero di credenziali ricordandosene solo una, quella per accedere al Password Manager stesso. Visto che un utilizzatore di un Password Manager non deve più ricordarsi le password gestite da questo, i Password Manager generano e gestiscono password molto lunghe, pseudo-casuali, con alta complessità, e diverse per ogni account.

I Password Manager sembrerebbero essere gli strumenti che risolvono i problemi di gestione delle password, ma in realtà han-

no alcune notevoli limitazioni (e per le versioni online, le sempre possibili vulnerabilità comuni a qualunque servizio web). Il principale problema di utilizzo dei Password Manager è dovuto alla disponibilità dei dati ed alla praticità di utilizzo. Ovviamente non è possibile autenticarsi ad un sistema se non si ha accesso prima al proprio Password Manager, e vista la complessità dei dati, è sempre necessario copiare e incollare (o trascrivere quando questo non è possibile) le credenziali dal Password Manager al sistema ove ci si vuole autenticare. Personalmente utilizzo un Password Manager per gestire alcune centinaia di credenziali, sia per scopi privati che di lavoro, e mi rendo conto quotidianamente di queste limitazioni pratiche che lo rendono uno strumento utile, ma non per tutti.

Per quanto riguarda i servizi online, in Cloud ecc., lo sviluppo degli ultimi anni è stato quello di ridurre il numero di autenticazioni richieste per l'accesso utilizzando l'equivalente di un SSO aziendale, ovvero la Federazione tra organizzazioni (o SSO Federato). Più precisamente, il Federated Identity Management (FIM) permette ad una organizzazione, l'Identity Provider, di registrare, identificare ed autenticare gli utenti facendo in modo che tale autenticazione sia riconosciuta valida anche da altre organizzazioni. L'Identity Provider garantisce ad altre organizzazioni l'identità e la validità dell'autenticazione dell'utente in modo che non sia necessaria una ulteriore registrazione ed autenticazione. Questo è quello che succede quando si accede ad un servizio web utilizzando il bottone "Autenticati con ..." (ad esempio Facebook o Google). Sommariamente, l'utente prima si autentica con l'Identity Provider, poi può accedere senza autenticarsi di nuovo e senza

ulteriori credenziali o password, a servizi web Federati a questo Identity Provider. Infatti L'Identity Provider dimostra al servizio web Federato che l'autenticazione è valida inviando token crittografici riconosciuti. Vi sono alcuni protocolli crittografici che permettono di fare questo, tra cui SAML, OpenID, Oauth, WS-Federation ecc. Ad esempio in Italia il Sistema Pubblico di Identità Digitale (SPID) si basa sul protocollo SAMLv2. Si noti come l'accesso Federato copra tipicamente le fasi di identificazione e autenticazione dell'utente, mentre l'autorizzazione rimane in carico ad ogni organizzazione ed applicazione.

La Federazione tra organizzazioni permette quindi di uniformare e semplificare gli accessi ai servizi web, riducendo il numero di credenziali che l'utente finale deve gestire.

Va però tenuto conto anche di un altro aspetto importante che riguarda la Privacy degli utenti: un Identity Provider traccia gli accessi dei propri utenti a tutte le organizzazioni e applicazioni Federate, ed è quindi importante che queste informazioni, che potrebbero portare ad un monitoraggio delle attività delle persone, siano gestite nel rispetto della Privacy degli utenti.

4.1.3 COME SUPERARE LE PASSWORD

Ovviamente la ricerca di diverse modalità di autenticazione è vecchia quanto quasi le password stesse. E' ben noto che in generale il processo di autenticazione si può basare su tre tipi distinti di informazioni segrete o uniche:

1. qualche cosa che si sa: ad esempio la password
2. qualche cosa che si è: una caratteristica biometrica
3. qualche cosa che si ha: una chiave od altro oggetto caratteristico.

L'utilizzo di tutti e tre i tipi di informazioni per l'autenticazione è ben noto nella vita di tutti i giorni. Sino a poco tempo fa, nei sistemi informatici l'autenticazione biometrica e con una chiave fisica sono state ristrette ad applicazioni ed esigenze particolari.

4.1.4 BIOMETRIA

L'autenticazione biometrica [Rif. 4] si basa sul riconoscimento di una caratteristica unica del corpo umano quale ad esempio l'impronta digitale, la forma del volto, la voce, l'iride, la geometria della mano eccetera. L'autenticazione biometrica richiede una fase iniziale in cui viene registrato un campione della caratteristica biometrica, detto template, ed associato all'identità da autenticare. Il template è archiviato sul sistema che esegue l'autenticazione. Quando poi l'utente vuole accedere al sistema, viene rilevato un altro campione della caratteristica biometrica dell'utente che viene confrontato con il template registrato per quell'utente (processo di "verifica biometrica").[41] Se il confronto è positivo, ov-

41) Invece per la "identificazione biometrica" viene fornito solo il campione biometrico e tramite una ricerca sui template disponibili si vuole scoprire l'identità dell'utente a cui il campione appartiene.

vero il campione ed il template sono sufficientemente simili, l'autenticazione biometrica è riuscita.

L'autenticazione biometrica comporta molte problematiche che ne hanno reso difficile l'adozione diffusa. Le seguenti sono alcune delle principali problematiche.

Mentre la verifica della password è svolta con una procedura che non ha errori od incertezze, o la password inserita è identica a quella registrata o è diversa, nel caso di autenticazione biometrica due campioni della stessa caratteristica non sono mai identici.

Ogni sistema biometrico deve quindi definire una soglia di somiglianza/errore tra il campione ed il template. Questo però porta a due tipi di errori:

- i Falsi Positivi, ovvero un campione dichiarato simile al template ma appartenente ad un'altra persona (False Match Rate, FMR)
- i Falsi Negativi, ovvero un campione dichiarato diverso dal template ma appartenente alla stessa persona (False Non-Match Rate, FNMR).

Alzando la soglia di somiglianza/errore, i Falsi Positivi diminuiscono ma al contempo aumentano i Falsi Negativi, ovvero chi non riesce ad autenticarsi pur avendone diritto. Abbassando la soglia di somiglianza/errore avviene il contrario, si veda Fig. 4.1.1.

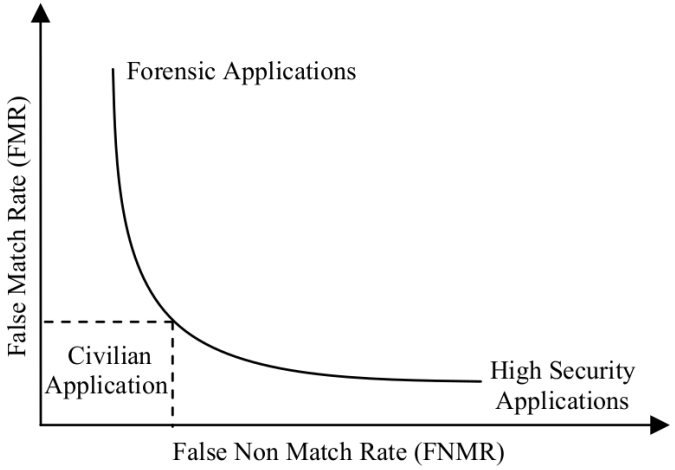

Figura 4.1.1: Diagramma di FMR e FNMR al variare della soglia di somiglianza/errore [fonte Rif. 4]

Inoltre i rilevatori biometrici possono essere soggetti a vulnerabilità che possono portare a Falsi Positivi, come ad esempio la ricostruzione di impronte digitali con gelatina o la ricostruzione di volti a partire da immagini ad alta risoluzione che sono state riconosciute come corrette verifiche in noti esperimenti.

Ma le principali problematiche della biometria risiedono sui rischi della gestione dei template e delle informazioni biometriche. Infatti una caratteristica biometrica non può essere modificata o sostituita, come invece facciamo regolarmente con le password. Pertanto un Data Breach di template biometrici potrebbe avere conseguenze molto gravi per la Privacy degli utenti. Quindi un sistema informatico che gestisce centralmente l'autenticazione biometrica richiede misure di sicurezza molto avanzate [Rif. 5]. E'

quindi meno rischioso archiviare i template biometrici localmente, presso l'utente stesso, come per esempio nel passaporto o nei più recenti smartphone.

Infine vi sono stati anche casi estremi ove, ad esempio, pur di riuscire a rubare un'automobile protetta con autenticazione biometrica ad impronte digitali, i ladri hanno amputato un dito al proprietario [Rif. 6].

4.1.5 SISTEMI A CHIAVE PUBBLICA+PRIVATA

La crittografia a chiave pubblica+privata offre un'altra possibile alternativa alla password. L'idea è che l'utente ha una chiave privata, che deve mantenere segreta, alla quale è associata una chiave pubblica. Le due chiavi sono legate univocamente una all'altra da un algoritmo crittografico, ma dalla chiave privata è possibile ottenere quella pubblica mentre il contrario non è possibile, ovvero non è possibile ottenere la chiave privata conoscendo quella pubblica. Inoltre quanto cifrato con una chiave (pubblica/privata) può essere decifrato solo con l'altra chiave (privata/pubblica). Sui sistemi a cui l'utente vuole connettersi, la chiave pubblica viene associata allo username intestato all'utente.

L'idea del meccanismo di autenticazione è molto semplice, anche se i protocolli crittografici reali sono più complessi a garanzia della sicurezza: quando l'utente si collega al sistema con il proprio identificativo (username), il sistema invia all'utente una stringa pseudo-casuale unica. L'utente cifra con la propria chiave priva-

ta la stringa e la invia cifrata al sistema. Il sistema decifra con la chiave pubblica i dati ricevuti e verifica che la stringa ottenuta sia la stessa che ha inviato. In questo modo l'utente dimostra al sistema di essere in possesso della chiave privata, in quanto solo quanto cifrato dalla chiave privata può essere decifrato dalla corrispondente chiave pubblica.

Mentre l'autenticazione con password è una condivisione di un segreto in quanto la password deve essere inviata ed archiviata dal sistema, anche se protetta come hash, il sistema di autenticazione a chiave pubblica+privata non invia o archivia nulla di segreto sul sistema. Pertanto i rischi di Data Breach sono molti inferiori.

L'autenticazione a chiave pubblica+privata ha però alcuni svantaggi notevoli rispetto alla password. L'utente deve mantenere segreta la chiave privata, questa però è una stringa molto lunga (sino anche a mille caratteri), pseudo-casuale e con particolari strutture matematiche che dipendono dall'algoritmo crittografico utilizzato. E' ovviamente impossibile mantenere a memoria le chiavi private, che sono sicuramente impossibili da indovinare ma anche impossibili da ricordare.

Il problema quindi si sposta interamente su come l'utente può mantenere segreta e personale la chiave privata. L'idea di protocollo descritta precedentemente prova solamente che l'utente che si autentica è in possesso della chiave privata. L'utente deve quindi mettere in pratica delle misure di sicurezza che garantiscano che nessuno possa copiargli la chiave privata.

La soluzione per la gestione sicura della chiave privata è la smart-card (o chip-card, o integrated-circuit-card ICC). Ben conosciamo questi oggetti in quanto costituiscono la base ormai della maggior parte del commercio, ovvero le carte di credito, debito, bancomat ecc. con "chip", anche contactless, e le SIM telefoniche.

La smart-card è fondamentalmente un circuito integrato con particolari caratteristiche di sicurezza: al suo interno è archiviata una chiave privata che non può essere estratta né letta anche con attacchi fisici. Il processore incluso nella smart-card permette la firma o cifratura di dati in input con la chiave privata archiviata in esso. Fondamentalmente una smart-card attesta che al suo interno è presente una particolare chiave privata, e garantisce che nessuno possa copiarla dal suo contenitore.

Le misure di sicurezza che un utente deve mettere in pratica per la chiave privata archiviata in una smart-card devono garantire che la smart-card non sia persa o rubata. Queste misure sono le stesse che bisogna adottare per proteggere una chiave. Infatti una smart-card implementa il terzo tipo di informazione per l'autenticazione, basato su "qualche cosa che si ha".

Come puro strumento di autenticazione la smart-card ha avuto una adozione e diffusione limitata. Vi sono esempi di adozione anche su larga scala, ad esempio il Servizio Sanitario Nazionale e la Tessera Sanitaria / Carta Regionale dei Servizi personale. L'utilizzo della smart-card è però limitato a servizi che richiedono un alto livello di sicurezza, come appunto quelli sanitari, ove il costo della carta, del lettore da utilizzare e del software da installare su apposti computer può essere sostenuto in vista dei rischi a cui si

possono esporre i dati.

Tenuto conto del rischio di furto o smarrimento di una smart-card, per servizi che richiedono un alto livello di sicurezza è necessario associare l'autenticazione con smart-card con un altro tipo di autenticazione, ad esempio la tradizionale password, realizzando un sistema di autenticazione a multi-fattori (MFA). Infatti per utilizzare sia le carte di credito/debito che le SIM telefoniche è necessario l'inserimento di un codice segreto di qualche cifra (PIN) tenuto a memoria come una password. Questo serve a garantire che l'utilizzatore della smart-card sia proprio l'utente legittimo proprietario ed a concludere la catena di autenticazione dalla persona al sistema informatico.

4.1.6 AUTENTICAZIONE A FATTORI MULTIPLI (MFA)

Come appena descritto, la problematica principale è non tanto nel definire delle tecniche di autenticazione teoricamente sicure, quanto di trovare dei processi di autenticazione che siano facilmente adottabili e che garantiscano un livello di sicurezza adeguato ai rischi dei relativi servizi informatici.

La possibilità di utilizzare tipi di autenticazione diversi (qualcosa che si ha, si sa, si è) unita alle criticità dell'utilizzo della password e della necessità per alcuni servizi di aumentare il livello di sicurezza del processo, porta a considerare l'utilizzo di più tipi di autenticazione per la stessa richiesta di accesso. Nella pratica si presentano principalmente due tipologie di MFA:

1. vengono richiesti in serie due (molto raramente tre) fattori di autenticazione al momento del primo accesso al servizio o sistema: ad esempio prima una richiesta di username+password e, se questa ha successo, la richiesta di autenticazione biometrica o di autenticazione tramite un token che si ha, ad esempio tramite una smartcard; se entrambe le autenticazioni hanno successo, viene stabilita una sessione e si ha accesso senza restrizioni al servizio

2. al momento del primo accesso al servizio o sistema viene richiesto un fattore di autenticazione, ad esempio username con password o autenticazione biometrica; se l'autenticazione ha successo, viene stabilita una sessione e si ha accesso al servizio ma limitatamente a funzioni con un livello di rischio basso, ad esempio solo in lettura; per poter eseguire delle attività a maggior rischio (scrittura, modifica dei dati o attività dispositive) viene richiesta una ulteriore autenticazione al momento stesso dell'attività e per ogni esecuzione di attività a rischio elevato.

Ovviamente queste sono solo due tipologie di base di MFA ed in pratica sono spesso adottate molteplici varianti.

Un concetto importante da sottolineare è la necessità per attività che comportano un livello di rischio non basso, di autenticare la singola transazione e non basarsi unicamente sull'autenticazione dell'intera sessione. Per capire la necessità di autenticare la singola transazione è necessario descrivere brevemente alcuni tipi di attacchi.

4.1.7 ATTACCHI DI SESSION HIJACKING, REPLAY, MAN IN THE MIDDLE (MITM) E PHISHING

Quando ci si autentica presso un servizio viene usualmente creata una sessione, ovvero all'utente vengono assegnati dei codici temporanei che lo identificano per la durata della sessione in modo da non dover ripetere il login ad ogni transazione. Nel caso di navigazione Web questo è tipicamente realizzato con dei Cookie (SessionID) contenenti del materiale crittografico creato dal Server, inviati e archiviati temporaneamente nel Browser. Ad ogni richiesta al Server, il Browser allega i Cookie di sessione in modo che il Server identifichi ed autentichi la richiesta senza dover ripetere il processo di login. Ma diversi tipi di Malware sul dispositivo dell'utente possono intercettare i Cookie e, in assenza di altre misure di sicurezza, utilizzare i dati di sessione per accedere al servizio al posto dell'utente, in pratica una sottrazione e appropriazione della sessione utente ("hijacking").

Un altro tipo di attacco, ormai raramente efficace, consiste nell'intercettare i dati o il traffico di autenticazione dell'utente, anche se cifrato, e riutilizzarlo (attacco "replay"). Per evitare questo tipo di attacco tipicamente si allega del materiale unico per ogni autenticazione che non può essere riutilizzato.

Per evitare questi tipi di attacchi bisogna quindi non solo autenticare con materiale unico ogni sessione, ma anche autenticare indipendentemente, meglio con un diverso tipo di autenticazione, ogni singola transazione con livello di rischio non trascurabile. L'autenticazione di una sessione deve permettere l'accesso per

un periodo di tempo esteso (ma non illimitato) che può andare dai pochi minuti a giorni od anche mesi a seconda del livello di rischio dell'applicazione, mentre l'autenticazione di una singola transazione deve essere molto limitata nel tempo e non riutilizzabile.

Infine, l'attacco più pericoloso e difficile da sconfiggere consiste nell'impersonare il servizio ("Phishing") o riuscire a intercettare la connessione al servizio inserendosi al suo interno ("Man in the Middle").[42] In entrambi i casi l'effetto è che l'attaccante vede passare (in chiaro) le informazioni di autenticazione, ovvero lo username + password, i Cookie e qualunque altro dato, e può utilizzarli a suo piacimento.

4.1.8 AUTENTICAZIONE E CODICI ONE TIME (OTP)

L'evoluzione degli attacchi e delle contromisure di sicurezza ha portato ad una specie di gara di velocità ("race condition"). Visto che username + password sono a relativamente alto rischio di essere intercettate e riutilizzate anche a distanza di giorni, mesi o anni, per transazioni a rischio non basso è necessario autenticare ogni singola transazione con un codice valido solo per quella transazione (OTP) e per un limitato periodo di tempo. Il primo metodo adottato a questo scopo è quello delle chiavette (Token) che

42) Alcuni Malware (ad esempio bancari) si inseriscono tra il Frontend del Browser ed il sistema operativo sul dispositivo dell'utente per accedere, ed anche modificare, i dati della connessione al servizio remoto.

generano codici pseudo-casuali comuni per gli accessi ai servizi online bancari, anche se ormai in via di dismissione.

L'idea di base è abbastanza semplice: il Token mantiene in hardware un numero segreto ("seed") a partire dal quale un algoritmo crittografico, tipicamente di Hash, periodicamente (ad esempio ogni 30 o 60 secondi) genera un numero pseudo-casuale di cui mostra all'utente una parte (usualmente 6 cifre). Un'elaborazione parallela viene eseguita da Hardware collegato al Server che eroga il servizio di accesso.

Quando è necessario autenticare una transazione, l'applicazione richiede all'utente l'inserimento del codice prodotto in quel momento dal Token, che viene verificato con il corrispondente codice prodotto sul Server. Il Token fornisce un secondo tipo di autenticazione ("qualche cosa che si ha") rispetto alla password ("qualche cosa che si sa"), è valido solo per quella transazione e la sua validità è limitata nel tempo.

Questa soluzione sembrerebbe a prima vista garantire ottimi livelli di sicurezza. Purtroppo presenta due debolezze: i costi dell'Hardware e di gestione, ed una "race condition". E' facile immaginare i costi di questa soluzione pensando ad un servizio, ad esempio bancario, con milioni di clienti disseminati geograficamente: anche se il costo di ogni Token fosse di pochi euro, con milioni di clienti il solo costo dei Token ammonterebbe a milioni di euro, a cui aggiungere i sistemi server e la gestione del supporto ai clienti. Non è detto quindi che questa sia la soluzione di sicurezza più efficace dal punto di vista della gestione dei rischi e dei costi/benefici.

Un'alternativa meno costosa è quella di inviare un codice OTP a tempo al telefono cellulare dell'utente via SMS. In questo caso è il telefono cellulare che ha il ruolo di "qualche cosa che si ha" anche se il codice è generato sul server ed inviato al telefono su di un canale diverso da quello utilizzato per l'accesso al servizio. Ovviamente molto meno costosa, questa soluzione è anche meno sicura[43] in quanto il protocollo SMS non garantisce né la consegna del messaggio né la sua confidenzialità. Infatti vi sono stati casi di attacchi alle infrastrutture telefoniche per intercettare gli SMS con codici di autenticazione OTP [Rif. 7]. Come sempre, il rapporto costi/benefici di sicurezza è l'argomento principale per scegliere una qualunque soluzione.

Abbiamo visto che il codice OTP può essere utilizzato una volta sola e dura un limitato periodo di tempo, ma gli attaccanti, a costo di spese maggiori, possono aggirare questa difficoltà: è sufficiente fare hijacking della sessione ed intercettare il codice OTP inviato dall'utente utilizzandolo nella propria sessione. Il tutto richiede tempi ben precisi, per transazioni su siti di banche online nell'ordine della decina di secondi, ma possibili.

Si noti come per l'OTP via SMS un punto di debolezza sia già l'invio del codice al telefono cellulare, mentre per la soluzione a Token la debolezza è nella possibile intercettazione del codice o direttamente sul Browser dell'utente oppure tramite siti di Phishing [Rif. 8, 9].

43) Per il momento non consideriamo il caso di accesso al servizio tramite App sullo stesso smartphone.

4.1.9 LA RINASCITA DELLA BIOMETRIA

Negli ultimi anni l'evoluzione dei processi di autenticazione ha avuto principalmente due obiettivi:

1. semplificare i processi di autenticazione per gli utenti
2. ridurre i costi.

E' sicuramente un problema per tutti ricordare una gran quantità di codici, PIN, username+password eccetera. Siamo tutti invece molto soddisfatti dell'utilizzo delle smartcard contactless, che ci permettono di fare acquisti per piccole cifre avvicinando la carta di credito al lettore, e null'altro. In questo caso ci si autentica unicamente con "qualche cosa che si ha" ed il rischio principale è la perdita o furto della carta di credito.

Nel caso di furto, questo particolare rischio economico per l'utente e per l'emittente della carta di credito è però basso ed ampiamente compensato dall'aumento delle transazioni.[44]

Al contempo, lo sviluppo negli ultimi anni dei lettori di caratteristiche biometriche ha permesso la produzione di dispositivi a basso costo e relativa buona efficienza che sono stati integrati nei prodotti per i consumatori. Ormai non solo i PC portatili di alta gamma ma anche moltissimi smartphone sono dotati di un let-

44) Per pagamenti di cifre superiori, sono in corso sperimentazioni di carte di credito Contactless con autenticazione biometrica delle impronte digitali, si veda ad esempio [Rif. 10], invece che della richiesta del PIN.

tore delle impronte digitali o delle caratteristiche del viso. Invece di ricordarci un username+password o un PIN, basta appoggiare il dito al lettore biometrico dello smartphone per aver accesso al telefono ed alle App che sfruttano questa funzionalità, prime di tutte le App bancarie ma anche molte altre [Rif. 11].

Le problematiche di Privacy sono per lo più risolte dal fatto che i dati biometrici sono raccolti e mantenuti da apposite componenti hardware sul dispositivo dell'utente, e non sono gestite da un sistema centrale.

Un possibile approccio è quindi quello di utilizzare username+password come metodo di autenticazione di riserva, che si può mantenere in busta chiusa in caso di difficoltà, ma utilizzare quotidianamente l'autenticazione biometrica per accedere a questi dispositivi e le relative applicazioni.

Ovviamente non stiamo facendo autenticazione a più fattori, ma sostituendo l'utilizzo di un fattore scomodo con uno più funzionale. E' importante notare anche che quanto appena descritto riguarda l'accesso al dispositivo, lo smartphone, non l'accesso a servizi remoti come quelli descritti nella sezione precedente.

Accedere ad uno smartphone con l'impronta digitale non risolve direttamente il problema di autenticare le transazioni economiche sul sito della banca.

4.1.10 AUTENTICAZIONE MUTUA

Il vero problema dell'autenticazione delle transazioni online risiede nella reale autenticazione reciproca tra client e server: ovvero l'applicazione deve essere sicura di chi è l'utente che la contatta e viceversa l'utente deve essere sicuro di comunicare direttamente con l'applicazione, senza alcuna interferenza.

Il protocollo SSL/TLS prevede a questo scopo l'utilizzo di certificati client, similmente ai certificati server che ben conosciamo, ma la complessità della loro gestione, ad esempio tramite smartcard, non ne ha favorito l'adozione.

Un recente approccio alternativo è quello di utilizzare lo smartphone come oggetto di seconda autenticazione tramite un App Authenticator, quali quelle di Google, Microsoft, eccetera o della propria banca. L'idea è quella di sostituire il Token o il SMS descritto precedentemente con lo smartphone sul quale è installata una App di autenticazione. Questa App associa lo specifico smartphone, con il proprio numero di telefono, ad un utente del servizio. Inoltre questa App si autentica mutualmente con il servizio online. Quando l'utente esegue tramite PC una transazione sul servizio online, il servizio online richiede alla specifica App associata all'utente la conferma della transazione. Grazie alla mutua autenticazione, il servizio online è sicuro di comunicare con l'App sullo smartphone dell'utente. L'utente non deve far altro che avere a disposizione lo smartphone e confermare la richiesta di autorizzazione inserendo il PIN di accesso all'App, oppure l'impronta digitale o il riconoscimento facciale. Facendo in questo modo si

può anche raggiungere un'autenticazione a tre fattori: username+password per l'accesso al servizio online ("qualche cosa che si sa"), lo smartphone ("qualche cosa che si ha") e l'autenticazione biometrica ("qualche cosa che si è") per accedere allo smartphone.

Le Authenticator App, seppur molto più sicure dell'invio di un SMS, hanno però anch'esse alcune limitazioni: a parte la possibilità di smarrimento, furto o anche solo scaricamento della batteria dello smartphone, il sistema operativo e le applicazioni per smartphone non sono esenti da vulnerabilità, malware ecc. Ad esempio in caso di rooting, jailbreaking o unlocking, un'App malevola potrebbe interferire con l'esecuzione dell'App di autenticazione portando comunque alla esecuzione di frodi.

Ma forse la principale vulnerabilità degli smartphone è quella di sostituire in molti casi il PC e di diventare l'unico dispositivo di accesso ai servizi online: in questo caso sia la transazione che tutte le autenticazioni sono svolte sullo stesso dispositivo. Accedendo ad esempio con l'impronta digitale, abbiamo un solo tipo di autenticazione ("qualche cosa che si è") eseguita localmente dal dispositivo e non dal servizio online, che deve basarsi solo sull'associazione del dispositivo all'utente fatta al momento di installazione dell'App. In caso di furto, cloning o presenza di malware a questo punto è teoricamente sempre possibile interferire con il processo di autenticazione.

Ovviamente l'adozione di una Authenticator App ha dei costi, in particolare per lo sviluppo e la gestione del software, dei vantaggi, svantaggi e rischi sia per il fornitore di servizi che per gli utenti. E'

comunque da notare che l'utilizzo dello smartphone dell'utente, a la BYOD, rende il fornitore di servizi dipendente da qualche cosa, lo smartphone scelto dall'utente, che non può gestire né realmente controllare, introducendo un ulteriore fattore di rischio.

4.1.11 SECURITY KEYS E FIDO2/U2F

La FIDO Alliance ha proposto un approccio all'autenticazione sicura online alternativo o complementare al Single Sign On (SSO) Federato gestito da un Identity Provider (come discusso precedentemente), puntando alla realizzazione di un sistema diffuso, non centralizzato, e garante della Privacy dell'utente. La soluzione proposta dalla FIDO Alliance [Rif. 12] che va sotto il nome di FIDO2, comprende un gruppo di standard tra cui U2F (CTAP1), UAF, CTAP2 e W3C WebAuthn, ed è basata su molte delle tecnologie di cui abbiamo trattato, dalle smartcard, ai Token fisici ponendosi come obiettivo l'autenticazione mutua utente-servizio.

Il processo di autenticazione è basato su di una Security Key fisica con delle caratteristiche interne simili a quelle di una smartcard, ma che può comunicare a seconda del modello sia tramite USB, sia NFC (come le carte di credito contactless) che Bluetooth Low Energy (BLE). Lo scopo delle Security Key è molteplice [Rif. 13, 14]: sostituire i Token OTP descritti precedentemente, ridurne i costi di gestione, semplificare e automatizzare la gestione per l'utente, aumentare il livello di sicurezza implementando anche l'autenticazione mutua.

Per l'utente l'utilizzo di una Security Key comporta molti benefici: la stessa Security Key può essere utilizzata per autenticarsi a molti diversi servizi online ed in qualunque momento può essere svolta la registrazione di un nuovo servizio; può essere utilizzata indifferentemente con PC desktop, portatili, tablet e smartphone senza necessità di particolari dispositivi di collegamento. Svolge quindi le funzioni di una molteplice chiave fisica per il mondo digitale. Inoltre tipicamente una Security Key presenta un bottone, od un lettore di impronte digitali, ed agisce solo se attivata da questo.

Ad alto livello il protocollo FIDO2 è abbastanza semplice [Rif. 13], le principali funzioni sono due: registrazione e autenticazione.

Lo scenario è quello di un utente che si collega ad un servizio online Web ed effettua prima la registrazione e poi successivamente utilizza il servizio.

Consideriamo il caso più semplice: quello di un servizio che utilizza la Security Key come secondo fattore di autenticazione per transazioni a rischio non basso (Universal Second Factor, U2F).

L'utente crea inizialmente la sua utenza con username+password e poi procede ad aggiungere il secondo fattore di autenticazione. Per far questo è in possesso di una Security Key che connette al dispositivo che sta usando per accedere al servizio online. L'utente attiva nel servizio online la registrazione della propria Security Key. Le principali attività che seguono sono queste (Fig. 4.1.2):

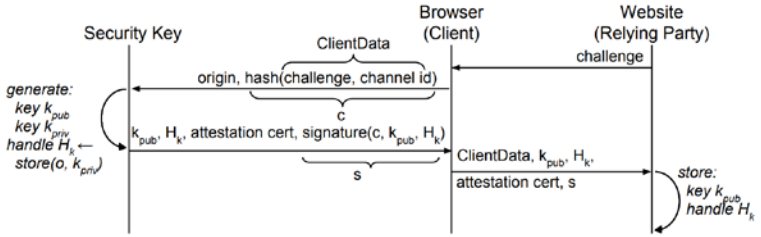

Figura 4.1.2: Registrazione di una Security Key [Fonte Rif. 13]

- il servizio online invia al Browser (o App) dell'utente una stringa pseudo-casuale ("challenge") e l'istruzione di registrare la Security Key
- Il Browser (o App) identifica la Security Key connessa al dispositivo, le invia i dati ricevuti dal servizio online e delle informazioni sul servizio quali il suo indirizzo ("origin"), e richiede la creazione di una nuova chiave pubblica+privata
- La Security Key, dopo la pressione del bottone, crea una chiave pubblica+privata con un identificativo ("handle"), con la chiave privata firma i dati ricevuti ed invia la chiave pubblica ed i dati firmati al Browser (o App)
- Il Browser (o App) inoltra i dati ricevuti dalla Security Key al servizio online
- Il servizio online verifica con la chiave pubblica ricevuta che la firma effettuata dalla Security Key sui dati (ed in particolare sul "challenge") sia corretta, archivia la chiave pubblica con il suo identificativo associandoli all'utenza.

D'ora in avanti il servizio online può utilizzare la chiave pubblica ed il suo identificativo per richiedere una seconda autenticazione all'utente. Questo procede in maniera simile come segue (Fig. 4.1.3):

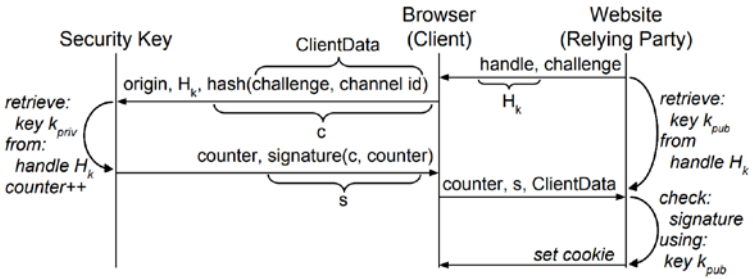

Figura 4.1.3: Autenticazione tramite Security Key [Fonte Rif. 13]

- Il servizio online invia al Browser (o App) dell'utente una stringa pseudo-casuale ("challenge") e l'identificativo ("handle") della chiave pubblica associata all'utente che vuole autenticare
- Il Browser (o App) identifica la Security Key connessa al dispositivo, le invia i dati ricevuti dal servizio online e delle informazioni sul servizio quali il suo indirizzo ("origin"), e richiede l'autenticazione
- La Security Key, dopo la pressione del bottone, verifica che l'identificativo corrisponda ad una propria chiave, che corrisponda ai dati del servizio per cui l'ha creata, firma i dati ricevuti con la chiave privata ed invia i dati firmati al Browser (o App)

- Il Browser (o App) inoltra i dati ricevuti dalla Security Key al servizio online
- Il servizio online verifica con la chiave pubblica dell'utente che la firma effettuata dalla Security Key sui dati sia corretta, e quindi procede con l'esecuzione della transazione.

Si noti come in questo processo il servizio online si fidi solo della firma della Security Key, ovvero dell'attestazione della presenza della chiave privata nella Security Key. D'altra parte la Security Key verifica sia i dati inviati direttamente dal servizio online che quelli forniti dal Browser: la Security Key esegue la firma solo se tutti i dati ricevuti sono consistenti. In questa maniera abbiamo due attori che in maniera automatica verificano reciprocamente la propria identità: il servizio online e la Security Key. Questo indipendentemente da chi possa cercare di intercettare o modificare la comunicazione.

Ovviamente questa soluzione non riesce a coprire tutti i possibili scenari di attacco, la fase più delicata è sicuramente quella della registrazione iniziale ove un attaccante potrebbe cercare di immedesimare il servizio online. Si noti anche che quando l'utente autorizza la firma della Security Key premendo il suo bottone (o verificando l'impronta digitale) non ha la possibilità di verificare il contenuto dei dati firmati su di un display sicuro; un attaccante potrebbe quindi cercare di modificare i dati della transazione, ad esempio cambiandone il destinatario, prima della firma.

Rimane da chiarire un aspetto comunque importante: come revocare le chiavi generate da una Security Key in caso ad esempio

di furto o smarrimento della Security Key stessa o per rimuovere una chiave generata dalla Security Key ma che non si vuole più utilizzare. Il protocollo al momento non prevede la possibilità di cancellare chiavi dalla Security Key né di notificare la perdita (od il ritiro) di una Security Key. Per garantire la Privacy degli utenti, non vi è un identificativo univoco di ogni singola Security Key che altrimenti permetterebbe di tracciare le attività dell'utente. Lo scenario più critico è sicuramente quello del furto di una Security Key ed al momento il protocollo lascia al gestore di ogni servizio online decidere come gestire questo evento.

Un'ultima osservazione su questo protocollo: le Security Key permettono di gestire un numero arbitrario di servizi online, si pone pertanto il problema della conservazione delle chiavi private sul piccolo dispositivo poiché potrebbero occupare parecchio spazio all'interno del processore sicuro. Lo standard non impone una soluzione ma offre un approccio: mantenere nel processore sicuro una (o due) master key private ed esportare in maniera sicura le altre chiavi private. In questo caso tutte le chiavi private delle coppie pubblico+privato generate dalla Security Key, sono cifrate e decifrate con la master key privata solo all'interno del processore sicuro e possono essere esportate solo se cifrate. Le chiavi private cifrate possono quindi essere archiviate od in un'area di memoria apposita sulla stessa Security Key, o inviate al servizio online che le archivia insieme alla chiave pubblica. La sicurezza è fornita dal fatto che solo la master key privata all'interno della Security Key può decifrare ed utilizzare queste chiavi.

L'adozione delle Security Key da parte di Google Inc. [Rif. 13, 14] e

di molte altre aziende sta dando risultati soddisfacenti sia per la semplicità d'uso che per i livelli di sicurezza raggiunti. Il fatto di essere una "chiave" che possiamo aggiungere al nostro portachiavi, utilizzabile con molteplici nostri dispositivi e diversi servizi online, sicuramente può semplificarci la vita oltre ad aumentare la sicurezza dell'accesso ai servizi.

4.1.12 VERSO L'ELIMINAZIONE DELLE PASSWORD

Ma come fare a eliminare le password?

L'approccio vincente [Rif. 15] sembra sarà quello di individuare la combinazione corretta tra le tecnologie (alcune delle quali discusse in questo capitolo), i processi e gli strumenti da utilizzare.

L'approccio che al momento è considerato più promettente consiste principalmente nell'aggiungere due ulteriori componenti a quanto già descritto:

1. l'utilizzo della biometria locale (ad esempio quella che ci permette l'accesso allo smartphone) come fattore di autenticazione di base (o "roaming authenticator"); ovvero l'autenticazione biometrica su di un dispositivo personale e locale diventa il punto di partenza che permette di garantire l'identità dell'utente al servizio, senza utilizzare alcuna password;
2. la possibilità di gestire in maniera sicura credenziali di accesso (in questo caso chiavi private) su sistemi di archi-

viazione online, o meglio servizi online, in modo che queste siano accessibili da qualunque dispositivo di un utente e solo all'utente che ne è proprietario (anche chiamate "multi-device credentials" o semplicemente "passkeys").

Un semplice caso d'uso è il seguente: un utente vuole accedere tramite un PC ad un servizio online. Per far questo può aprire sul PC la pagina web di accesso al servizio online (eventualmente inserendo la propria username o email), connettere il proprio smartphone via Bluetooth al proprio PC e confermare sullo smartphone la propria identità tramite biometria locale (es. impronta digitale o immagine del viso). In questo processo non è richiesta alcuna password.

Inoltre la possibilità di gestire online le credenziali di accesso (o "passkeys") permette di autorizzare facilmente altri propri dispositivi con autenticazione locale a funzionare da "roaming authenticator", risolvendo in questo modo il problema del guasto o perdita del dispositivo di autenticazione (ovvero risolvendo il problema del recupero delle proprie chiavi private, che sono la propria identità digitale, come discusso in §4.1.5, nel caso si guasti il dispositivo che le archivia e protegge).

Secondo FIDO i protocolli CTAP e WebAuthn, nel caso con minori estensioni, sono in grado di implementare questi processi in maniera sicura proteggendo al tempo stesso contro attacchi di Phishing, MitM ecc. E' però troppo presto per dichiarare che abbiamo finalmente individuato la strada per abbandonare definitivamente le nostre vecchie e care password.

4.1.13 RIFERIMENTI BIBLIOGRAFICI

Rif. 1: Dlgs 30 giugno 2003, n. 196, Allegato B

Rif. 2: Per un recente esempio si veda Wired "Hackers are passing around a megaleak of 2.2 billion records", 2019/01/30, https://www.wired.com/story/collection-leak-usernames-passwords-billions/

Rif. 3: NIST Special Publication 800-63B "Digital Identity Guidelines: Authentication and Lifecycle Management", giugno 2017, in particolare la sezione 10.2.1, https://pages.nist.gov/800-63-3/sp800-63b.html

Rif. 4: Per una introduzione alla biometria si veda A.K. Jain, A. Ross, S. Prabhakar "An Introduction to Biometric Recognition", IEEE Transactions on circuits and systems for video technology, Vol. 14, n. 1, Jan. 2004, https://www.cse.msu.edu/~rossarun/pubs/RossBioIntro_CSVT2004.pdf

Rif. 5: Garante per la Protezione dei Dati Personali "Provvedimento generale prescrittivo in tema di biometria", 12 novembre 2014, Pubblicato sulla Gazzetta Ufficiale n. 280 del 2 dicembre 2014, https://www.garanteprivacy.it/web/guest/home/docweb/-/docweb-display/docweb/3556992

Rif. 6: BBC News "Malaysia car thieves steal finger", 31 marzo 2005, http://news.bbc.co.uk/2/hi/asia-pacific/4396831.stm

Rif. 7: Per un recente caso si veda

- "Criminals Are Tapping into the Phone Network Backbone to

Empty Bank Accounts" https://motherboard.vice.com/en_us/article/mbzvxv/criminals-hackers-ss7-uk-banks-metro-bank,

- "SS7 exploited to intercept 2FA bank confirmation codes to raid accounts" http://www.scmagazine.com/home/security-news/cybercriminals-are-exploiting-flaws-in-ss7-a-protocol-used-by-telecom-companies-to-coordinate-how-they-route-texts-and-calls-around-the-world-to-empty-bank-accounts/

Rif. 8: B. Schneier, "Hacking Two-Factor Authentication",

https://www.schneier.com/blog/archives/2009/09/hacking_two-fac.html

Rif. 9: Si veda ad esempio

- Ars Technica "Iranian phishers bypass 2fa protections offered by Yahoo Mail and Gmail", https://arstechnica.com/information-technology/2018/12/iranian-phishers-bypass-2fa-protections-offered-by-yahoo-mail-and-gmail/

- The Hacker News "WARNING – New Phishing Attack That Even Most Vigilant Users Could Fall For", https://thehackernews.com/2019/02/advance-phishing-login-page.html

Rif. 10: "Intesa Sanpaolo and Mastercard introducing the first contactless biometric payment card in Italy", https://www.world.intesasanpaolo.com/hp-news/intesa-sanpaolo-mastercard-introducing-first-contactless-biometric-payment-card-italy/

Rif. 11: si veda ad esempio “WhatsApp can now be locked using Face ID or Touch ID”, http://www.theverge.com/2019/2/4/18210197/whatsapp-touch-id-face-id-security

Rif. 12: https://fidoalliance.org/ , https://www.w3.org/TR/webauthn/

Rif. 13: J. Lang et al. “Security Keys: Practical Cryptographic Second Factors for the Modern Web”, Google Inc., http://fc16.ifca.ai/preproceedings/25_Lang.pdf

Rif. 14: Ars Technica “This low-cost device may be the world’s best hope against account takeovers”, https://arstechnica.com/information-technology/2016/12/this-low-cost-device-may-be-the-worlds-best-hope-against-account-takeovers/

Rif. 15:

- FIDO “How FIDO Addresses a Full Range of Use Cases”, Marzo 2022, https://media.fidoalliance.org/wp-content/uploads/2022/03/How-FIDO-Addresses-a-Full-Range-of-Use-Cases.pdf
- Wired “A Big Bet to Kill the Password for Good”, https://www.wired.com/story/fido-alliance-ios-android-password-replacement/

05

L'EVOLUZIONE DELLA CRITTOGRAFIA

5.1 Elaboratori Quantistici e Crittografia Post Quantum

Quando si pianificano o si valutano i sistemi di sicurezza IT, è importante considerare non solo le minacce e le vulnerabilità odierne, ma anche possibili scenari futuri. Un campo abbastanza interessante che molto probabilmente influenzerà il panorama futuro della sicurezza IT è la teoria dell'Informazione Quantistica. La teoria dell'Informazione Quantistica può essere brevemente descritta come la disciplina che studia come le informazioni possono essere codificate in particelle elementari e cosa si può fare quando le informazioni vengono codificate in questo modo. La teoria dell'Informazione Quantistica oggi è per lo più ancora una branca di pura ricerca in fisica fondamentale, ma ci sono alcune sorprendenti applicazioni pratiche.

Sin dalla fine degli anni '60, i fisici hanno studiato come usare le particelle elementari direttamente per applicazioni informatiche,

partendo da come codificare bit di informazione nelle particelle elementari. L'interesse principale nasce dal fatto che le particelle elementari seguono le leggi dettate dalla teoria della Meccanica Quantistica e il loro comportamento sotto molti aspetti è molto diverso da quello di qualsiasi oggetto che affrontiamo quotidianamente. Ci sono profonde differenze tra le leggi Classiche della fisica che tutti conosciamo bene, quali la dinamica di Newton, l'elettromagnetismo di Maxwell e persino la relatività di Einstein, e la fisica Quantistica. Ad esempio, di solito non è possibile misurare una particella elementare senza modificarla, il che implica anche che è impossibile farne copie esatte. Ciò è abbastanza diverso dalla nostra esperienza quotidiana con oggetti macroscopici, che possiamo misurare senza modificare e di cui possiamo fare copie identiche. E' possibile sfruttare questa differenza intrinseca tra le particelle elementari ed il mondo macroscopico per fare qualcosa di veramente nuovo anche nel campo dell'informatica.

5.1.1 GLI ELABORATORI QUANTISTICI

Lo studio della teoria dell'Informazione Quantistica per lo sviluppo di elaboratori quantistici cominciò negli anni '70 [Rif. 1]. In particolare all'inizio degli anni '80 un contributo importante fu dato dai fisici Manin, Benioff e Feynman che proposero l'idea che un elaboratore quantistico potesse eseguire simulazioni non possibili per i normali elaboratori classici [Rif. 2]. La speranza iniziale dei ricercatori era che un elaboratore quantistico potesse facilmente svolgere calcoli (simulazioni) di sistemi elementari complessi,

quali atomi e molecole, poiché non avrebbe dovuto risolvere numericamente le equazioni della Meccanica Quantistica in quanto queste sarebbero state intrinsecamente codificate nell'elaboratore stesso. In linea di principio gli elaboratori quantistici potrebbero contribuire in maniera ancor oggi impensabile alla chimica elementare, alla farmaceutica, alla creazione di nuovi materiali ecc.

I computer quantistici si basano sull'idea di codificare il valore di un bit in una proprietà di una particella elementare. Come esempio banale e non realistico, potremmo supporre che se una particella gira in senso orario, il valore del bit è 1 e se gira in senso antiorario il valore è 0 (in realtà il valore 0 o 1 è assegnato ad una caratteristica quantistica della particella). Pertanto, invece di avere correnti e tensioni elettriche a codificare il valore del bit all'interno delle CPU, abbiamo particelle elementari, ognuna con il valore di un bit. Un bit codificato in una particella elementare viene chiamato "bit quantico", in breve qubit, e le operazioni in queste CPU quantistiche vengono eseguite trasformando lo stato delle particelle elementari.[45]

Per eseguire un calcolo con un elaboratore quantistico bisogna per prima cosa preparare le particelle elementari nello stato iniziale in modo che rappresentino i numeri di partenza. Poi bisogna eseguire delle trasformazioni fisiche sulle particelle, ad esempio tramite campi magnetici o l'interazione con altre particelle, che

45) Per maggiori dettagli sulla teoria dell'Informazione Quantistica, gli elaboratori ed il calcolo quantistico si fa riferimento alla letteratura specialistica quale ad esempio [Rif. 3].

realizzano il calcolo stesso. Infine bisogna effettuare una misura dello stato fisico finale delle particelle per ottenere il risultato numerico del calcolo. In analogia alla struttura degli elaboratori Classici, per eseguire i calcoli negli elaboratori Quantistici sono state definite teoricamente delle "porte quantistiche" (o "Quantum Gates") che permettono di realizzare qualunque calcolo possibile su di un elaboratore quantistico.

La definizione di "(Universal) Quantum Gates" permette più facilmente di scrivere programmi ed algoritmi per elaboratori quantistici, e la creazione di ambienti di simulazione ove poter verificare la correttezza dei programmi e degli algoritmi.

5.1.2 ALGORITMI QUANTISTICI E SICUREZZA INFORMATICA

Gli elaboratori quantistici sono balzati all'attenzione in particolare di chi si occupa di sicurezza informatica alla metà degli anni '90 quando Peter Shor (nel 1994) e Lov Grover (nel 1996) proposero due importanti algoritmi quantistici: il primo permette la fattorizzazione di un numero intero in tempi polinomiali, il secondo velocizza la ricerca di un valore ("database search") o l'inversione numerica di una funzione. L'algoritmo di Shor è sicuramente quello più famoso in quanto permette di risolvere facilmente il problema matematico alla base di molti algoritmi di crittografia asimmetrica, od a chiave pubblica-privata, quale l'algoritmo RSA. In pratica l'avvento di elaboratori quantistici in grado di eseguire l'algoritmo di Shor renderebbe del tutto insicuro l'utilizzo di algoritmi asim-

metrici quali RSA, ECC, DH[46] ecc., ovvero tutti gli algoritmi crittografici basati sui problemi matematici di fattorizzazione degli interi, dei logaritmi discreti e dei logaritmi discreti su curve ellittiche. Per l'utilizzatore verrebbe a mancare la sicurezza fornita dalla crittografia ai certificati digitali, alle firme digitali, alla cifratura con algoritmi asimmetrici, alla navigazione in Internet, in pratica alla grande maggioranza degli utilizzi quotidiani della crittografia.

L'implementazione dell'algoritmo di Grover avrebbe delle conseguenze meno catastrofiche, in quanto permette di "raddoppiare" la velocità di enumerazione delle possibili chiavi segrete di un algoritmo simmetrico quale AES. Quindi per mantenere lo stesso livello di sicurezza basterebbe utilizzare chiavi lunghe il doppio. In pratica per proteggerci oggi dall'algoritmo di Grover, invece di utilizzare AES con chiavi di 128 bit dovremmo adottare chiavi di 256 bit.

Dal punto di vista della sicurezza informatica, un aspetto molto importante è individuare quando sarà necessario sostituire gli algoritmi impattati dall'algoritmo di Shor e quando sostituire con chiavi più lunghe quelli impattati dall'algoritmo di Grover. Il primo aspetto da considerare è quando saranno realmente disponibili elaboratori quantistici in grado di eseguire questi algoritmi, esamineremo questo punto nella prossima sezione. E' fondamentale però anche valutare quanto prima di questo momento debbano essere implementate le contromisure [Rif. 4].

46) RSA è l'acronimo di R. Rivest, A. Shamir, L. Adleman, ECC di "elliptic-curve cryptography", DH di W. Diffie, M. Hellman.

Consideriamo due esempi: nel primo si supponga che delle informazioni estremamente riservate siano state trasmesse a gennaio 2020 cifrate in Internet e che un attaccante abbia registrato tutto il traffico in rete. Queste informazioni devono rimanere riservate almeno per 10 anni, quindi la cifratura deve garantire la sicurezza della confidenzialità almeno sino a dicembre 2029. Se l'avvento degli elaboratori quantistici fosse prima del 2030, allora in questo scenario già a gennaio 2020 è necessario cifrare le comunicazioni con algoritmi resistenti all'algoritmo di Shor.

Il secondo esempio riguarda la progettazione di un apparato, come ad esempio un aereo, un'automobile, un microprocessore integrato come quelli per le carte di credito o le SIM, o anche un elaboratore specializzato, che implementi in maniera embedded algoritmi crittografici. Nel momento in cui si sceglie quale algoritmo crittografico implementare, bisogna valutare la vita utile dell'apparato e la possibilità dell'avvento degli elaboratori quantistici entro il periodo di tale vita utile. E' da notare che apparati complessi come un aereo possono avere periodi di vita utile di 30 o più anni.

5.1.3 QUALI ELABORATORI QUANTISTICI OGGI? E DOMANI?

Sin dai primi anni 2000 l'interesse agli elaboratori quantistici si è in parte spostato dall'accademia alle aziende, sia per la speranza di fare un investimento economicamente vantaggioso, sia per la necessità di grandi finanziamenti disponibili solo a grandi

aziende. Ad oggi i principali laboratori nei quali sono sviluppati gli elaboratori quantistici sono quelli di Google, Microsoft, Intel, IBM ecc. Se da un lato le grandi aziende hanno le risorse principalmente economiche per procedere con la ricerca e lo sviluppo delle tecnologie necessarie per costruire un elaboratore quantistico, d'altra parte per ovvie necessità di business, le informazioni sull'andamento degli sviluppi sono meno dettagliate. E' comunque possibile fare un punto approssimativo partendo dalla road-map stabilita da un gruppo di esperti convocati dalla ARDA (Advanced Research and Development Activity, un'agenzia di finanziamento della comunità di intelligence degli Stati Uniti) agli inizi degli anni 2000 [Rif. 5]. Secondo questa road-map entro il 2012 sarebbe stato costruito un elaboratore quantistico con circa 50 qubit che avrebbe permesso di eseguire una semplice istanza di un algoritmo quantistico rilevante. Sino a settembre 2019 non era stata fornita alcuna evidenza definitiva che questo traguardo fosse stato raggiunto, anche se ad esempio Intel stava lavorando ad un elaboratore quantistico con 49 qubit, IBM con 53 e Google con 54 e 72 (si veda anche [Rif. 7]).

Ad ottobre 2019 Google ha annunciato [Rif. 6] di aver raggiunto la "Quantum Supremacy" ovvero di essere stato in grado di eseguire un calcolo sul proprio elaboratore quantistico a 54 qubit in 200 secondi con un algoritmo quantistico mentre il corrispondente algoritmo classico avrebbe impiegato probabilmente almeno 10.000 anni sugli elaboratori Classici più potenti oggi esistenti. Simili risultati sono stati annunciati anche da IBM. Possiamo dire quindi che con 7 anni di ritardo è stato raggiunto il traguardo della road-map.

Uno tra i principali motivi della difficoltà nella costruzione degli elaboratori quantistici risiede nei fenomeni usualmente denominati con il termine di "Decoerenza". L'idea di base è abbastanza semplice: come già descritto, gli elementi costitutivi di un elaboratore quantistico sono delle particelle elementari che al contempo:

1. non devono interagire con qualunque altra particella o campo che potrebbe modificare il loro stato e quindi introdurre errori (in pratica cambiare il valore dei qubit)

2. devono interagire unicamente con le particelle ed i campi necessari ad eseguire l'algoritmo e solo nel momento di esecuzione.

Per oggetti macroscopici, le leggi della meccanica classica garantiscono che è possibile separare e tenere separati due oggetti, mentre le leggi della meccanica quantistica garantiscono solo che si possono preparare particelle elementari, ad esempio a temperature prossime allo zero assoluto, in modo che la probabilità di interazioni sia molto bassa, ma non nulla. Perciò in un elaboratore quantistico è necessario implementare degli appositi algoritmi quantistici di correzione degli errori, che però a loro volta richiedono l'introduzione di ulteriori qubit per essere eseguiti.

Quanti qubit sono necessari per eseguire su dati reali l'algoritmo di Shor[47] ed altri algoritmi quantistici rilevanti? Non vi è una

47) Già nel 2001 un elaboratore quantistico a 7 qubit di IBM riuscì a fattorizzare 15 in 3 * 5. Nel 2012 fu fattorizzato 21 in 7 * 3.

risposta precisa anche perché molto dipende dalla tecnologia adottata, dagli algoritmi di correzione degli errori ecc. Le stime più ottimiste indicano alcune decine di migliaia di qubit, ma i più ritengono che saranno necessari decine di milioni se non miliardi di qubit fisici, ovvero particelle elementari [Rif. 8].

Sarà possibile costruire un vero elaboratore quantistico di queste dimensioni? E nel caso quando?

Le opinioni sono molto diverse, ad esempio M. Mosca in [Rif. 4] stima una possibilità del 50% che per il 2031 sia costruito e funzionante un elaboratore quantistico in grado di decifrare RSA (facendo anche ricorso ad una possibile "legge di Moore" per gli elaboratori quantistici), mentre M. Dyakonov in [Rif. 8] ritiene che non sarà mai possibile far funzionare un tale elaboratore quantistico.

5.1.4 CRITTOGRAFIA POST-QUANTUM

Per i nostri scopi, possiamo dividere gli algoritmi crittografici in due grandi classi: quelli che sono rotti dall'algoritmo di Shor e quelli che non lo sono. La principale caratteristica degli algoritmi crittografici a rischio è quella di basare la propria sicurezza su alcuni problemi matematici di difficile soluzione. Più precisamente, sono problemi matematici facili da risolvere quando alcune particolari informazioni sono note, ma difficili o in pratica impossibili da risolvere se i numeri sono grandi e se alcune particolari informazioni non sono note. Solo per dare un'idea, è possibile fare l'esempio del prodotto di due numeri primi, a cui abbiamo già ac-

cennato: se dato 15 è facile identificare 3 e 5 come i suoi fattori primi, la matematica ci insegna che dato un numero prodotto di due numeri primi di centinaia o migliaia di cifre, anche con i più potenti elaboratori Classici esistenti sono necessari almeno migliaia di anni per calcolare i due numeri primi fattore.

In generale, un algoritmo crittografico è ritenuto sicuro quando l'attacco più efficace ed efficiente è quello di provare tutte le chiavi segrete possibili. Quindi per utilizzare l'algoritmo in maniera sicura è sufficiente scegliere chiavi segrete lunghe a sufficienza (e pseudo-casuali, cioè impossibili da indovinare). In questo caso, l'avvento degli elaboratori quantistici richiede solo il raddoppio della lunghezza delle chiavi per rimanere in sicurezza rispetto all'algoritmo di Grover.

Molti algoritmi crittografici, quali quelli simmetrici come AES o di impronta come SHA256, sono basati su trasformazioni ripetute dei dati quali sostituzioni, permutazioni ed XOR. Questi algoritmi sono soggetti solo all'algoritmo di Grover. Al contrario, la quasi totalità degli algoritmi in uso a chiave pubblica-privata ed utilizzati quotidianamente per la navigazione web, le firme digitali ecc., sono basati su problemi matematici complessi ad "una via", ovvero impossibili in pratica da invertire, e questi sono soggetti all'algoritmo di Shor che li rende del tutto insicuri.

E' chiara quindi la necessità di sviluppare nuovi algoritmi in particolare a chiave pubblica-privata per sostituire quelli soggetti all'attacco di Shor. Sull'effettivo rischio di sicurezza, i crittografi non sono unanimi nel considerare realistico l'avvento degli elaboratori quantistici all'inizio degli anni 2030. Alcuni, seguendo ad

esempio l'analisi di M. Dyakonov, ritengono più opportuno indirizzare gli sforzi della ricerca nel miglioramento degli algoritmi attuali quali RSA, ECC, DH ecc. piuttosto che dedicarsi allo sviluppo di nuovi algoritmi crittografici.

Altri invece ritengono assolutamente necessario sviluppare algoritmi che non siano soggetti all'attacco di Shor. Questa direzione di ricerca è chiamata Crittografia "Post-Quantum", cioè che ha lo scopo di identificare e creare algoritmi crittografici che rimarranno sicuri dopo l'avvento degli elaboratori quantistici.

Si può far risalire l'avvio della ricerca di questi nuovi algoritmi al 2006 quando si svolse la prima conferenza PQCrypto presso la Katholieke Universiteit di Leuven in Belgio [Rif. 9]. L'interesse e la necessità di sviluppare questi nuovi algoritmi è stata supportata ad esempio anche dall'Unione Europea che negli anni 2015-2018 ha fornito 3,9 M€ per il progetto PQCRYPTO [Rif. 10]. Il NIST nel 2016 ha avviato un processo di selezione di nuovi algoritmi post-quantum [Rif. 11] con l'obiettivo di sostituire in pratica entro il 2030 gli algoritmi suscettibili all'attacco di Shor. Il processo di sviluppo e selezione degli algoritmi post-quantum è in corso ed è prevista la standardizzazione degli algoritmi selezionati entro il 2024. In questo modo rimarrebbero almeno 5 anni per implementare i nuovi algoritmi in tutti i sistemi IT, sia hardware che software prima dell'avvento degli elaboratori quantistici. Alla ricerca post-quantum partecipano non solo ricercatori universitari ma anche le principali aziende informatiche a livello mondiale, a partire da Microsoft, Google[48] ecc.

48) Solo come un esempio, Google ha svolto alcuni test sul campo dell'algoritmo post-quantum "New Hope" inserendolo nel suo browser Chrome.

5.1.5 ALGORITMI POST-QUANTUM

Ad oggi i ricercatori hanno proposto molti diversi algoritmi crittografici post-quantum con diversi approcci e basati su problemi matematici diversi. Una caratteristica abbastanza comune di questi algoritmi è la lunghezza delle chiavi e delle firme digitali, come si può evincere ad esempio da questa tabella:

Algorithm	Type	Public Key	Private Key	Signature
NTRU Encrypt[37]	Lattice	766.25 B	842.875 B	
Streamlined NTRU Prime[citation needed]	Lattice	154 B		
Rainbow[38]	Multivariate	124 KB	95 KB	
SPHINCS[19]	Hash Signature	1 KB	1 KB	41 KB
SPHINCS+[39]	Hash Signature	32 B	64 B	8 KB
BLISS-II	Lattice	7 KB	2 KB	5 KB
GLP-Variant GLYPH Signature[10][40]	Ring-LWE	2 KB	0.4 KB	1.8 KB
NewHope[41]	Ring-LWE	2 KB	2 KB	
Goppa-based McEliece[14]	Code-based	1 MB	11.5 KB	
Random Linear Code based encryption[42]	RLCE	115 KB	3 KB	
Quasi-cyclic MDPC-based McEliece[43]	Code-based	1,232 B	2,464 B	
SIDH[44]	Isogeny	564 B	48 B	
SIDH (compressed keys)[45]	Isogeny	330 B	48 B	
3072-bit Discrete Log	**not PQC**	384 B	32 B	96 B
256-bit Elliptic Curve	**not PQC**	32 B	32 B	65 B

Figura 5.1.1: Confronto tra algoritmi Post-Quantum [fonte Rif. 12]

In questa tabella sono raffigurate le lunghezze delle chiavi e delle firme digitali di alcuni algoritmi post-quantum necessarie per ottenere una sicurezza equivalente a quella offerta oggi con chiavi simmetriche a 128 bit. Per confronto le ultime due righe riportano gli stessi dati per due algoritmi attuali soggetti all'attacco di Shor.

Come si vede, molti algoritmi utilizzano chiavi e producono firme digitali anche mille volte più grandi di quelle prodotte dagli attuali algoritmi. Questo avrebbe almeno due conseguenze:

1. l'aumento di dati da trasferire e quindi del traffico in rete
2. l'utilizzo di maggiori risorse dei computer, smartphone, IoC ecc. sia di memoria sia della CPU per l'esecuzione degli algoritmi.

Quello ancora da fare da parte dei ricercatori non è poco: oltre a creare algoritmi post-quantum che permettano prestazioni compatibili con i dispositivi su cui saranno utilizzati e sufficientemente simili alle prestazioni degli algoritmi in uso oggigiorno, devono verificare che questi nuovi algoritmi siano "sicuri" anche rispetto a tutti i requisiti della crittografia non quantistica. Dato che alcuni di questi algoritmi sono basati su problemi matematici alternativi a quelli utilizzati in crittografia sinora, sono necessari studi approfonditi per poter raggiungere una sufficiente certezza dell'assenza di debolezze che possano portare alla "rottura" dell'algoritmo. Sarebbe grave e probabilmente molto costoso, se dopo aver adottato e implementato a livello mondiale un algoritmo post-quantum si scoprisse una sua falla rispetto ad attacchi effettuati con elaboratori Classici e si fosse obbligati a sostituirlo in tempi rapidi.

È da notare inoltre che sono ormai passati più di 25 anni dalla pubblicazione dell'algoritmo di Shor e che la ricerca nel campo dello sviluppo di nuovi algoritmi per elaboratori quantistici è stata ed è ancora molto attiva. Malgrado ciò, sino ad ora non sono stati trovati altri algoritmi quantistici con conseguenze simili per la

crittografia a quello di Shor. Si ritiene pertanto difficile che venga scoperto nel prossimo futuro un altro algoritmo quantistico con simili conseguenze per la crittografia.

Nella prima parte di questo capitolo si è mostrato come possa essere necessario richiedere che i requisiti di sicurezza forniti dalla crittografia durino anche per 10 o più anni. Pertanto, nel caso dell'avvento di un elaboratore quantistico nel 2030, ove necessario dovremmo già oggi utilizzare algoritmi resistenti all'attacco di Shor. Questo però non è possibile, anzi il NIST prevede che i nuovi algoritmi post-quantum saranno disponibili non prima del 2024 e completamente implementati e distribuiti per il 2030. Nell'incertezza di se e quando arriveranno gli elaboratori quantistici, questo ci lascia con un rischio sicuramente difficile da valutare e rispetto al quale ad oggi abbiamo poche contromisure possibili da adottare.

5.1.6 RIFERIMENTI BIBLIOGRAFICI

Rif. 1: si veda ad esempio https://en.wikipedia.org/wiki/Timeline_of_quantum_computing per una timeline dell'evoluzione degli elaboratori quantistici.

Rif. 2: per la nascita della teoria degli elaboratori quantistici si fa spesso riferimento alla presentazione di Richard Feynman intitolata "Quantum mechanical Hamiltonian models of discrete processes that erase their own histories: application to Turing machines" presentata alla prima conferenza in "Physics of

Computation", maggio 1981, MIT (USA).

Rif. 3: M.A. Nielsen, I.L. Chuang "Quantum Computation and Quantum Information", Cambridge University Press, 2010

Rif. 4: M. Mosca "Cybersecurity in an era with quantum computers: will we be ready?", https://eprint.iacr.org/2015/1075.pdf ; si veda anche https://post-quantum-cryptography.com/#tg

Rif. 5: Report of the Quantum Information Science and Technology Experts Panel "A Quantum Information Science and Technology Roadmap", 2/4/2004, https://qist.lanl.gov/pdfs/qc_roadmap.pdf

Rif. 6: F. Arute et. al (Google AI Quantum), "Quantum supremacy using a programmable superconducting processor", Nature 574, pag. 505 (2019), https://www.nature.com/articles/s41586-019-1666-5 ; "Google's Quantum Tech Milestone Excites Scientists and Spurs Rivals", IEEE Spectrum, https://spectrum.ieee.org/tech-talk/computing/hardware/googles-quantum-tech-milestone-excites-scientists-and-spurs-rivals

Rif. 7: "Qubit Count", https://quantumcomputingreport.com/scorecards/qubit-count/

Rif. 8: M. Dyakonov "The Case Against Quantum Computing", https://spectrum.ieee.org/computing/hardware/the-case-against-quantum-computing

Rif. 9: Per le conferenze e workshop PQCrypto si faccia riferimento a https://www.pqcrypto.org/conferences.html

Rif. 10: https://ec.europa.eu/digital-single-market/en/news/pqcrypto-eu-funded-project-success-story , https://pqcrypto.eu.org/

Rif. 11: Post-Quantum Cryptography, NIST Computer Security Resource Center, https://csrc.nist.gov/projects/post-quantum-cryptography

Rif. 12: https://en.wikipedia.org/wiki/Post-quantum_cryptography