Impatto pratico del GDPR sull’architettura Software

Giuseppe “Pino” De Francesco, SMIEEE

Abstract - Durante i due anni di periodo di grazia dopo che il General Data Protection Regulation (GDPR) fu pubblicato, la maggior parte delle aziende non erano preparate a diventare conformi. Il 25 maggio 2018 il GDPR entra in vigore, e il tipico patchwork dell'ultimo minuto è quello che vediamo succedere quasi ovunque, specialmente nell’industria della processazione dei dati e del software. Lo sforzo di tutti i responsabili della protezione dei dati dell'ultimo minuto è di assicurare il patchwork meno costoso da implementare, lasciando di fatto l’azienda esposta a grandi multe stabilite dal GDPR. In questo paper identifico i passi obbligatori per fare un’architettura software conforme al GDPR, guardando agli strumenti legali e tenendo a mente che siamo tenuti ad implementare “protezione dei dati in base alla progettazione e per impostazione predefinita”.

1. Introduzione

Il giorno 26 Aprile 2016 venne adottato il GDPR e venne pubblicato sul Giornale Ufficiale dell’Unione Europea il 4 Maggio 2016 [1]. L’orologio partì il 24 Maggio, contando alla rovescia fino al 25 Maggio, quando la regolamentazione entrò a pieno regime in tutti gli stati membri dell’UE.

La prima cosa da chiarire è che il GDPR è una regolamentazione Europea, non una direttiva. Le differenze sono le seguenti:

DIRETTIVA: Definisce che gli stati membri la devono inserire nella loro legislazione. Una direttiva è ricevuta ed implementata in vari modi differenti in ogni stato membro come ogni governo crede sia meglio.

REGOLAMENTAZIONE: È uno strumento legale che sostituisce ogni legge conflittuale sullo stesso campo in tutti gli stati dell’UE. Ha la forza di legge senza intervento del governo degli stati membri.

La differenza è sostanziale, e ancora, nel Marzo 2018 molti responsabili della Sicurezza mi dicevano che il 25 Maggio non era importante, perché “comunque i nostri governi devono implementarla nel nostro paese prima che abbia effetto”. La sensazione è che la maggioranza delle aziende rifiutano che la GDPR entri in vigore e possano prendere multe fino a 20 milioni di Euro per quelle trovate non conformi.

Una volta che hanno compreso che sta succedendo ed è solamente in un mese, dopo il rifiuto si muovono a fare il necessario per diventare conformi, provando a trovare ogni possibile scusa per non intervenire correttamente.

Questa nuova forma di rifiuto è di gran lunga la più pericolosa perché introduce false fiducie in quella che l’interpretazione legale dovrebbe essere.

In questo paper, stiamo iniziando adesso un viaggio pratico e pragmatico in quelle parti degli strumenti legali che influenzano l’architettura software. Io eviterò tutti i problemi legati all’IT, lasciandoli per un diverso studio, dato che loro meritano un’analisi separata data dalla complessità di fare una infrastruttura IT conforme al GDPR.

Mentre leggete questo paper, c’è una cosa che dovete tenere a mente ogni volta che dite “Devo realmente fare così?”: immaginate di essere in un’aula di tribunale, in piedi, e citato a giudizio perché avete violato il GDPR, e l’avvocato vi pone la grande domanda, “Hai o non hai

implementato le ultime leggi sulla protezione dei dati? Avresti o non avresti potuto, nella tua opinione professionale, fare di più per evitare la perdita dei dati?". Ricorda che in una corte di legge non puoi dire bugie a meno che non vuoi sopportare grandi conseguenze, e la GDPR è tutta riguardo all'implementazione e alla tenuta dell' essere all'avanguardia nel campo della Protezione dei dati con le nostre soluzioni. Ricorda anche che l'avvocato assumerà un consulente per provare che tu non hai implementato quelle norme, quindi è meglio che ci pensi due volte ogni volta che sei tentato di respingere uno standard di protezione dei dati.

2. Protezione dei dati

Quando parliamo di protezione dei dati, spesso non riusciamo a smettere di pensare sull'attuale significato né della relazione con la privacy. Inoltre, c'è una tendenza a proteggere i dati dopo una violazione, e questo approccio non è più accettabile dopo la GDPR.

2.1 Privacy vs Protezione dei dati

La prima nozione che dobbiamo digerire è la differenza tra Privacy e protezione dei dati. C'è spesso qualche confusione intorno questi concetti, quindi la miglior opzione che ho è di iniziare il nostro viaggio chiarificando entrambi.

2.1.1 Privacy

Noi, soggetti dei dati, abbiamo il diritto alla privacy, e questo è garantito da vari strumenti legali in tanti gusti, a seconda della giurisdizione, ma in generale La Privacy è un diritto. Tutti gli strumenti legali stabiliscono questo diritto fondamentale, dai convegni internazionali fino agli statuti locali, hanno provato a garantire in qualche modo la protezione della nostra privacy.

2.2.2 Protezione dei dati

C'è un solo modo per garantire il nostro diritto alla privacy: proteggere i nostri dati. Questo è un semplice concetto, facile da digerire: La protezione dei dati è il mezzo con cui la nostra privacy è garantita. Adesso, un mezzo è normalmente realizzato usando strumenti, e nel nostro caso, questo si applica. La protezione dei dati è implementata nell'architettura software e nella progettazione selezionando strumenti, pattern e standard che facilitano la protezione dei dati, e dove lo strumento manca, ne creiamo uno noi.

2.2 Protezione dei dati nella progettazione e per impostazione predefinita

Questo obbligo stabilito dall'art. 25 del GDPR è il pezzo per noi più cruciale della legislazione per capire e digerirlo completamente. Questo è anche dove quasi tutti gli attori di business con cui ho parlato (proprietari di prodotti, gestori di conti, amministratori delegati etc.) tendono a cercare vie di uscita. Buttiamoci nella prima frase dell'art. 25(1):

"Tenendo a mente le tecniche di avanguardia,

il costo di implementazione e di natura,

scopo, contesto e obiettivi del processamento,

così come i rischi di varie probabilità e gravità

dei diritti e libertà delle persone fisiche

poste dal processo, il controllore deve…

Notiamo tutti allo stesso modo la parte che recita “...il costo di implementazione…”? Ora, il mantra delle persone di business ancora recita così: “Costa troppo proteggere questi dati usando la tecnologia avanzata! Anche il GDPR lo dice!” Ok, questo è perché abbiamo considerato tutta la complessa legislazione, per aiutarci a capire quei passaggi che potrebbero portarci a malinterpretare la legge e le conseguenti multe. Il resoconto 26 chiarisce questo punto sui costi da considerare, e ha senso da un punto di vista tecnico come:

“Per accertare se è probabile che i mezzi

siano ragionevolmente utilizzati per identificare la persona fisica,

è necessario tenere conto di tutti i fattori oggettivi,

quali i costi e il tempo necessario per l'identificazione,

tenendo conto della tecnologia disponibile

al momento della elaborazione e sviluppi tecnologici."

Come possiamo vedere, i costi da considerare sono quelli rappresentati dallo sforzo di un attaccante per “identificare la persona fisica”. Noi, persone specializzate, usiamo questo approccio tutto il tempo ed il nostro obiettivo è sempre quello di assicurare che rompere le nostre misure di sicurezza è così costoso e comporta così tanto sviluppo che nessuna persona sana di mente inizierebbe mai a provarci. Il resoconto 83 usa una forma simile, come Art. 25(1), Art. 17(2) e 32(1).

Naturalmente come è chiaro al lettore, qui sto giocando a fare l’avvocato del diavolo. Certamente ha senso considerare i costi attuali di implementazione delle misure necessarie per proteggere i dati, ma questo vale per bilanciare le scelte, riducendo il costo implementando un livello di sicurezza adeguato ma fra quelli che possono essere implementati con il minor costo possibile.

Da nessuna parte nel GDPR, ci lasciano scegliere per non proteggere i dati perché costa troppo per noi: un simile atteggiamento costerebbe pesantemente all'azienda nel caso venisse citata in giudizio per non conformità. Quindi, ora che abbiamo chiarito il punto critico, capiamo cosa significa la protezione dei dati “nella progettazione e per impostazione predefinita”.

2.2.1 Nella progettazione

La formulazione dell’ art. 25 tratta di stabilire in base alla progettazione tutte le misure necessarie per proteggere i dati personali. Questo chiarisce, prima di discutere di ogni implementazione, la progettazione della soluzione deve preventivamente considerare come proteggere i dati, e specificatamente applicare i sei principi stabiliti nell’articolo 5(1):

- Liceità, equità e trasparenza
- Limitazione delle finalità
- Minimizzazione dei dati
- Precisione
- Limitazione dell'archiviazione
- Integrità e riservatezza

I mezzi tecnici per raggiungere tale conformità spesso menzionati nel GDPR sono:

- crittografia
- pseudonimizzazione
- sicurezza del trattamento

Noi, gli architetti, abbiamo il compito di progettare le nostre soluzioni usando quei principi, non andando avanti se non siamo soddisfatti di quello che abbiamo fatto facendo tutto il possibile nel contesto e all'avanguardia.

2.2.2 Per impostazione predefinita

Affermando "per impostazione predefinita" sembra ridondante, ma è un altro importante concetto che i legislatori hanno stabilito.

Nella nostra progettazione, tutti i comportamenti devono predefinire alle più strette regole della protezione dei dati. Molto spesso nel passato, il comportamento predefinito nel software era rappresentato dal minimo insieme, le basi. Ora noi siamo chiamati ad assicurare che se noi non conosciamo o non abbiamo le informazioni necessarie ad ogni punto nella nostra soluzione e nei suoi processi, allora dobbiamo applicare le regole più ferree, questo per garantire che nessun dato sarà mai processato o violato per errore.

"L'approccio Privacy by Design è

caratterizzato da proattivo piuttosto che

misure reattive. Anticipa e previene

gli eventi invasivi sulla privacy

prima che essi accadano.

PbD non aspetta che i rischi di privacy

si materializzino, né offre rimedi per la risoluzione delle

infrazioni alla privacy una volta che si sono verificati,

ha lo scopo di impedire che si verifichino.

In breve, Privacy by Design viene prima di tutto, non dopo." [2]

3. Processamento dei dati

Una menzione speciale è necessaria per processare i dati. Facciamo prima chiarezza su cosa *processare* significa, specialmente quando l'interpretazione della legge entra in gioco.

Se io prendo nota su un pezzo di carta un nome di una persona e un indirizzo, sto processando i dati della persona. Se prendo nota e li metto nel mio portafogli, sto ancora processando quei dati. Quando entrano in gioco i computer, la nozione diventa semplice, prendendo la definizione dal dizionario di Oxford: Operare sui dati mediante un programma.

Ogni operazione riguardante i dati in ogni modo rapprenta un processamento dei dati, non solamente l'utilizzo dei dati in un algoritmo computazionale relativo a quei dati.

Questo è enfatizzato nel GDPR in molti passaggi: dobbiamo limitare il processamento dei dati al minimo necessario per effettuare le operazioni che sono state autorizzate dal soggetto dei dati.

In linea di principio, se per una data operazione ho bisogno solo del nome e del cognome, io dovrei acquisire solo quei dati dal database, non tutto il record contenente email ed indirizzo: Dovrei creare una vista dei dati che soddisfi l'accesso minimalistico ai dati.

Questo perché più dati portiamo fuori dalla memoria, più dati mettiamo a rischio nel caso di un attaccante che guarda le nostre operazioni in memoria, e questo ci porta alla crittografia dei dati, ma prima vale la pena chiarire il significato di Stato dell'arte.

3.1 Stato dell'arte

Abbiamo visto che il GDPR obbliga l'utilizzo dello stato dell'arte della tecnologia corrente. Questo è menzionato in quattro punti: Resoconto 78 e 83, Articolo 25(1) e 32(1).

La definizione dello stato dell'arte è un bersaglio mobile: cambia nel tempo.

Il GDPR stabilisce che il processamento dei dati deve garantire lo stato dell'arte della tecnologia al momento del processamento. Perciò noi abbiamo degli obblighi legali a tenere le nostre architetture software aggiornate, completamente. Implementando lo stato dell'arte della tecnologia della protezione dei dati oggi non garantisce che in tre mesi la soluzione non sia obsoleta da nuove trovate.

3.2 Crittografia del database

Il fatto che dobbiamo disegnare, progettare, costruire e mantenere usando le ultime tecnologie ha un enorme impatto sul database che usiamo.

Ci sono due livelli importanti di crittografia del DB: a riposo(at rest) e in memoria(in-memory). La crittografia a riposo fu lo stato dell'arte finché la crittografia live DB fu introdotta, rendendo obsoleta di fatto l'opzione semplice a riposo perché comprende entrambi, a riposo e crittografia dei dati in memoria.

Mentre scrivo questo paper, ci sono solo due database che conosco che hanno queste due funzionalità insieme, Microsoft SQL Server dalla versione 2016 e Oracle Database Server con il modulo di crittografia dei dati trasparente, quindi ora (mentre scrivo), gli altri database sembrano non conformi al GDPR. Questo rappresenta un grande problema per noi perché molte soluzioni sono basate su database non conformi, e non c'è nessuna tabella di marcia per quelli di diventare conformi. In verità, alcuni DB gratuiti non hanno nemmeno i fondi per diventare conformi, quindi sono necessariamente destinati ad essere abbandonati dalle soluzioni software per i dati personali dei cittadini europei.

La situazione è differente per i database NoSQL: nessuno di essi, in mia conoscenza, ora offrono entrambe le crittografie. Perciò la crittografia a riposo è lo stato dell'arte per ora, che comprende molti dei più comuni NoSQL DBs utilizzati, come Mongo DB (solo Enterprise con WiredTiger engine) e Cosmos DB.

3.3 Struttura del database e pseudonimizzazione

Una volta che abbiamo selezionato il database corretto, dobbiamo pensare alla struttura. Quando parliamo di sicurezza, tutti sappiamo il vecchio detto di non tenere tutte le uova in un paniere, e questo si applica in questo caso.

Progettando database utilizzando la Terza Forma Normale (3NF) o la Boyce-Codd forma normale (BCNF) sembra non essere abbastanza. Il principio della pseudonimizzazione stabilisce che nella GDPR, il DB non può essere facilmente implementato con le nostre usuali pratiche di progettazione.

Il principio della pseudonimizzazione è da noi conosciuto più familiarmente come steganografia.

Qullo che dobbiamo fare è nascondere le informazioni sostituendole con uno pseudonimo, così nascondendo i dati in bella vista in un modo che l'attaccante vede al suo posto informazioni diverse e significative.

Prendiamo questo numero, immaginiamo che sia un valido numero di sicurezza: 1228475.

Salvando questo come una informazione steganografica può essere rappresentata per esempio come una lista di città americane: Rome, Denver, Washington, Arlington, Lebanon, Madison, Greenville.

Questo risultato è ottenuto applicando la seguente tavola di traduzione:

Originale	**Prima occorrenza**	**Seconda occorrenza**
0	Clayton	Auburn
1	Rome	Hudson
2	Denver	Washington
3	Springfield	Franklin
4	Lebanon	Clinton
5	Greenville	Bristol
6	Fairview	Salem
7	Madison	Georgetown
8	Arlington	Ashland
9	Dover	Oxford

Ovviamente, necessitiamo di molte più colonne per le occorrenze di cifre, ma questo da un esempio pratico di come utilizzare la steganografia per applicare pseudonimi ai dati, rendendoli inutili per un intruso.

Questo implica che la colonna dove salviamo i numeri di sicurezza sociale dovrebbero essere definiti nel DB come per esempio *PostiVisitati*, quindi chiudiamo il cerchio su come far disorientare l’intruso.

Dobbiamo anche affrontare il problema di dove salvare la tabella di traduzione. Salvandola nello stesso DB insieme ai dati su cui abbiamo applicato la traduzione sarebbe un errore. Quindi dovremmo metterla in posti separati, possibilmente un NoSQL DB crittografato.

Un’altra cosa che va detta per rendere sicuro il sistema steganografico: ricorda di sostituire la tabella di traduzione regolarmente. Per rendere questa operazione efficiente hai bisogno di una colonna per segnare quale tavola di traduzione è stata usata per crittografare la colonna, e questo può essere un qualunque valore convenzionale.

Per esempio, se sostituisci la tabella ogni settimana, allora puoi avere il numero di settimana salvato. Questo è necessario perché sostituire la tabella comporta l’analisi di tutti i record nella tabella, e potrebbe essere una lunga operazione eseguita a bassa priorità. Quindi se la tabella è segnata, a qualche punto avremo qualche valore ancora crittografato utilizzando la vecchia tabella, e dobbiamo saperlo, a meno che non blocchiamo l’intera tabella fino a che tutti i record siano sostituiti, ma potrebbe essere una brutta pratica, specialemente per grandi tabelle analizzate a bassa priorità.

3.4 Problemi con i dati in-memory

Sappiamo come spesso un intrusore è nient’altro che un osservatore di memoria, un programma che monitora la RAM; cercando dati può riconoscere e rubare, spesso conosciuti come *memory sniffers*

o *memory scrapers*. Contro questi attacchi, abbiamo il DB in memoria crittografata, ma ad un certo punto dobbiamo recuperare i dati per utilizzarli, e questo è il momento in cui diventiamo vulnerabili ancora. Questo accade sempre prima quando usiamo un NoSQL DB perché, come abbiamo visto, loro hanno solo crittografia a riposo.

È evidente che non possiamo usare i dati se non li abbiamo in modo leggibile, quindi ad un certo punto, saremo comunque vulnerabili se uno *sniffer* sta guardando i nostri processi, quindi il trucco è mantenere i dati in modo chiaro per il minor tempo possibile. Nel caso abbiamo bisogno di quei dati per più di un processo allora li dobbiamo tenere in memoria crittografati e decriptarli solo per i microsecondi necessari al nostro bisogno computazionale.

Al giorno d'oggi la tecnologia ci consente di criptare e decriptare al volo in un tempo insignificante per la prospettiva di un utente umano. Quindi questo è il miglior approccio che possiamo applicare utilizzando le librerie dello stato dell'arte della crittografia odierne.

3.5 Dati sensibili

I dati sensibili sono elencati nell'articolo 9(1):

"origine etnica, opinioni politiche,

credi filosofici o religiosi,

appartenenza sindacale, e il processamento di

dati generici, dati biometrici per

lo scopo di unicamente identificare una

persona fisica, dati riguardanti la salute

o dati riguardanti la vita sessuale o

l'orientamento sessuale di una persona fisica."

Quando abbiamo bisogno di processare questi tipi di dati dobbiamo assicurare una migliore e più sicura crittografia, quindi la coppia di chiavi RSA non deve essere minore di 2048 bit, e 512 bit per il blocco di cifre.

La cosa più importante riguardo i dati, e specialmente riguardo quelli sensibili, è di chiedersi: Abbiamo bisogno di processarli? Sono dati necessari per eseguire il servizio? Abbiamo un consenso esplicito per processare i dati sensibili del soggetto dei dati?

Ricorda che non possiamo collezionare dati con consensi generici o impliciti: dobbiamo assicurare che i soggetti dei dati siano completamente a conoscenza del fatto che stiamo processando i loro dati, e loro devono esplicitamente acconsentire al processamento utilizzando un modulo che spiega anche lo scopo ed il lasso di tempo del processamento dei dati. Inoltre, dobbiamo in modo sicuro salvare il consenso, ed in caso dobbiamo processare i dati per un periodo più lungo, un nuovo consenso è necessario, specialmente per i dati sensibili.

3.6 Generatori di numeri casuali reali

Ogni volta che menziono la crittografia, stiamo implicando la generazione di numeri casuali, e quelli devono essere crittograficamente sicuri.

Lo stato dell’arte per la generazione di numeri casuali sicuri per la crittografia sono schede hardware che utilizzano rumore impulsivo, effetti fotoelettrici ed altri sistemi basati su elettroni o fotoni.

Queste schede dovrebbero essere conformi con NIST SP800-90 B [3] e C (ancora in bozza) [4], come il Comscire PQ4000KS [5] o SwiftRNG Pro [6]. Dato il basso costo e l’alta disponibilità raggiunta da questi generatori di numeri casuali, non c’è nessuna giustificazione per evitare l’implementazione della tecnologia: possiamo sicuramente affermare che l’era dei numeri pseudo-casuali basati sugli algoritmi invece di hardware dedicato è finito, quindi dobbiamo adottare questa tecnologia per essere conformi con il GDPR.

3.7 Integrità e sicurezza del processamento

Quattro sono i principali elementi che devono essere considerati per garantire l’integrità dei dati e la sicurezza del loro processamento:

- Sempre implementare vincoli di integrità referenziale a livello del database
- Mai eliminare fisicamente alcun dato a meno che hanno raggiunto il limite del loro periodo di salvataggio: usare solo eliminazione logica (una colonna booleana is_Deleted)
- Definire una politica chiara di backup e rispettarla
- Usare Sviluppo guidato da test e avere una squadra di QA che lavora isolata dalla squadra di sviluppo

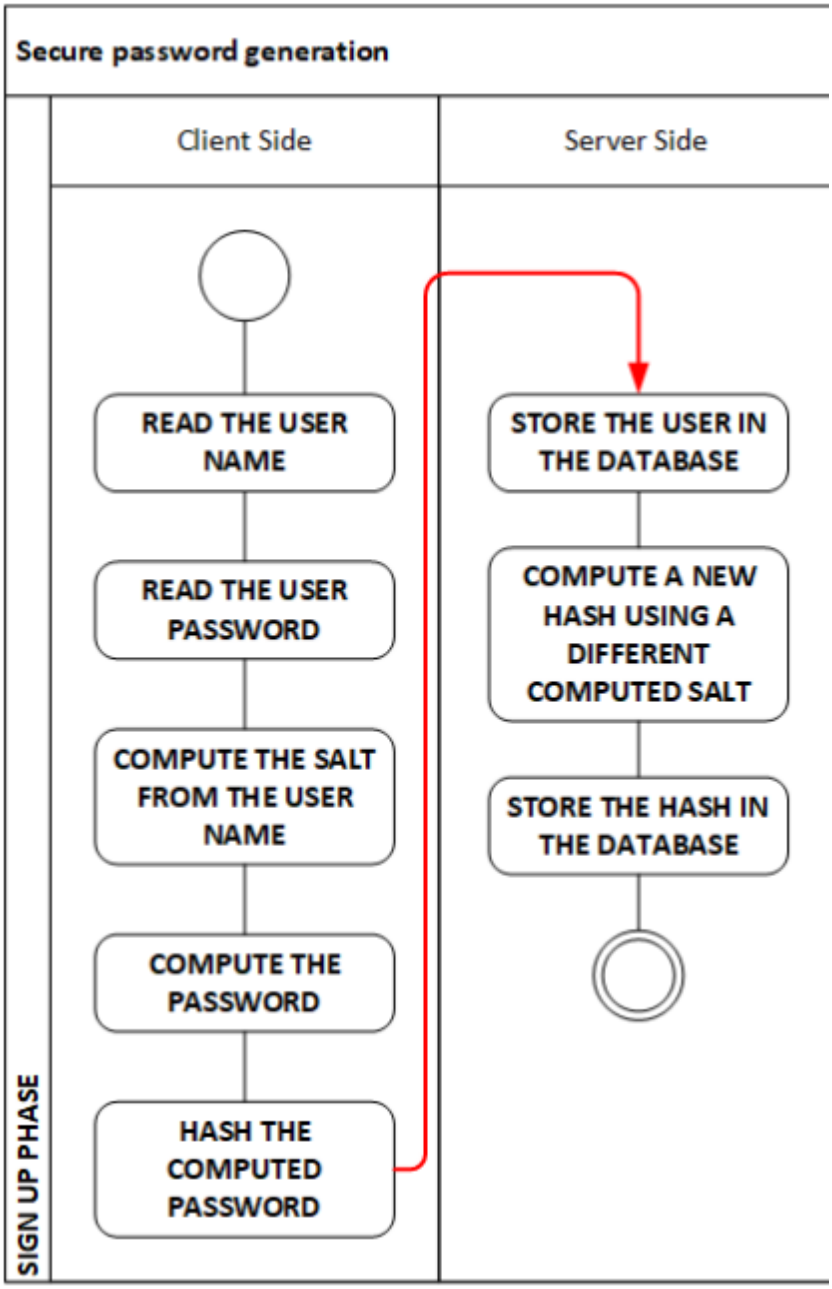
Nessuno dei quattro punti sopra descritti è più importante di un altro: tutti devono avere un impatto considerevole per essere in grado di garantire che il sistema che abbiamo progettato, disegnato ed implementato è robusto come il GDPR richiede.

Se in una corte di legge sarà provato che il prodotto ha violato l’integrità dei dati e non c'è stato uno standard QA durante tutte le fasi del ciclo di vita del software, sarà molto più probabile che la tua azienda venga trovata in violazione del GDPR.

3.8 Salvare password in modo sicuro

Le password sono state un problema serio e sempre sottostimato quando dovevano essere salvate in un modo sicuro. Molti utenti hanno l’abitudine di utilizzare la stesse password a rotazione su tutte le loro risorse protette da password, e questo è un comportamento incomprensibile perché altrimenti, l’utente medio finirebbe con una media di 20 password da gestire, e c’è ancora una piccola fiducia nel pubblico nei gestori di password sul mercato.

Data a questa abitudine, se una password viene violata, l’utente deve trattare con una violazione in molti servizi, non semplicemente quello che ha ricevuto direttamente la violazione. È nostra responsabilità di fare in modo che sia difficile violare le password, quindi non possiamo solo creare l’hash come è abitudine comune nella maggioranza delle architetture: ci sono troppi software eccellenti di brute force disponibili che risolvono gli hash, e perché la forza computazionale disponibile ad ogni hacker sta crescendo esponenzialmente, non possiamo lasciare le password esposte semplicemente salvando il loro hash con un algoritmo di hash; non contando le risorse di dizionari per brute force online come https://hashkiller.co.uk, https://crackstation.net e altri centinaia, quasi tutti disponibili online gratuitamente. Quindi, se dovessimo ricoprire il ruolo di provider di identità, dovendo archiviare le credenziali degli utenti? Sto offrendo qui



una soluzione semplice, un algoritmo che ho pubblicato due anni fa. L'hash della prima password avviene sempre sul lato client, quindi la password originale scelta dall'utente non è mai inviata nei cavi in modo chiaro. In un contesto di un processo di registrazione, noi abbiamo insieme il nome utente e la password; quindi nell'algoritmo, ho proposto di derivare il 'salt' dal nome utente in modo creativo, come, per esempio, il seguente (preso dal mio precedente articolo).

Immagina un Mr John Doe che si registra e l'email è il nome utente (per semplicità):

john.doe@somedomain.com

Inoltre, lui decide di usare una password

MyPassword

Il primo passo sul client dove il nostro nuovo utente si sta registrando è di processare il salt. Per farlo, notiamo il valore ASCII della prima lettera del nome utente:

j = 152

Essendo 152 un numero pari abbiamo arbitrariamente deciso di prendere dal nome utente tutti i caratteri posti nelle posizioni dispari, quindi abbiamo il nostro salt “crudo”(raw):

jh.o@oeoancm

Quindi, calcoliamo ora l’hash MD5 di questo salt per avere una migliore forma di esso per usarla:

c4211ead299f2bd80a3465ab9be18c05

Ora abbiamo “salato” la password, ottenendo il seguente

*MyPassword*c4211ead299f2bd80a3465ab9be18c05*

Aggiungo “*” in mezzo proprio come una complessità aggiunta e una migliore presentazione in questo esempio. Abbiamo ora un qualcosa di abbastanza complesso: traduciamo questo in una stringa codificata in Base64 dell’hash SHA256 di essa:

723b6f133818b87215e9f476350d06e55815c8de00fc

13af58aa94dee4c66398b441b2a1060dd1e2f7f82dd3a

b2a420ee4245b943fc7721ab59b765f84eefa26

Questa è una tipica stringa in Base64, con tutti i caratteri alfabetici in minuscolo. A questo punto facciamolo più interessante, quindi, ancora perché j è un carattere pari, decidiamo arbitrariamente che tutti i caratteri in posizioni dispari devono essere maiuscoli, ottenendo il seguente:

723b6f133818B87215E9F476350d06E55815C8De00Fc13A

f58Aa94DeE4C66398B441B2A1060dD1E2F7F82dD3Ab2

a420eE4245b943fC7721aB59b765f84EeFa26

Questo rappresenta la password attuale che il software del client invierà al server.

Come potete vedere dalla figura precedente, sul server facciamo qualcosa di simile, ma non identico, ma in un modo diversamente creativo, come prendere la undicesima lettera della password hashata e fare qualcosa basata su quella essendo pari o dispari in ASCII, o qualsiasi altra regola che ti viene in mente. Il miglior approccio è creare un unico algoritmo per la tua azienda così anche dando l’accesso al database sarà impossibile provare a computare in modo inverso il valore originale che l’utente ha scelto come password.

3.9 Informazioni sulle librerie software e l'accesso

Ho fatto riferimento ad algoritmi per la steganografia, crittografia in-memory e protezione di password, e finirete con molti altri prima che la vostra soluzione sia completa, e questo farà apparire un altro problema: la sicurezza degli algoritmi implementati.

Va da sé che le librerie che offrono le funzionalità di protezione dei dati devono essere fortemente offuscate, quindi il costo per smontarle potrebbe essere troppo alto per ogni hacker o organizzazione. Ma che dire della persona più pericolosa che possiamo affrontare? Ci si potrebbe

chiedere "Chi è quella persona?"; solo una è la peggiore riguardo la pericolosità: l’impiegato scontento.

Diciamo che Mr. Doodle è un capo analista nel nostro dipartimento di analisi dei dati e lui viene licenziato causa riduzione della forza lavoro. Lui è arrabbiato e ha accesso ai dati, quindi lui scarica tutti i database che amministriamo su Torrent. Bene, perché noi abbiamo implementato le misure descritte sopra, da una prospettiva di protezione dei dati non dobbiamo preoccuparci molto perché non incorreremo in conseguenze nel rispetto del GDPR: tutti i dati sono stenograficamente protetti e nessuna password è salvata in chiaro da nessuna parte. Così nessuno può essere identificato. Bene, questo è vero se il nostro Mr Doodle non ha accesso agli algoritmi usati e al codice sorgente della loro implementazione. Altrimenti, lui può metterli su Torrent e dopo noi saremo in guai seri.

È veramente importante che prendiamo tutte le precauzioni rilevanti per assicurare che nessuna singola persona ha tutta la conoscenza degli algoritmi della protezione dei dati, né accesso alla completa implementazione del codice sorgente. Questo implica che lo sviluppo delle implementazioni deve avvenire atomicamente, avendo squadre differenti che sviluppano parti differenti di esse, sfruttando le funzionalità offerte da altre librerie di offuscamento, quindi sarebbe impossibile capire cosa il codice della protezione dei dati stia facendo. Questo significa che in caso di violazione degli algoritmi solamente l’architetto che li ha progettati o il custode dei file potrebbe essere la sorgente della violazione, rendendo semplice identificare il colpevole in tali casi.

Qui sono sicuro che voi staiate pensando “Non è troppo?” e ancora vi invito ad immaginarvi in piedi in una corte di legge in cui vi chiedono “Hai o non hai implementato lo stato dell'arte nella protezione dei dati? Potresti o non potresti, secondo la tua opinione professionale, fare di più per prevenire la perdita di dati?”.

Impostare il processo e i livelli di accesso corretti è una cosa semplice; quindi, non è giustificabile non mettere queste misure di sicurezza in campo in ogni azienda dalle medie aziende alle più grandi. Una piccola compagnia dovrebbe prenderle seriamente, ma adattarle ad una ragionevole infrastruttura considerando i mezzi disponibili.

3.10 Cancellazione dei dati (diritto all'oblio)

Un altro problema tecnico viene dal diritto all’oblio stabilito nell’articolo 17 del GDPR. Questo è un problema che non ha una soluzione semplice perché dobbiamo avere un piano di backup dei dati implementato per essere conformi con l’articolo 32 del GDPR, quindi quando un soggetto dei dati richiede di essere dimenticato (completa cancellazione dei suoi dati personali) abbiamo un problema.

La soluzione, in questo caso, non può essere tecnica, perché il costo di una cancellazione sicura dei dati è molto alta. Il GDPR ammette eccezioni per il tempo che l’utente ha acconsentito a quelli in un primo tempo, quindi, mia opinione, quando collezioniamo il consenso del soggetto dei dati, dobbiamo avere una clausola chiara, esplicitamente accettata, dove chiariamo che in caso di consenso alla revoca dell'esercizio del diritto all'oblio, i dati sul backup resteranno e saranno distrutti quando il tempo limite sarà raggiunto.

È necessario un processo chiaro nel caso in cui i dati vengano ripristinati per garantire che nessuno di questi dati venga ripristinato. È ragionevole ripristinare completamente un backup e disporre di un processo batch immediatamente dopo l'esecuzione dell'operazione di ripristino per eliminare i dati che devono essere cancellati.

So che questa non è una soluzione pulita, ma l'alternativa sarebbe processare tutti i backup che abbiamo nell'ambiente, ripristinarli, dopodiché eliminarli e fare un backup ancora e distruggere l'originale: il costo di questa operazione sarebbe inimmaginabile in alcuni casi.

3.11 Diritto alla portabilità dei dati

Il GDPR stabilisce per i soggetti dei dati il diritto ad avere i loro dati esportati in uno standard portabile (Articolo 20). Va ancora più lontano, stabilendo che, laddove tecnicamente possibile, se l'interessato ha bisogno che i dati vengano trasferiti ad un altro responsabile del trattamento, tale processo deve essere eseguito (articolo 20, paragrafo 2).

In parole povere, questo significa che se ho un account Google Plus e voglio che tutto sia mosso su Facebook, dove Facebook ha un endpoint per consentire così un trasferimento dei dati, Google sarà costretto dal GDPR ad eseguire il trasferimento di tutti i dati su Facebook.

Questo non presenta nessun problema tecnico, ma, poiché tutti i dati sono criptati, dobbiamo mettere in campo un processo per permettere l'esercizio del diritto di portabilità dei dati, e deve essere messo in atto prima che nessun utente eserciti questo diritto, altrimenti il tempo di esecuzione rischia di andare oltre ogni ragionevole ritardo e il soggetto dei dati avrà diritto ad un compenso mentre è probabile che la nostra azienda venga multata per non aver stabilito un mezzo adeguato per esportare i dati in modo tempestivo in conformità con il GDPR.

4. Client e Server

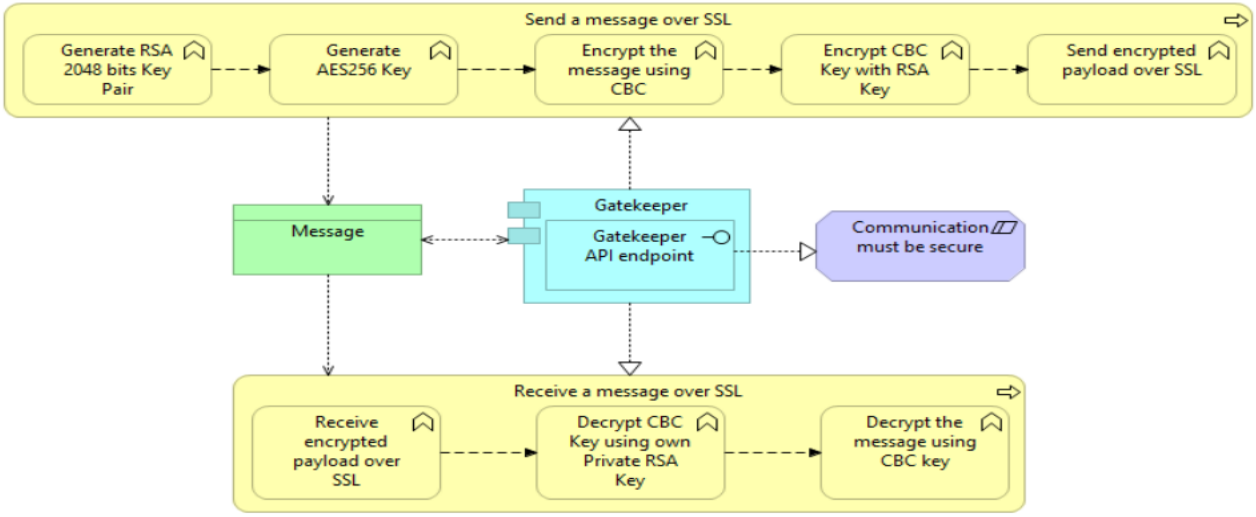


Figure 2 – Enhanced Gatekeeper implementation

La maggior parte, se non tutti, dei processamenti dei dati coinvolgono due livelli: un client, usualmente che opera su un dispositivo dell'operatore, abilitando l'operatore ad interrogare, aggiungere, modificare, o eliminare dati; e un server, dove i dati sono salvati e dove la maggior parte della computazione viene eseguita.

Alcune volte ci sono molteplici server, dove di solito il server che interfaccia il client umano è, a sua volta, un client di un altro server. Tutto questo parlare tra le macchine ha una maggiore implicazione: i dati vengono scambiati su una rete, la quale, per sua natura, non è quasi mai sicura.

4.1 Un gatekeeper potenziato

Come ho menzionato nell'introduzione, in questo paper non starò a contatto con il problema di rendere sicura un'infrastruttura IT, quindi andiamo un pò più su con i livelli logici, fino al livello Applicativo, dove illustrerò come rendere sicuri lo scambio dei dati.

Il primo passo è ovvio, disegnare l'applicazione in modo da sfruttare il Transport Layer Security (TLS, ex SSL, Secure Socket Layer), quindi assumiamo che la nostra architettura Client-Server è basata su un approccio RESTful, non su una connessione di livello di trasporto, altrimenti dovremmo reinventare la ruota, e questo non ci piace farlo, specialmente dovuto ai costi coinvolti nello stabilire e mantenere un protocollo di sicurezza customizzato basato sullo scambio crudo di pacchetti TCP / IP.

Nella figura sopra possiamo vedere il diagramma di processo della mia implementazione (la quale ho messo a dominio pubblico qualche anno fa) del Pattern di progettazione del famoso Gatekeeper. Nella mia visione il Gatekeeper ha un solo REST endpoint, che aspetta un messaggio POST strutturato come segue:

Campo	**Valore**
Chiave pubblica del mittente	La chiave pubblica del mittente che il destinatario userà per criptare la chiave CBC per criptare la risposta.
Chiave CBC criptata	Questa è una chiave SHA256 usata per criptare il corpo del messaggio. Questa chiave è criptata utilizzando la chiave pubblica RSA dell'altra parte.
Corpo	Questo contiene il carico attuale (chiamata API e corpo) criptato con la chiave CBC.

La chiave CBC (Catena di blocchi di cifratura) SHA256 è generata nuovamente ogni volta che mandiamo un messaggio: mai riciclare questa chiave è fondamentale per proteggere i dati trasferiti tra client e server.

Due sono le assunzioni che ho fatto in questo algoritmo:

- La chiave pubblica RSA del server può essere ottenuta da una posizione nota al client: questo permette di rigenerare questa chiave ogni giorno
- Il client genererà una nuova coppia di chiavi RSA ogni volta che viene eseguito

Il corpo è la chiamata API effettiva che il Gatekeeper trasmetterà, fungendo da proxy. La struttura usuale del corpo criptato è una struttura JSON che porta quattro campi, rappresenta la chiamata API da essere inoltrata al Gatekeeper, come mostra la seguente tabella

Campo	**Valore**
API query	La stringa legale di interrogazione, qualcosa tipo /api/user&param=xyz
Corpo	L'eventuale carico da passare all'endpoint API
Metodo	Il metodo HTTP da usare (GET, POST...)
Headers	Il valore degli headers per l'endpoint, specialmente necessari per i servizi di autenticazione

Il Gatekeeper aggiungerà l'URL base, prepara un nuovo messaggio usando le informazione dell'header e del corpo provviste e effettua la chiamata per conto del client sul server API interno.

Naturalmente, il Gatekeeper farà affidamento sul risultato al client usando lo stesso approccio. Quindi cripterà la risposta cruda dal server API interno usando una nuova chiave CBC, e la chiave sarà criptata usando la chiave pubblica RSA.

Questo approccio garantirà lo stato dell'arte nella protezione dei dati, almeno mentre sto scrivendo questo paper.

Naturalmente il Gatekeeper può essere esteso anche per giocare nel ruolo di API Gateway semplicemente implementando un insieme di Access Control List (ACL) alle API interne disponibili, quindi il Gatekeeper garantirà o rifiuterà l'accesso basandosi sui ruoli del suddetto ACL.

4.2 Mai affidarsi al client

Siamo tutti stati allenati a ripetere il mantra "Mai affidarsi al client", ancora è comunque bene toccare questo argomento, specialmente nel contesto di responsabilità stabilita dal GDPR.

Dobbiamo ricordare che nella nostra progettazione ogni client può essere violato, un dispositivo client può essere perso o rubato, quindi dobbiamo progettare la nostra soluzione assumendo che i nostri client cadranno in mano ad un attaccante.

La conseguenza di questo è che la logica di business sarà eseguita su un client, e specialmente nessun dato sarà salvato sul dispositivo client. Dobbiamo anche porre attenzione a criptare sempre le stringhe in-memory ed anche progettare l'interfaccia utente in un modo che in ogni schermo ci siano minor dati identificativi possibili e il modo di visualizzarli dovrebbe essere semplice da capire solo per un umano, questo in previsione del fatto che il dispositivo sia stato contaminato da uno screen-scraper, un software dannoso che cattura schermate e cerca di estrarre informazioni significative da ciò. La protezione da screen-scraper può essere implementata progettando l'interfaccia utente usando font non comuni, variabili per ogni tipo di dato, quindi sarà facile e significativo all'utente ma avrà poco senso per un programma provare a decodificare il contenuto dello schermo.

Un'altra misura, opzionale ma veramente efficace, che vale la pena di implementare nel caso in cui il client invii dati sensibili, è di assicurare l'uso di schermi con un basso angolo di visione e sfruttare la distorsione dei colori LCD come suggerito da Harrison e Hudson nel loro paper [7], quindi per camuffare il contenuto dello schermo quando visualizzato da ognuno che non sia di fronte allo schermo.

5. Conclusioni

In questo paper, ho esplorato la portata dell'impatto del GDPR sull'architettura del software e ho proposto soluzioni pratiche per garantire la piena conformità al regolamento. Le soluzioni proposte sono state verificate nelle applicazioni del mondo reale. Perciò rappresentano un approccio pragmatico per implementare la conformità. Poiché il legislatore nello scrivere il GDPR ha usato una forma che considerava il ritmo dell'evoluzione tecnologica, dobbiamo stabilire un protocollo per delle revisioni periodiche delle nostre soluzioni per assicurare che loro siano ancora conformi. Le aziende dovrebbero riportare questi protocolli nei loro processi di certificazioni ISO e nei loro budget di mantenimento software.

Riferimenti

[1] European Union (2016). Regulation (EU) 2016/679. Brussels: European Union, pp.1-88. (URL link http://bit.ly/GDPR_Pdf).

[2] A. Cavoukian, “Privacy by Design - The 7 Foundational Principles,” Internet Architecture Board, 02-Nov-2010. [Online]. Available: http://bit.ly/AnnCavoukianPhDPbDD. [Accessed: 05-Feb-2018].

[3] M. S. Turan, E. Barker, J. Kelsey, K. McKay, M. Baish, M. Boyle, NIST, and NSA, “SP 800-90B, Recommendation for the Entropy Sources Used for Random Bit Generation,” SP 800-90B, Entropy Sources Used for Random Bit Generation | CSRC. [Online]. Available: http://bit.ly/SP800-90B. [Accessed: 17-Feb-2018].

[4] E. Barker, J. Kelsey, and NIST, “SP 800-90C (DRAFT), Recommendation for Random Bit Generator (RBG) Constructions,” SP 800-90C (DRAFT), Recommendation for RBG Constructions | CSRC. [Online]. Available: http://bit.ly/SP800-90C-DRAFT. [Accessed: 02-Apr-2018].

[5] “PureQuantum® Model PQ4000KS,” ComScire. [Online]. Available: https://comscire.com/product/pq4000ks/. [Accessed: 17-Jan-2018].

[6] “SwiftRNG Pro - TectroLabs,” - TectroLabs. [Online]. Available: https://tectrolabs.com/swiftrng-pro/. [Accessed: 17-Feb-2018].

[7] C. Harrison and S. E. Hudson, “A New Angle on Cheap LCDs: Making Positive Use of Optical Distortion,” Chris Harrison | A New Angle on Cheap LCDs. [Online]. Available: http://chrisharrison.net/index.php/Research/ObliqueLCD. [Accessed: 14-Apr-2018].