1° LEZIONE

CyberSecurity presuppone attaccanti, protegge sia i dati che le risorse computazionali (ciò che si può fare), spazio di archiviazione contro utilizzatori indebiti. Differenza tra bug (proprietà funzionale inattesa) e vulnerabilità (sfruttamento del bug da un attaccante). Non sempre vuln deriva da bug, ffma anche da problemi design/config. La sicurezza ha ripercussioni sulla safety (riguarda la salvaguardia delle persone). Data protection (GDPR) sono misure per garantire privacy (diritti alla segretezza di informazioni sensibili. CyberSecurity aiuta Data Protection, sono due insiemi che si intersecano.

PROTOCOLLI DI SICUREZZA (FRONTIERA)

SYMMETRIC NEEDHAM-SCHRODER

Si può notare come per crittografia simmetrica si necessita di una terza parte fidata che generi le chiavi di sessione, e di conseguenza necessiti di conoscere le chiavi a lungo termine. Questo perché A potrebbe creare una chiave di sessione e inviarla a B, ma con cosa la cifra/protegge? La si vuola segreta. E non si ha a disposizione, per esempio, DH per crearla e scambiarla con un parziale segreto da ambo le parti. TTP (Trusted Third Part) conosce le chiavi a lungo termine degli utenti. Obiettivi: confidenzialità della chiave di sessione scambiata e autenticazione tra A e B con freshness (con nonce). Da considerare che le nonce non hanno un significato in sé per sé, sono numeri da usare una sola volta e danno la garanzia di freshness di una qualche proprietà a chi la crea e invia, una volta che la riceve (cifrata/modificata e cifrata), quindi a due vie a differenza di timestamp.

Se attaccante modifica Na iniziale, fa solo attacco di negazione del servizio perché Alice vede indietro una nonce diversa e fa abort. La nonce Na dà garanzia ad A che il server sta rispondendo a quel particolare messaggio (precedente), cioè che la chiave di sessione generata è fresca. Passi 4 e 5 servono per garantire freshness

dell'autenticazione di A a B (B non ha ancora parlato infatti con TTP, ma con A). Al passo 5 sarebbe meglio (Nb, Nb cifrato kab) al posto di Nb - 1 (perché il tentativo è dimostrare di modificare il contenuto del passo 4). Bob rivede la sua nonce e autentica con freshness A. C'è proprio un handshake tra A e B. Quindi infine si sa che dall'altra parte è attivo chi conosce la stessa chiave (adesso con freshness, autenticazione c'era già). B autentica A al passo 3, ma senza freshness, mentre autenticazione con freshness avviene al passo 5. A autentica B al passo (con freshness? Ma senza sua nonce? Forse A controlla che la chiave con cui decifra il 4 funziona e quindi è uguale a passi prima). Il ticket dentro la cifratura ka potrebbe essere ridondante (?) come in kerberos IV (anche se lì il ticket ha il timestamp), serve solo per proprietà di key confirmation (vedi kerberos), infatti in ogni caso poi il ticket esce fuori senza doppia cifratura ai passi successivi in entrambi i protocolli. Al passo 4 si ha cifratura di nonce rispetto passo 1 perché così A autentica B, ma la modifica in ricezione (necessaria perché nonce è per freshness legata a una qualche proprietà) sta nel -1 o come ha detto il prof.

Attacco Denning and Sacco ma con modello di attaccante che conosce una chiave di sessione vecchia (attacco ancora più significativo rispetto alla conoscenza di una chiave fresca, il problema è piu grave) (in Dolev Yao in realtà non c'è questo attacco) qua si viola proprietà di forward secrecy (se qualcosa del passato va compromesso, non deve avere conseguenze in futuro), si può usare una chiave vecchia ed essere autenticata. Fix parte dalla consapevolezza che l'handshake in realtà riguardava solo un utilizzo fresco della chiave, ma una chiave generata fresca. La nonce è solo un marcatore personale. L'idea è che dovremmo fare inserire a B la sua nonce prima, così che possa essere a conoscenza della freschezza della generazione della chiave, così come A nella vecchia versione (fix Yahalom). In definitiva andrebbe riprogrammata la progettazione del protocollo. In modo alternativo usare i timestamp, molto più facile (Kerberos: si necessita orologi sincronizzati (NTP), il progetto è a una via, basta che chi riceve il timestamp lo controlli).

2° LEZIONE

Chi inserisce il timestamp deve essere fidato, perché il problema del timestamp è di integrità già all'origine, oltre che in transito. In Kerberos quelli cruciali vengono inseriti dal TTP, non è un problema perché già ci fidiamo della sua generazione delle chiavi di sessioni. I 2 requisiti sono quindi la generazione del timestamp da TTP e la protezione di integrità in transito.

KERBEROS

Ha come obiettivi autenticazione, segretezza e freshness dell'autenticazione. Per usabilità si usa single sign on tramite Kerberos, se si ha una suite di servizi, si accede una sola volta a uno e non agli altri, ma va contro la sicurezza dell'autenticazione, deve essere sicuro. Anche nei siti web il browser manda il token per noi allo stesso server, anche se non è proprio single sign on. Vedi suite pacchetto microsoft.

Kerberos (a tre teste) perché il Server TTP sdoppiato (autenticazione e autorizzazione) per accedere ad una risorsa. L'autenticazione è funzionale per l'autorizzazione, che è lo scopo finale. Alice vuole parlare con un server (o sfilza di servizi) e di mezzo c'è TTP (ci parla una sola volta). La terza testa è il server con cui si vuole interagire. Kerberos ha 3 fasi: autenticazione, autorizzazione e servizio, le ultime 2 sono trasparenti nel single sign on. Ogni fase fornisce credenziali (coppia chiave di sessione e crittotesto, il ticket di ns simmetrico, authK è chiave di sessione per autorizzazione, servK uguale ma per servizio) per la fase successiva. Auth key dura per tutta la sessione di autenticazione (ore). Servkey ha molta meno durata perché sono chiavi di sessione per l'autorizzazione dei servizi. Anche qui entra in gioco bilancia usabilità sicurezza. Authkey può criptare più servKey perché la prima è dura più delle seconde.

As è authentication server, il TGS (ticket granting service) è il server di autorizzazione che dà i ticket per i servizi. T1 è timestamp di A,

falsificabile ma non è un problema. Nel passo 2 si cita TGS (dentro ticket per distinguerlo dall'altro ticket, a grandi linee uguali), la chiave di sessione, il timestamp per rendere fresca la chiave generata, c'è integrità data dalla chiave esterna. Il ticket dovrà essere inviato da A al TGS. Nelle versioni di Kerberos semplificate il server è uno solo, mai implementato. Adesso ci sono 2 server non per i timestamp, ma per risolvere anche il problema del single sign on, fattibile sia con nonce (Yalohom) o timestamp. Passo 2 e 4 sono analoghi, cambia il destinatario dei messaggi seguenti (prima TGS, poi B). Il passo 6 si completa con il solo timestamp. I messaggi dispari sono tutti richieste di A. Ta è il timestamp fidato prodotto da AS riferito alla chiave di autenticazione. Il timestamp al passo 1 può essere alterato perché in chiaro, ma non è un problema perché al massimo se viene alterato all'indietro si ha la stessa situazione con la nonce di prima, al più è negazione di servizio. Al passo 2 A che riceve il timestamp fidato da AS, controlla se va bene, altrimenti rigetta se troppo passato. Al passo 3 A produce un altro timestamp che però protegge, perché si deve garantire la freshness del ticket inviato, si vuole che la richiesta arrivi subito, in caso contrario si fa abort, perché potrebbe capitare che Ta è buono, ma T2 è lento, nel dubbio si potrebbe fare abort. Gli autenticatori sono fatti a parte perché concatenarli al ticket e fare cifratura esterna con authk/servk, comporterebbe che il ricevente non può decifrare in quanto riceve la rispettiva chiave di decrittazione proprio da dentro il ticket, che farebbe però parte della cifratura ancora più interna. Volendo A può usare il ticket per un altro servizio rispetto B a patto che abbia chiave authk fresca, per questo si diceva prima che authk potesse cifrare più servk (1 a molti). Ovviamente se si perde la prima chiave si rompono anche le altre, ma è causato dal single sign on (in realtà c'è 2FA). Gli autenticatori servono, oltre ad autenticare, all'interlocutore per verificare eventuali latenze di rete che potrebbero lasciar pensare manomissioni. Nel passo 6 si dà conferma ad Alice, ci potrebbe stare un altro timestamp. In fase 2 e fase 3 è il processo

utente a fare ciò per Alice, per questo sono fasi trasparenti, Alice inserisce la password solo all'inizio.

Gli attacchi ci sono ma sono limitati temporalmente in base alla validità dai timestamp. I timestamp danno anche una durata di validità a differenza delle nonce, c'è scadenza.

C'è attacco anche con timestamp scaduti. Authk è sempre protetta da ka che è ok, mentre servk è protetta da authk che non è a lungo termine (al passo 4). C'è da giocare con gli aspetti temporali dei timestamp. Ta viene prodotto prima ma ha una scadenza più lunga, Ts dopo ma ha scadenza molto prima.

3° LEZIONE

authK: tA (tempo di gen.), lA (lifetime); tA viene prima di ts; servK: ts (tempo di gen.), ls (lifetime); ts viene prima di ta+la

ts+ls potrebbe andare dopo tA+lA, se va prima non è un problema perché una chiave auth scaduta implica che le servk già generate sono sicuramente scadute, e con questa non se ne possono richiedere altre servk. L'attaccante ha a disposizione una chiave authKey scaduta, la proprietà che viene meno è forward secrecy. Auth key anche se scaduta fa rivelare servKey perché serve per decifrare dove dentro c'è servkey (al passo 4). Però se siamo nella fase dopo, c'è autorizzazione senza autenticazione, perché decifro messaggio e trovo dentro servk ancora valida, nella finestra di tempo critica in cui ts+ls è dopo ta+la e prima del tempo ts+ls. È un attacco più forte rispetto ad aver rubato una authK valida e non scaduta, ma molto più sottile da essere notato.

Il fix lo fa il TGS che può anche non emettere la chiave se ts+ls va a destra di ta+la, ta+la deve essere >= ts+ls. Attacco scoperto con fdr. Solo nell'ipotesi in cui l'attaccante conosca una authK vecchia potrebbe scoprire una servK buona se non c'è fix. Si cambiano le

ipotesi, il modello di attaccante, perché di per sé a cascata authK è confidenziale. Comunque, è attacco di replay attack perché il servTicket viene replicato e poi opportunamente costruito autenticatore2, impersonando A. Causa è assenza di forward secrecy. Realm tra macchine (non vuole fare questa parte). Slide prima di limiti tecnici no, limiti tecnici sì.

Kerberos eredita la doppia cifratura nei ticket da NS (anche lì è ridondante), in realtà si usava solo perché DES non era proprio sicuro e si mise una pezza così, ma non è necessario (l'eventuale replicazione del ticket deve essere accompagnata da un autenticatore cifrato con la chiave dentro il ticket, comunque non decifrabile da un attaccante); infatti, Kerberos V mette fuori dalla cifratura il ticket. Adesso però non si ha più la certezza che sia stata Alice a estrarre quel ticket e averlo inviato a TGS, così invece qualcuno vede il ticket cifrato però è ticket, non è bloccato dentro cifratura di A. Mentre in Kerberos IV TGS formalmente non poteva certamente ricevere il ticket da user diversi da A, ciò in Kerberos V non è dimostrabile dato che chiunque può avere il ticket se lo intercetta, non ha secondo strato di cifratura. Vi è un attacco alla proprietà di key confirmation, cioè di essere certo che il ticket arrivato (con solo 1 strato di cifratura) provenga da A.

In realtà dietro A, c'è identità, chi c'è dietro pc, e chi esegue il processo col protocollo, a noi interesserebbe solo l'ultimo. Cos'è Ka? In Kerberos IV, per esempio, proviene dalla password in qualche modo. Password è weak secret perché è corta. Serve sistema che lo trasformi in forte contro attacchi di forza bruta, con salting ed entropia.

SMARTCARD

Un altro approccio (per evitare brute force sulla password, Kerberos V) è usare un dispositivo, la smartcard, dove dentro mettiamo direttamente segreto molto più grande, la chiave direttamente.

L’efficienza del processo dentro la smartcard è importante perché dentro c’è una decodifica da fare. O la workstation decripta, o gli dà la chiave così decripta la smartcard, ma non è sicuro, si potrebbe cifrare la chiave ma con un'altra chiave, spostando però il problema nello stesso (Problema di root of trust, ad una certa qualcosa non è protetto, se mettessimo un pin che usiamo per cifrare). Quindi la smartcard deve essere computazionalmente efficiente perché deve decriptare il segreto da mandare. Concetto generale è tirare fuori il segreto.

4° LEZIONE

Ma il segreto dentro smartcard come è protetto? Ad oggi sono tutte col microchip, prima c’era banda magnetica (e anche numeri a rilievo?), tutti questi dati diversi sono tutti strumenti di lettura di dati della carta. Non è possibile clonare la carta (Che vuol dire clonare?). Man mano si hanno sempre meno dati sulle carte per velocità di lettura, usabilità. Il pin è il secondo fattore di autenticazione, ora con RFID non c’è pin per acquisti sotto i 50 euro o circa, è più comodo, ma più insicuro. Prima era solo autenticazione per possesso, con carta, adesso è per possesso fisicamente e online è un segreto per conoscenza.

Differenza tra banda magnetica e chip (smartcard)? La banda magnetica è più semplice da leggere (con una testina), non ha interfaccia funzionale, a differenza della smartcard che ha interfaccia funzionale (firmware) e risponde a un certo range di comandi, una maschera di funzionalità, mentre la banda va strisciando. Il pin stava sulla banda in alcuni casi, non c’era capacità di calcolo; quindi, il segreto doveva essere conosciuto o dal sistema (non sicuro) o appunto nella banda.

È solo il chip che può avere la capacità di tenere un qualche segreto nella carta, protegge i segreti grazie a interfaccia funzionale che emette. Se è programmata bene non dà il segreto. Quindi è possibile

anche metterci chiave per crittografia, e usare Kerberos, dato che c'è computazione possibile. Quindi clonare dipende dal protocollo, basti pensare al man in the middle tra protocolli tipo NS o Kerberos, ottieni qualcosa, ma basta per rompere autenticazione? Prima clonare sì perché era solo banda magnetica. Bisognerebbe forzare con qualche attacco l'interfaccia a farsi dare i dati (non per niente banale), o leggere proprio elettronicamente le celle di memoria.

Le prime smartcard erano solo memory card, adesso hanno microprocessore dentro. Le smartcard sono microcomputer. Contact plate è contatto tipo interruttore, guardare struttura smartcard. Javacard permette di eseguire bytecode, quindi è programmabile in java, gli passo il bytecode e JVM lo leggerà. File System. Ha standard proprietà fisiche. La chiave privata sta nell'EEPROM, l'interfaccia sta in CPU.

Tassonomia di attacchi:

Microprobing: tramite ponticelli si causano dei glitch nel flusso di esecuzione in modo da fare uscire il segreto da EEPROM. Software attacks tipo se ci sono backdoor che rilevano segreto. EavesDropping significa intercettare, Fault induction possono intervenire le contromisure di reset della carta. Microprobing è invasivo, processo visto velocemente. I microponticelli alterano il disegno per fare il dump della memoria. Molto più difficile che leggere la banda magnetica. Oppure alterando il PC per saltare magari istruzioni di controllo.

Eavesdropping: Le linee rosse corrispondono a linea di tensione, più è alta più è il lavoro attuale della carta, si potrebbe capire che operazione si sta facendo tramite pattern (canale subliminale).

Fault Induction vista velocemente, nel Probe ci sono n linee, non una sola. Se si altera la cpu in modo che il controllo sia sbagliato nel ciclo, cioè che deve uscire ma non esce, SPOF nell'if, se si riscaldasse a tot gradi il controllo va ancora bene? O potrei rimanere dentro ciclo e leggere tutta la memoria.

5° LEZIONE

CRITTOGRAFIA VISUALE

Si ha per esempio nelle buste che arrivano a casa con i pin (delivery pin). Ci sono state diverse versioni, quella tamper evident, che, se qualcuno apriva busta e leggeva pin si vedeva, o difesa da attacchi di trasparenza luce con adesivo, o il metodo di grattare che risolve entrambi.

La chiave o il crittotesto non rivelano da sole il testo in chiaro. La sovrapposizione ottica delle 2 share (immagini) rivela l'immagine originaria. La divisione iniziale (encryption) avviene tramite SW, share sono digitali. Viceversa, l'operazione di rivelare l'immagine (decryption) si fa con l'occhio umano (Cyberfisica). Inizialmente genero una chiave (che sarà insieme di bit che saranno immagine senza senso), poi crittogramma in qualche modo, tale che sovrapponendoli portino a immagine in chiaro.

Applicazioni: 1) per esempio sovrapposizione del pin di un adesivo su un altro specifico quadratino (due overlay non banali, overlay implicano trasparenza dello strato sopra, quello sotto deve essere visto in un certo modo), invece se questo adesivo è da mettere su sfondo bianco, è una semplificazione in cui "tutti hanno la chiave", lo sfondo chiaro (basta perdere il primo pezzo). 2) OTP visivi per spionaggio. 3) Voto elettronico, ricevuta per il votante che andava a sovrapporre nel display per confermare che avesse votato.

È cifratura simmetrica (con OTP, op. XOR). Trasposizione di bit 1 e bit 0 a bianco (trasparente scritto) e nero (trasparente vuoto), pixel. Ma se si facesse la tabella con le caselle sarebbe OR vi è un problema intanto funzionale, OR non invertibile, se m = 0 k = 1 m OR k = c = 1, ma c OR k = 1 OR 1 = 1, ma m era 0, di conseguenza dato m = 0 necessariamente c e k devono essere 0, specchiando dato C = 1, sicuramente m = 1 (probl. di sicurezza per probabilità). Data un'immagine si calcola

randomicamente lo share chiave (prima bit poi immagine), poi XOR tra bit e si ottiene crittotesto (da portare a immagine). La chiave la uso una sola volta e quindi la posso mandare tranquillamente, basta che siano inviati per canali diversi. Il problema è che la sovrapposizione visiva di per sé è OR.

Soluzione: l'occhio umano può essere ingannato, Schema a due livelli. L'andamento può anche essere orizzontale o verticale o diagonale quando creo matrice 2x2. Nel lucido del dubbio, le due condizioni intermedie (di bianco/nero) sono in teoria diverse, ma l'occhio li interpreta entrambi grigi allo stesso modo, non vede che proviene da operazioni diverse. Non sono 3 e 1 per ogni pixel perché le condizioni di migliore contrasto per l'occhio sono con 2 e 2. Se vogliamo dare toni di grigio all'occhio, va arricchito il dato, con matrice più grande. 3 e 6 perché per sovrapposizione ottengo 3 neri, 4 neri, 5 neri o 6 neri, il resto sarà bianco. È il miglior equilibrio di contrasto per l'occhio come scala di grigi.

NON RIPUDIO

Non è possibile negare una transazione, un'azione. Applicazione con la firma digitale, PEC (questa funziona come la posta SMTP, semplicemente i server sono fidati, e concedono PEC, c'è fiducia). C'è sempre un'autorità fidata (cifratura simmetrica -> TTP, cifratura asimmetrica -> catena di certificazioni). La differenza tra e-mail e PEC è solo il non ripudio. La PEC usa la firma digitale. Il problema è l'equità, perché non ripudio è one way. Gestore Alice, Rete, Gestore Bob, Bob. Il valore del non ripudio è dato dalla fiducia sui provider.

6° LEZIONE

Nei protocolli per non ripudio, tutti possono mal comportarsi, eventualmente a spese dell'altro, questo è il modello d'attaccante più naturale. A invia PEC a B per volere garanzia del protocollo stesso del non ripudio, cioè di avvenuta consegna, e inoltre B vuole essere certo che sia stato A. La PEC è un caso particolare di applicazioni di proprietà di non ripudio alla mail. La PEC si basa su atti di fiducia o protocolli di sicurezza sui server PEC certificati? Vedere flusso.

Non ripudio è misura di sicurezza di 2° livello. Si basa sulla firma digitale, ma la garanzia da essa data la voglio 2-way. Vedere definizione. Serve alla controparte. Non basterà cifrare con un altro strato come per segretezza. Vedere esempi di casi d'uso. Delega per esempio ritiro di pensione, delega è passaggio di permessi, non obbligo di andare a prendere la pensione. I problemi potrebbero essere disonestà, canale inaffidabile. Compravendita: primi protocolli con chiavi asimmetriche e TTP.

Idealizzazione A (protocollo giocattolo): Fnro è flag, etichetta, significa non ripudio di origine, esplicita il senso della firma. Il primo messaggio cita destinatario e messaggio (e-mail?). B segue il protocollo perché abbiamo assunto che è onesto, in teoria potrebbe già fare abort, ha le sue garanzie tramite firma ricevuta. B manda ad A una firma da parte sua su flag di non ripudio di ricezione. I due messaggi sono omologhi, cambia a chi si dà garanzia.

Idealizzazione B: agenti disonesti (potrebbero rompere schema di prima, nel senso che rompe proprietà di equità). TTP è unico sicuramente onesto. Il TTP ha il compito di inviare a B tutto firmato da lui con il non ripudio di submission (Alice non può negare submission). L'altro flag è di delivery. TTP segue protocollo e quindi certamente manda messaggi 2 e 3 (atomicamente, nello stesso momento, perché altrimenti per un certo lasso di tempo uno dei due avrebbe vantaggio sull'altro).

Idealizzazione C: mezzo inaffidabile. Non serve TTP. Se 1° messaggio non arriva, il protocollo termina senza dispute (proprietà non inficiata). Se è il 2 a non andare a buon fine la proprietà inficiata perché Bob ha evidenza mentre Alice no, sarebbe attacco al protocollo. B manda il messaggio 2 fino a quando non riceve ACK. Se salta il solo ACK in realtà la proprietà c'è comunque, entrambi hanno non ripudio. Un'alternativa è quella di evitare il "bombardamento", inviamo a server (come posta) rendendo disponibile perché comunque presto o tardi si può scaricare ed è del ricevente l'onere di scaricare la mail.

Nel caso reale, posso prendere idealizzazione B e fare in modo che è onere di B e A scaricarsi la posta da TTP, e si ottiene equo non ripudio (PEC fa così), sarebbe equa compravendita con 3 messaggi a differenza di Zhou-Gollman (per il prof).

ZHOU-GOLLMAN

Zhou-Gollman (complicazione per il prof, non necessaria): il tentativo non va a buon fine, poe è promise of exchange. Vengono mandati crittotesti. B se chiude al passo 1 non ha nulla, ha prova solo un crittotesto. B ha interesse a continuare, non ha ottenuto nemmeno una funzione da A, manda flag di accettazione. A, ricevuto il messaggio 2 ha conferma, ma di un crittotesto, il protocollo termina senza dispute (nessuno dei due ci guadagnerà qualcosa se indebito). All'ultimo passo Bob può benissimo decifrare e andare via, non inviare ack. La chiave dovrebbe essere confidenziale in tunnel TLS perché, se un attaccante intercettasse la chiave e la desse prima a B, potrebbe andarsene prima dal protocollo, questo problema è in Zhou-Gollmann. (In realtà no, gli servirà anche con_k)

Zhou-Gollmann usa TTP, ma perché hanno usato anche rilascio posticipato? (non lo sanno nemmeno loro). FTP serve per scaricarsi i messaggi dal server (prima non c'erano siti web, git ecc..). sub_k submitted quando viene inviata la chiave da A a TTP, l'altro è firmato da

TTP perché la chiave viene inviata da TTP. L (lable, link) è id della sessione. Se Bob è DY intercetta il messaggio 3 e prende chiave, il messaggio non arriva a TTP, lui decifra e frega tutti, ma in realtà B non otterrà certificato da TTP. Il non ripudio non viene intaccato. Inoltre, siamo in misure di secondo livello, k potrebbe viaggiare su TLS. Se invece intercetta e arriva a TTP, poi entrambi hanno evidenza, non ci ha fatto nulla. Al passo 5 k che arriva ad Alice non serve a nulla, l'importante è con_k. Non ci interessa della segretezza. Bob può decifrare crittotesto, ma senza con_k non può dimostrare nulla.

Risoluzione dispute: Se B nega, A ha NRR e con_k, NRR da solo non fa nulla, ma con_k associato con la stessa Lable L di NRR, garantisce. Stessa cosa d'altra parte.

7° LEZIONE

Non ripudio in e-mail: obiettivo è robustezza crittografica e dimostrabilità legale. In quello debole il problema è che il sender ha la ricevuta, mentre al receiver manca la ricevuta di spedizione (legale), la mail non è non ripudiabile, al massimo è autenticabile. È alla base delle spedizioni di poste con raccomandate di ritorno (in realtà l'autenticazione non c'è alla posta, nemmeno qua in quella debole). In quello forte il ricevente ottiene anche attestazione, quindi per modificare quello di prima serve intanto autenticazione (cond. necessaria per il non ripudio). Il receiver autentica il mittente a livello applicativo, controllando il contenuto del messaggio, perché i canali sotto potrebbero essere insicuri (in quella debole). La differenza tra raccomandata e PEC è fare la stessa cosa digitalmente e nella realtà.

ABADI

Abadi: TTP stateless implica che è più difficile trovare sicurezza. TTP ha due coppie di chiavi, una per firma, una per codifica. Ogni agente deve

registrarsi su TTP tramite password dato che non c'è PKI, però ciò porta SPOF e non scalabilità, tutte le password stanno in TTP. Le funzioni sono costanti, anche le password dovrebbero esserlo, sul piano logico. La differenza tra le due è l'arietà, ack ha arietà 1, un solo parametro, la password ha arietà 0, non ha parametri di input. È un altro modo di autenticarsi diciamo, le due non hanno differenze, solo entropia (potenzialmente). La differenza è che non mando password, ma una sola query, avendo poi una response; quindi, metterei a rischio solo una query. Ma in realtà potrei condividere 20 password condivise (nullarie). Allora cambia solo il fatto che con password non aspetto challenge (che sarebbe la funzione), metto direttamente password. Ottiene equo recapito debole. Usa rilascio posticipato. Hash serve per integrità, confronto poi fra i due, prodotti da S e R. Crittotesto è sorta di caratterizzazione della sessione, q e r stanno insieme perché password la abbiamo su due livelli. Q primo e c primo sono quelli che il receiver ha dopo il passaggio via rete, r primo lo costruisce in base alla q ricevuta.

I controlli potevano anche essere fatti tra variabili singole, con l'hash funziona uguale però l'hash può crearlo solo S e R dato che q e r, q primo r primo li conoscono solo S e R, tutti possono fare controlli di integrità (soprattutto il TTP) senza fare disclosure su q e r, segreti tra S e R. È un meccanismo tipo firma digitale.

TTP adesso può controllare gli hash e capire se i due utenti sono d'accordo, e sblocca le funzioni per essi. Non c'è misura di autenticazione del mittente, c'è una chiave dentro (S to TTP), questo messaggio può riceverlo solo TTP. La ricevuta di ritorno è DR, in cui mette timbro, k è la chiave di sessione per rilascio posticipato (dov'è autenticazione?).

La query in chiaro non è un problema tanto potrei addirittura dare input fuori dal dominio, la password in sé è coppia input output. S2TTP è cifrato con chiave pubblica, sicuro per proteggere la chiave. R rispedisce messaggio a TTP tramite SSL, per proteggere password di R.

Se attaccante modifica messaggio 1, TTP se ne accorge confrontando gli hash e potrebbe fare abort in caso di modifica di c e q. Al messaggio 2 non ci possono essere dispute (chiave non è ancora stata rilasciata da TTP). Le funzioni sbloccate dal TTP si hanno dopo i controlli di integrità tra gli hash, e autenticazione di R tramite password. Non c'è enfasi su mezzo inaffidabile dato che non c'è un sistema di reinvio messaggi, o scaricamento di chiave dal server da parte degli user. DR è delivery ricevuta. Il quarto passo si potrebbe sistemare (dato che non è su SSL che sta su TCP), con i metodi di reiterazione o scaricamento da server TTP della ricevuta. TTP ricevendo i due hash li confronta e capisce se s e r sono sintonizzati su una funzione ack.

8° LEZIONE

Si vuole minimizzare impegno del TTP, infatti è stateless, ma per far ciò si è dovuto far autenticare S e R fra di loro, ma senza PKI, ma ciò porta SPOF in TTP che conosce tutte le password. r' è calcolato in locale da receiver. Pwd R serve per fare autenticare receiver al TTP. Si va al passo 3 se controlli hash e autenticazione R è ok. È equo recapito debole, il receiver ha solo l'e-mail da poter leggere, non un'evidenza da portare in tribunale. Hr è una sorta di identificazione di sessione per R, perché q e r potrebbero essere uguali, c'è una sorta di id con encryption non deterministica (una qualche randomness, perché addirittura potrei voler inviare lo stesso messaggio e quindi c sarebbe uguale), e viene accoppiato alla chiave, così R capisce a quale c associare k. Funge come la lable di Zhou-Gollman.

Raggiungiamo quello forte: serve evidenza da dare a R che il messaggio sia costruito da S. Manca, intanto, autenticazione esplicita di S verso R. Dentro S2TTP c'è S ma chiunque potrebbe metterla. Però se tutto va bene (gli hash ok, 3° potere qui degli hash) allora S dentro S2TTP è autenticato, è ok. Il mittente non può che essere quello dentro S2TTP. C'è attacco banale di freshness ma è proprietà di 1° livello, è dato per

assunto che ci sia freshness senza dirlo. È una meta-autenticazione. TTP estende il messaggio 3 mettendo S, e firmandolo. Questo considerando la meta informazione di deduzione di TTP, oppure si potrebbe anche aggiungere autenticazione di S mettendo la sua password dentro S2TTP, che può leggere solo TTP. Equa compravendita corrisponde a equo recapito forte.

CRISPO

Non-ripudio in delega: Task (delega) è Omega, il ricevente non è obbligato a farlo, ma ha l'autorizzazione e la responsabilità.

Grantor è chi dà la delega, grantee è chi la riceve. Entrambi hanno possibilità di firmare digitalmente. Ogni coppia di chiavi ha il suo scopo. No TTP, ma allora serve ipotesi che tutti sono non disonesti, ma a noi serve qualcosa di debole -> equa delega. Corrisponde a equo recapito forte, ma in realtà la sottigliezza sta nel fatto che Omega è qualcosa di pubblico, meno interessante. End-point non fa da TTP che potrebbe agevolare in caso di dispute. Solo la firma ha la certificazione esterna, le altre sono certificate internamente al protocollo. Le chiavi certificate internamente non sono un problema, basta allungare la catena PKI, io certificato esternamente firmo un qualcosa di interno al protocollo, che verrà usato dentro. La delega è l'evidenza del permesso. L'accettazione è quella di chi riceve la delega.

Al passo 1 il grantor dà chiave pubblica di delega con certificazione interna. Al passo 2 g dice che accetta omega per mezzo di pubAK e pubEK, firmando con la sua chiave privata di firma (quella certificata esternamente). Col passo 3 il grantor sigilla le 3 chiavi pubbliche cert. internamente, per mezzo di una chiave cert. anche internamente al passo 1, questo realizza il token di delega. Queste cert. interne sono una sorta di chiavi di sessione. Deve esserci compartecipazione da entrambe le parti. Il grantee che va dall'endpoint deve presentare token di delega. Le parti in chiaro sono omesse, quindi end-point ha chiavi pubbliche per controllare firme. La chiave di accettazione (?) è fuori dal

protocollo. Le chiavi firmate dentro in ogni passo sono più che altro delle ridondanze che fungono per la cerimonia di delega stessa. Togliendo tutte quelle chiavi interne, problema, messaggio 1 e 3 sarebbero uguali. Ridondanza funzionale.

Analisi di equità: al passo 1 g non ha ancora token (delega è contratto a due vie, prima va accettata per poter dire che si è ricevuta); quindi, non può eseguire accettazione (riscossione dell'oggetto di cui si è delegato il permesso). In realtà al passo 2 se G chiude, ha un qualcosa di debole, che g ha ricevuto delega, ma non può dire che l'ha esercitata, qui l'end-point può fare la differenza controllando eventualmente se l'ha esercitata o meno. La vera importanza è l'esercitazione di omega. Se g nega di aver esercitato omega, G ha bisogno oltre del messaggio 2 (dimostra solo l'accettazione della delega), ma l'endpoint può verificare se il token è stato esercitato, può anche non farlo. Ci si affida molto sulla onestà dell'endpoint (sovraccaricato), altrimenti problemi in dispute, da entrambe le parti. Negli altri protocolli era il TTP a verificare eventuali dispute.

9° LEZIONE

ASPETTI NORMATIVI

ALLEGATO B

Fino al 2016 c'era solo normativa 196 del 2003. GDPR dal 2016 è legge 679, entrato in vigore 2 anni dopo. Ad oggi ci sono molte normative dal 2018 in avanti (NIS, ora NIS2, Eur Cybersecurity act, Dora, Perimetro nazionale di sicurezza cibernetica). GDPR è basato su consapevolezza dell'utente sul dare dato.

Vedere allegato B di 3 pagine (abrogato), per questo la sicurezza è una misura organizzativa (oltre che tecnica). Ci sono misure universali (da considerare anche attualizzando e rivedendo incarnazione odierna,

esempio di password robusta), e altre obsolete (rimpiazzabili con cose diverse più facilmente e che hanno lo stesso scopo).

Per trattamento si intende qualsiasi cosa (scrittura, lettura, divulgazione, storage ecc..). L'incaricato è chi è autorizzato (ha ricevuto la nomina). Serve autenticazione per trattare dati.

Ogni passaggio del dato deve essere contrattualizzato, e serve consapevolezza dell'importanza del dato al fornitore. È anche una questione di bilancia rischio/beneficio di compliance per le aziende che trattano i dati. Non ci sono ancora dei controlli per l'utente dopo, per esempio, aver chiesto cancellazione dei dati propri, l'azienda può dire che è compliant (per legge più che per etica), ma la certezza non si può avere. Questa si chiama soft privacy (consenso utente, contrattualizzazione (nomina, chiamato incarico in GDPR) utente-servizio, e speranza che sia compliant), non c'è enforcment dell'attuazione delle regole. Hard privacy invece si basa su zero trust, infatti è attuata al contrario dall'utente (a differenza di soft in cui si fida), e si attua dando dati offuscati, pseudonimizzati, con tecniche tipo di crittografia visuale a soglie, per raggiungere oblio hardenizzato, verificabile. Sono tutte misure organizzative, ma sono necessarie e a priori dell'implementazione tecnica. Data breach è un trattamento indebito, di chi non è autorizzato, a prescindere se è malevolo o onesto.

In 2 dice servono username e password (segretezza) oppure autenticare su possesso, o eventualmente a 2 fattori. Oppure con biometria. In definitiva serve login univoco, che sia combinazione o meno di uno dei 3 tipi di autenticazione (è regola attuale). In primo c'è permesso, può (consentito), nelle altre non è specificato, mancano verbi servili. Nella 3 dovrebbe essere un può perché dipende se servono o meno un set o una sola credenziale (per esempio servono solo se sia necessario fare attività con privilegi differenti). In 4 (obbligo) dice che il malloppo di istruzioni (che deve essere dato a incaricato) dice che l'incaricato deve proteggersi la password e i dispositivi che usa per autenticarsi.

Nella 1 è tutto fattibile tecnicamente, nella 2 anche tranne utente che non deve condividere password con nessuno, potrebbe essere fatto enforcment con doppia autenticazione di cui una è possesso di un badge aziendale per esempio. La 3 potrebbe essere fattibile anche se comunque magari c'è qualcuno che lavora con privilegi superiori rispetto a quello che deve fare, enforcment con IPS (?). La 4 (attuale) è una metamisura, dice che devono esserci le regole nella policy, l'enforcment è che deve esserci la policy, è un livello logico differente, a cascata tecnicamente si rifà alle altre che alcune non sono rinforzabili.

Nella 5 (non attuale, possibilmente 8 non va bene, infatti è anteriore a NIS che nel 2000 e qualcosa ha detto car. Alfanumerici, o addirittura si permette pochi caratteri) la password ha almeno 8 caratteri, anche se ci sono dei casi in cui se non è possibile, si permette meno (problema). Ok parte successiva, tranne durata di 6 mesi (perché proprio 6?). Poi i 3 mesi significa che più l'account è per accesso a trattamento dati, più deve essere frequente cambio, hardenizzazione autenticazione (perché proprio 3?). Si può fare enforcment sia della complessità della password, che del cambio, può essere imposto. Definizione dati sensibili in wikipedia, GDPR li chiama dati particolari.

Nel 6 il login non può essere assegnato ad altri incaricati, neppure in tempi diversi (non va più bene, problema bruciamento login per esempio per persone con nomi uguali, ad oggi c'è change management per questo, ISO, misura tecnico-amministrativa).

Nella 7 la gestione tecnica è funzionale. Il motivo è per evitare account zombie, rimpiazzabile dal change management. La durata potrebbe essere diminuita. Non va bene perché sembra dire metti ordine quando c'è disordine, serve più organizzazione.

Nella 9 è metaregola sulla policy, come la 4. È facile verificarla ad alto livello (controllare se è stato impartito libretto istruzioni), ma scendendo a basso livello, c'è enforcment a differenza della 4, cioè cambio durata screenlock, tipo a 1 minuto. È attuale.

10° LEZIONE

Nella 10 quando vuol dire se. Se autenticazione è con password, sono impartite disposizioni (metaregola, policy di per sé). Portano confusione (si potrebbe essere compliant a essa già copiando e incollando la stessa nella propria policy), ma si dovrebbe lavorare a livello tecnico, oggetto. L'azienda deve garantire disponibilità ad accesso a dati e strumenti per operatività/sicurezza sistema (in base al ruolo) di un dipendente che va via per un po' (metaregola, deve impartire istruzioni al riguardo, non è spiegato invece come). In modo operativo si dovrebbe fare: ogni utente che ha password per dati, l'azienda deve poter garantire l'accesso ai dati, prima deve esserci stata una nomina a qualcuno che ha la copia delle credenziali (fiduciario) del dipendente assente. Il fiduciario, se necessario, farà login con la copia di credenziali, salva la situazione e deve tempestivamente informare l'incaricato che è indisponibile. Fasi temporali non evidenti da articolo. Importanza attuale. Aspetto di qualità un po' meglio delle altre metaregole (custodia per segretezza). La misura è ad oggi obsoleta, esistono gruppi di utenti con risorse/spazio condiviso (dove mettere quelle critiche per continuità). Con aziende piccole ci sarebbe ridondanza, ma sarebbe maggiore con misura dell'allegato, ogni utente dovrebbe avere un fiduciario. La 11 era stata abrogata per via del discorso della diffusione.

Nel complessivo è vago, ma anche GDPR lo sarà, non ci saranno tecnicismi da applicare (quello è dato da standard come NIS, o nostra esperienza).

Autorizzazione:

nel 12 quando sta per se, se ci sono mansioni differenti a livello organizzativo (non tecnico), allora devono esserci autorizzazioni (tecniche), es. ACL, YAML. In genere è sempre necessario, e verificarlo è facile (booleano). Importanza odierna, ad oggi necessariamente

tecnologico, un po' obsoleto dato che non è specificato. Non è metaregola.

Nella 13 si dice che le utenze vanno configurate prima di consentire trattamento. Ancora valido, ma scontato ormai.

Nella 14 è giusto, ma obsoleta (ad oggi sistemi migliori). Per esempio, firewall che dice c'è una regola non triggerata da tanto tempo, o profilo di utenza non usato da tanto tempo.

La 15 dice che le utenze singole possono essere organizzate in ruoli. Il può potrebbe non essere applicato ed essere compliant (dove non c'è verbo servile è un deve).

Nella 16 non va bene la cadenza semestrale, ma una volta non c'era sistema automatico di aggiornamento. L'idea è ancora attuale però. Idonei dovrebbe essere valutato quali sono, non è specificato da nessuna direttiva (non ce n'è perché sicurezza generica senza conoscenza degli aspetti funzionali specifici non ha senso). Rischio di intrusione è risk assessment, anch'essa soggettiva e con possibilità di errore.

La 17 è odierna, ma non va bene cadenza annuale, né semestrale, simile alla 16.

La 18 è metaregola, sui backup, non va bene la cadenza. Gestire backup è complicato, e sono importanti RTO (tempo massimo di recupero da backup, entro il quale la perdita di servizio/dato è accettabile) e RPO (frequenza di backup).

19. Registro dei trattamenti del GDPR ha sostituito documento programmatico sulla sicurezza, doveva essere fatto ogni anno dalle sole aziende che trattavano dati sensibili o giudiziari. 19.1 dice di elencare tutti i trattamenti sui dati. 19.2 profili autorizzativi, 19.3 risk assessment (sulla sicurezza, non privatezza), 19.4 anche dal punto di vista fisico, 19.5 come ripristinare i dati. 19.8 è pseudonimizzazione.

Ulteriori misure. 20. Non c'è standard per farli, l'esperto deve conoscere lo stato dell'arte, ciò che è idoneo. Non è dettagliato, è odierno. Si potrebbe estendere a dati personali. Cifratura a dati su memoria.

21. Metaregola, pendrive, non dà criteri di qualità. Facile da soddisfare. 22. Le pendrive con dati e non usate, vanno distrutte o rese inutilizzabili, cancellate (????), vanno sanitizzate. In generale il problema è che non ha interfaccia di accesso, è una memoria che basta inserire ed è facile da perdere, a meno che non sia cifrata.

23. si rifà al backup, metaregola. 24. Nel mondo sanitario i dati sensibili vanno trattati separatamente (pseudonimizzati) rispetto i dati personali (nome, cognome, CF). I dati genetici sono ancora più delicati. Odierno e ok. 25. importante che se ci si avvale di installazioni esterne di mezzi di sicurezza, si abbia una garanzia nei confronti dell'esterno, tutti i controlli visti da ribaltare al fornitore.

27 (metaregola) 28 (se sensibili, io autorizzato non devo darlo a nessuno) (hanno senso) e 29 (dice obbligo di ricevere autorizz., ma manca controllo fisico (???)) sono sulla protezione dati fisici (necessaria perché ancora tutto viene stampato, importante al pari).

11° LEZIONE

GPDP

Garante per la protezione dei dati personali (per essere compliant con garante privacy): ha a che fare con gli amministratori di sistema, è ancora vigente a prescindere dal GDPR. Dice come comportarsi da sys admin. Ha un certo insieme di privilegi superiori di gestione risorse rispetto alle singole utenze, tutti i sys admin devono avere una nomina, e deve esserci una lista (dei sys admin che trattano dati personali di dipendenti o non, quindi tutti praticamente) che li elenchi e dica su che risorse hanno privilegi (lista deve essere conosciuta a tutti in azienda).

Una segretaria che genera pwd e le dà alle utenze è sys admin, se dà i pizzini con le pwd no, se dà pwd in comune a tutti no. Amministratori di db, amministratori di reti, amministratori di firewall o mezzi di sicurezza, chi fa backup, sono sys admin. Chiunque al di fuori delle utenze che fanno il loro e basta, gli altri sono sys admin, dipende dai privilegi che hanno su determinate risorse rispetto le generiche utenze. Gran parte dei compiti dell'allegato B sono svolti dal sys admin. Prima di essere assegnati come tali, devono essere valutati su esperienza e capacità, e poi va messo scritto e firmato da entrambe le parti. La nomina deve essere individuale con relativo privilegio di autorizzazione. C'è possibilità di avere sysadmin outsourcing (per esempio SOC che mi fa anche solo firewall), per anch'esso i suoi identificativi devono essere conservati dall'azienda. Almeno ogni anno l'azienda deve rivalutare le competenze del sys admin, e come ha trattato i dati personali. Devono esserci log di accesso con possibilità di verificare integrità (firma digitale) di essi (la soluzione migliore è avere SIEM esterno con SOC, senza una terza parte è discutibile), dei sys admin quando si autenticano al sistema che amministrano.

REGULATION 2016/679 EUR LEX (GDPR)

Regolamento per la protezione delle persone con riguardo al trattamento dei dati personali e circolazione. Non c'è parola privacy perché è data protection, misura per proteggere dati personali; quindi, ha a che fare con la privacy. È costituito da 173 considerando, 99 articoli (con definizioni, diritti (privacy), security (+ DPIA), multe). La privacy si espleta con obiettivi funzionali, cosa si può dover fare, per far ciò serve sicurezza da attaccanti (chiedo di cancellare dati ma attaccanti li hanno). Tra i diritti c'è il diritto alla cancellazione nel caso in cui un utente chiede di fare ciò sui propri dati. Anche security senza diritti non ha senso, perché è data protection. Articolo 4: qualsiasi informazione che riporti in modo diretto o indiretto a una persona (PII, Personally Identifiable Information, univocità) (CF è PII, id univoco per le persone, anche IP (anche dinamico)). Definizione di trattamento

(tutto), profilazione: uso algoritmi per elaborare i dati di una persona fisica e trasformarli in informazioni (do mutuo sì o no, vedo affidabilità), ci sarà articolo su consenso sulla profilazione, molto importante. Pseudonimizzazione. Consenso dell'interessato. Violazione dei dati personali (data breach, è trattamento non autorizzato, indebito). Articolo 5, il più importante (da fare bene): correttezza e trasparenza, e descrivere la limitazione delle finalità. Minimizzazione dei dati. Dati esatti, dati conservati solo per il tempo del servizio (limitazione conservazione). Trattati con sicurezza (mezzo tecnico/amministrativo) (integrità e sicurezza). Tutto il GDPR si fonda su punto 2: azienda che tratta i dati è accountable, responsabile in modo negativo, colpevole, se ci sono problemi, responsabile nel bene e nel male su tutti i principi sul dato.

12° LEZIONE

Fino ad ora solo aspetti teorici, manca il come si fa concretamente. Sul punto del trattamento dati (Articolo 6) è importante sapere che il titolare dei dati può trattare i nostri dati solo se diamo il consenso, però alcuni siti, nonostante non diamo consenso, possono darli a siti terzi per legittimo interesse. C'è da vedere cosa è di legittimo interesse per le aziende e cosa è nostro diritto non dare come dati personali con consenso. Il filo di differenza è sottile e lo discutono gli avvocati dell'azienda caso per caso (a ed f, consenso e legittimo interesse).

Articolo 7: Condizioni consenso: 1. Il titolare deve essere in grado di dimostrare il consenso dell'utente.

Articolo 9: ridondanza di esplicazione, consenso necessario per dati particolari. C'è un'estensione riguardo i dati sensibili con l'aggiunta di genetici e biometrici. Vedere a), g) i) per esempio covid, se riguarda sanità pubblica, i dati particolari possono essere trattati.

Diritti dell'interessato: articolo 15: l'interessato ha il diritto di chiedere al titolare che dati personali hanno e per quale finalità li tengono.

Articolo 16, diritto di rettifica se inesatti, articolo 17, diritto all'oblio se, secondo interessato, i suoi dati non sono necessari… per la cancellazione bisogna fidarsi che siano compliant, cioè che avvenga effettivamente.

Articolo 20: portabilità dati. Prima del 2006 non c'era portabilità dati, per esempio per cambio SIM con stesso numero, gestori energetici. La novità è che GDPR la estende a tutti i tipi di dati, tra cui quelli di portabilità medica, dovrebbe essere possibile senza impedimenti, deve essere leggibile da calcolatore.

Articolo di opposizione 21 in generale, anche se per scopi pubblici di ricerca o storici l'opposizione è possibile.

Articolo 22 (importante, profilazione): profilazione è trattamento automatizzato, algoritmo, l'interessato ha il diritto di non essere sottoposto a una decisione basata solo su profilazione, per esempio per assunzione in azienda, salvo alcuni casi sotto.

Articolo 25, protezione del dato by design by default: 1. l'azienda mette in atto misure tecniche e org. Adeguate ad attuare i principi dell'articolo 5, tenendo conto dello stato dell'arte, ma dipende dal costo, in base all'azienda se è grossa o meno (differenza che nell'allegato B non c'era) by design. 2. By default.

Articolo 30: deve esserci un registro di dati trattati, nell'allegato B invece no.

Sicurezza di dati personali (importante):

Articolo 32: si ripetono i parametri di valutazione a priori, è importante valutare il rischio e poi fare delle scelte, è il motore per essere compliant con articoli come GDPR. Infatti, si parla di sicurezza (tecnica e organizzativa) adeguata al rischio, ci sono degli esempi che potrebbero essere opportune o meno, dipende sempre dal rischio (poche righe appunto per questo, a differenza di allegato B in cui c'erano tante pagine). Tra gli esempi: CIA (ISO 27001), disponibilità (disaster recovery), vapt, resilienza, cifratura. È importante la

premessa, le misure tecniche devono essere adeguate a garantire un livello di sicurezza adeguato al rischio, per mitigarlo. Non ci sono più i requisiti minimi per tutti, si è passato a misure risk-based. In allegato B c'erano le misure di sicurezza, adesso invece dipende dal rischio e requisiti astratti e dipendenti dal rischio. Si vanno a vedere gli standard adatti, tra ISO, ENISA, e altri. Audit è insieme di controlli, ogni controllo è verificare una misura di sicurezza. Va considerato anche il rischio di data breach riguardo privacy. ISO 27005 è risk assessment. ISO 27002 è controlli di sicurezza.

Articolo 33: se c'è data breach, l'azienda informa il garante (su tutto) senza ritardo, entro 3 giorni, a meno che sia improbabile che la violazione dei dati è un rischio per i diritti delle persone fisiche. Serve DPO (Data Protection Officer) in azienda, da comunicare al garante in avviso di data breach, cosa ho fatto durante l'evento.

Articolo 34: comunicazione di data breach a clienti/customer se c'è rischio a persone fisiche, linguaggio semplice.

Sezione 3, finita sicurezza: valutazione d'impatto su protezione dati (DPIA, Data Protection Impact Assessment): prima di procedere al trattamento va valutato il rischio, è risk assessment.

13° LEZIONE

ISO 27001

Vedere qualche esempio e struttura generica del sommario, flusso struttura e il perchè del legame per esame.
Dall'indice si vede un'orientamento police-based: va analizzato il contesto, il risk assessment e i modi per fare enforcment della policy. Nel contesto interno c'è CISO, DPO mentre esternamente il garante. Discutibile il fatto che la policy è stabilita dal top management,

potrebbe non capirne nulla. Diffusione policy, distribuzione ruoli. Vanno valutati i criteri del rischio: il criterio di accettazione del rischio, e di valutazione del rischio. Ripetizione di risk assessment per limitare la forte soggettività a cui è legata questa analisi. È bene considerare gli owner dei rischi (ogni processo ha il suo responsabile, es. backup). I rischi non sono eventi valutabili (es. firewall è aggiornato si/no, 0 o 1), ma difficilmente valutabili, meno tangibili e misurabili, più basati sulla probabilità (in realtà cvss in vapt è basato su indicatori oggettivi, come indicatori di benessere ecc...). Va analizzato il rischio determinandone il livello, conseguenze, verosimiglianze. Evaluation è confronto di risk analysis e criteri di rischio (risk assessment spiegato meglio in 27005). Il rischio è quantificabile tramite un livello (in realtà è analisi qualitativa tramite un aggettivo, l'importanza è il rapporto con la scala usata, il range, quantitativa è con unità di misura come denaro rischiato di danno), mentre threat (minacce) no.

La verosimiglianza si differenzia dalla probabilità perché quest'ultima è assoluta, mentre nel primo caso il valore è relativo al significato assunto (es. 3 di verosimiglianza, 3 su 7 auto), probabilità tipo 100% può essere qualsiasi cosa. Nel caso qualitativo si rientra nel caso della verosimiglianza. La verosimiglianza è soggettiva, io posso dire 2 ma un altro 3. Il livello di rischio dipende da quello di verosimiglianza e delle conseguenze, entrambe dipendono da dibattiti soggettivi.

La valutazione (esito booleano) è output dell'analisi (confrontare dati analizzati che si hanno con criteri), e quest'ultima dipende da criteri. Prima si fa analisi, poi evaluation. Ma in analisi c'è assess(??) (a sua volta è analisi e evaluation??).

Trattamento del rischio: mitigare, un primo modo è accettarlo, la scelta più saggia è cercare di risolvere il problema, ma senza avere mai la certezza di aver applicato le giuste misure, le ISO non si prendono questa responsabilità.

ISO 27002

controlli: controlli organizzativi (paragrafo 5) appunto perché sicurezza è una faccenda anche organizzativa, deve esserci policy. Posiziona ontologicamente il controllo. Guidance: la policy deve avere una serie di parametri. Devono essere definiti ruoli e responsabilità. Isolare aree di conflitto (come in WATA insegnante e visionatori). Vedere esempio di controlli, uno, due per categoria e lista di categoria, struttura e perché è così per esame. Ci sono anche controlli per mitigare errori umani, come consapevolezza, training. Vedere esempio 6.3. 6.4 è processo disciplinare interno se qualcuno nonostante tutto sbaglia. Remote working 6.7 altrettanto importante. Paragrafo 7 è sui controlli fisici, non tutti dovrebbero potere entrare dentro determinate aree. 7.1 e 7.2 generici. 7.7 sul non lasciare informazioni in giro (come in allegato B) o sul display (lockscreen). Paragrafo 8. 8.17 Clock synchronization. Per esempio, in Kerberos, non avrebbero senso nemmeno i log. Guidance dice con NTP o PTP, anche questi è importante siano sicuri. 8.24 è use of crypto in cui è importante key management.

ISO 27005

Risk management: Glossario al paragrafo 3: le conseguenze del risk assessment dipendono da un evento; controllo è modificatore di rischio; evento è es. firewall c'è o non c'è; contesto interno (governance, patrimonio tecnologico) e esterno (GDPR); livello di rischio è un valore associato al rischio, sia se proviene da aspetti qualitativi o quantitativi espresso in funzione di conseguenze e verosimiglianze; likelihood è possibilità che qualcosa accada; rischio residuo è quello che resta dopo le mitigazioni; rischio è evento con likelihood non assoluta (?). Risk analysis è produrre il risk level, poi la valutazione è confrontare questo con i criteri. L'assessment è risk identification e poi analisi e evaluation; risk communication; risk criteria; evaluation è comparare analisi di rischio e criteri; management è gestione di tutto il processo; risk treatment.

Immagine processo gestione rischio: è un ciclo da fare periodicamente o se cambio qualcosa nel sistema (cambio firewall esempio), stabilisco il contesto (es. da noi siamo molto attenti) per stabilire i risk criteria, assessment è identifico rischio, analizzo, valuto. Alla fine, tratto, cioè, accetto o mitigo. Alla fine in ogni caso accetto il rischio residuo. 7.2.2 sui criteri di rischio 7.2.4 su accettazione di rischio, potrebbero esserci soglie multiple.

Come calcolare effettivamente il rischio sulla base di verosimiglianze e conseguenze? Vedere Annex C pagina 37, c'è esempio di lista di minacce, anche se variano nel tempo, per risk identification. In Table E.1 ci sono esempi per analisi. La E.2 è la più comune (manca asset di proposito, potrei non considerarli), ci sono fasce di rischio basso, medio e elevato (È evaluation, vedere in che fascia ricadiamo (?)). Lo scenario più tipico è con moltiplicazione tra likelihood e conseguenze, ottenendo il risk level. Risk assessment è soggettivo e va fatto in team. Vanno mitigate le minacce più rischiose prima.

14° LEZIONE

METODI FORMALI

Sono importanti, per esempio, considerando che l'autenticazione è interpretabile a diversi livelli (1° livello di aliveness, A c'è, c'è stata, 2° livello è basato sulla sessione, se effettivamente A ha sessione con B), per esempio attacco di Lowe a NS c'è se considero autenticazione 2° livello, per 1° livello no (perché anche se C intercetta sessione, comunque A c'è stata).

Principio di induzione. Perché vale. Un principio non è un teorema, quindi non è basato su una dimostrazione sulla quale tutti possiamo fare verifica. Infatti, ci si affida ad esso. I principi non sono dimostrabili. Principio di induzione dice che, se vale [P(0) caso base; e P(n) -> (implicazione logico, a livello oggetto) P(n+1)] allora => (implicazione a

un altro livello, metalivello) per ogni n vale P(n). Il ; è AND logico. Le quadre racchiudono le ipotesi.

[P(0); P(n) -> P(n+1)] => per ogni n. P(n). Il primo n è una costante, per un generico numero, vale anche per il successore della costante. Per tutti gli n, questo n ha significato di tutti gli n, sta nello scope.

Applicazione del principio è verificare le ipotesi e quindi aspettarsi di avere la conseguenza.

Il principio di autenticazione va portato a una dimostrazione matematica.

I numeri naturali sono definiti per induzione: Base 0 appartiene a N, Ind n appartiene a N, allora succ(n) appartiene a N. Allora gli elementi sono regolamentati dal principio di induzione. Questo è un modello formale per i numeri naturali, una rappresentazione, modello perché magari ce ne sono altri, formale perché c'è un linguaggio da usare.

I passi induttivi possono essere più di 2 (compreso caso base), non di meno. Ciò implica semplicemente che per ogni caso induttivo c'è una regola, un'ipotesi in più, una certa precondizione.

Proiettato alla sicurezza, il modello per il protocollo sarà orientato a tutte le possibili esecuzioni, che costituiscono il dominio. Esistono tool chiamati theoremProver o ProofAssistent.

Il model checking è nato verso gli anni 80-90, è un metodo di verifica di un modello. Il model checker è un software che utilizza tecniche di ricerca automatica per testare stati, percorsi di un programma, le possibili istanze. Model checking equivale a enumerazione a stati finiti. Uno stato è la fotografia di un programma, nel nostro caso è di tipo epistemologico (di conoscenza a un certo stato). Il modello è differente da quello induttivo. L'esplorazione consiste nel verificare gli stati di passaggio dell'istanza, se si arriva per esempio a conoscere una chiave (se l'obiettivo è la segretezza della chiave). Aumentando i possibili stati, la quantità combinatoria aumenta, quindi per questo ci sono i tool che automatizzano l'analisi. C'è il problema di state explosion, se

aumentano troppo le sorgenti di finitezza (agenti del protocollo e numero messaggi del protocollo, lunghezza messaggi, numero di nonce) il tool non funziona (problemi di risorse computazionali). Per piccoli problemi è molto utile. Il symbol model checker allarga questo campo avendo come stati a sua volta un'istanza con tanti stati possibili. Quindi il model checker è limitato a ipotesi ristrette, se riesce a dimostrare che non ci sono attacchi (semplici), non necessariamente è detto che non ce n'è con altre casistiche non considerate, allargando le sorgenti di finitezza. Questo perché non sono dimostrazioni matematiche.

Dualismo tra metodo induttivo/matematico e a stati finiti. FDR4 model checker. Dato che l'induzione è assunta vera, se associamo una proprietà a un modello induttivo e riusciamo a verificarla con l'induzione, allora abbiamo dimostrato che la proprietà è vera all'interno di quel modello.

15° LEZIONE

Nelle macchine a stati finiti il lato positivo è che c'è la possibilità di vedere i singoli stati, ma dall'altro lato rende limitato il risultato, in quanto non ho lavorato nella rappresentazione generale. Ci piacerebbe avere un modello che analizzi infinite casistiche, unbounded.

I metodi formali li abbiamo già usati, perché sono una qualunque tecnica per rappresentare un sistema reale in modo astratto tramite un rigore matematico. Per esempio, se devo progettare un'auto prima faccio un modello e ne testo la safety per evitare incidenti. Un altro esempio è l'integrale. Matematica è inerentemente modellazione di sistemi reali, a parte certi casi come integrali tripli o altro. Sono modelli in quanto indipendenti da sistemi reali. Vogliamo estendere il tutto ai protocolli di sicurezza.

Inizialmente si ebbero logiche di autenticazione come BAN. Process calculus c'è lambda calcolo. Approccio è metodo generale. È molto più

difficile dire che un protocollo non ha attacchi tramite dimostrazione formale rispetto a trovare un attacco.

Il model checking ha un grafo degli stati finiti in cui ogni nodo è uno stato (fotografia). Ci sono due problemi che affronta: specification (ottenere modello formale del target protocol, tramite process calculus), verification (vuole trovare attacchi o mostrare proprietà sul modello, tramite model checking). Definizione induttiva di numeri naturali è specification.

La rappresentazione degli stati in slide è epistemologica, si vede cosa conosce A eccetera. È Needham-Schroder, man mano le conoscenze crescono. Ci si concentra non sulle spedizioni di messaggi, ma sugli stati di conoscenza. Per verificare segretezza basta controllare tutti gli stati e vedere cosa conosce un agente. Vedere pro e contro rapidamente. DY+ non potrebbe essere specificato in rappr. stati, perché poteva fare messaggi di lunghezza a proprio piacere, c'è una sorgente di finitezza da rappresentare come macchina a stati infiniti, ma non è possibile. Per questo MC più usato per trovare attacchi, Theorem Proving per dimostrare correttezza.

Livello	Quantificazione	Esempio	Uso nei protocolli
Propositional	Nessuna	$A \land B \to C$	proprietà elementari, verità/falsità di proposizioni
FOL	Su variabili (oggetti)	$\forall x.\exists y.\mathbf{Knows}(x, y)$	agenti, chiavi, messaggi
HOL	Su variabili, funzioni e predicati	$\forall P.\forall x.P(x) \lor \neg P(x)$	proprietà astratte, principi generali di sicurezza

Il metodo induttivo è basato su logica di ordine superiore, c'è esistenziale e universale da applicare a variabili e funzioni. Verificare tramite logica prop. vero o falso costa esponenzialmente (problema di soddisfacibilità SAT (esiste ass vero/falso a var. che rende formula vera?) è NP-completo, cresce esp. con numero variabili). Salendo al primo ordine non si può più dimostrare che una formula sia vera o falsa. La macchina da sola potrebbe non farcela. C'è sempre bisogno di

analista umano per le cose più difficili, le porzioni più semplici saranno gestibili dal tool, per esempio A&&B (perché è proposizionale, decidibile, non è del 1° ordine). Induzione è usata nella parte di definizione del protocollo (specification) e nella verifica. Induzione → usata per ragionare su insiemi infiniti; i tool la applicano solo se l'umano guida.

Nel model checking si vuole negare la formula per trovare attacco, il motore di ciò è il SAT solver (???). ProVerif permette di specificare protocollo di processo ed è automatico. Grazie alla magia dell'induzione (fede) ci servono pochissime semplificazioni. Il theorem prover è interactive per il discorso di prima della logica di 1° livello che ha bisogno dell'analista umano.

Per rappresentare modello di protocollo di sicurezza (specification) devo rappresentare la sessione tra gli agenti tramite un protocollo. Durante la verification studio il modello per vedere se c'è una violazione di confidentiality, se no, potrebbero comunque esserci per esempio due sessioni, quindi serve approccio generale, basta pensare attacco di Lowe (due sessioni interleaved, con 3 agenti), oppure ad attacco temporale su Kerberos IV (piu agenti e fattore temporale).

Modellare un protocollo è come scrivere una biografia, rappresentando un fatto dopo l'altro sul soggetto, nel mentre possono subentrare altri personaggi. Il protocollo porta benefici per chi vi partecipa. A differenza del mondo reale, tutti si relazionano internamente solo sfruttando quella precisa sequenza di messaggi di un unico protocollo, invece che tantissimi protocolli. Ma potenzialmente gli agenti con cui interagisce sono infiniti, e noi vogliamo rappresentare le interazioni con tutti, ma significherebbe scrivere biografia dell'umanità (= lista infinita di agenti). Non va bene fissare un intervallo di tempo, perché ci potrebbe essere attacco fuori. Voglio infinitezza temporale e di dimensione (vedi sopra).

Quindi voglio modello per uno specifico protocollo, da istante 0 a unbounded, per tutti gli agenti. Vorremo insieme di tutte le possibili biografie, con tutte le scelte possibili, per poter fare affermazioni totali

anche sul futuro. La differenza tra i modelli (?) sta nel tipo (linguaggio di progr.) base, nei numeri naturali per esempio li evolvo solo tramite successore, nel nostro caso delle biografie il mattoncino è l'interazione. Il tipo è la lista, una lista di eventi, la posizione di essi dice cosa viene prima e dopo, quindi non servono operatori temporali. Le liste di eventi si chiamano protocol trace. Voglio tutte le possibili trace (per il modello del protocollo), basta fare induzione sul singolo trace.

Event.html di Isabelle: in datatype ci sono 3 eventi, 3 solo perché vogliamo il focus su questi, dipende dove vogliamo metterlo.

Theorem prover Isabelle: Ho assiomi da quale voglio provare regola, mi servono due cose: classical reasoning (forme di deduzione banale, regola conjE1: A&&B => (metaimplicazione) A, non è safe perché non c'è piu b, infatti c'è anche conjE2: A&&B => B, mp (modus ponens): [|A -> B; A|] ==> B). Isabelle tramite la deduzione automatizza queste regole (che assumiamo sound da precondizioni). Mi serve anche fare term-rewriting, riscrittura di termini, esempio in slide. Vedere metodi utilizzabili in tool, clarify fa reasoning, proof state è l'insieme dei casi che si hanno (base e induttivo), il goal è la tesi, che si suddivide in subgoal che sono i sottocasi.

Per usarlo in pratica, metto dentro theory file il modello, con gli script (es. applico princ. di induzione, chiudo caso base) di dimostrazione, verifica. Computazione in programmazione funzionale non basata su variabili, perché la variabile è il proof state (file che descrive la funzione, da utilizzare poi), anche applicazione del principio di induzione è fare programmazione funzionale, a causa della riscrittura, come le espressioni matematiche, nel linguaggio parlato.

16° LEZIONE

Non esistono modelli giusti e sbagliati, ma realistici o meno in base alle ipotesi di partenza per applicare la definizione. Il lato positivo del modello rispetto la realtà è che l'unico verbo è limitato all'esecuzione del protocollo, anche se poi la popolazione è unbounded. Per arrivare a definire tutte le possibili combinazioni, useremo l'induzione. A crescere per l'unbounding è la lista, può essere allungata da un singolo evento. La struttura è come i numeri naturali, cambia solo il tipo base. Le liste stesse di per sé sono definite per induzione, caso base è vuota.

Con i protocolli di sicurezza voglio introdurre l'insieme come free type che va definito, come i numeri naturali con la loro definizione e N come nome, ma può essere qualsiasi altro. Per esempio, per NS, Paulson ha scelto NS_public (public sta per asymmetric crypto). Il caso base è [] appartiene a ns_public, non succede nulla. Poi ci sono 3 passi induttivi quanti i passaggi del protocollo. 1. A->B: {A,Na}kb, 2. B->A: {Na,Nb}ka, 3. A->B: {Nb}kb. Quindi gli unici casi possibili sono uno di questi 3. Passi induttivi: NS1. Evs (Events) appartiene a ns_public => # (cons, inserimento in lista) appartiene a ns_public). Says (costruttore del tipo evento, spedizione) AB(Crypt(pubkB){|AgentA, Nonce Na|}). AB sono agenti, il resto è messaggio, dentro il messaggio abbiamo esplicitazione dei tipi, per evitare esfiltrazione dati (attacchi a incrocio tipi), per questo è un linguaggio fortemente tipato.

Per il messaggio 2 ci sarà un ulteriore precondizione cioè B deve aver ricevuto prima il messaggio 1. Le precondizioni del messaggio 1 potrebbero essere qualsiasi (accendere pc, rete stabile), ma non possiamo considerarle tutte, quindi trascuriamo quelle che vogliamo, in base all'importanza relativa che gli si dà. La precondizione della chiave viene messa a monte, non qui, e A non ha precondizioni (ci può essere anche attaccante) ma è importante precondizione su nonce, deve essere usata una sola volta, e generata ora fresca, se non la aggiungiamo non si può chiudere il subgoal, ci sarebbe il caso aperto non verificato di nonce già usata. Quindi aggiungo tra le precondizioni

che Nonce Na non appartiene a used evs. -> [|evs appartiene ns_pub; Nonce Na... |].

Isabelle è l'interprete delle espressioni di ML (Meta Language). È scritta in ML per elaborare formule logiche che sono espressioni da semplificare riscrivendole. Le 3 logiche messe insieme a Isabelle danno HOL. Il protocollo NS__public quindi in primo luogo è implementata su HOL. Sets (insieme non ordinato e senza duplicati) e lists (sequenze ordinate) le ritroviamo già dentro main, sono statement fatti da altri che si possono usare già pronte. Le 3 teorie su cui basarsi per fare NS_public oltre a quelle già date è message (per protocollo, ha mittente e destinatario, i passi possibili), event (per trace, invio, ricezione ecc..), public (per crittografia asimmetrica, in realtà include anche shared per quella simmetrica).

Questa è sola parte di specification. Modello di protocollo giocattolo per esempio. In FLP1 (FLP per) è molto simile a NS ma senza cifratura. Importante evs1 appartiene a flp. È induzione strutturale sulla lunghezza delle tracce. In realtà evs1 e evs2 potrebbero avere lo stesso nome anche se rappresentano due tipi di messaggi diversi del protocollo, lo scope è diverso e funziona, è sempre un nome di variabile. Le etichette devono essere distinte (FLP1 E 2), per storare senza ambiguità. La parentesi graffa più barra è la formalizzazione dell'insieme contenente il messaggio. Inoltre [| ... |] è per le precondizioni. Caso base non ha precondizioni perché certamente esiste lista vuota. Va letto: "Data trace del modello, quando nonce Na è fresca su events, l'estensione della testa della lista con un messaggio da A a B di quel tipo, può essere estesa come una lista di eventi del modello specifico.

Per il messaggio FLP2 ci dovrebbe essere firma digitale, ma in modelli formali non si esplicita mai, solo per convenienza, è al di fuori dell'interesse. Fake vuole modellare l'attaccante, recp la ricezione, necessaria perché il MITM potrebbe bloccare l'arrivo e quindi il sending di per sé non è sufficiente. X non è esplicitato (va bene qualsiasi

struttura), è regola generica per tutti i messaggi. Il verbo può estendere la lista è perché le precondizioni devono essere valide, ci potrebbero essere ricezioni avvenute, e quelle non avvenute che non si trovano. Così ho anche nel modello casistica di DY+.

Va definita knows, analz, spy (? È attaccante), used. Come prima versione di events aveva solo invio, non ricezione. Senza esso (senza gets??), potremmo modellare il messaggio 2 aggiungendo la precondizione che basta la spedizione dell'1 e si suppone il poter inviare il 2. Assumo che spedire implica ricevere. Ma voglio eliminare tutta questa supposizione. Lo ha risolto aggiungendo primo in A, A primo, è chiunque, può essere uguale ad A o a chiunque altro, importante che qualcuno ha mandato, anche se non è molto readable (generalizza mittente, inserendo anche DY+ come possibile). Non avere la gets implica minore conoscenza degli agenti nel modello. Al posto di says A' mettere gets. Knows dice chi conosce cosa. Non ha ritenuto necessario ricezione perché, se c'è ricezione è perché deve prima esserci stato un invio nel traffico del messaggio. La ricezione senza gets non avrebbe senso, è modellata proprio per questo.

17° LEZIONE

È unbounded, flp è il modello. Nella prima versione non c'era gets, ma tutto diventa più leggibile. Nel modello a rettangoli con i frame, solo la ricezione non si poteva modellare, Larry ha detto però che, se sta nel traffico, allora è stato spedito. Per autenticazione però non è detto che sia effettivamente di alice, poteva essere di un attaccante. Per la segretezza è encoding in questa struttura, c'è quando attaccante non lo può ricavare dalla sua analisi del traffico. Gets accettata successivamente, utile quindi per autenticazione (?). C'è esempio generale di trace di un modello ma non di quel protocollo in alto, è al contrario, si inserisce in testa nella lista. Nel modello di questo

protocollo (è un insieme di eventi, trace) non devono esserci eventi di tipo notes. L'evento notes non è stato definito. Nelle post-condizioni non c'è mai l'inserimento di un evento notes, non è una questione di precondizione. Se c'è nella precondizione ma in nessuna post-condizione, quella precondizione non sarà mai verificata.

Per la segretezza, data una traccia di esempio chiamata evs = [Says AB__X, Gets _ X] va definito knows -> knows Spy evs, va definito analz -> analz (quello prima), parts (che prende ancora quello prima). Spy avrà sempre volontà di arrivare al messaggio. Se c'è è una gets, è perché c'era stata una says (dalle precondizioni scaturisce la gets), quindi non ha bisogno di estrarre il messaggio. L'attaccante vuole anche un set di messaggi. Per quanto riguarda la knows, gli altri rispetto la spia l'unico modo per apprendere che hanno è inviare e soprattutto messaggi ricevuti; quindi, per questo ha importanza esplicitare la ricezione. Analz serve per vedere i messaggi, li spezza e li scompone, estrae contenuti di crittotesti ammesso che abbia la chiave (Knows è avere il blocco di eventi). Parts è simile da analz, per estrarre contenuto crittotesto da insieme di partenza, senza chiave, è più potente, l'insieme di input possibile è più grosso. Parts serve soprattutto se manca la gets che estrae il messaggio. Parts riesce a vedere se dentro il calderone di tutti i messaggi c'è un certo messaggio (quindi riguarda solo esplicazione di ricezione, allora se quel messaggio c'è, questo implica says AB X).

La confidenzialità si esprime quindi con key K non appartiene a analz (knows Spy evs). La regola fake dice che evsf se appartiene al modello, estende con un altro evento di Spy che manda a un generico B un messaggio X, quello di prima delle precondizioni. Synth è l'operatore che costruisce il messaggio malevolo (concatenazione e/o cifratura e/o hash) da componenti singoli che ricava da analz ottenuti accorpati in knows.

Theory message -> HOL_AuthMessage.html. key = nat è riscrizione di termine. Nelle const ci sono funzioni con corpo costante, invKey è una

chiave che implica la sua inversa, è solo una definizione per dire che una chiave privata implica una chiave pubblica. Nelle definizioni c'è il . che è tale che. Dice se la chiave è simmetrica, la sua stessa inversa è sé stessa per dire che non c'è. Questo aiuta nella riscrittura, in questo caso invk verrà riscritto come k. Datatype agenti: c'è il costruttore per agenti per evidenziare Spy. In Says AB non è escluso niente, non sono limitati di tipo, inoltre per esempio A può anche essere uguale a B, il modello è più forte. Spy è variabile o costante? Costante, A e B sono invece variabili (che assumeranno il valore di costante server, spy o friend nat). Il messaggio ha definizione ricorsiva per fare più parti.

Parts prende un insieme di messaggi e lo trasforma in un altro, costruito per induzione come si vede sotto. I lemmi servono per riscrittura. Analz sarà come parts ma in body avrà in più precondizione di conoscenza chiave. Vedere lemma parts_insert_Number, scompone e poi induttivamente può andare su parts(H), semplifica molto. In analz si usa invKey che genericamente cattura sia chiave di sessione per cripto simmetrica, sia chiave privata per asimmetrica. Synth, in crypt Key(K) appartiene a H, e perché non ha messo synth(H), perché non stanno mai a destra di queste regole induttive, non si può mai costruire da sé chiave, ma data a priori. Se non sta in H, non sta in synth di H. Può costruire agenti e numeri da sé senza precondizioni, sono pubblici. Nonce e chiavi invece sono sempre oggetto di confidenzialità, analisi, non sono messi a prescindere.

In Event c'è Notes e Gets, c'è anche costante per indicare insieme di agenti collusi, che verrà riempito man mano grazie all'espressione funzionale definita. Quelli in bad sono quelli che hanno dato chiave a lungo termine ad attaccante, si ammette che siano compromessi.

Funzione knows (?? Sotto specification bad): è funzione currificata, di due tipi, dato agent e event list restituisce msg set. A è chiunque. initState è quello che sa (insieme di chiavi a lungo termine, non è riferito a enumerazione a stati finiti). Knows_cons è caso induttivo: # si legge cons. si vede inserimento ricorsivo in insieme più grande. A' per

indicare chiunque come mittente. In notes si evidenzia che, se A' è bad, allora Spy conosce note quindi contenuto messaggio.

Funzione used: si potrebbe usare parts, però il server deve generare chiavi fresche, va formalizzata, poteva essere messo nelle precondizioni, used è parts + tutti gli stati iniziali di tutti, per evitare di riscrivere tutto quello in precondizioni. Scremo caso estremi (che chiavi di sessioni corrispondo a chiavi a lungo termine) delle chiavi già presenti in init_state a lungo termine, è una cosa grossa tramite questa funzione. La vera differenza sta nel caso iniziale base.

In initState (dentro Public) c'è il server che conosce tutte le chiavi possibili, la Spia conosce le sue chiavi, e quelle che sanno i bad. I bad danno sia note che sue chiavi alla spia, collaborano alla collusione in 2 casi. Public_Bad, lemma A_trusts_NS2_lemma per autenticazione, guarda nel traffico senza Gets: Spies evs è il traffico delle 3 casistiche figura. Se A e B non sono collusi e c'è quel messaggio nel traffico, allora sicuramente è partito da quel mittente. In bad è inclusa la spia per semplificare subgoal fake. Anche il server non sta in bad.

Theorem Spy_not_see_NB: se ci sono quelle ipotesi, e in più A non ha mai mandato il messaggio 3, allora è confidenziale. La dimostrazione dei teoremi, la chiusura di essa, serve per dimostrare sicurezza. L'attacco è quello di lowe, non chiude subgoal, falsifica congettura iniziale, si trova assurdo (conseguenza false che provengono quindi da precondizioni false).

Se A, B non sono compromessi, e B ha mandato il messaggio 2 (con NB), all

$$Nonce\ NB \notin analz(spies\ evs).$$

Ricorda la tabella di verità:

A	B	A → B
Vero	Vero	Vero
Vero	Falso	**Falso**
Falso	Vero	Vero
Falso	Falso	Vero

Quindi:

- Se **A è vero** e **B è falso**, allora $A \longrightarrow B$ è **falso** → cioè siamo in contraddizione.

18° LEZIONE

Purple team si occupa sia di difesa che di attacco, più presenti in aziende. Le vulnerabilità sono contestualizzate a un applicativo, a differenza delle debolezze che sono problemi più generici (es. SQL Injection). Attacco a tls heartbleed è di tipo read out of bounds.

L'exploit sfrutta la vulnerabilità (es. il punto e virgola per concatenare successiva richiesta in input), il payload decide cosa fare dopo aver sfruttato la vulnerabilità (richiesta aggiuntiva dopo il punto e virgola). Un esempio di utilizzo di encoder è per facilitare l'invio, per esempio, di codice c in cui le stringhe devono terminare con \0. I nop sled servono nei buffer overflow per fare slittare l'indirizzo di esecuzione di return al quale andare, sta per no operation, perché magari non si sa a che indirizzo mettere il codice eseguibile, si sovrascrive indirizzo di ritorno con un punto dentro NOP sled (opportunamente inserito prima dello shellcode, cioè apertura e comandi shell via codice assembly iniettato).

Search type:'modulo'. Oppure search type:'modulo' 'vulnerabilità'. Lcd e lls in meterpreter permettono di eseguire ls e cd in macchina locale. Idletime mostra da quanto tempo l'utente non usa quella macchina.

19° LEZIONE

Lo svantaggio degli staged payloads è che insieme occupano più memoria di un unico payload, però questi ultimi sono più rilevabili da antivirus.

MsfVenom usato per social engineering, oppure se ho violato una macchina ma ho una shell normale, tramite questo si ottiene shell root. Generiamo un file malevolo in vittima che dovrà però eseguire, a

differenza di msfconsole che riesce a fare eseguire il payload direttamente per esempio tramite buffer overflow.

In testTime sono staged, hanno tutti lo stesso peso perché scarichiamo solo lo stager.

Red Team ha obiettivi simili, ma in particolare non si vogliono trovare tutte le vulnerabilità, ma testare le abilità difensive della vittima, l'obiettivo è farsi buttare fuori dalla rete dal blue team, a differenza di PT in cui si cercano più vulnerabilità possibili. Il successo si misura in base a quanto tempo si riesce a stare dentro il sistema attaccato. Ammette uso di phishing e social engineering.

In MSFVenom l'encoding ha più senso per offuscare il payload più che per evitare problema null byte.

Tryhackme.com/room/blue

20° LEZIONE

Passo 2 di shikata ga nai perché mentre prima gli eseguibili venivano caricati sempre nella stessa porzione di memoria dello stack (causava buffer overflow molto più semplici perché si sapeva con certezza l'indirizzo a cui jumpare), adesso vengono messi in modo random; per risolvere ciò si cerca l'indirizzo di un leak, una sola informazione del nostro eseguibile con la sua posizione, il resto della posizione di tutto sarà nell'intorno dello stack, e poi addr che è posizione esatta. Può essere eseguito ciclicamente per evitare detection (offuscazione su offuscazione, come hash di hash ecc..., ma lascia il tempo che trova).

Se il processo di partenza non ha privilegi SEDebugPrivilege, può migrare solo verso processi dello stesso utente (vedere usi legittimi, perché esiste).

post/windows/manage/sshkey_persistence crea coppia di chiavi, la pubblica la mette nelle chiavi riconosciute dalla macchina remota, mentre la privata nella nostra macchina attaccante, per poter instaurare una connessione permanente via SSH. Vedere post/windows/manage/sticky_keys: anche se l'utente fa logout, c'è la combinazione di tasti 5 volte shift (SETHC) che fa un qualcosa, potrebbe essere sostituito per aprire una shell (?). È backdoor. Il problema di questa vulnerabilità è che è un comando eseguibile anche senza aver effettuato accesso al sistema con una determinata utenza.

Di default si ha un solo canale di trasporto. In slide uso del trasporto si capisce che è reverse tcp perché c'è IP della nostra macchina come punto di ritorno, altrimenti con bind-tcp ci sarebbe stata un'altra porta. Il cambiamento di trasporto ha chiuso la sessione perché il punto di ingresso dalla nostra macchina non era stato aperto, cosa che viene fatta nella successiva slide (?)

21° LEZIONE

Tryhackme.com/room/ice

23° LEZIONE

Articolo 32 GDPR, Sicurezza del trattamento: il titolare è chi detiene i dati, a chi li stiamo dando (es. unict); il responsabile è colui che è responsabile del trattarli (es. gomp per unict). Rischio = probabilità*impatto, il livello è in base alla scala scelta. L'impatto è molto soggettivo, è difficile calcolare la probabilità esatta, e per questo poi l'analisi del rischio è soggettiva. Servono diverse misure di sicurezza riguardo i dati in base al tipo, sempre più restrittive con l'aumentare della delicatezza dei dati. Misure tecniche è implementazione HW/SW, misure organizzative è policy aziendale (sia regole hardcodabili (quindi con misure tecniche) che no. Il rischio può

essere mitigato, delegato da fare mitigare esternamente o accettato, in base al rischio ottenuto dopo l'analisi, c'è da capire se il rischio è accettabile. Mitigato eventualmente il rischio, accetto il rischio residuo. Tra le tecniche citate, la a) dipende se ha senso cifrare tutti i dati personali (perché comprendono ogni tipo di dato, dipende anche dalla potenza computazionale, in base alla macchina se non è efficiente). Per i dati particolari e genetici è del caso, se personali generici dipende. La b) con disaster recovery plan, e business continuity plan per casi meno gravi tipo blackout. La c) perché l'utente a cui appartengono i dati ha molti diritti sui dati che devono essere garantiti per il GDPR, che implicano la necessità di accesso. D) è PT, ma misure organizzative sono valutate da Red Teaming.

Nel punto 2 dice che il risk assessment deve comprendere distruzione, perdita, modifica, ecc.., cioè data breach (trattamento indebito). Punto 3 dice che c'è possibilità di certificazioni per i punti sopra, il 4 dice che chi tratta i dati deve avere almeno un'informativa sul come farli.

L'articolo 33 dice che, se c'è data breach bisogna notificarlo al garante della privacy, a meno che non si valuti che ciò non comporta rischi per i diritti delle persone fisiche. Dipende dalla convenienza dell'azienda, perché notificarlo comporta controlli ed eventualmente multe e dovute adeguazioni. Articolo 34 dice che, se c'è un rischio elevato di danni alle persone fisiche, vanno notificati anche i diretti interessati.

Pseudonimizzazione è associazione dati a uno pseudonimo del possessore. Lo pseudonimo è considerato personale perché permette di fare l'associazione tramite tabella di coppie. Anche il dato cifrato è personale.

Il dato anonimato non è personale, perché non è riconducibile a una persona univoca. Vedere k-anonimato wikipedia su esempio di tabella (per ogni "gruppo" di righe che condividono la stessa combinazione di quasi-identificatori, ci sono almeno k righe identiche su quei campi). Altro esempio di anonimato è l-diversity (l misura quanta varietà c'è nel valore sensibile dentro ciascun gruppo, così che sapere il gruppo non

riveli comunque l'informazione sensibile) e T-closeness (la distribuzione dei valori sensibili nel gruppo non si discosta troppo da quella del dataset intero). Anche i dati statistici sono anonimi. Un dataset che sembra anonimo al tempo t, potrebbe non esserlo al tempo t+1, quindi i dati anonimi non sono personali ma comunque da saper trattare.

PET (Privacy enhancing technologies): tecniche per proteggere dati. Per esempio autenticazione e autorizzazione. Si potrebbe fare offuscazione (per esempio mostrando solo una parte, come ultima parte carta di credito per capire se è propria). Anche ZK proof, cifratura omomorfica e secure multi party computation: Shamir Secret Sharing, ognuno condivide la parte di share, mentre un'altra la tiene segreta, poi tutti sono in grado di vedere per esempio la media, ma non i singoli dati di ognuno. Vedere federated learning (è PET perché elimina per i dispositivi la necessità di inviare i dati al server), si sposta il learning del modello dal singolo server a tutti i dispositivi in locale, con modelli molto più leggeri. Poi questi vengono reinviati al server e vengono aggregati.

24° LEZIONE

In risk assessment, evitare il rischio è differente dalla mitigazione, in quanto si rimuove la minaccia dagli asset in modo che la minaccia non si possa più verificare, per esempio fascicoli cartacei con dati personali a rischio incendio, evito togliendo i dati personali da carta e tratto solo su digitale. In strategia finale non c'è mai mitiga, perché altrimenti significa che dovremmo fare qualcos'altro, alla fine un qualche rischio va accettato.