TULUX

MacSec & Key Agreement

MacSec è un protocollo utilizzato per garantire confidenzialità ed integrità a livello 2. I canali di connessione sono unidirezionali, e a ciascun canale si associa una propria chiave di sessione **SAK**. La gestione delle chiavi non rientra strettamente in MacSec, bensì si parla esplicitamente di **MacSec Key Agreement** (**MKA**), che è un altro standard. **MKA** veniva usato per *determinare* chiavi sicure per le associazioni **SAK** con una stessa **Connectivity Association key CAK**. Essa è una chiave *pre-shared* (protegge durante l'autenticazione) usata per derivare:

- **ICK** - Chiave unica generata dal server di autenticazione per ogni sessione di autenticazione. *Utilizzata per proteggere l'integrità del traffico di rete con l'algoritmo di integrità MACsec.*
- **KEK** - Chiave unica generata dal server di autenticazione per ogni sessione di autenticazione. *Utilizzata per crittografare la chiave di sessione (SAK) prima della trasmissione al client.*

Questa derivazione appena vista, che parte da **CAK**, è un approccio statico.
Esiste anche un approccio dinamico, in cui si parte da una **MSK** pre-shared da cui poi si ottiene **CAK**.
L'associazione si annulla se non ci sono aggiornamenti per un certo periodo di tempo.

Quando l'utente è autenticato, può negoziare la **SAK** tramite **MKA**, la SAK viene generata da un **Key Server** eletto, il quale può anche revocare tali chiavi. La SAK deve essere presente in entrambi gli endpoint comunicanti, in quanto usata per proteggere il traffico di rete.

La **SAK** è una chiave unica generata dal server di autenticazione per ogni sessione di autenticazione. Utilizzata per proteggere il traffico di rete con la crittografia MACsec.

Quindi, un host che vuole creare una *Security Association* con un'altra entità (eletta a **Key Server**), la contatta, e in modo sicuro scambiano la **CAK**, da cui entrambi ricavano **KEK** e **ICK** mediante funzioni di derivazione delle chiavi (**KDF**). Il Key server genera **SAK** (tramite RNG), lo invia cifrato con **KEK**, e quindi nello scambio di messaggi si userà **SAK** per la crittografia e **IDK** per l'integrità.

DNS-SEC & PKI

DNSSEC (Domain Name System Security Extensions) è un'estensione del protocollo DNS che fornisce un meccanismo di autenticazione e sicurezza per la risoluzione dei nomi di dominio.

1. Vulnerabilità

In particolare, abbiamo che nel DNS i pacchetti viaggiano su UDP in chiaro, un attaccante in questo caso un potrebbe avvelenare la cache del name server DNS rispondendo con associazioni errate alle richieste DNS. La difficoltà per l'attaccante in questo caso sta nel fatto che i DNS server accettano risposte solo se loro hanno avviato le query. In più un attaccante ha la possibilità (Kaminsky's attack) anche di sostituirsi a un server DNS. Per fare entrambi questi attacchi però serve conoscere Query ID.

In più un attaccante può fare attacchi DoS, vedere il traffico rete di chi utilizza DNS (è in chiaro) e DNS hijacking (DNS risponde con siti malevoli) e DNS tunneling (traffico rete contenuto nei pacchetti DNS).

2. Meccanismi di DNSSec

In DNSSec abbiamo che per risolvere questi problemi i record vengono inviati cifrati. La **Public Key Infrastructure (PKI)** è utilizzata per garantire l'autenticità delle chiavi pubbliche utilizzate per firmare i record DNS.

Ogni zona DNS ha **due coppie di chiavi ZSK (zone signing key) e KSK (key signing key)**. Oltre ai record standard di DNS si aggiungono RRsig (per verificare KSK e ZSK pubbliche), DNSKey (contiene chiave ZSK di firma pubblica), NSEC, NSEC3, DS, CDS, CDNS.

raggruppati per "tipologia"

I record inviati (RRSet) vengono firmati con la **ZSK privata**, per verificarli si può ottenere la **ZSK pubblica** attraverso DNSKey. In questo modo sappiamo che il record viene dalla zona desiderata, ma non sappiamo se effettivamente la zona è autentica. Per confermare ciò si usa la coppia di **KSK**. Con la **KSK privata** vengono firmati i record DNSKey, attraverso questi record è possibile conoscere la KSK pubblica per verificare la firma. Per verificare poi che la chiave pubblica appartenga effettivamente a quella zona, quello che si fa in DNSSec è andare al name server di livello superiore e ottenere il record DS relativo alla zona desiderata, questo record è l'hash della DNS key della zona. Quindi si ottiene il DNS key della zona, si calcola l'hash e si confrontano.

KSK: firmano la chiave ZSK pubblica (memorizzata in un record DNSKEY) e creando un RRSIG per il DNSKEY.

Questo passaggio può essere fatto per ogni name server fino ad arrivare al root. Questo ha una chiave KSK che viene stabilita durante la cerimonia della selezione della chiave (in cui un gruppo di persone importanti si riuniscono in una stanza piena di telecamere per far vedere che tutto vada secondo quanto stabilito).

Questa chiave KSK root può essere utilizzata per stabilire l'autenticità dei vari name server sottostanti (è il termine della chain).

Proprio come la ZSK pubblica, il nameserver pubblica la KSK pubblica in un altro record DNSKEY

In particolare, dopo aver selezionato la chiave vengono condivisi DNS key e RRsig anche della root che possono essere utilizzati per le verifiche.

Sia la KSK che la ZSK pubbliche sono firmate dalla KSK privata

Modificare la ZSK è semplice ogni zona lo può fare in qualsiasi momento, mentre modificare la KSK è più complesso perché bisogna modificare il record DS nella zona di livello superiore.

3. Altre criticità

DNS sec presenta comunque delle vulnerabilità in quanto i record NSEC e NSEC3 vengono utilizzati per rispondere nel caso in cui non venga trovata l'associazione per una certa richiesta DNS. In questo caso questi rispondono con il nome host più vicino a quello effettivamente richiesto (NSEC in chiaro e NSEC3 lo protegge con hash). Questo dà informazioni all'attaccante che potrebbe riuscire a ricostruire come è fatta la zona. Anche con l'hash infatti potrebbe portare avanti un dictionary attack provando a ricostruire il risultato coperto con hash, oppure potrebbe portare avanti un CPA. In più poiché calcolare l'hash è un'operazione pesante potrebbe fare anche un attacco di tipo reflection e amplification portando ad un attacco DoS in cui il sistema smette di funzionare. Per cercare di evitare questo problema si utilizzano le curve ellittiche al posto delle normali funzioni di hash in modo che con meno bit e meno calcoli danno lo stesso livello di sicurezza.

BGP

4. Introduzione

BGP è un tipo di **Border gateway protocol**: Un protocollo di routing progettato e destinato all'uso tra diversi sistemi autonomi AS. (in questo caso è Exterior Gateway Protocol, visto che è applicabile anche internamente ad AS). È un **Distance (path) vector routing protocol**, in quanto tutti questi annunci mantengono la lista degli AS attraversati per quella specifica rotta. BGP permette agli AS di scambiarsi delle informazioni di routing. BGP gira su TCP. Ogni AS deve fornire le proprie rotte assegnate (non tutte, dipende dall'AS policy). Se è un AS di transito devo inviare gli annunci che ricevo da altri AS. *BGP sincronizza tutte le rotte per tutti gli AS*. E' possibile avere più rotte per la stessa sottorete imparate attraverso diversi routing protocols, quale ha la precedenza? Dobbiamo vedere la distanza amministrativa che serve per assegnare priorità alle rotte. Ricevo una rotta tramite BGP, la filtro, in base a delle policy decido se inserire nella mia routing table l'informazione che è arrivata. Dopo posso decidere se inoltrare l'annuncio ad altri vicini (anche chiamati peer).

Classifichiamo:

- Internamente ad un AS si parla di *iBGP*. Questa categoria si suddivide in:
  - Tipologia **Distance vector**: RIP
  - Tipologia **Link state**: OSPF
- esternamente si parla di *eBGP*.

Due router instaurano una connessione scambiandosi le *routes* e mandandosi *heartbits*.

I messaggi vengono scambiati dai router mediante UPDATE.

Il path migliore è basato su *shortest path*. Per evitare *loop* interni, un router appreso tramite *IBGP* non viene propagato agli altri router.

I *loop esterni* si identificano da AS-PATH.

Interessante è il legame con **MPLS - Multi protocol Label Switching**, con cui BGP scambia messaggi di controllo. trattato successivamente

BGP Security

Non siamo più nella nostra rete locale e la sicurezza è un grande issue:

1. Minacce relazionate ai protocolli ai livelli sottostanti (TCP/IP).
2. *Minacce al BGP ROUTING CONTROL PLANE*: hijacking (dirottamento) del prefisso BGP, route leaks, miss-routing, dirottamento del traffico non voluto, degradazione delle prestazioni.
3. *Attacchi DATA PLANE*: Attacchi DoS (DDoS) distribuiti su larga scala ai server che utilizzano indirizzi IP (Internet Protocol) contraffatti, amplificazione della riflessione.

5. Vulnerabilità di TCP/IP che influenzano BGP

5.1. Peer spoofing e tcp reset

Il "*peer spoofing*" è un attacco che mira a fingere l'identità di un peer in una comunicazione di rete, in modo da poter inviare messaggi fraudolenti o ricevere informazioni riservate.

Il *TCP reset*, invece, è un tipo di attacco informatico che mira a terminare una connessione TCP in corso. Esistono diverse contromisure che possono essere adottate per prevenire o mitigare questo tipo di attacco.

I messaggi BGP possono essere falsificati (anche se risulta più difficoltoso) in modo da apparire come un messaggio proveniente da un peer valido e inserire informazioni false in una tabella di routing BGP. Gli indirizzi IP possono spesso essere trovati utilizzando la funzione traceroute ICMP, quindi le implementazioni BGP dovrebbero includere contromisure contro questo attacco. Un caso speciale di peer spoofing, chiamato attacco di reset, comporta l'inserimento di messaggi TCP RESET in una sessione in corso tra due peer BGP. Quando viene ricevuto un ripristino, il router di destinazione interrompe la sessione BGP ed entrambi i peer ritirano i percorsi precedentemente appresi l'uno dall'altro interrompendo, così, la connettività di rete fino al ripristino. TCP ha un numero di sequenza: se inserisco un numero nella finestra corretta (provo tutti i numeri con brute force, ho al piu 65k guesses da fare) posso mandare un finto messaggio RESET con TCP. Posso anche usare messaggi ICMP per causare un reset della sessione TCP.

perchè limitarmi a chiudere la connessione?

5.2. Dirottamento della sessione hijacking

Il dirottamento della sessione comporta l'intrusione in una sessione BGP in corso. L'attacco può essere progettato per ottenere qualcosa di più che il semplice abbattimento della sessione tra peer BGP. Ad esempio, l'obiettivo potrebbe essere quello di modificare i percorsi utilizzati dal peer, al fine di facilitare le intercettazioni, blackholing, o analisi del traffico.

5.2.1. Contromisure

- **strong sequence number randomization**: Questa tecnica consiste nell'utilizzare numeri di sequenza casuali e imprevedibili per l'invio dei messaggi, in modo da rendere più difficile per un attaccante l'intercettazione e l'inserimento di messaggi fraudolenti nella comunicazione. In pratica, ogni pacchetto inviato dal mittente contiene un numero di sequenza casuale, che viene verificato dal destinatario per garantire l'autenticità del pacchetto. In questo modo, un attaccante che cerca di intercettare la comunicazione e inviare messaggi falsi non potrebbe conoscere il prossimo numero di sequenza da utilizzare.

- **TCP sequence number checking** in entrata può rendere più difficile un attacco di reset, in quanto l'attaccante deve eseguire un attacco brute force sul numero di sequenza per far si che la connessione venga terminata.

in realtà è già implementato, quindi non si sa che dovrebbe controllare...

- Un'altra precauzione è il **TTL hack**. Nella configurazione di BGP, i peer sono solitamente adiacenti, richiedendo un solo hop per un pacchetto inviato in un messaggio BGP. In condizioni normali, ci si aspetta che il TTL non cambi quando si comunica due volte con lo stesso peer BGP. L'utilizzo del meccanismo di sicurezza TTL generalizzato (TTL Hack) consiste nell'impostare il TTL a 255 sui

pacchetti in uscita. I peer adiacenti dovrebbero quindi ricevere i pacchetti in entrata con TTL = 255. Questo approccio è efficace a meno che non ci siano attacchi di tipo MITM in grado di modificare i pacchetti. Durante l'implementazione dell'hack TTL, è possibile specificare anche un valore atteso per il TTL in ingresso.

- MD5 viene utilizzato per proteggere le sessioni BGP tramite un hash con chiave per **autenticare i messaggi e l'header**. Tuttavia, MD5 è considerato obsoleto e debole. Ogni segmento inviato su una connessione TCP per prevenire lo spoofing contiene un digest MD5 a 16 byte, ottenuto applicando l'algoritmo in un ordine specifico: pseudo-header TCP, header TCP (senza opzioni), dati del segmento TCP (se presenti), e una chiave o password condivisa da entrambe le parti TCP specifica della connessione.

- Infine, IPsec può essere utilizzata per garantire l'autenticità, integrità e confidenzialità dei pacchetti TCP. IPSec fornisce meccanismi di autenticazione e crittografia per i pacchetti IP, garantendo che i pacchetti provengano da fonti affidabili e non siano stati modificati durante il trasporto.

5.3. Route flapping

Il continuo cambiamento delle tabelle di routing BGP, noto come "**route flapping**", è problematico. In questo scenario, un nodo BGP annuncia e ritira la stessa rotta ripetutamente, propagando queste modifiche attraverso gli AS e le loro tabelle di routing BGP. Il flapping di rotta ad alta velocità può causare seri problemi ai router, in quanto ogni flapping provoca cambiamenti o ritiri delle rotte che si diffondono nella rete degli AS.

Quando i cambiamenti delle rotte avvengono rapidamente, ad esempio da 30 a 50 volte al secondo, i router possono essere sovraccaricati, ostacolando la convergenza su rotte valide. Questo fenomeno rappresenta un rischio di attacco che può portare a un Denial of Service (DoS), sia accidentalmente, ad esempio a causa di un link difettoso, sia da parte di un attaccante esterno.

5.3.1. Contromisure

Smorzamento del Route Flap: è un metodo per ridurre i route flap implementando un algoritmo che ignora il router che invia aggiornamenti di flapping per un periodo di tempo configurabile. Ogni peer ha un penalty value che viene incrementato ogni volta che avviene un evento di flapping (la penalità decade esponenzialmente nel tempo). Se questo valore supera una soglia prestabilita, allora non processo gli update che il peer malevolo annuncia.

6. Vulnerabilità BGP Control Plane

6.1. Dirottamento del prefisso e annuncio di un indirizzo non allocato - prefix hijacking

Si verifica quando un Autonomous System (AS) genera, sia accidentalmente che intenzionalmente, un prefisso che non è autorizzato dal legittimo proprietario di quel prefisso. In questo caso, l'annuncio è considerato illegittimo. La selezione del percorso IP preferisce sempre la regola della specificità, ovvero privilegia il prefisso più lungo.

NB: C'è differenza tra AS_PATH (preferisco il più breve) e PREFISSO (192.168.178.1/30 è preferito, perchè più specifico, di 192.168.178.1/24)

Quando un AS annuncia in modo errato o malevolo un prefisso più specifico rispetto a un prefisso annunciato da un AS autorizzato, il prefisso più lungo e non autorizzato può essere ampiamente accettato. Questo accade perché la regola della specificità favorisce l'indirizzo più lungo e, quindi, più specifico. Allo stesso modo, un AS potrebbe anche erroneamente originare uno spazio di indirizzi assegnato ma attualmente non utilizzato.

6.2. Modifica del messaggio BGP UPDATE

Gli AS_PATH sono utilizzati per implementare politiche di routing che riflettono accordi commerciali.

- **MODIFICA DELL' AS_PATH**:
  Rimuovo qualche AS nell'AS_PATH in modo da ricevere il traffico verso di me. Questo può essere fatto da alcuni AS malevoli in modo da:
  - Sembrare più ricercati e farsi pagare di più;
  - Intercettare i pacchetti che, altrimenti, avrebbero preso altre strade.
- **MODIFICA DEL PREFISSO BGP**:
  Modifico il prefisso per renderlo più lungo (e specifico). Il risultato è lo stesso di sopra … avrò più traffico verso di me.

6.3. ROUTE LEAK e definizione delle peering relationship

Il problema nasce tra le policies che ci sono tra deu AS che non vengono rispettate. Una route leak è la propagazione degli annunci di routing con fini al di la dello scopo previsto. Per esempio con uno fine malevolo. Ne derivano:

- **Eavesdropping o analisi di traffico** a causa della ridirezione del traffico attraverso un percorso non voluto. Sarebbe un ascoltare come MITM.
- Quando un gran numero di percorsi è leaked contemporaneamente, l'AS offending è spesso sopraffatto dal conseguente traffico di dati inaspettato e perde gran parte del traffico che riceve. Ciò causa **blackholing** e **denial-of-service DOS** per i prefissi interessati.

6.4. Soluzione per queste vulnerabilità

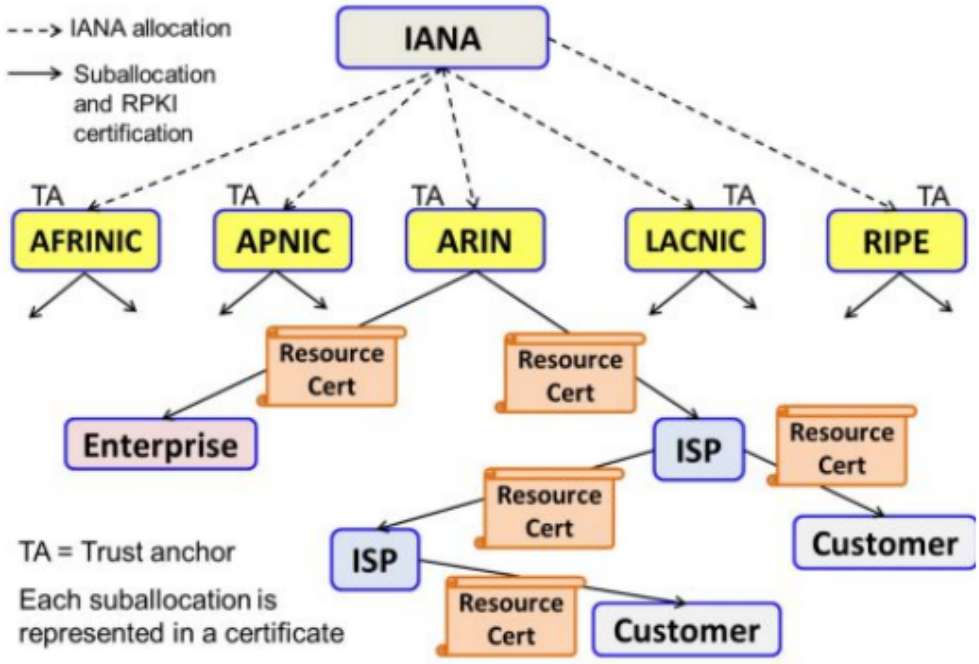
6.4.1. RPKI

Esistono dei registri *RIR* e *IRR* che memorizzano i dati sulle allocazioni di risorse politiche di routing, origine dei prefissi. I dati non sempre sono affidabili. La soluzione è nell'approccio **Resource Public Key Infrastrcture RKPI**, framework standard per fornire *registri crittograficamente protetti* e *certificazioni/autorizzazioni di routing*. **RPKI** è un'autorità CA e un servizio di registro offerto dai *registri internet regionali*.

La IANA assegna risorse ai registri RIR, che a loro volta subassegnano risorse agli ISP.

Si ha quindi una catena di certificazioni *RPKI*. I *RIR* mantengono una chiave pubblica (*trust anchor*) per stabilire fiducia nella gerarchia *RPKI* e rilasciano certificati delle risorse agli ISP.

RIR



La certificazione delle risorse avviene secondo due modelli:

- *Hosted*: il RIR conserva le chiavi RPKI e le gestisce sul proprio server.
- *Delegated*: una ISP riceve certificato dal proprio RIR, lo ospita ed esegue operazioni RPKI. Può quindi firmare autorizzazioni, rilasciare certificati. Questo significa che ogni organizzazione che gestisce ROA per i propri Autonomous System (AS) deve ottenere un certificato da un'organizzazione di livello superiore. In sintesi, la catena di certificati RPKI nel modello delegated viene generata attraverso la creazione di certificati di organizzazioni di livello superiore che poi firmano i certificati delle organizzazioni downstream e le loro ROA. In questo modo, la catena di certificati RPKI garantisce la provenienza e l'autenticità delle ROA.

6.4.2. BGP OV - Origin Validation

grazie ad RPKI, "so di chi fidarmi"

Un router BGP riceve un *elenco validato* di tuple (*prefix*, *maxlenght*, *originAS*) da **RPKI cache server** (trattato in un'altra domanda, qui diciamo che associa in modo sicuro risorse IP ad una CA). Il router, per determinare lo stato di *una rotta che gli arriva*, usa **BGP Origin Validation - BGP-OV**:

- Una rotta ha origine valida se (prefix, originAS) corrisponde con l'elenco.
- La rotta è *NotFound* se il prefisso non è coperto da altri prefissi nella white-list.
- Se la sequenza di AS_PATH è *non ordinata*, non si può determinare l'origine da *AS_PATH*.

6.5. Altra vulnerabilità: Forged-Origin Hijacks

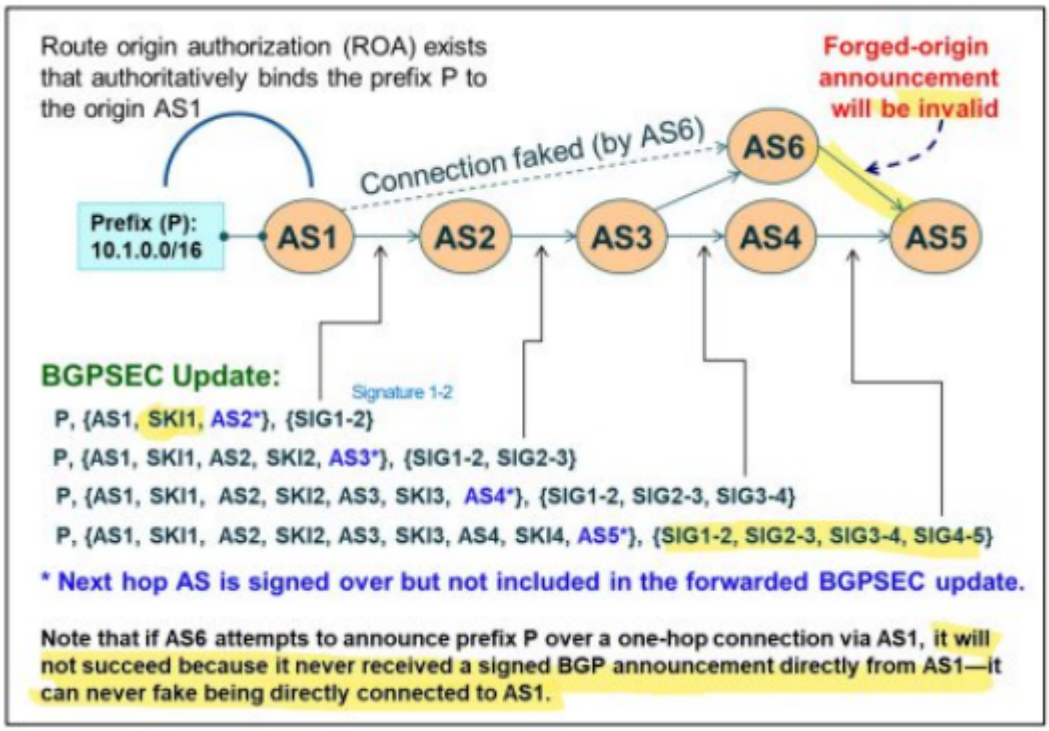
La **BGP origin validation** è *necessaria* ma, di per sé, non è sufficiente per proteggere completamente il prefisso e il percorso AS negli annunci BGP. Con la validazione dell'origine basata su ROA un AS non

può annunciare un prefisso a meno che non abbia un ROA valido. Tuttavia, è possibile falsificare l'AS di origine (AS «proprietario» del prefisso) per un qualsiasi BGP UPDATE prendendo l'ASN in un ROA valido (per il prefisso target) e anteponendolo al nuovo messaggio di UPDATE non autorizzato. In questo nuovo UPDATE posso anche sostituire il prefisso nel percorso con uno più specifico con lunghezza non superiore alla massima specificata nel ROA.

6.5.1. Soluzione: BGP PV - Path Validation

BGP PV è un meccanismo per proteggere l'annuncio BGP da modifiche dei prefissi.

- Ogni AS ha un certificato di risorsa per il proprio ASN.
- Ogni router ha certificato e chiave privata per firmare gli update.
- Ogni AS usa la chiave privata del router per firmare l'aggiornamento BGP, i**ncludendo nei dati da firmare il prossimo AS.** Questo è l'aspetto importante, infatti nella figura AS6 non può far finta di aver comunicato con AS1.
- Ogni AS riceve più firme da verificare. Se tutto ok, l'aggiornamento è valido.



Packet Classification

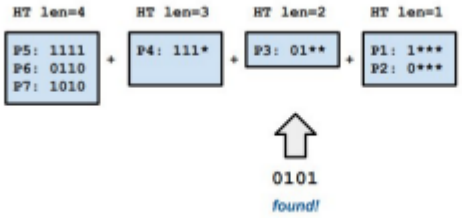
E' processo che consiste nell'identificare e categorizzare i pacchetti di rete in base alle loro caratteristiche, come l'*indirizzo IP di origine* e destinazione, la *porta* di origine e destinazione, il protocollo di rete, e così via. La packet classification è un'attività cruciale per molti sistemi di rete, come firewall, router, gateway, e così via. Possiamo fare tale classificazione in diversi modi, identifichiamo principalmente *schemi ad una dimensione*, *schemi a due dimensioni* e *schemi multidimensionali*.

7. Schemi ad una dimensione [non fatti]

Basati un solo attributo, facili ma con traffico elevato inadeguati, perchè aumentano regole di classificazione.

Per il matching identifichiamo:

- **Trie**: Rappresento con un albero che non ha una chiave in ogni nodo. Può essere usato per il prefix lookup: la ricerca parte dalla root. La priorità è associata alla LUNGHEZZA del prefisso.
- **Binary Search sui range**: ogni prefisso è come un range (ad esempio da 10*00* a 10*11*, cambiano gli ultimi 2), e quindi cerchiamo l'indirizzo di destinazione nel range opportuno.
- **Hashing Exact Match**: Creo tabelle hash basate sulla lunghezza (*ovvero il numero di bit che non cambiano*), collegate come una chain. Tipicamente, si fa *scansione lineare* prendendo la tabella hash con lunghezza maggiore.



  Genera problemi nella collisione.

8. Schemi a due dimensioni [non fatti]

Due dimensioni perchè si fa il matching sia su prefisso di destinazione, sia su quello della sorgente. Usano le stesse metodologie viste prima, anche se la migliore è *trie*.

Con **SET-PRUNING TRIES**: inizio a creare un trie per la destinazione e, in ogni prefisso valido di destinazione, appendo un “source trie”. Il problema è che un prefisso può appartenere a più trie, ad esempio, se la regola R_1 è associata a destinazione $D = 0*$, questa regola R_1 appartiene anche a $D' = 00*$. La soluzione più semplice è lasciarle nella destinazione più generica.

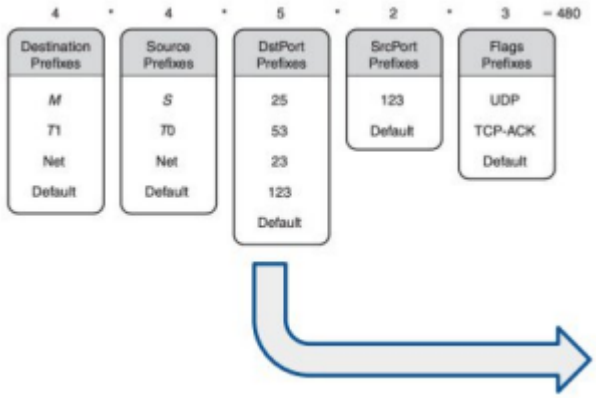
9. Schemi multidimensionali [fatto solo bit vector]

In generale sono molto costosi (crescono esponenzialmente con l’aumentare delle dimensioni) quindi sono nate delle semplificazioni. Per estendere gli schemi a due dimensioni potremmo:

- *cercare linearmente*, magari partendo dai ragionamenti fatti su due dimensioni.
- *divide et impera*, gestendo ogni dimensione separatamente, unendo poi i risultati. Esistono tre metodi che differiscono per come combinano i risultati:
  - **Bit Vector Linear Search**: Ho un database per ogni campo su cui fare il lookup. Faccio l'AND dei matching, quindi se trovo un matching tra fields `T1` e colonne `b4,b5`, negli altri DB mi concentrerò su chi ha questi campi. Associamo 1 nelle colonne che corrispondono a righe

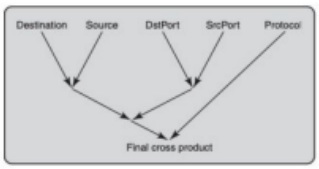
matchate con un certo campo nel firewall, 0 altrimenti. Se alla fine ho più matching, prendo il meno costoso.

- **Rule cross-producting**: Nella variante *Naive rule Cross Producting* si fa il prodotto cartesiano e si pre-calcola la matching rule. Espansione della memoria importante. Data l'intestazione di un pacchetto, si determina la regola di corrispondenza meno costosa, e poi nella tabella prodotta si fa la ricerca.



Number	Cross Product	Matching Rule
1	*M, S*, 25, 123, UDP	Rule 1
2	*M, S*, 25, 123, TCP-ACK	Rule 1
3	*M, S*, 25, 123, *default*	Rule 1
4	*M, S*, 25, *default*, UDP	Rule 1
5	*M, S*, 25, *default*, TCP-ACK	Rule 1
6	*M, S*, 25, *default*, *default*	Rule 1
...	...	...
479	*default, default, default, default*, TCP-ACK	Rule 8
480	*default, default, default, default, default*	Rule 8

Nella variante *Equivalenced Cross-producting*, si cerca di fare compressione della tabella prodotta, a costo di cercare in modo più lento. Come? Mappando prodotti piccoli in *classi di equivalenza*, che avranno un *numero di classe* usato per le combinazioni.



Destination–source prefix pairs	Rule bitmap	Class number
M, S	11110011	C1
M, 70	11010011	C2
M, Net	11010111	C3
*M, **	11010011	C2
71, S	00000011	C4
71, 70	00001011	C5
71, Net	00000111	C6
*71, **	00000011	C4
Net, S	00000011	C4
Net, 70	00000011	C4
Net, Net	00000111	C6
*Net, **	00000011	C4
**, S*	00000001	C7
**, 70*	00000001	C7
**, Net*	00000100	C8
**, **	00000001	C7

partial cross product dest X src

A sinistra abbiamo una possibile combinazione in coppie.

A destra abbiamo la combinazione di destination e source. Per ogni coppia (ad esempio, M,S) calcoliamo l'insieme di regole compatibili esattamente con una tale coppia di corrispondenze. Per esempio M,S è l'AND del bit vector di M per destination e S per source.

Le coppie che hanno rule bitmap uguale fanno parte della stessa classe di equivalenza: poiché la bitmap della regola per M è 11110111e la bitmap per S è 11110011, la bitmap di intersezione per M , S è **11110011**

Architettura VXLAN (come fa forwarding) + cos'è un VTEP e differenza tra L2VNI e L3VNI

VXLAN (Virtual eXtensible Local Area Network) è una tecnologia di rete utilizzata per estendere e scalare le reti virtuali. VXLAN è una *tecnologia di virtualizzazione di rete* che consente di estendere le reti Layer 2 (*VLAN*) su una rete Layer 3 (*IP*). In pratica, VXLAN consente di *creare una rete virtuale su una rete fisica esistente*, consentendo a diverse macchine virtuali (VM) di comunicare tra loro come se fossero sulla stessa rete IP locale. VXLAN è progettato per supportare un gran numero di VM e di siti.

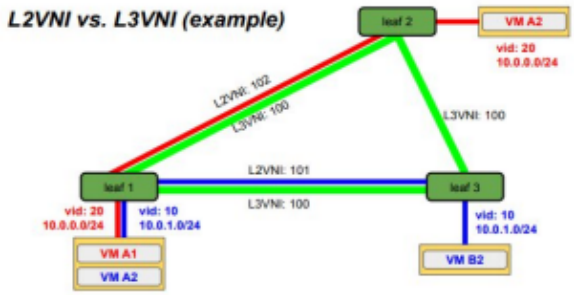
Ciò viene realizzato permettendo di inserire frame L2 in datagrammi IP/UDP, in cui solo i nodi *edge* devono supportare l'incapsulamento, gli altri vedono datagrammi IP.

VTEP (VXLAN tunnel end point) è il terminale che deve supportare VXLAN encapsulation/decapsulation, oltre che avviare e terminare il tunnel VXLAN. Nel *VXLAN header*, un

campo particolarmente interessante è *VNI*, ovvero l'identificatore della VXLAN. Nello specifico:

- **L2VNI** è VNI associato ad un solo *broadcast domain* (*stessa* sottorete IP) di uno specifico tenant.
- **L3VNI** è VNI associato ad un "tenant per layer 3 routing" in *diverse* sottoreti IP, quindi usato per fare tunneling tra braodcast domain diversi.

Esempio:



Le access port si trovano in VLAN differenti. Tra VM A1 e VM A2 c'è un tunnel di L2VNI (notare che hanno lo stesso broadcast domain:10.0.0.0/24).
L3VNI permette di attraversare un BD (da 10.0.1.0/24 a 10.0.0.0/24) per raggiungerne un altro, per esempio per andare da VM B2 a VM A1 .

In una rete VXLAN, i pacchetti LAN sono incapsulati all'interno di pacchetti IP, che consentono di trasportare il traffico LAN attraverso una rete IP. Il formato dei pacchetti VXLAN è composto dai seguenti campi (riporto quelli interessanti):

- *VXLAN Network Identifier (VNI)*: un campo di 24 bit che contiene l'ID di rete VXLAN. Questo campo consente di distinguere le diverse reti VXLAN all'interno della stessa rete fisica (discrimino i due tenants che appartengono alla stessa sotto-rete IP).
- *VXLAN header*: un campo di 8 byte che contiene le informazioni di controllo per la comunicazione VXLAN.

Il forwarding in una rete VXLAN è basato sull'ID di rete VXLAN (VNI). Quando un pacchetto VXLAN arriva ad uno switch di accesso, lo switch decapsula il pacchetto e legge l'ID di rete VXLAN (VNI). Lo switch quindi utilizza l'ID di rete VXLAN per instradare il pacchetto verso la destinazione corretta. Se il destinatario si trova in un'altra rete VXLAN, lo switch di accesso incapsula il pacchetto VXLAN in un nuovo pacchetto IP e lo invia all'altro switch di accesso tramite la rete IP.

10. Data Center Topologies delle VXLAN: standard e CLOS topology.

Parliamo di **Data center topologies**.

In un **basic tree** le limitazioni principiali sono:

- A livello *core network tier*, gli switch principali, deono supportare la piena larghezza di banda aggregata dei livelli inferiori.
- *Bassa tolleranza ai guasti*: se un nodo nell'albero fallisce, tutti i suoi "rami" più piccoli sono fuori servizio
- Gli switch di livello superiore sono molto costosi.

Oggi vogliamo *scalabilità* (espansione rete senza toccare architettura), *ridondanza* (no single point of failure), *latenza* (riduzione hop), *capacità della rete* e *isolazione dei tenants*.

Oggi i datacenter hanno una topologia Clos (da Charles Clos 1952). La topologia di rete Clos è una architettura di rete comune per i data center, utilizzata per garantire una maggiore scalabilità, affidabilità e flessibilità.

- **Scalabilità**: la topologia di rete Clos può essere facilmente scalata aggiungendo più switch di accesso o di aggregazione. Inoltre, i flussi di traffico sono distribuiti in modo uniforme attraverso i diversi switch, evitando congestionamenti della rete.
- **Affidabilità**: la topologia di rete Clos è progettata per garantire la ridondanza, in modo che la rete continui a funzionare anche in caso di guasti di uno o più dispositivi di rete.
- **Flessibilità**: la topologia di rete Clos può essere utilizzata con diverse tecnologie di rete, tra cui Ethernet, Fibre Channel e InfiniBand. Inoltre, la topologia di rete Clos consente di implementare diverse politiche di rete, come il load balancing, la QoS (Quality of Service) e la sicurezza.

Nei data centers tipicamente abbiamo:

- ***Two-tier topology (aggregation - core)*** $\rightarrow$ **leaf-spine topologies**
- ***Three-tier topology (access – aggregation – core)*** $\rightarrow$ **fat-tree topologies**

11. Leaf spine topologies

Ogni foglia ha una connessione per ogni SPINE.

- Se si necessita di una maggiore larghezza di banda, è possibile aggiungere un'altra SPINE alla configurazione.
- Per ottenere una maggiore potenza computazionale, si può aggiungere un'altra foglia e collegarla alle spines esistenti.
- In caso di guasto nelle SPINE, la raggiungibilità tra i nodi è comunque garantita dalle spines rimanenti. I protocolli di routing rilevano gli errori nei collegamenti o nei nodi e possono adattare l'inoltro di conseguenza.
- Se si verifica un guasto nelle foglie, i server rack a esse collegati vengono dichiarati fuori servizio.

Per migliorare la tolleranza ai guasti, è possibile implementare la tecnologia LAG o MLAG.

- **LAG (Link Aggregation Group)**: Una tecnica di multiplexing inverso in cui un gruppo di collegamenti viene aggregato per garantire sia la ridondanza che la moltiplicazione della larghezza di banda, creando un singolo collegamento virtuale ad alta velocità.
- **MLAG (Multi-Chassis Link Aggregation Group)**: Aggiunge la ridondanza non solo a livello di link ma anche a livello di switch, fornendo una soluzione di aggregazione di link multi-chassis.

12. Fat-tree topologies

Il Fat-tree è una topologia che estende la struttura leaf-spine della topologia Clos aggiungendo un livello. Concettualmente, può essere visualizzato come una foglia esplosa in quattro foglie, e questo gruppo di quattro foglie è denominato "pod".

Ethernet LAN e vulnerabilità

Ethernet è una tecnologia LAN definita dallo standard IEEE 802.3, utilizzata per trasmettere dati su reti cablate. Questa tecnologia facilita la connessione e la comunicazione tra dispositivi attraverso un cavo condiviso.

Vulnerabilità: I problemi di Ethernet derivano dalla sua natura di autoconfigurazione. Le caratteristiche come la *MAC table learning*, *Spanning tree protocol*, e *ARP*, insieme al meccanismo di broadcast, sono le principali fonti di vulnerabilità.

Accesso al Segmento Ethernet Target: L'aggressore, potendo autoconfigurare, può ottenere accesso al segmento Ethernet target. Questo accesso può essere ottenuto da un insider con pieni diritti di accesso, attraverso una connessione Ethernet pubblica, o prendendo il controllo di una postazione di lavoro tramite malware o altri metodi.

Possibili Attacchi:

1. **Apprendimento della Topologia della Rete:**
    - Un attaccante può analizzare le trame Ethernet per apprendere la topologia della rete e il traffico, ottenendo informazioni utili per attacchi futuri.
2. **Ottenere Controllo dei Dispositivi di Rete:**
    - Sfruttando l'accesso a qualsiasi trama Ethernet, l'attaccante può cercare di assumere il controllo di switch, router o server nella LAN.

13. Vulnearibilità di accesso alla rete

L'accesso non autorizzato a un segmento Ethernet può avvenire attraverso diverse modalità:

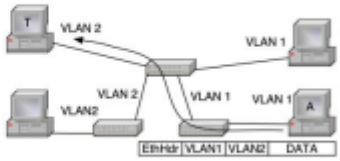
- **Unioni non autorizzate**: Qualsiasi individuo può connettersi a un segmento Ethernet ottenendo l'accesso a una porta non utilizzata su uno switch. Questo può essere fatto tramite l'accesso fisico allo switch, se la porta è attiva, oppure tramite l'accesso a una presa a muro. Altre tecniche includono rimuovere il cavo da un PC e collegarlo altrove, o inserire uno switch tra il PC esistente e la presa.
- **Espansioni della rete non autorizzate**: L'architettura di Ethernet consente agli utenti di estendere la rete installando il proprio switch o punti di accesso wireless, permettendo così ad altre persone di accedere alla rete senza autorizzazione.
- **VLAN join**: Se uno switch ascolta i protocolli di gestione VLAN sulle porte host, un host può agire come uno switch unendosi a tutte le VLAN presenti.

- **VLAN tagging and hopping**: Un utente malintenzionato può creare frame Ethernet con un tag VLAN e iniettare così frame in VLAN a cui non dovrebbe avere accesso, sfruttando la manipolazione dei tag VLAN. Un esempio di attacco è *double tagging attack*:

  Un esempio è il **«double tagging» attack**:

  A è l'attacker, T è il target.

  1. L'aggressore crea un frame che ha l'indirizzo MAC dell'host di destinazione come destinatario e contiene un tag VLAN 1 seguito da un tag VLAN 2
  2. Lo switch rimuove il tag e invia il frame al collegamento trunk della VLAN 1, dove lo switch ricevente rileva il secondo tag ed elabora il frame come appartenente alla VLAN di destinazione
  3. NESSUNA capacità di traffico di ritorno, ma lo spoofing aggiuntivo può farlo
  4. Vari attacchi possono essere eseguiti sul flusso unidirezionale

- **Accesso remoto alla LAN**: Ottenere accesso a un segmento Ethernet può avvenire attraverso l'accesso di livello superiore a un host sulla LAN, ad esempio tramite servizi offerti da un end-host nella stessa rete.

- **Scoperta della topologia e delle vulnerabilità**: Un utente malintenzionato può esplorare la rete per individuare host e servizi, inviando messaggi e analizzando le risposte. Questo può includere la rivelazione degli indirizzi IP tramite richieste broadcast ARP, la determinazione dell'intervallo di indirizzi IP in uso tramite richieste al server DHCP e la scansione dettagliata della rete per identificare sistemi operativi, servizi e versioni, rivelando potenziali vulnerabilità.

- **Attacchi software di rete (break-ins)**: Un utente malintenzionato può sfruttare la rete Ethernet per attaccare altri host e switch, mirando alle vulnerabilità del software di rete, come lo stack TCP/IP e le applicazioni server. Questi attacchi possono portare alla cattura di un host o di uno switch, che poi può essere utilizzato per ulteriori attacchi.

- **Controllo dello switch**: Gli switch spesso hanno password predefinite o nessuna password, e queste possono essere reimpostate fisicamente. Se un utente malintenzionato ottiene il controllo di uno switch, può dirottare il traffico disattivando collegamenti, rivendicando la radice del protocollo STP aumentando la priorità dello switch o causando un Denial of Service (DoS) su collegamenti specifici.

14. Confidenzialità del traffico

prima l'hub era dummy

L'Ethernet coassiale originale era un bus facilmente intercettabile, dove ogni stazione riceveva ogni frame. La moderna Ethernet con bridge/switch filtra la maggior parte del traffico e un host riceve solo il proprio traffico, trasmissioni e frame casuali inondati sullo switch dopo un timeout della tabella MAC (non c'è piu il mapping nella tabella MAC).

- **Passive eavesdropping / sniffing**: Una volta che si è ottenuto l'accesso alla rete, si può fare lo sniffing (passive eavesdropping) del traffico di rete. L'intercettazione passiva è possibile se un utente malintenzionato può collegare un dispositivo di ascolto a un cavo che collega un host a uno switch o tra due switch.

- **Eavesdropping with MAC flooding**: Se uno switch non sa dove inoltrare un frame, lo espelle da tutte le sue porte. Con il software, un utente malintenzionato può facilmente generare abbastanza frame con indirizzi casuali per sovrascrivere un'intera tabella MAC e fare in modo che lo switch invii tutti i frame di dati a tutte le porte per l'intercettazione. Sulla maggior parte degli switch questo colpisce tutte le VLAN, anche se l'attacco ha origine all'interno di una VLAN.

Tuttavia, il problema risiede anche nella procedura di *learning dello switch*: l'invio di un frame con un indirizzo mittente contraffatto sovrascrive la voce corretta nella tabella MAC, reindirizzando il traffico verso l'aggressore. Di conseguenza, l'attaccante può dirottare il traffico trasmesso dalla vittima verso se stesso, sebbene sia più teorico, poiché la vittima invia anch'essa messaggi. Questo può essere realizzato generando un pacchetto con un indirizzo falso o intercettando un pacchetto e modificandolo, poiché a livello 2 non esiste un controllo di integrità o un tag HMAC.

15. Vulnerabilità associate all'integrità

15.1. Mac flooding

qui parliamo dell'integrità della tabella MAC

L'attacco MAC Flooding consiste nell'inondare uno switch con numerose trame contenenti diversi indirizzi MAC. L'obiettivo è saturare la memoria dello switch, costringendo l'invio di tutti i pacchetti in broadcast su tutte le porte. Le conseguenze di questo attacco variano, ma l'intento dell'attaccante è rimuovere gli indirizzi MAC legittimi dalla tabella, causando il flooding di numerosi frame su tutte le porte.

L'esito dell'attacco include l'esclusione degli indirizzi MAC legittimi dalla tabella, costringendo l'invio broadcast su tutte le porte. Un utente malintenzionato può sfruttare questa situazione con un analizzatore di pacchetti per acquisire dati sensibili trasmessi tra altri computer, normalmente non accessibili quando lo switch opera normalmente.

Successivamente, l'attaccante può anche eseguire uno spoofing ARP per mantenere l'accesso a dati privilegiati dopo che gli switch si sono ripresi dall'attacco iniziale di MAC flooding.

15.2. ARP e DHCP poisoning

- **ARP Poisoning:** L'attacco può avvenire poiché è uno *stateless protocol* e molti OSs accetteranno replies anche se non richieste (unsolicited). ARP ha il compito di mappare un indirizzo IP su un indirizzo fisico (indirizzo MAC) su una rete locale. Un attaccante può utilizzare il poisoning di ARP per mappare il proprio indirizzo MAC all'indirizzo IP di un dispositivo di destinazione. Ciò consente a un host di acquisire il traffico destinato a un altro host semplicemente inviando un messaggio ARP al mittente con l'indirizzo IP del destinatario previsto e l'indirizzo MAC dell'attaccante In questo modo, quando il dispositivo di destinazione invia i dati, l'attaccante li intercetta perché i dati vengono effettivamente inviati al proprio indirizzo MAC. Il poisoning di ARP è anche conosciuto come spoofing di ARP.

DHCP Mi dà l'IP, ARP lo associa al mio MAC

- **DHCP Poisoning**: DHCP viene utilizzato per assegnare dinamicamente indirizzi IP ai dispositivi su una rete. Un attaccante può utilizzare il poisoning di DHCP per causare che un server DHCP assegni un falso indirizzo IP a un dispositivo di destinazione, reindirizzando il traffico di rete del destinatario al dispositivo dell'attaccante. L'attaccante può quindi intercettare o manipolare il traffico di rete del destinatario. Questo tipo di attacco è anche conosciuto come spoofing di DHCP.

La conseguenza di questi attacchi è la realizzazione di un MITM. Un attacco **MITM (Man-in-the-Middle)** attraverso il poisoning di ARP è un tipo di attacco in cui un attaccante intercetta e manipola il traffico di rete tra due dispositivi su una rete locale. Questo attacco si basa sul fatto che il protocollo ARP funziona come una tabella di mappatura tra indirizzi IP e indirizzi MAC su una rete locale.

15.3. Session Hijacking

Ethernet è un protocollo stateless, ma molti protocolli superiori creano sessioni. Una volta stabilita una sessione, si presume spesso che sia affidabile senza ulteriori verifiche del traffico. Un utente malintenzionato che acquisisce informazioni su una sessione (indirizzi IP, porte TCP, numeri di sequenza e dati dell'applicazione) può ricrearla e agire come un endpoint.

Se un endpoint della sessione non può essere deviato, potrebbe partecipare alla comunicazione, interrompendo le sessioni. L'ARP gratuito può indirizzare il traffico dell'endpoint locale a un indirizzo MAC falso, instradando il traffico in entrata dal router del gateway all'host dell'aggressore. Un attacco DoS può silenziare un endpoint, mentre, con tempistiche adeguate, una sessione può essere manipolata con messaggi di applicazione corretti o facendo affidamento su TCP per scartare pacchetti apparentemente duplicati in base ai numeri di sequenza.

15.4. Replay

Un messaggio intercettato in precedenza può essere ritrasmesso. Poiché il messaggio non è stato modificato, può essere autenticato o crittografato dall'originale mittente senza influire sull'attacco: l'attaccante deve solo indovinare il contenuto del messaggio per valutare se vale la pena ritrasmetterlo. All'interno del dominio Ethernet, i messaggi utili da ritrasmettere sarebbero piccoli, senza stato, e controlli che si adattano a un singolo frame. Tipici messaggi per un attacco di ritrasmissione potrebbero essere notifiche di routing o messaggi SNMP "set" o "trap".

16. [non è integrità] Denial of Service

L’obiettivo è quello di negare il servizio.

- A livello 1 è basato sul tagliare i collegamenti fisicamente o danneggiare i circuiti con l'elettricità (ovvio).
- A livello 2: Abbiamo due possibilità:
  - **Resource Exhaustion Attacks**:
    - mirare ai piani di controllo e gestione di uno switch inviando frame che richiedono elaborazione e gestione aggiuntive (registro, configurazione VLAN)
    - Inondazione unicast sconosciuta → BROADCAST: come il MAC flooding, ma l'intenzione è quella di congestionare la rete e il successo dipende dalla capacità di causare traffico sufficiente.
  - **Protocol based DOS**: Lo Spanning Tree Protocol (STP) che crea un albero da una rete mesh è progettato per essere *auto configurante*. Un utente malintenzionato che controlla un nodo sulla rete può inviare messaggi STP e fingere di essere uno switch. L'intera rete di commutazione può essere arrestata inondandola di messaggi di controllo STP.

17. Contromisure adottate

Per risolvere i problemi, è stato deciso di contrassegnare tutte le trame Ethernet come non sicure. Per attuare questa decisione, è necessario inserirle in un ambiente protetto, come un firewall. In alternativa, è possibile adottare soluzioni crittografiche. In ogni caso, esistono quattro categorie principali di contromisure: sicurezza basata sui router, controllo degli accessi, utilizzo di protocolli sicuri e monitoraggio della sicurezza.

17.1. Router Based Security

La sostituzione di uno switch Ethernet centrale con un router IP influisce sulla sicurezza. Un router IP suddivide il **resto della rete Ethernet in diversi segmenti**:

- Ogni nuovo segmento è un broadcast domain separato: o ARP, STP, VLAN e attacchi basati sulla tabella degli indirizzi MAC non sono più possibili tra i segmenti.
- All'interno dei segmenti gli stessi attacchi rimangono fattibili, a meno che ogni switch non venga sostituito con un router multi-porta.
- Il traffico tra i segmenti diventa quindi impossibile da intercettare da altri segmenti o da reindirizzare per un attacco MITM. Le intestazioni MAC di Ethernet sono «droppate» sul router e il traffico è guidato dagli indirizzi IP e dalla tabella IP del router.

Il router **blocca i protocolli del piano di controllo di Ethernet (ARP e STP) e DHCP**:

- In questo modo gli attacchi MiTM sono limitati a broadcast domain piu piccoli. Se non soni supportati dal router ARP e STP vengono bloccati.
- Un host può spostarsi nella rete Ethernet e mantenere i propri indirizzi IP e MAC, le tabelle degli indirizzi MAC negli switch vengono aggiornate automaticamente.

17.2. Access Control

Un utente malintenzionato necessita dell'accesso prima di poter eseguire qualsiasi attacco. Le entità non attendibili possono essere tenute fuori limitando l'accesso alla rete o richiedendo l'autenticazione.

Strategie

- **Protezione fisica della rete**: Per impedire l'accesso non autorizzato, le apparecchiature di rete possono essere posizionate in armadi e rack chiusi a chiave, con il cablaggio installato all'interno delle pareti.
- **Segmentazione e VLAN**: Ridurre le dimensioni di un segmento Ethernet limita l'area vulnerabile agli attacchi. La segmentazione può essere realizzata esternamente a Ethernet mediante dispositivi di livello superiore come router o firewall. All'interno di Ethernet, la tecnologia VLAN offre un modo per limitare le trasmissioni e altri tipi di traffico a segmenti specifici.
- **VLAN individuali**: La creazione di VLAN separate contribuisce a isolare il traffico e a migliorare la sicurezza della rete.
- **Controllo degli accessi basato sull'autenticazione e 802.1X port authentication**: Le porte di uno switch vengono aperte solo se il client si autentica correttamente presso un'entità esterna, come un

server di autenticazione.

- **Liste di controllo degli accessi (ACL)**: Sebbene le ACL non facciano parte della specifica Ethernet, gli switch avanzati consentono di filtrare i protocolli di livello superiore. Questo può includere la limitazione basata sull'indirizzo MAC o sull'Ethertype.

- **Sicurezza LAN gestita centralmente**: L'adozione di un'entità centralizzata per garantire l'accesso in rete è un approccio di ricerca. Le software define network (SDN) sono un esempio di questa architettura, utilizzando un controller SDN per gestire il flusso di traffico di rete in modo separato dal livello fisico della rete.

PELLEGRINI

Least Privilege e Need to Know Principles. Come incrementano la sicurezza?

Con il principio *Least Privileges* l'idea è di assegnare agli utenti solo i privilegi necessari per svolgere le loro mansioni specifiche. Un controesempio è accedere ad un DB come root, in quanto si ha accesso a tutto, mentre, secondo tale principio, l'utente dovrebbe interfacciarsi solo con risorse e informazioni inerenti allo scopo perseguito.

Il principio *Need to know* si applica principalmente alle informazioni riservate e si basa sull'idea di limitare l'accesso solo a coloro che hanno un bisogno diretto per quelle informazioni specifiche per svolgere le loro responsabilità lavorative.

Spectre e Retpoline

(Meltdown funziona solo su Intel)

Spectre è una vulnerabilità che forza i programmi in esecuzione ad accedere a locazioni di memoria che in teoria dovrebbero essere inaccessibili al programma. A differenza della vulnerabilità Meltdown, Spectre non si basa su una caratteristica tecnica di un singolo processore o di una famiglia di processori ma è un'idea generale applicata a un ambito molto ampio e che potenzialmente può essere utilizzato in diversi modi.

La versione 1 di Spectre in particolare prevede prima di tutto di inizializzare le pagine dell'array_2 (altrimenti per le ottimizzazioni dei SO sarebbero tutte sovrapposte in un'unica pagina)

Nella versione V1, l'idea è eseguire più volte lo stesso codice:

```
1 if (x < array1_size){
2 y = array2[array1[x] * 4096];
3 }
```

ripetutamente, con x < array1_size, fa confronto e si convince del branch taken

quando x eccede array1_size

`array1_size` non deve essere in cache, in modo che, ripetutamente, la BPU dica sempre *branch taken*. A noi interessa il byte `array1[x]`, ma se questo eccede la size (speculativamente), stiamo saltando da qualche parte nell'address space. L'attaccante deve utilizzare il proprio codice.

con array2 di size 256 (pari ai caratteri ASCII) riusciamo a ricostruire un char.

Nella variante V2, il codice per l'attacco non è nel processo dell'attaccante, bensì cerchiamo gadget nell'address space della vittima (ad esempio librerie usate).

librerie condivise

18. Soluzioni

In generale, gli attacchi basati su timing sono complicati da patchare perché si usano i meccanismi interni della cache. Identifichiamo:

- **Detection Way**: In cache vengono fatte molte operazioni (e quindi dei pattern noti). E' quindi possibile, tramite degli *Hardware Performance Counter* "vedere" cosa sta succedendo nella cache, e capire se il pattern di utilizzo della CPU. Non sicurissimo.
  (questa è per Meltdown)
- **Kernel Isolation**: Tramite CR3 si switcha tra kernel level e user level, puntando a tabelle diverse. Si flippa un singolo bit perchè le due page table sono contigue. Così si cambia la "visione", e se siamo users non possiamo accedere a indirizzo kernel per via di questo bit che non mapperà pagine kernel.

- **Retpoline**: Si evita di fare esecuzione speculativa di qualunque cosa che possa fare leaking dei dati.

- ```
  jmp *%r11       //jmp retpoline_r11_trampoline;
  call *%r11      //call retpoline_r11_trampoline;

  retpoline_r11_trampoline:
   call set_up_target

  capture_spec:
   pause                    speculativamente non fa niente!
   jmp capture_spec

  set_up_target:
   mov %r11, (%rsp)
   ret
  ```

Quando facciamo `call set_up_target` salviamo sullo stack l'address dell'istruzione successiva, ovvero `capture_spec`. Seguendo il percorso di `set_up_target` modifichiamo il target sullo stack, mettendoci `r11`, e facciamo `ret`. Se, speculativamente, dovessimo entrare in `capture_spec`, stiamo "controllando" la speculazione, e questo la porterà in loop finchè non sarà flushata (ovvero quando il corretto flusso di esecuzione la porterà ad eseguire l'altro branch.)

Meltdown

Ogni processo ha il proprio address space suddiviso in zone di memoria dedicate all'esecuzione user mode e zone di memoria dedicate all'esecuzione kernel mode. L'attacco ha come scopo ultimo quello di *accedere a un'area di memoria situata nel kernel anche se non si hanno i permessi*, magari per andare a leggere delle informazioni sensibili di un utente differente del sistema. Andando nel dettaglio, l'attacco viene eseguito nella maniera spiegata qui di seguito. Anzitutto si definisce un `array A` nella memoria *user space* e si *svuota la cache tramite l'istruzione cflush*. Dopodiché, si preleva un particolare byte `x` (e.g: è un indirizzo, diciamo di averlo) situato nel kernel e si memorizza il byte in un registro; naturalmente questa è un'istruzione offending. Si accede alla entry con spiazzamento `x` rispetto all'indirizzo base dell'array A (e.g. caricandone il contenuto in un registro) in modo da farla salire in
  ```

cache:
Tale aggiornamento della cache rappresenta proprio il side effect lasciato sullo stato micro-architetturale da parte dell'istruzione successiva a quella offending che, chiaramente, viene eseguita solo speculativamente. A tal punto, poiché l'array A si trova in user space, è possibile accedere a tutte le sue entry e, per ogni entry, cronometrarne il tempo necessario per l'accesso: la entry relativa al tempo di accesso minore sarà chiaramente quella di spiazzamento `x`, perché si tratta dell'unica entry il cui contenuto era stato memorizzato in cache. Ed ecco qui: ora è noto lo spiazzamento `x`, *che era proprio il byte di memoria prelevato inizialmente dal kernel space*. Ricapitolando, il **valore** di `x` (noi siamo partiti **dall'indirizzo** di `x`) è stato ottenuto in maniera indiretta misurando i tempi di accesso alle informazioni presenti nell'array A. Di conseguenza, abbiamo a che fare con un side-channel (o covert-channel) attack, ovvero con un attacco che fa uso di un canale laterale per andare a rubare delle informazioni. Io parto da un `byte B` presente nella zona kernel, lo salvo in un registro (non potrei), accedo a `array[B]` che va in cache. Successivamente accedo a tutte le entry di array (in un loop continuo, indipendente da B). L'accesso più veloce sarà quello ad `array[B]`, e quindi tra tutti gli accessi effettuati, riesco a capire cosa c'è in `array[B]`, perchè più veloce. Quindi prendendo un byte nella zona kernel, riesco da user a leggere qualcosa a livello kernel. Meltdown può essere esteso anche al caso in cui si vuole scoprire un'intera stringa all'interno del kernel space, che può essere una password, una chiave segreta e così via.

L'attacco sfrutta un bug nella CPU, dove la traduzione indirizzo virtuale-fisico richiede tempo, e il controllo dell'accesso al kernel viene effettuato a runtime anziché a tempo di progettazione. In CPU come ARM e alcune AMD, questo controllo avviene immediatamente, rendendoli non vulnerabili a questo tipo di attacco, a differenza delle CPU Intel.

La cache nella CPU, sebbene non sia direttamente leggibile da processi non autorizzati, è suscettibile a timing attack. Sfruttando il probe array, l'attaccante può misurare i tempi di accesso e determinare se una specifica locazione di memoria è presente nella cache. In particolare, l'attacco di tipo Meltdown, associato alle peculiarità del set di istruzioni X86, consente un accesso completo a tutta la memoria mappata dal sistema.

DRAM

La RAM è costituita da celle, ciascuna delle quali può memorizzare un bit e può essere selezionata tramite un indirizzo.

La DRAM (Dynamic Random-Access Memory) è un tipo di RAM che immagazzina ogni bit in un condensatore separato. I bit di controllo indicano se si sta leggendo o scrivendo in una cella. Per la scrittura, la CPU utilizza un bus di dati bidirezionale per scrivere il valore nella cella, mentre per la lettura viene effettuato il "sensing" del contenuto della cella, trasferendo poi il valore sul bus.

Internamente, la DRAM utilizza condensatori per mantenere i dati. Tuttavia, i condensatori tendono a disperdere carica nel tempo, quindi è necessario eseguire periodicamente un refresh, cioè leggere e riscrivere i dati per evitare la perdita di informazioni.

Durante la lettura, il transistor associato alla cella viene chiuso (cortocircuito), scaricando così il condensatore. Questa operazione è distruttiva, poiché leggere un valore distrugge il contenuto della cella. Per memorizzare più di un bit, si utilizza un array di celle, dove ciascuna cella è composta da transistor e condensatori. Le linee di indirizzo vengono attivate dall'indirizzo per selezionare le celle che si desidera accedere.

19. Operazioni sulla DRAM

Le celle di memoria vengono selezionate tramite linee di indirizzo. Per un'operazione di scrittura, viene applicato un voltaggio alla bitline e si cortocircuita il transistor (segnalando la bitline). Durante una lettura, si seleziona la linea di indirizzo, si misura il voltaggio del condensatore e si confronta con una soglia per determinare se il bit è uno 0 o un 1. La lettura è distruttiva, poiché scarica il condensatore, richiedendo quindi operazioni di refresh periodiche per riscrivere i dati letti.

Il problema principale è la rapida scarica dei condensatori, specialmente all'inizio, richiedendo frequenti operazioni di refresh. Queste vengono eseguite ogni 64 ms, creando un divario significativo tra le prestazioni della CPU e della memoria. Questa inefficienza energetica e l'indisponibilità della memoria durante il refresh rendono le DRAM inefficienti.

Per affrontare questo problema, si introducono le gerarchie di cache, che sono più veloci e non soffrono degli stessi tempi di refresh. L'approccio dei "refresh distribuiti" suddivide il periodo di 64 ms in diversi momenti, eseguendo il refresh in zone diverse della DRAM, riducendo la probabilità di richiedere una parte di memoria indisponibile.

L’idea è che non sappiamo quando un refresh viene eseguito, quindi non sappiamo se accedere una cella di memoria porterà a un hit o meno. Dato che all’aumentare della capacità di memoria il refresh introduce un overhead sempre più significativo, per questo aggiungiamo sempre più livelli di cache con livelli inferiori sempre più grandi, che significa che la CPU costa di più ma questo è l’unico modo per avere performances decenti nei nostri sistemi.

20. Tipi di attacchi

20.1. Row Hammer Attack

Siamo in grado di scrivere dei bit random in memoria. Il problema è la densità dei chip: i condensatori sono molto molto vicini, organizzati in matrice e che vengono caricati / scaricati ed ogni volta che si legge una riga vanno ricaricati i condensatori. Ogni volta che si legge quindi, la corrente che scorre c’è fluttuazione del voltaggio, che genera un campo indotto che si espande fra le diverse linee. Quindi, questo genera effetti nei capacitori vicini, si possono quindi

- aumentare lo scaricamento del capacitore
- ricaricare

Quindi, se si scarica prima, la deadline è più stretta ed il memory controller legge uno 0 piuttosto che un 1. Se si ricarica invece, si riesce ad andare oltre la threshold e si può ricaricare il capacitore. Quindi fare letture continue permette di flippare bit, nessuno se ne accorgerà e la cosa da fare è trovare delle righe adiacenti in memori fisica.

21. Memory Performance Attacks e Raw buffer

L'obiettivo è implementare denial of service memory al livello della RAM. Ogni volta che si legge qualche riga della memoria si scarica qualche condensatore, quindi bisogna trovare un modo per non

perdere i dati, per questo c'è un **row buffer**, che è essenzialmente un registro, quindi non fatto con i condensatori. Il **row buffer** inizialmente è vuoto, poi quando viene richiesto l'accesso ad una colonna di una certa riga la riga viene portata nel row buffer, tutti gli accessi a colonne diverse della stessa riga risulteranno in un hit. Ad un certo punto si chiede l'accesso ad una nuova riga, si ha un miss nel raw buffer, quindi si esegue il "write back" in memoria. La memoria serve cache multiple. Abbiamo due diversi attori che ad un certo punto iniziano a sperimentare conflitti. C'è una latenza ridotta ogni volta che si ha un hit nel raw buffer, ma si vuole anche ottimizzare il modo in cui si accedono i vari banchi di memoria, perché non si vuole mettere molto sforzo nell'arbitraggio per accedere ai diversi banchi. Dato che le richieste possono essere ritarddate introduciamo una coda associata a richieste diverse che vengono da thread diversi. Se si mandano continuamente richieste da uno stesso thread si continua a servire lui e gli altri threads non riescono ad accedere la memoria. Non ci sono fix per questo problema al momento.

Meccanismi adottati dai malware per raggirare i controlli statici basati sulle signature e perché le behavioral signature sono migliori

Gli antivirus in generale funzionano facendo scan della memoria per andare a cercare pattern nel codice dei processi, andare a vedere se l'hash del malware corrisponde con l'hash di qualche malware noto.

22. Tecniche di evasione

Per bypassare questi controlli il malware ha 3 possibili modi:

- **utilizzare un packer**: in questo caso il malware è formato da una parte di codice che serve per fare l'unpacker. In particolare, lo stub si occupa di allocare un'area di memoria in cui andare ad inserire il codice dopo aver fatto l'unpacker, di ricostruire la IAT (import address table) e di saltare poi all'entry point. In più nei malware moderni potrebbe succedere di avere encryption. L'analista non può analizzare il malware direttamente ma deve partire analizzando lo stub anche per trovare eventuali chiavi se c'è bisogno. Dopo aver fatto l'unpacker e trovato l'entry point può cominciare l'analisi vera e propria. Per accorgersi del fatto che il malware sia compresso si può notare:
    - una differenza tra le stringhe in memoria e quelle del processo;
    - differenza grande tra memoria occupata su disco e memoria virtuale;
    - un long jump (di solito lo stub è una porzione di codice piccola);
    - Sezioni rwx non tipico del codice comune (solitamente si trovano o rw o rx);
    - Mancanza di IAT
    - Nomi delle sezioni non standard.
- **ineffective code**: si va ad inserire del codice che non fa nulla nella logica del malware per fare in modo che non vengano riconosciuti i pattern nel codice.

- **spaghetti code**: si vanno ad inserire diversi jump nel codice per andare a modificare il flusso di esecuzione senza però modificare la logica eseguita dal malware.

23. Behavioural signature

La behavioural signature è meglio di quella classica perché va a vedere effettivamente come funziona il malware, ossia il *comportamento* andando a studiare la sequenza di API utilizzate oppure la frequenza degli op-code in modo da capire appunto cosa fa. In questo modo anche aggiungendo operazioni che non fanno nulla o facendo molti jump il comportamento rimane lo stesso e diventa più semplice identificare il malware con questo approccio.

24. Tipologie di virus

In generale il concetto di virus che modifica se stesso è interessante.

- **Polimorphic virus**: cerca di nascondersi con packing e cifratura, ogni volta che infetta una nuova macchina utilizza un "packing" differente. Si possono avere diversi cifratori e decifratori e chiavi, combinabili tra loro in molteplici modi.

- **Metamorphic virus**: il Malware stesso ha della logica che permette di generare una versione totalmente differente di alcuni algoritmi nella prossima generazione: algoritmi totalmente diversi, ma con lo stesso comportamento: la semantica resta la stessa. Solo una parte è il payload del virus, il resto è ciò che serve per modificarsi e replicarsi.

25. Firme comportamentali

Dobbiamo focalizzarci su cosa fa il mw nel payload, quindi farne un'analisi comportamentale; essa ha due nature:

1. Ispezionare la sequenza di systemcall
2. Ispezione a livello di istruzioni

Se osserviamo la sequenza di istruzioni guardandole una per volta non capiamo molto, dobbiamo correlarle per trovare pattern comportamentali.
Detection di codice malevolo tramite istogramma:

- Tracciare l'occorrenza di istruzioni
- Tracciare l'occorrenza di coppie di istruzioni
- Usare l'istogramma per matchare le signatures

Anche se il codice è mescolato implementa una certa logica. Questo approccio si usa in ambienti simulati: eseguo il mw in una sandbox affinché si spacchetti da solo.

26. Heuristic engines:

Si basano su modelli di machine learning. L'applicazione viene eseguita in una sandbox e l'AV osserva come il MW interagisce con l'ambiente esterno.

1. **Sequenze di chiamate API** : l'interazione di un malware con il sistema operativo può essere un'indicazione delle azioni che l'applicazione sta compiendo. Tipicamente basato sulla classificazione: milioni di malware e benignware utilizzati come training set.
2. **Sequenze di opcode**: gruppi di istruzioni macchina possono essere un'indicazione delle operazioni che l'applicazione sta eseguendo. Le applicazioni vengono analizzate per determinare la frequenza degli opcode nei binari. La frequenza dei termini ponderata (weighted term frequency) è utilizzata per trovare le sequenze più rilevanti di opcodes
3. **Call function graph**: cerca di estrarre blocchi di base dall'applicazione, se troviamo connessioni tra questi blocchi possiamo capire la semantica del codice. I blocchi di base delle applicazioni sono quelle che, qualora invocati, indicano che è stato invocato qualsiasi altro blocco. Due versioni della stessa applicazioni avranno lo stesso CFG, anche se pensiamo all'ineffective code perchè i basic blocks saranno gli stessi. Quando costruiamo un CFG consideriamo solo 4 tipi di istruzioni, quelle che cambiano il flusso di esecuzione:
   non-conditional jumps (jmp), conditional jumps (jcc), function calls (call) e function returns (ret).
   Ci focalizziamo solo su queste perché vogliamo essere veloci. Se pensiamo allo spaghetti code due blocchi in cui da uno si salta all'altro diventano un solo blocco. Vogliamo ridurre la complessità del codice; rimuoviamo i nodi relativi a salti non condizionali (jmp) e tutti i loro predecessori vengono collegati al suo successore. Stiamo facendo una trasformazione che potrebbe non riflettere totalmente l'esecuzione originale. Il grafo risultante funge da signature. Visitando il grafo lo si confronta con la signature che si ha, questo è un processo molto rapido. Possiamo avere falsi positivi e falsi negativi, ma comunque risulta essere una tecnica molto efficace.
4. **n-grams**: sequenze continue di parole che si possono trovare in un testo. Si confrontano diverse sottosequenze del testo e si cercano quelle che occorrono molteplici volte. Si osservano le parole che si ripetono: si cerca di rendere più prolisso il testo originario. Si usa una finestra scorrevole parola per parola e si raggruppano così n parole per volta. Quando applichiamo questa tecnica ad un file binario la applichiamo ai bytes. Applichiamo poi dei modelli di machine learning per classificare il codice malevolo: si estraggono gli n-grams dal codice e li si inseriscono nel classificatore. Per migliorare le performances si usano tecniche di boosting. Ne risultano dei falsi negativi, ma pochi falsi positivi. Le DNN si comportano bene in questo contesto; dato che lavorano bene con immagini una tecnica usata è quella di convertire il binario in un'immagine (byteplot). Tipicamente l'entropia del mw è elevata, soprattutto se impacchettati, ciò è dovuto ai bytes randomici relativi alle tecniche di cifratura.
   Esempio:
   "*The cow jumps over the moon*", con n = 2:
   "the cow" "cow jumps" "jumps over" "over the" "the moon".

Come funzionano i comandi su, sudo, perché potrebbero rompere il principio dei privilegi minimi e quali potrebbero essere le contromisure da adottare

L'utente fa il login tramite una password, il sistema operativo mantiene le password in due pseudofile: `/etc/passwd` (accessibile dall'utente) e `/etc/shadow` (accessibile solo da root).

Se nel primo file si potesse effettivamente accedere alla password di ogni utente, avremmo un problema nella sicurezza; per questa ragione le password sono mantenute nell'altro pseudofile in maniera cifrata. L'utente che esegue un programma è identificato tramite *UID*. Quando si lancia un processo vengono mantenuti 3 UID:

- `real UID` : UID dell'utente che lancia l'applicazione.
- `effective UID` : dice che cosa si può fare, ad un certo punto si può diventare un altro utente. In generale è "0", ovvero si possono eseguire delle capabilities di root. Ad un certo punto si chiede al SO di rimpiazzare il *real UID* con l'*effective UID*, seppure si lancia l'applicazione con privilegi amministrativi non si possono eseguire azioni come root finché l'applicazione non lo richiede internamente.
- `saved ID` : è il vecchio ID, che viene salvato quando si cambiano privilegi, serve per poter tornare indietro.

27. Come funziona lo switch?

su: eseguo come root
sudo: eseguo come altro utente (tra cui root)

Si basa su due systemcall: *setuid()* e *seteuid()*.

- `setuid()` è non reversibile, si possono cambiare tutti e tre gli UID.
- `seteuid()` è reversibile, e sta per "effective".

Si può cambiare lo UID anche ad un intero che non corrisponde ad alcun utente, dal punto di vista del SO un utente è solo un intero. *Si possono usare queste systemcall solo se si esegue come root*. Per essere root bisogna lasciare l'applicazione con "sudo" e "su". C'è un bit nei metadati associati ad un file che dice se si può o meno cambiare il "real UID". Le capabilities associate ad un file sono "*rwsm*", dove c'è uno sticky bit che indica che l'applicazione può essere lanciata solo dal proprietario del SO. Il SO interagisce con il filesystem e lancia un'applicazione per conto di un utente; ogni utente può lanciare l'applicazione sudo con privilegi di root, il processo viene forkato così da poter essere lanciato con privilegi da amministratore.

27.1. Esempio montaggio di un modulo

Per Unix tutto è un file, quindi se rimpiazzi le systemcall intercettandole con un modulo si può fare tutto per cambiare i privilegi utente: questo è possibile solo se si monta il modulo come super user. Quindi tutta la sicurezza è associata ai privilegi di super user, è questo il punto in cui un sistema sicuro si trasforma in un sistema non sicuro. Se si è amministratori del sistema si può fare tutto.

28. Rompere la sicurezza

L'account root rompe il principio dei privilegi minimi. nel file " `/etc/sudoers` " si può specificare cosa un utente può fare nel sistema, si possono dare privilegi a grana fine. Un "God administrator" può causare problemi perchè se viene compromesso allora tutta la macchina viene compromessa. Anche lo sticky flag può essere problematico: se lo si concede a molte applicazioni; un'applicazione con un bug potrebbe lanciare una shell e tutti possono fare tutto eseguendo come root. I binary con sticky bit sono vettori per attacchi.

Bisognerebbe eseguire come amministratore solo quando necessario, questo ci porta al concetto dei privilegi a grana fine, che controllano l'abilità di un soggetto di eseguire un'azione su un oggetto.

29. Politiche di sicurezza

Principali classi di politiche di sicurezza:

1. **Discretionary Access Control DAC** → la definizione di attributi di sicurezza avviene da parte degli utenti regolari delsistema. Gli utenti con privilegi possono cambiare le politiche.
2. **Mandatory Access Control MAC** → c'è una distinzione tra utenti e "policy makers". Le politiche sono sotto il controllo degli amministratori di sistema, diverso dagli utenti del sistema. **Chi usa il sistema è diverso da chi lo amministra.** Gli utenti non posso cambiare le politiche. le politiche MAC sono particolarmente importanti in sistemi critici dal punto di vista della sicurezza.

Quando si configura una politica di sicurezza bisogna tener conto della distinzione tra:

1. *External security* → nessun utente esterno al sistema deve poter accedere-
2. *Internal security* → nessuna persona deve poter attaccare il sistema dall'interno, neppure gli utenti temporanei (e.g. un utente attaccato tramite cavo ethernet alla rete di un azienda). Non va MAI trascurata.

29.0.1. Politiche a grana fine

I controlli di accesso possono essere di tipo **DAC** (Discretionary Access Control) o **MAC** (Mandatory Access Control), definendo la granularità dei permessi concessi a un soggetto su un oggetto. La granularità può variare a seconda della complessità delle politiche di sicurezza. Alcune basi su cui possono essere basate includono:

- **Grandezza del sistema**: Ad esempio, la gestione completa se si è un amministratore di sistema (root), è la politica più semplice.
- **Directory**: Protezione di file o dispositivi specifici.
- **File**
- **Azioni su un file**: Utilizzo dei classici permessi rwx per specificare le azioni che un soggetto può compiere su un file.
- **Porzioni di file**: Protezione di specifiche sezioni di un file.

Il costo di non configurare attentamente le politiche di sicurezza può essere minore rispetto a quello che si affronterà in caso di attacco. Pertanto, la corretta configurazione delle politiche di sicurezza è essenziale per prevenire potenziali minacce.

SERIE DI ARGOMENTI MAI CHIESTI TULUX

VLAN

Una rete Ethernet ha il problema della comunicazione broadcast (a causa del fatto che le richieste ARP, DHCP inondano la LAN, soprattutto in reti LAN con molti nodi) e questo è vero sia con gli hub che con gli switch: il dominio broadcast è condiviso fra tutte le stazioni collegate. Se la topologia è composta da vari bridge, l'intera rete vede la richiesta broadcast. Ci piacerebbe partizionare la rete Ethernet in segmenti indipendenti. Quando si gestisce una rete LAN di grandi dimensioni utilizzando physical IP subnets, ci sono alcuni problemi che possono rendere difficile la gestione e la scalabilità della rete:

1. **Esaurimento degli indirizzi IP**: La suddivisione della rete in physical IP subnets richiede l'assegnazione di una sotto-rete di indirizzi IP per ogni sotto-rete. Questo può causare l'esaurimento degli indirizzi IP disponibili e rendere difficile l'aggiunta di nuovi dispositivi alla rete.
2. **Configurazione complessa**: La configurazione di molteplici physical IP subnets può essere complessa e difficile da gestire, soprattutto per reti di grandi dimensioni con molte sotto-reti.
3. I problemi di avere un **router** è che dobbiamo avere diversi switch per ognuna delle stanze, ma ciò che vorremmo è un approccio più flessibile che non richiede di avere uno switch per ogni segmento Ethernet Per questo motivo, è importante considerare l'utilizzo di tecnologie alternative, come le **VLAN**, per gestire in modo più efficiente una rete LAN di grandi dimensioni.

Le **VLAN (Virtual Local Area Networks)** sono una tecnologia di rete che permette di suddividere una rete fisica LAN in più reti virtuali logicamente separate, note come VLAN. Ciò significa che i dispositivi sulla stessa VLAN possono comunicare tra loro come se fossero sulla stessa rete fisica, anche se in realtà potrebbero essere fisicamente disposti in diverse parti della rete.

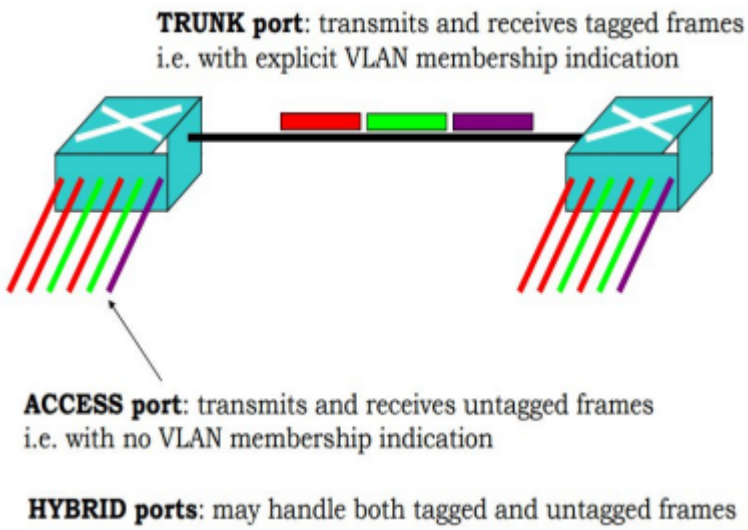
30. Componenti

Le porte TRUNK, ACCESS e HYBRID sono concetti utilizzati nelle reti di computer.

- Una porta **TRUNK** viene utilizzata per collegare uno switch a un altro switch o ad un router. Lo scopo di una porta TRUNK è consentire la trasmissione di più VLAN su un singolo collegamento, consentendo un uso efficiente della larghezza di banda della rete. Supporta tagged Ethernet frames.

- Una porta di **ACCESSO** è una porta utilizzata per collegare un dispositivo come un computer o una stampante a uno switch. Una porta ACCESS è in genere configurata per essere un membro di una singola VLAN e tutto il traffico ricevuto su una porta ACCESS è contrassegnato con l'ID VLAN associato a quella porta. Supporta untagged Ethernet frames

- Una porta **HYBRID** è una porta che può funzionare come porta TRUNK o ACCESS, a seconda della configurazione. Una porta IBRIDA è utile in situazioni in cui si desidera la flessibilità necessaria per modificare il ruolo della porta secondo necessità. Supporta sia tagged che untagged Ethernet frames.



31. Sicurezza VLAN - attacchi

- **MAC Attack:** E' basato su MAC flooding: è un tipo di attacco informatico che sfrutta le vulnerabilità nel protocollo di rete ARP (Address Resolution Protocol) per generare un elevato numero di richieste ARP false e sovraccaricare la tabella di inoltro dei dispositivi di rete.
  Mitigazione: L'attacco MAC flooding può essere mitigato utilizzando le funzionalità di port-security. Ciò consente di specificare gli indirizzi MAC per ogni porta o per apprendere un certo numero di indirizzi MAC per porta.

- **VLAN Hopping attack:** Questo attacco si basa su Dynamic trunk protocol (DTP). DTP viene utilizzato per *negoziare il trunking* su un collegamento tra due dispositivi e per negoziare il tipo di incapsulamento del trunking da utilizzare. Nell'attacco, una stazione può eseguire lo spoofing come fosse uno switch con segnalazione 802.1Q (utilizzando un frame DTP rogue). La stazione è quindi membro di tutte le VLAN.

- **Double encapsulation VLAN hopping attack:** L'attaccante inserisce un pacchetto che contiene due tag VLAN, che identificano due VLAN diverse. Il pacchetto con due tag VLAN viene inviato a uno switch che accetta pacchetti con più tag VLAN e lo inoltra a un altro switch, che tratta il pacchetto come se fosse parte della seconda VLAN.
  Soluzione: Per evitare l'attacco, bisognerebbe usare VLAN ID per tutte le porte trunk.

- **ARP attack:** L 'attaccante può utilizzare un attacco di spoofing per inviare pacchetti ARP falsificati attraverso la rete e ottenere informazioni sulle macchine nella rete VLAN.

- **Spanning tree attack:** STP viene utilizzato per mantenere topologie prive di loop in un'infrastruttura di livello 2 ridondante. L'attaccante si fa eleggere come nuovo *root bridge*, modificando la topologia della rete. Oppure potrebbe fornire informazioni false, o mandarne troppe!

Attacchi DOS e rilevamento

Gli attacchi DoS impediscono l'utilizzo autorizzato di una risorsa esaurendola. Le categorie includono:

1. **Banda di Rete:** Satura la capacità dei collegamenti.
2. **Risorse di Sistema:** Invia pacchetti di rete per esaurire il software di gestione della rete.
3. **Risorse Applicative:** Tentativo di esaurire la memoria di una specifica applicazione.

Gli attaccanti spesso usano lo spoofing dell'indirizzo sorgente per evitare il tracciamento. Gli ISP dovrebbero filtrare gli indirizzi IP "spoofed" per mitigare questo problema. L'attacco **SYN spoofing** è comune: l'attaccante invia SYN con un indirizzo sorgente falso, il server risponde, allocando memoria. Quando la tabella TCP è piena, le richieste legittime vengono scartate.

32. Attacchi HTTP

quando parlo di REFLECTION uso lo stesso protocollo sia per la richiesta sia per la risposta. quando parlo di AMPLIFICATION, partendo da pochi byte, ingrandisco la dimensione della risposta.

Reflection Attack: Questi attacchi sfruttano "middlebox" per riflettere l'attacco, consentendo di eseguire grandi DoS senza richiedere molta capacità propria. La reflection richiede "source address spoofing".

Amplification Attack: Questo tipo di attacco non richiede source address spoofing e utilizza fattori di amplificazione. Ad esempio, nell'attacco DNSsec, una richiesta di pochi byte genera una risposta notevolmente più grande. La migliore mitigazione è bloccare i pacchetti in entrata "spoofed".

Esistono due tipi principali di attacchi HTTP:

1. **Flooding di Richieste HTTP:** Questo tipo di attacco è un "amplification attack", poiché pochi byte di richiesta generano molte byte di risposta per caricare la pagina web.

2. **"Slowloris":** Questa tecnica monopolizza il server HTTP inviando richieste che non si completano mai. Inviano pochi byte in ogni richiesta, consumando gradualmente la banda. Le firme vengono utilizzate per mitigare questo tipo di attacco.

33. Attacchi a DNS

DNS si basa su UDP, un protocollo non basato sul concetto di connessione, consentendo sia attacchi di reflection che amplification.

- **Amplification:** Una richiesta di record è di circa 10 B, ma inviando una richiesta con tipo "ANY", la risposta contiene tutti i record nel NS, risultando molto più grande. Questo tipo di attacco inonda il target con risposte.

DNS

Gli attacchi memcached rientrano in questa categoria e sono attacchi di reflection ed amplification.

34. DDOS

Il termine DDoS, acronimo di "Distributed Denial of Service", implica l'uso di diverse macchine compromesse, controllate da un attaccante tramite controller, per sovraccaricare un sistema target. La mitigazione di questo attacco è difficile a causa della distribuzione geografica delle macchine infette. Rilevare un DDoS può essere complicato, ma la forma e il contenuto dei pacchetti possono fornire indizi.

34.1. Prevenzione e mitigazione

1. **Blocco degli indirizzi IP "spoofed".**
2. **Utilizzo di tecniche specifiche:** Ad esempio, l'uso di "SYN cookies" per prevenire gli attacchi SYN spoofing.
3. **Utilizzo di server replicati.**
4. **Utilizzo di servizi cloud:** Servizi come Cloudflare sono esperti nel rilevare attacchi DDoS e verificare l'autenticità delle connessioni.

L'**attacco detection e filtering** durante l'attacco può essere applicato a livello di singolo IP o di un'intera sottorete.

35. BOTNETS

Le botnets sono gruppi di dispositivi connessi a Internet che eseguono software malevolo, spesso utilizzati per compiere attacchi DDoS, spam, phishing e furto di dati. I proprietari delle botnets le controllano attraverso il software di "Command and Control". Mentre i bot tradizionali seguono uno schema client-server, i nuovi botnet utilizzano reti P2P, che non richiedono un controllo centralizzato.

- **Modello client-server:** Si basa su strutture come IRC (Internet Relay Chat), dove i bot si iscrivono a canali e ricevono comandi da un controller centrale.
- **Modello P2P:** Consente a tutti i bot di agire sia da client che da server, garantendo una maggiore resilienza in caso di guasti.

I bot più comuni utilizzano meccanismi di discovery per unirsi a una botnet, contattando indirizzi IP casuali fino a trovare un bot che risponda con i comandi corretti.

Componenti principali:

1. **Bot master:** L'iniziatore che controlla la botnet attraverso diverse modalità come IRC, SSH, Telnet, pagine web o social media.
2. **Macchine zombie:** Dispositivi compromessi senza la consapevolezza del loro utilizzo malevolo, sfruttati per risorse, noto come "scrumping".

35.1. Detection di botnets

- **Dal punto di vista del sistema:** Utilizzo di software antivirus per rilevare il malware.
- **Dal punto di vista della rete:** Monitoraggio delle attività inusuali tramite sistemi di rilevamento delle intrusioni.

35.2. Fast Flux Botnets

Le botnet Fast Flux si distinguono in "Single Fast Flux" e "Double Fast Flux" e sono utilizzate per rendere difficili da individuare i server di comando e controllo (C&C) del malware.

- **Single Fast Flux:** Un host infetto contatta agenti Fast Flux tramite la risoluzione di un nome di dominio fisso. Gli agenti Fast Flux fungono da proxy e contattano il server C&C, nascondendo così gli indirizzi IP del server.

- **Double Fast Flux:** In questo caso, cambia anche il nameserver autoritativo. Il client infetto manda una richiesta al TLD, che risponderà con l'IP del nameserver autoritativo, facendo da proxy per l'inoltro della query.

Entrambe le varianti risolvono il problema dei nameserver autoritativi fissi, ma il domain name per i domini Fast Flux rimane ancora fisso. Questo porta all'uso di algoritmi per la generazione dinamica di nomi di dominio, rendendo più difficile il rilevamento.

35.3. Sistemi di Rilevamento delle Intrusioni (IDS)

I sistemi di rilevamento delle intrusioni sono funzioni HW o SW che analizzano le informazioni da diversi utenti per identificare intrusioni. Ci sono tre tipi principali:

1. **Host-based IDS:** Analizza l'attività su un singolo host.
2. **Network-based IDS:** Utilizza sensori nella rete per analizzare il traffico.
3. **IDS ibridi:** Combinano caratteristiche degli IDS host-based e network-based.

Gli IDS vanno più a fondo nei pacchetti e utilizzano tecniche avanzate. Due approcci comuni sono:

1. **Anomaly Detection:** Basato su statistiche raccolte dal comportamento di utenti legittimi o approcci basati sulla conoscenza, utilizzando sistemi esperti o tecniche di machine learning.

2. **Signature/Heuristic Detection:** Utilizza firme o pattern per identificare comportamenti sospetti. Le euristiche sono regole definite per individuare tali comportamenti, applicabili anche al traffico cifrato.

MPLS (+ iBGP)

MPLS, acronimo di Multi-Protocol Label Switching, rappresenta un meccanismo di forwarding che si discosta dal tradizionale inoltro IP. Questa tecnologia consente di instradare flussi di **traffico multiprotocollo** tra un nodo di origine (ingress node) e uno di destinazione (egress node). La concezione fondamentale di MPLS consiste nell'associare un breve identificatore, noto come "*label*", a ciascun pacchetto, posizionandolo tra i layer 2 e 3 del modello OSI. Tale approccio consente l'applicazione di un "fast forwarding" basato sul label switching. *Da sottolineare che MPLS è indipendente sia dalla sottorete di trasporto utilizzata sia dai protocolli di rete adottati*. Prevede due componenti indipendenti:

- **Control Component**: Gestisce l'allocazione delle label e della loro associazione tra nodi *adiacenti*, chiedendo la label al suo nodo adiacente.
- **Forwarding Component**: Gestisce il forwarding basato sullo "*swap*" delle label. Ovvero attuo la sostituzione. cioè ad un certo punto al posto della label metto l'informazione reale.

36. LDP

L'obiettivo principale di LDP (**Label Distribution Protocol**) in MPLS è automatizzare e rendere scalabile la distribuzione delle etichette utilizzate per instradare il traffico nelle reti MPLS. Questo processo avviene attraverso le seguenti fasi principali:

1. **Discovery (Scoperta):** I router LDP cercano i loro vicini di rete scambiando messaggi di "Hello" per annunciare la loro presenza e condividere informazioni sulle capacità e le etichette che possono distribuire.

2. **Label Distribution (Distribuzione delle Etichette):** Una volta scoperti i vicini di rete, i router iniziano a distribuire le etichette. Questo processo implica lo scambio di messaggi per creare una tabella di label che associa un'etichetta numerica univoca ad ogni prefisso di rete conosciuto dal router.

3. **Label Retention (Mantenimento delle Etichette):** Dopo la distribuzione delle etichette, i router continuano a scambiare messaggi di controllo per garantire l'aggiornamento costante e la coerenza delle informazioni sulle etichette. Questo monitoraggio assicura che le informazioni siano sempre aggiornate.

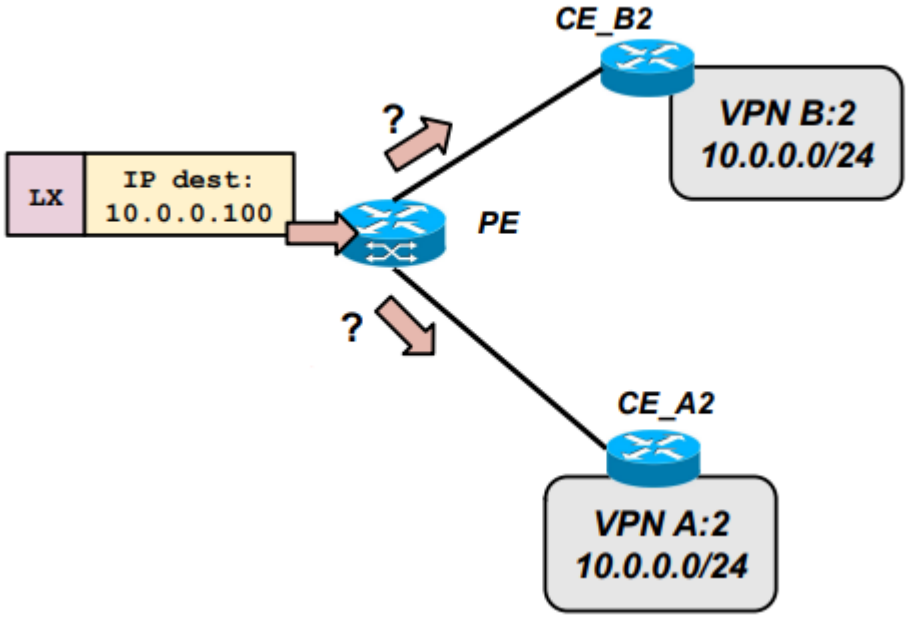
Durante la distribuzione delle etichette, i router utilizzano vari tipi di messaggi, come le `Label Request` per richiedere nuove etichette e i `Label Mapping` per associare un'etichetta a un prefisso di rete specifico. Una volta distribuite, le etichette vengono utilizzate dai router MPLS per instradare il traffico, determinando la prossima etichetta MPLS da assegnare ai pacchetti in base alla destinazione e alle informazioni ricevute dai vicini di rete.

La **combinazione di BGP e MPLS** semplifica la gestione delle rotte esterne, riducendo la complessità delle tabelle di routing sui router interni e migliorando l'efficienza complessiva del forwarding del traffico nella rete.

Attraverso BGP/MPLS, è possibile creare una rete di tunnel basata su LSP (Label Switched Paths) tra i router di bordo. In questo modo, il traffico in transito passa attraverso questa "mesh" di LSP senza la necessità di replicare tutte le rotte esterne su ogni router interno. Ciò consente ai router interni di instradare il traffico in base alle etichette MPLS, senza la necessità di vedere gli indirizzi IP sottostanti.

37. Vulnerabilità nelle VPN

L’indirizzamento nelle VPN non è coordinato e posso avere lo stesso indirizzo IP su VPN_A e VPN_B. PE ha lo stesso IP per entrambe. Se ho più clienti potrei avere una sovrapposizione degli indirizzi VPN (privati).



37.1. Soluzione: Doppio Incapsulamento

Il PE associa il pacchetto in entrata alla VPN del cliente semplicemente abbinando l'interfaccia di ingresso. La tabella di forwarding MPLS cambia in base alla VPN specifica a cui appartengono i clienti. Il PE deve supportare tante tabelle di forwarding quante sono le VPN dei clienti ad esso connesse. Tali tabelle sono chiamate **VRF (VPN Routing and Forwarding)****, $1\forall$ VPN collegata al Provider Edge.

Viene memorizzata anche una **Global Forwarding Table GFT**, che aggiunge una label esterna (indipendente dalla presenza della VPN) per collegare due Provider Edge.

SERIE DI ARGOMENTI MAI CHIESTI PELLE

Predizione dei salti

La *speculazione* nella pipeline della CPU è fondamentale per migliorare le prestazioni, ma può comportare costi elevati in caso di predizioni errate. Se la predizione è *corretta*, non paghiamo penali, altrimenti il costo è il flush della pipeline. Altro fattore è associato a *quando* verifichiamo la predizione.

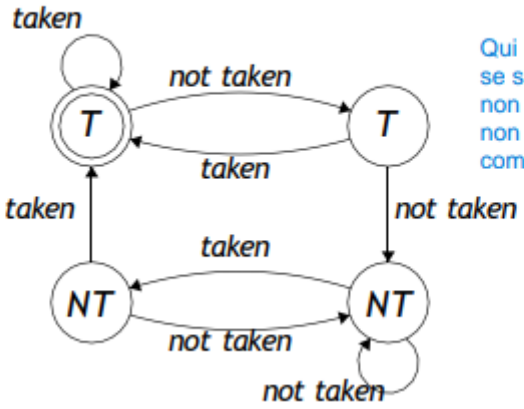
Le predizioni del salto possono essere *dinamiche* (basate sulla storia passata, a livello hardware) o *statiche* (a livello di compilazione). La "**Branch Prediction Unit**" utilizza una tabella di predizione del salto, una piccola memoria associata agli indirizzi delle istruzioni di salto condizionale.

Nella pipeline, durante la fase di *fetch* della *prossima* istruzione, il processore decide se eseguire o saltare l'istruzione corrente. Se la predizione è errata, si verifica un flush della pipeline con una penalità di un ciclo ed *impariamo* qualcosa sul salto. Se è corretta, non ci sono penalità.

38. Two bit saturating counter - cicli scorrelati

La "**Two-bit Saturating Counter**" è una automa a stati finiti con quattro stati aventi due valori, "preso" e "non preso", che decide in base alle predizioni degli altri salti. Le transizioni avvengono solo se la predizione è errata, creando una correlazione tra le istruzioni. Funziona bene coi cicli annidati, sicuramente meglio della versione con solo due stati, in quanto i cicli annidati sono scorrelati. E per cose *correlate?*

Dipende dagli eventi che occorrono, a differenza di usare 1 byte (2 stati), qui ne uso 2 (4 stati). Questo perchè qui c'è il concetto di "rinforzo della predizione". Non è detto che, se sbaglio una predizione, allora con la stessa predizione non posso aver ragione dopo.
(Se io prevedo X, esce Y. Dopo potrebbe uscire veramente X, e quindi in questo secondo caso avrei ragione, e non dovrei cambiare.



Qui ho T "rinforzato" (il primo T), se sbaglio predizione vado in T " non rinforzato", e a meno che non sbaglio nuovamente farò comunque T.

39. Correlated Predictor - correlazione tra salti

Cerchiamo di individuare correlazioni tra salti, ad esempio quando una variabile è presente nella condizione di due salti o quando una variabile viene modificata all'interno del corpo di un'istruzione `if` dopo essere stata utilizzata come condizione per un altro `if`. Utilizziamo una storia dei passati m salti che rappresentano un percorso all'interno del programma.

La **Pattern History Table** è una serie di Finite State Machines (FSM) progettate per prevedere l'esito di un salto basandosi sulla storia recente dei salti. I bit cercano di identificare pattern nella storia dei salti, rappresentando tracce di diverse parti del programma con comportamenti distinti. Quando viene intrapreso un nuovo salto, shiftiamo a sinistra il registro scartando i bit più vecchi.

Tuttavia, questo meccanismo non è privo di errori a causa della presenza di istruzioni non correlate. Per affrontare ciò, utilizziamo **Tournament Predictors**, che consistono in due predittori: locale e correlato. Il predittore locale si basa sull'indirizzo di una singola istruzione di salto, mentre il predittore correlato si basa sugli ultimi m salti, accedendo alla storia locale. Un ulteriore contatore a due bit (Choice Predictor) indica quale dei due funziona meglio per ciascuna istruzione.

40. Branch target buffer - salti indiretti

I salti indiretti rappresentano un tipo di salto in cui la destinazione non è nota in modo diretto, come nel caso delle istruzioni `return` o dei salti ai registri come `jmp *rax` o `ret`. La difficoltà nella loro predizione deriva dal fatto che possono avere molteplici destinazioni possibili.

Per affrontare questa sfida, le CPU utilizzano un componente chiamato **Branch Target Buffer (BTB)**, che funge da piccola cache accessibile nella fase di fetch delle istruzioni. Il BTB tiene traccia di alcuni target predetti per i salti indiretti. Inoltre, per distinguere tra i diversi target prefetched, vengono utilizzati bit di previsione.

La particolarità è che, in alcune situazioni, la predizione può essere semplificata. Ad esempio, nel caso delle istruzioni "return", dove si ritorna dalla chiamata di una funzione, già si conosce il chiamante. In questo contesto, non è necessario fare affidamento su statistiche o complesse analisi di pattern, poiché il chiamante è noto e può essere utilizzato per prevedere il target del salto. Questo scenario specifico rappresenta una semplificazione nella gestione dei salti indiretti. In effetti, circa l'85% dei salti indiretti sono associati a istruzioni di "return", rendendo la predizione più agevole in questa specifica categoria. Ciò si fa mettendo l'indirizzo del chiamante sullo *stack* (**Return address stack**), da usare insieme a BTB.

Side Channels

Un **side channel** è un metodo che consente di accedere a informazioni sensibili o rivelare pattern di accesso alla memoria. Alcuni tipi di attacchi side channel includono:

- **Prime + Probe**
- **Flush + Reload**
- **Flush + Flush**
- **Evict + Time**
- **Evict + Reload**
- **Prime + Abort**

Nel contesto di un attacco alla gerarchia di cache come side channel, l'obiettivo è portare la cache in uno stato noto e osservare gli effetti generati da altri processi.

Il processo di attacco si suddivide in due fasi: la fase di preparazione (pre-attack) e l'effettiva esecuzione dell'attacco.

41. Fase di Preparazione (Pre-attack):

1. Acquisire il target: l'area di memoria desiderata.
2. Comprendere il funzionamento di hit e miss nella cache per l'applicazione specifica.
3. Avere una conoscenza dettagliata dell'architettura del sistema.

Nella fase di preparazione, è essenziale misurare la cache per determinare quanto tempo impiega un hit o un miss. Si cerca di portare il canale in uno stato noto, si attende che la vittima acceda al sistema, si osserva l'accesso della vittima e si ottiene informazioni da ciò.

Dato che più entità possono accedere alla cache, è necessario tenere conto di questo fattore. Inoltre, è importante considerare la possibilità di essere deschedulati durante l'attacco.

42. Attacco Prime+ Probe

L'attacco **Prime+Probe** è stato inizialmente progettato per operare sulla cache L1, ma è stato successivamente adattato anche per la cache L3. Questo consente di ottenere dati da altre Virtual Machines (VMs). L'attacco rileva quando la vittima accede a un valore specifico. "Prime" indica che l'attaccante riempie la cache con un valore noto. Dopo che la vittima ha eseguito l'accesso alla linea di cache, se si verifica un miss, indica che la vittima ha acceduto a quella linea di cache, causando un

Riempio varie LINE DI CACHE con valore noto. Una volta fatta girare la vittima, questa deve invalidare delle linee di cache e carica i suoi valori, a questo punto possiamo rigirare e fare il probing del cache set caricato prima tramite una operazione di probing: se il tempo di accesso aumenta, sappiamo che la vittima ha toccato quelle linee di cache.

conflitto e l'eliminazione della linea dalla cache. Viene utilizzato un "array di probe" per evincere il valore. Tuttavia, in memorie set-associative, è necessario evincere l'intero set.

43. Attacco Flush+Reload

L'attacco **Flush+Reload** sfrutta la condivisione di memoria e si concentra su una libreria condivisa come target. Se l'attaccante può controllare la memoria, effettua un "flush" di una linea di memoria utilizzata dalla vittima. Successivamente, rieseguendo la vittima, l'attaccante monitora il tempo di "reload". Un reload veloce indica che la vittima ha acceduto alla linea di memoria. L'obiettivo è ottenere non solo l'identificazione della variabile ma anche il suo valore, sfruttando il side channel di Flush+Reload.

44. Attacco Flush + Flush

L'attacco **Flush+Flush** è una variante di Flush+Reload. Se una linea è in uno stato valido, il flush richiede più tempo poiché il memory controller deve determinare il proprietario. Se la linea viene ricaricata, significa che la vittima l'ha acceduta. Identificare la malizia di un flush è difficile, e se si fa il flush di una linea già flushata, non si crea alcun "rumore" sulla cache.

In scenari reali, l'attuazione di questi attacchi richiede tempo e una conoscenza dettagliata dell'architettura del sistema. La dinamicità della mappatura memoria-cache può rendere complesso il successo di tali attacchi.

45. Attacco Evict+Time

Si sfrutta il fatto che ricaricare una linea di cache dopo la rimozione richiede tempo. L'attaccante fa eseguire la vittima, osserva il caricamento dei dati nella cache per calcolare un tempo di "baseline". Successivamente, l'attaccante elimina la linea di cache di interesse e misura il tempo che la vittima impiega per accedere nuovamente alla linea di cache. Se la linea eliminata è quella di interesse, il tempo sarà notevolmente superiore rispetto alla baseline, rivelando così informazioni sensibili attraverso il side channel.

46. Attacco Evict + Reload

Nell'attacco Evict + Reload, l'obiettivo è causare l'eliminazione (eviction) di una linea di cache e osservare il tempo necessario per ricaricarla. Se una linea viene ricaricata rapidamente, significa che la vittima ha acceduto a quella linea di cache, consentendo all'attaccante di inferire informazioni sensibili. Questo approccio differisce da Flush+Reload perché sfrutta il tempo di ricarica della cache dopo un'eliminazione e può essere più difficile da identificare, in quanto non genera "rumore" sulla cache se si effettua l'eliminazione di una linea già svuotata.

47. Attacco Prime+Abort

In questo attacco, si inizia una "transazione" mettendo un dato nella memoria cache e si aspetta che qualcuno acceda a quel dato. Quando ciò accade, la transazione viene interrotta. Questo tipo di attacco non richiede di misurare il tempo, poiché le transazioni hardware funzionano come avvisi automatici. Per prevenire questo attacco, è possibile disabilitare le transazioni hardware. L'attacco può essere eseguito sia

a un livello di cache più vicino al processore (L1) che a uno più lontano (L3), con alcune differenze nel modo in cui si spia la cache. Questo metodo sfrutta la dinamica delle transazioni hardware e della cache per ottenere informazioni sensibili in modo meno evidente.

Database

E' l'insieme di meccanismi e migliori pratiche che proteggono i dati memorizzati nel database da minacce intenzionali o accidentali.

- **Security policy**: descrive una misura di sicurezza applicata Bisogna utilizzare i meccanismi di sicurezza del DBMS sottostante per rafforzare le policy, ma anche usare policies di sistema e gestione esterne al DBMS per rafforzare la sicurezza dei dati.

Bisogna comprendere le questioni di sicurezza in:

- un ambiente di sistema di database generale e un'implementazione specifica del DBMS.
- considerando le problematiche di sicurezza del database nel contesto dei principi generali di sicurezza.
- considerare le problematiche relative sia all'archiviazione del database che alla comunicazione del sistema di database con altre applicazioni, la comunicazione è importante in quanto sottintende il coinvolgimento di terze parti.

SQL injection: attacco che deriva dal fatto che l'utente interagisce con un DB tramite stringhe. L'input utente viene inserito all'interno di queries SQL, ciò è totalmente insicuro. Anche se una stringa è corretta dal punto di vista dell'utente può non esserlo dalla prospettiva del db (piccolo Bobby Table docet). **Soluzione a questo problema**: anziché concatenare stringhe usare i "**prepared statements**", ovvero delle queries incomplete che utilizzano dei placeholders al posto dei parametri attuali. `e.g. INSERT INTO score (event_id,student_id,score) VALUES(?,?,?)` Non bisogna permettere all'utente di cambiare la semantica di una query. Il DBMS analizza, verifica e compila la query incompleta; poi quando i parametri effettivi sono disponibili, possono essere associati alla query pre-compilata. Quando il DBMS riceve i parametri effettivi, viene eseguita la query completata effettiva e vengono prodotti i risultati. L'altra soluzione per la SQL injection sono le "stored procedures" rendono il codice non manutenibile, perchè dobbiamo avere a che fare con tantissime procedure, che rendono il codice meno leggibile; per questo motivo sono meglio i "**prepared statements**".

48. Autenticazione e autorizzazione nei DB:

Il principio dei privilegi minimi va applicato anche ai DBMS. Se tutti eseguono sul DB con privilegi di root si può fare di tutto, ciò causa problemi dal punto di vista della sicurezza, perché se l'applicazione che si connette al DB è buggata può fare qualsiasi cosa. Bisogna dare i privilegi in base a ciò che un utente ha necessità di fare.

- **Autenticazione**: Un meccanismo che determina se un utente è chi dice di essere. Un amministratore di sistema è responsabile di concedere agli utenti l'accesso al sistema creando account utente individuali. La maggior parte dei moderni sistemi DBMS permette solo agli utenti autenticati di stabilire una connessione con il DBMS.

- **Autorizzazione**: L'assegnazione di un privilegio che consente a un utente autenticato di avere un accesso legittimo a un sistema. Sono al pari degli access control. Il processo di autorizzazione coinvolge l'autenticazione dell'utente che richiede l'accesso agli oggetti. Questa è una caratteristica importante dei moderni sistemi DBMS per implementare il PoLP.

49. SQL access control

Lo standard SQL permette di specificare chi può accedere una porzione del DB ed in che modo può farlo. L'oggetto dei privilegi sono le tabelle. Di default:

1. Chi crea una risorsa ha privilegi completi su di essa.
2. L'amministratore ha privilegi completi su tutto.

Il modo appropriato per progettare un database è creare diversi utenti, ognuno con dei permessi diversi su porzioni diverse del DB.

49.1. Viste

Esse sono delle particolari relazioni nel database richieste da uno specifico utente, non sono realmente esistenti in un DB. Sono anche un modo per garantire sicurezza, in quanto permettono di rendere visibile a ciascun utente solo ciò che deve poter vedere. Una query su una vista è identica ad una query su una qualsiasi tabella.

RAID

La ridondanza è necessaria per soddisfare la "dependability".

Cosa fare se una macchina si rompe?
La sicurezza ha a che fare con il fatto di essere "operativi". Un approccio alla ridondanza è il RAID, che può essere implementato a livello sw o hw. Si tratta di raggruppare diversi dischi poco costosi in un unico array: la probabilità di fallimento non si somma, in quanto sono indipendenti tra loro. Dobbiamo coordinare il comportamento di molteplici componenti poco efficienti. Le tecniche che si usano sono associate al "data striping". Sui dischi si possono fare due operazioni di base: read e write. Dobbiamo gestire la concorrenza. In cosa consiste il data striping? Distribuire su dischi diversi blocchi di file. Solitamente la stripe size coincide con la taglia del blocco. Ciò aumenta la disponibilità ed il throughput della concorrenza del sistema.
Classifichiamo:

- **Raid 0:** nessuna ridondanza. Il singolo file è diviso tra più dischi.
- **Raid 1:** Ogni disco ha una copia di alcuni file. (eg: file A1 presente su più dischi). Bene in parallelo, ma devo aggiornare tutte le copie.
- **Raid 2:** A livello di bit, con codici a correzione di errore. Hamming per ricostruire l'informazione
- **Raid 3:** Striping a livello di byte, usa la parità.
- **Raid 4:** Striping a livello di blocchi.

  I diversi blocchi sono scritti sui diversi dischi e ce n'è uno addizionale che mantiene informazioni sulla parità

- **Raid 5:** Usato dai data center. Sempre coi blocchi. il blocco di parità viene distribuito tra i vari dischi, quindi no bottleneck.
- **Raid 6**: C’è un’altra informazione di parità suddivisa tra i dischi. Tolleranza: fallimento di 2 dischi. Ci vuole un gran numero di dischi. 2 blocchi di parità per avere tolleranza a due guasti
- **Raid 10:** Unisce RAID-0 e RAID-1, ovvero i dischi del RAID-0 sono array di tipo RAID-1.

Horse Pill Attack a boot-time

Questo è un tipo di attacco che avviene durante il caricamento del sistema operativo e mira a modificare il contenuto del RAM disk. Il RAM disk, creato localmente ogni volta che il sistema è avviato, non ha firme digitali poiché è assemblato dinamicamente.

L'obiettivo principale è compromettere la sicurezza dello userspace manipolando il RAM disk. Utilizzando tecnologia basata su container, l'attacco può creare un ambiente isolato, in cui applicazioni ed utenti credono di utilizzare l'intera macchina. Il RAM disk infetto può caricare moduli di sicurezza, montare il filesystem, osservare i processi in esecuzione e avviare un container prima che lo userspace sia completamente avviato.

All'interno del container, il RAM disk può simulare l'ambiente di un sistema operativo legittimo, nascondendo ciò che accade all'esterno del container. Può rimontare il filesystem proc, creare finti threads di livello kernel, eseguire operazioni di cleanup e iniziare un processo init. Nel frattempo, al di fuori del container, può montare qualsiasi tipo di filesystem, eseguire fork per lanciare backdoor shell e osservare le azioni dell'utente.

Dopo aver orchestrato queste operazioni, il container viene spento e poi riavviato, apparentemente senza alcun problema dal punto di vista dell'utente. Tuttavia, tutto ciò che accade all'interno del container può essere osservato dall'esterno, consentendo di intercettare e rubare file scritti dall'utente durante il processo.

50. Mitigazione

verificare che le entries dei threads abbiano ppid = 0

Ci sono una serie di cose che si possono notare all’interno del container. Nel container possiamo vedere quali sono i namespaces associati alle applicazioni che lanciamo. I thread kernel non sono generati dal kernel iniziale con pid pari a 0. In generale bisogna fare un esame e revisione estensiva del sistema anche un esame esterno realizzato staccando l’hard drive.

51. Prevenzione

Non è semplice, bisognerebbe evitare di creare localmente il RAM disk, ma non è possibile far fronte alla complessità hardware in questo modo, i RAM disk sono l’unico modo in cui possiamo configurare il nostro sistema basato su diversi driver. Ogni volta che il RAM disk viene ricreato sulla macchina l’attaccante deve re-installare il malware.

Sicurezza a livello di rete

I sistemi operativi offrono alcune capacità interessanti per la sicurezza di rete, sebbene siano poco utilizzate al momento. Le Access Control List (ACL) possono essere impiegate per controllare i servizi di rete in base agli indirizzi di origine delle richieste.

Per implementare Mandatory Access Control (MAC), ci sono capabilities nei sistemi operativi POSIX basate sugli indirizzi di rete. L'uso di super server, come **inetd** (internet daemon) e **xinetd** (extended internet daemon), permette di spawnare server aggiuntivi o containers.

UNIX inetd: Inetd è un super daemon che controlla quali applicazioni dovrebbero ascoltare su una specifica porta. Ogni richiesta su una porta specifica non viene inviata immediatamente all'applicazione in ascolto, ma *inetd* decide quale applicazione invocare basandosi su configurazioni specifiche.

UNIX xinetd: Xinetd offre capabilities avanzate, essendo un'estensione, tra cui:

- Access Control basato su indirizzi IP.
- Controllo del tempo per limitare la durata di una connessione.
- Logging completo degli eventi a run-time.
- Prevenzione di attacchi di tipo Denial of Service (DoS) imponendo limitazioni sulle istanze di servizio, le dimensioni del file di log e altro.

La configurazione di xinetd è gestita dal file `/etc/xinetd.conf`. Ogni richiesta che arriva a *inetd* e *xinetd* passa attraverso controlli di accesso utente nello user space e può essere inviata al **demone TCP** (tcp daemon) per ulteriori processamenti.

È da notare che, sebbene in passato *tcp daemon* fosse ampiamente utilizzato, oggi l'approccio **Keep It Simple Stupid (KISS)** spinge verso soluzioni più semplici, evitando complessità come la redirezione dati attraverso file e privilegiando l'uso diretto dei socket. Quando una richiesta raggiunge inetd e xinetd, xinetd esegue controlli di accesso utente nello user space e quindi invoca il demone TCP (tcp daemon) per elaborare la richiesta. Il demone tcpd decide se far procedere una richiesta specifica o meno.

Tcpd esegue anche un ulteriore controllo di sicurezza per rilevare manipolazioni DNS, in particolare DNS tampering.

Il controllo rDNS (reverse DNS) consiste nell'eseguire una query inversa per ottenere il nome simbolico a partire dall'indirizzo IP. Questo controllo funziona poiché le query dirette e inverse vengono eseguite in zone diverse; la zona responsabile delle reverse queries è gestita dal proprietario stesso del record DNS associato all'indirizzo IP in questione.

Capabilities vs ACL

Le **ACL** sono una serie di passi che il SO compie per determinare chi si è e dare o meno accesso in base all'oggetto che si vuole accedere e alle azioni che si vogliono eseguire su di esso. Lavorano maggiormente sulle risorse. Questo ha delle limitazioni: se l'owner è root ci si ferma al primo controllo e si può fare qualunque cosa si vuole. Abbiamo visto che avere un god user rompe il sistema.

Le **capabilities** ono state progettate per evitare il problema del god administrator, dobbiamo definire cosa un amministratore di sistema può fare. Lavorano a grana ancora più fine, perché implementano privilegi a grana fine anche per ilroot account. Lavorano quindi sul tipo di utente. In Unix ci sono due categorie di processi:

- **Unprivileged process**: si possono gestire con le ACL
- **Privileged process:** possono bypassare qualsiasi check dei permessi, che è esattamente il problema che abbiamo osservato prima.

Le capabilities nascono per buttare tutto ciò che ha a che fare con privilegi di root. E’ un modo per ridurre i danni che possono essere fatti dagli utenti legittimi. In generale se si lavora in applicazioni o a livello kernel ci sono diversi tipi di capabilities.

Quindi come è possibile far girare una applicazione come root evitando che questo possa fare qualunque cosa: le capabilities possono essere usate per dividere i premessi di root in diversi domini, si possono avere dei processi che girano come root che possono configurare device di rete ma non cambiare i permessi sul FS.