
Security-by-design

Chi opera nel settore IT in qualsiasi azienda ha ormai sentito parlare di «security by design»

Se finora qualcuno l'ha considerata una semplice locuzione per sottolineare l'importanza di prendere in considerazione gli aspetti di cyber security nella gestione delle infrastrutture digitali, si è sbagliato di grosso.

Il concetto di security by design è qualcosa che va ben oltre una semplice dichiarazione di principio e rappresenta una vera rivoluzione sia a livello di processi, sia a livello di gestione della Information Technology.

Nei processi di sviluppo e progettazione, la sicurezza informatica è sempre stata considerata come una sorta di «brutto anatroccolo».

La verifica di eventuali problemi di sicurezza sono sempre stati relegati al termine dei processi e nella migliore delle ipotesi la verifica veniva fatta al termine della progettazione o sviluppo.

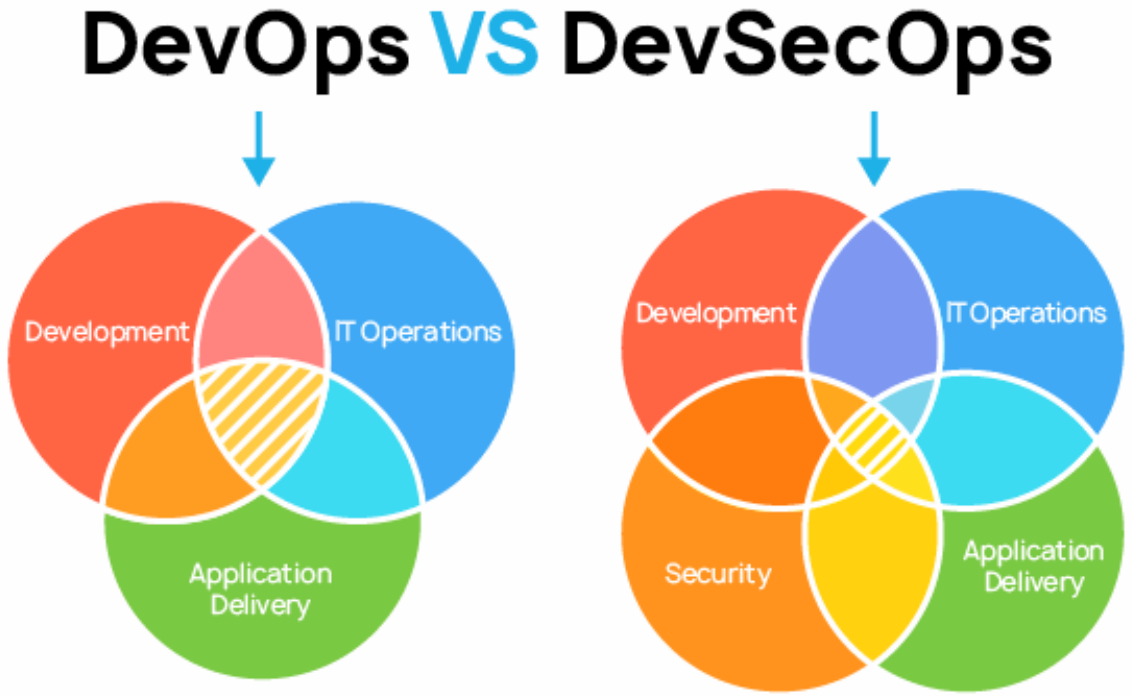
Nella peggiore, si lasciava al tempo il compito di far emergere eventuali problematiche.

# «I risultati ?»

Trascurare la sicurezza in fase di sviluppo e progettazione, oltre ad aprire vere praterie agli attacchi dei pirati informatici ha innescato una continua rincorsa ad aggiornamenti e modifiche di software e dispositivi, con il continuo rischio di emersione di incompatibilità e una crescita esponenziale dei costi.

Che si tratti del firmware di un dispositivo IoT o di un software dedicato all'erogazione di servizi aziendali, quello che viene richiesto oggi è di applicare la logica della security by design attraverso l'adozione di un processo definito DevSecOps

Neologismo che vuole indicare come, oltre agli sviluppatori (Dev) e a chi si occupa della parte di operations (Ops) sia necessario coinvolgere da subito anche i responsabili dei test di sicurezza (Sec).



La procedura, in pratica, prevede che i controlli di cyber security vengano effettuati già in corso di sviluppo del software, attraverso un continuo confronto tra sviluppatori e analisti.

L'obiettivo è quello di raggiungere un migliore livello di sicurezza e ottimizzare il processo.

In questa declinazione, anche se non siamo di fronte a una vera e propria fase di sviluppo, la collaborazione con gli esperti di sicurezza consente di adottare da subito una strategia orientata alla protezione dei dati e all'integrità dei servizi.

I vantaggi che ne derivano sono enormi.

Oltre a consentire di implementare strumenti di protezione più efficaci, l'adozione della filosofia security by design abbatte i costi per la sicurezza e riduce drasticamente il rischio di incidenti di security.

Più in generale, però, il concetto di security by design si può applicare a qualsiasi ambito IT, per esempio (e soprattutto) quando si affronta la pianificazione di reti e servizi digitali.

Alla luce delle premesse sin qui fatte, con l'espressione security by design si intende, un approccio per lo sviluppo software e hardware che cerca di mettere i sistemi in sicurezza rispetto ad attacchi e vulnerabilità impreviste, attraverso misure come il monitoraggio continuo, l'utilizzo di credenziali e l'aderenza a pratiche di programmazione migliori.

Si tratta di un approccio rivoluzionario, che capovolge completamente il punto di vista con cui si affronta un nuovo progetto.

Se di solito la sicurezza è un problema che si valuta a lavoro finito, con questa metodologia la questione security viene invece presa in considerazione subito, sin dalle fondamenta del progetto.

Questo cambio di prospettiva è fondamentale, poiché permette di mettere in campo strategie più efficaci, sviluppate in fase di progettazione, permettendo quindi di identificare le soluzioni migliori, adattando, se necessario, lo sviluppo ai parametri di sicurezza.

Seguendo il concetto di security by design, ogni programmatore e sviluppatore deve sempre tenere conto di determinati elementi o parametri durante ogni fase di scrittura del codice, da quella di progettazione a quella di testing.

Lo sviluppo software e hardware deve avere quindi come obiettivo la prevenzione delle vulnerabilità, possibile attraverso il costante monitoraggio e una programmazione efficace e attenta alla leggibilità.

Adottando questi accorgimenti si possono avere servizi, programmi, applicazioni e prodotti affidabili e progettati appositamente per essere sicuri.

Si possono prevedere attacchi e minacce fin dalle fondamenta del progetto e attuare scelte strategiche che garantiscono una base solida al sistema informatico.

Un esempio di security by design è la tecnica, oggi molto diffusa, di creare linguaggi formali specifici per determinate proprietà di un software.

In questo caso le specifiche vengono confrontate automaticamente con il codice che si sta scrivendo, assicurando così un'implementazione corretta e funzionale.

Un espediente importante per evitare falle e malfunzionamenti e creare un sistema di qualità che sia meno soggetto alle vulnerabilità.

# I vantaggi della security by design

La security by design è un vantaggio economico molto sostanzioso, capace di ottimizzare i costi e ridurre le spese dovute a sistemi difensivi retroattivi, oggi estremamente costosi.

In più è un ottimo modo per minimizzare i danni finanziari e d'immagine dovuti a un attacco, limitando il grosso impatto negativo che possono avere su un'azienda.

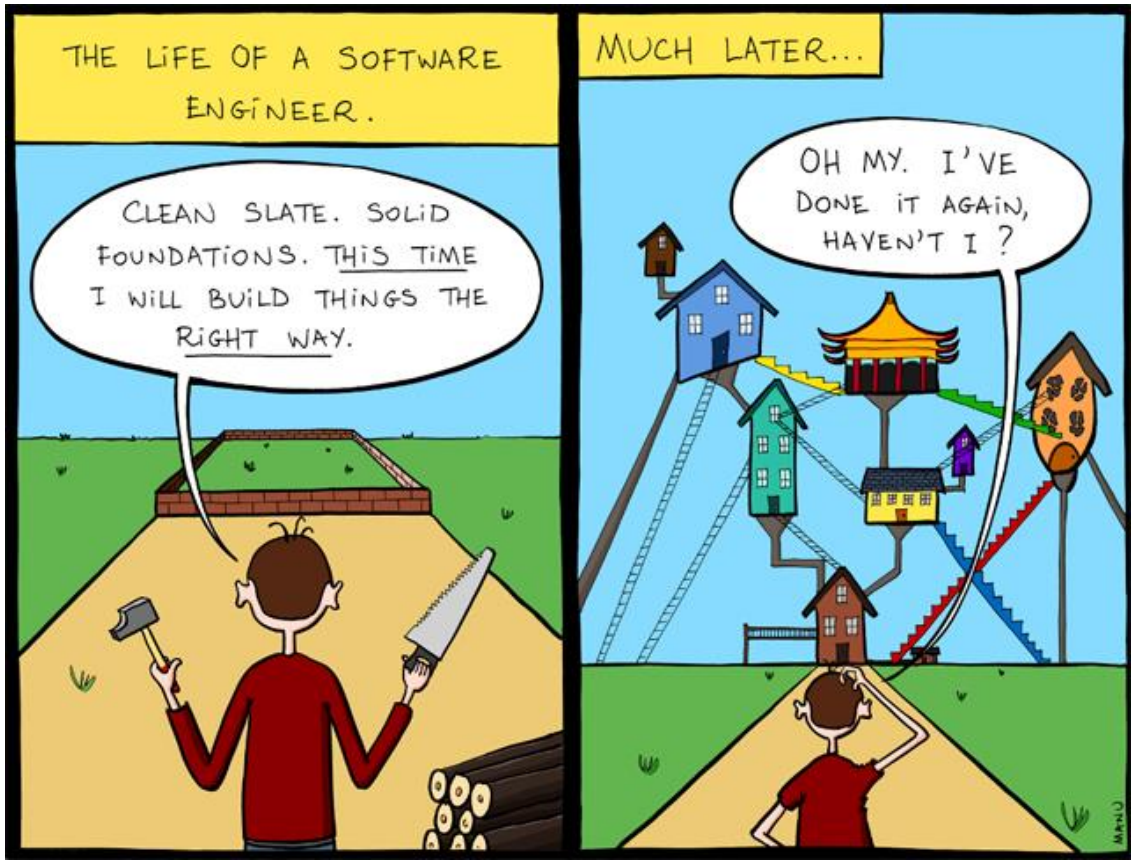
Un altro aspetto conveniente è la certezza, fin da subito, di affidarsi a un codice pulito, funzionale e di qualità: una sicurezza da non sottovalutare considerando quanto sia difficile rimediare alle vulnerabilità individuate dopo lo sviluppo del software.

# I vantaggi della security by design

Correggere gli errori richiede fatica, importanti operazioni di refactoring e tanto tempo a disposizione.

In più, se l'errore è strutturale, può risultare quasi impossibile rimediare e ci si trova obbligati a ritirare il prodotto.

Infine è importante ricordare che la security by design non si limita al settore IT, ma è applicabile in qualsiasi ambito, coinvolgendo i processi e le policy aziendali.



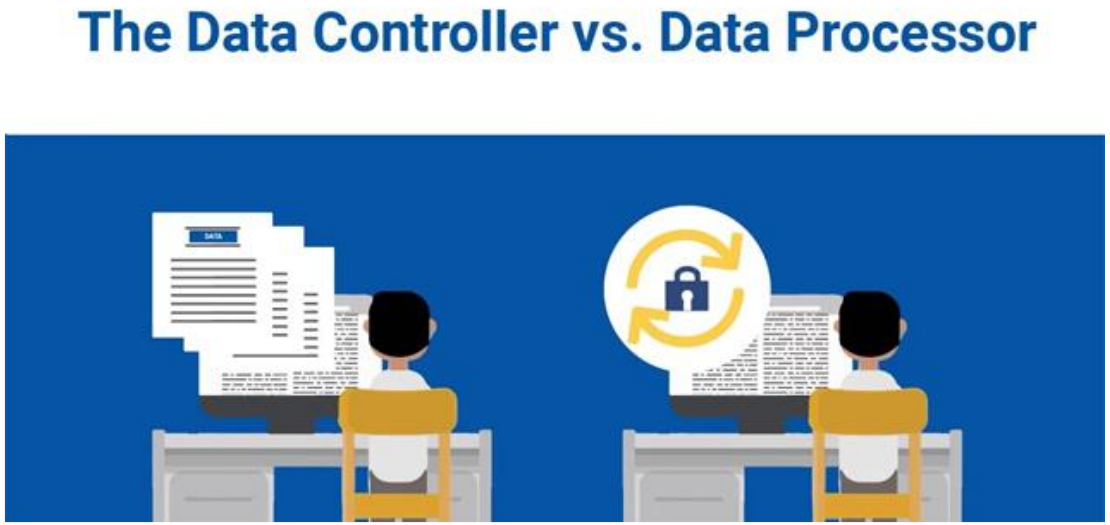
# La security by design nel GDPR

Recentemente, il concetto di sicurezza by design ha ricevuto un'importante spinta dall'introduzione del GDPR, il regolamento europeo sulla protezione dei dati personali in vigore dal 25 maggio 2018.

Il regolamento ha radicalmente rovesciato la filosofia delle precedenti norme in materia di Privacy.

Ha trasferito di fatto la disciplina del trattamento dati da un sistema normativo di tipo formalistico (basato sulla previsione di regole formali e su un elenco di adempimenti e misure minime di sicurezza da adottare), ad un sistema basato su un'alta responsabilizzazione sostanziale del Data Controller.

A quest'ultimo, in particolare, è richiesta (tra le altre cose) la capacità di prevenire (e non solo di correggere) gli errori (il cosiddetto principio di accountability).



# La security by design nel GDPR

Il GDPR ha inoltre introdotto un nuovo approccio metodologico completamente risk-based che considera la tutela dei dati personali un presupposto da considerare già nella fase di progettazione dei processi di trattamento degli stessi, dei servizi e dei prodotti.

Nell'art. 5 del GDPR, infatti, non solo si individua nel Titolare del trattamento il soggetto responsabile di garantire il rispetto dei principi applicabili al trattamento di dati personali ma si stabilisce che il medesimo debba essere altresì "in grado di comprovarlo".

In pratica dovrà essere in grado di:

- documentare il processo che ha portato alla definizione del registro dei trattamenti
- valutare un determinato rischio in materia di sicurezza
- decidere di notificare o meno agli interessati una violazione dei dati personali
- aver attuato in relazione ad un nuovo trattamento le necessarie valutazioni legate alla privacy «by design»

Con l'entrata in vigore del GDPR oltre al concetto di privacy by de, vengono introdotti nuovi principi e concetti in materia di tutela dei dati personali, in particolar modo, l'articolo 25 introduce i due principi di privacy:

- Privacy by Default
- Privacy by Design

Con privacy by default si intende indicare che la protezione dei dati personali deve avvenire di "default" cioè nei processi di raccolta e trattamento dati.

Devono sempre essere applicate delle procedure interne che ne regolino le modalità di acquisizione e gestione, in modo da garantirne sempre il rispetto della normativa sulla privacy, riducendone a priori il rischio di diffusione o trattamento illecito.

Con il termine privacy by design si intende invece

- la necessità di prevedere già in fase di progettazione dei sistemi informatici e delle procedure aziendali
- la necessità di tutelare i dati sensibili che vengono trattati, andando a ridurre a quelli effettivamente necessari
- la raccolta e l'utilizzo di dati personali
- Introduzione di sistemi di anonimizzazione dei dati, riducendo così i rischi legati a tale trattamento

Il concetto di privacy by design richiama l'articolo 3 del principio di necessità del codice della Privacy, che recita:

*"I sistemi informativi e i programmi informatici sono configurati riducendo al minimo l'utilizzazione di dati personali e di dati identificativi, in modo da escluderne il trattamento quando le finalità perseguite nei singoli casi possono essere realizzate mediante, rispettivamente, dati anonimi od opportune modalità che permettano di identificare l'interessato solo in caso di necessità*".

# Come funziona?

Per programmare in modo sicuro, si deve adottare un approccio sistematico basato su controlli, verifiche e audit.

La priorità è quindi ottenere un software che risulti sicuro già in fase di progettazione, piuttosto che affrontare le minacce al termine del processo in maniera retrospettiva.

Come si raggiunge questo obiettivo?

Innanzitutto bisogna valutare quali sono le vulnerabilità e i rischi a cui si può andare incontro realizzando un software.

L'analisi viene svolta a livello architetturale e implementativo e si adattano gli asset coinvolti alle possibili minacce.

# Come funziona?

Anche durante la produzione e la scrittura del codice si ipotizzano tutte le eventuali conseguenze e si predispongono delle contromisure di tipo tecnologico, organizzativo e procedurale.

Infine si eseguono tutti i security test necessari per valutare il livello di cyber security generale.

I test più frequenti sono il Penetration test, il Vulnerability Assessment e il Binary Code Review, tutti eseguiti in modalità black box testing e white box testing.

# black box testing

Il test black-box è un metodo di test del software che esamina la funzionalità di un'applicazione senza scrutare le sue strutture interne o il suo funzionamento.

Questo metodo di test può essere applicato virtualmente a ogni livello di test del software: unità , integrazione , sistema e accettazione.

A volte viene definito test basato sulle specifiche.

Non sono richieste conoscenze specifiche del codice dell'applicazione, della struttura interna e della programmazione in generale.

Il tester è a conoscenza di ciò che il software dovrebbe fare ma non è a conoscenza di come lo fa; ad esempio, il tester è consapevole che un particolare input restituisce un determinato output invariabile, ma non è a conoscenza di come il software produce l'output in primo luogo.

# white box testing

Il test strutturale, detto anche white box o verifica strutturale, è un particolare tipo di test che viene effettuato per rilevare errori in uno o più componenti (parte di codice, metodo, funzione, classe, programmi, ecc.) di un sistema software.

Il suo funzionamento si basa su alcuni criteri che hanno lo scopo di trovare dati di test che consentano di percorrere tutto il programma.

Per trovare un errore nel codice, infatti, bisogna usare dei dati che "percorrono" la parte erronea del programma.

Per testare una parte di programma si introduce il concetto di cammino: una sequenza di istruzioni attraversata durante un'esecuzione.

Naturalmente non esiste un criterio in grado di testare ogni singolo cammino dato l'elevato numero di questi ultimi (soprattutto in presenza di cicli), tuttavia, è possibile trovare un numero finito di cammini indipendenti che combinati tra loro forniscano tutti (o per lo meno la maggior parte) i restanti cammini.

Se eseguito correttamente e senza particolari eccezioni, il test può coprire fino al 90% delle istruzioni.

A differenza della metodologia black box, per cui non si ha a disposizione il codice, alla base della metodologia white box sta proprio l'analisi del codice

Ciò permette di identificare i cammini da percorrere e viene effettuata anche in modo manuale, con l'uso di diagrammi di flusso che mostrano i diversi cammini del programma e delle tabelle di traccia, che simulano l'esecuzione del codice.

# Dove si usa la security by design?

La security by design è applicabile con successo i diversi settori.

Con l'avvento dell'industria 4.0, tutto il settore secondario si è evoluto e informatizzato, incorporando macchinari e componenti connessi tra loro.

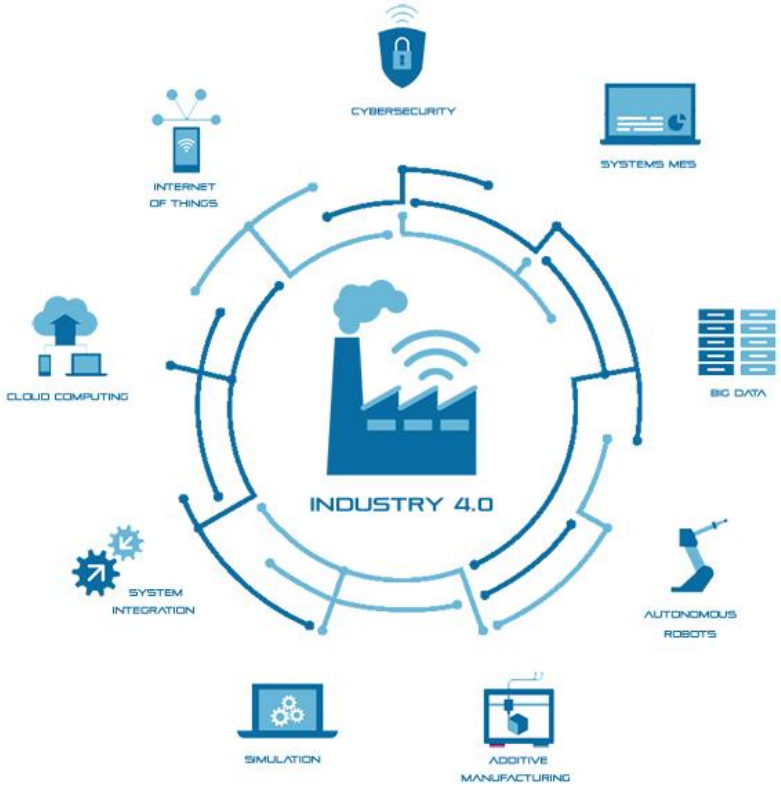
Chi progetta queste macchine punta di solito a progettare attrezzature che siano efficienti, solide e abbiano un buon rapporto qualità-prezzo.
Tralasciando gli aspetti legati alla cyber security.

È una svista che ci si può ancora permettere? La risposta è no, perché i rischi sono molto alti e i danni – economici e di immagine – possono avere conseguenze gravi.

# Dove si usa la security by design?

Per questo è utile seguire un approccio di security by design e progettare macchinari attenti alla sicurezza e conformi a certificazioni e caratteristiche specifiche.

Inoltre è importante adottare accorgimenti che agiscano su una realtà industriale nel suo insieme, senza intervenire su elementi singoli rischiando interruzioni di **business continuity**.



# Dove si usa la security by design?

Anche per le moderne architetture utili all'implementazione di soluzioni IoT (Internet of Things) la security by design, a maggior ragione, risulta essere l'approccio più corretto per mantenere in sicurezza l'intera infrastruttura, di solito molto complessa.

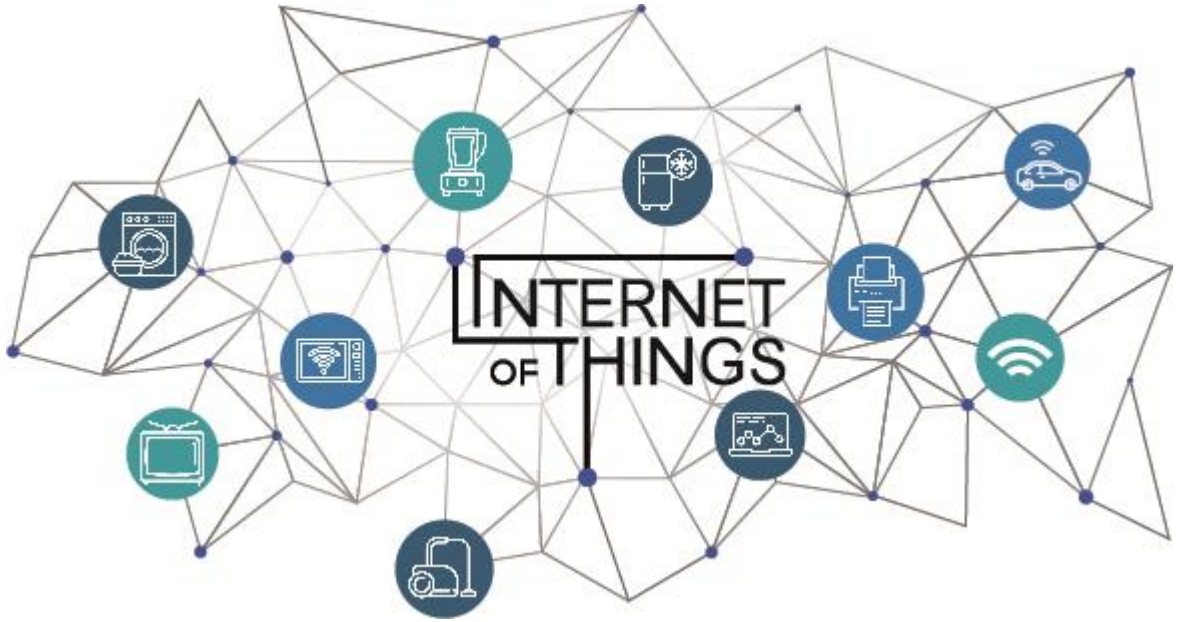
L'Internet of Things – o IoT – significa letteralmente "Internet delle cose" e indica una serie di oggetti considerati intelligenti perché connessi tra loro e capaci di scambiarsi informazioni.

Si parla quindi di smartphone, smart watch, smart car e molti altri.

# Dove si usa la security by design?

Pensando a quanto siano diffusi questi oggetti, è facile capire che i dati personali e sensibili veicolati in questa rete sono moltissimi ed è evidente quanto sia importante garantire sicurezza e massima protezione.

E' fondamentale progettare dispositivi che puntino alla sicurezza già in fase di sviluppo, affidandosi a programmatori che sappiano salvaguardare la privacy degli utenti finali e prevedere e anticipare le richieste dei consumatori.



# Strumenti e metodologie per lo sviluppo sicuro del software

Già da alcuni anni il termine security-by-design si è diffuso nella comunità degli sviluppatori per indicare il fatto che, oltre ai requisiti funzionali, la progettazione e lo sviluppo del codice deve tenere anche in considerazione la sicurezza.

La ricerca scientifica e quella industriale hanno prodotto negli ultimi decenni notevoli risultati in questa direzione.

Gli strumenti a disposizione degli sviluppatori sono molteplici e si applicano alle diverse fasi dello sviluppo, dalla progettazione fino al testing.

Nonostante questo, molti sistemi vengono compromessi con attacchi che sfruttano (anche e soprattutto) vulnerabilità del software.

La scarsa adozione degli strumenti che potrebbero incrementare la qualità del codice, però, non sempre è dovuta a ignoranza o pigrizia.

Comprendere le ragioni di questo fenomeno è un passaggio fondamentale per affrontarlo nel modo corretto.

Alcune delle tecniche che fino a qualche anno fa erano note solo nella comunità scientifica sono oggi diventate parte integrante dei principali linguaggi e sistemi per lo sviluppo del software.

Un caso interessante è rappresentato dai linguaggi formali per la specifica delle proprietà del software.

Questi linguaggi permettono di comparare (semi-)automaticamente le specifiche, tipicamente prodotte in fase di progettazione, con il codice sviluppato in fase di realizzazione. I benefici in termini di sicurezza sono significativi.

È noto infatti che molte falle derivano da implementazioni scorrette, cioè non rispettose delle specifiche.

Esistono ambienti per la scrittura e la verifica di specifiche formali per molti dei principali linguaggi di programmazione, come ad esempio C e Java.

Alcuni di questi, per esempio JML e Frama-C, sono strumenti realizzati e manutenuti da comunità di ricercatori e sviluppatori.

Altri sono invece veri e propri prodotti sviluppati da importanti attori industriali, come ad esempio Spec# di Microsoft.

Proprio Microsoft ha investito risorse significative per la creazione di strumenti di verifica che aiutino gli sviluppatori a produrre codice di maggiore qualità e, quindi, più sicuro.

Gli stessi linguaggi di programmazione sono stati influenzati significativamente dalle ricerche sulla sicurezza del software.

Un esempio interessante è quello dei permessi.

Storicamente, i permessi sono nati come meccanismo di gestione delle autorizzazioni per l'accesso dei programmi alle risorse sotto il controllo del sistema operativo, ad esempio i file.

Nel tempo i permessi, insieme alla loro gestione, sono diventati sempre più complessi e raffinati.

Ne ha esperienza chiunque usi uno smartphone.

È infatti comune, dopo aver installato una nuova app, ricevere un avviso su quali permessi l'applicazione richiede per essere eseguita.

È facile notare che i permessi riguardano molti aspetti, dall'uso della fotocamera fino alla possibilità di effettuare pagamenti online.

La richiesta di questi permessi è parte integrante del codice dell'applicazione e gli sviluppatori devono necessariamente gestire la logica di queste autorizzazioni come parte del proprio software.

Di conseguenza, tramite il controllo del sistema operativo, gli abusi nell'utilizzo dei permessi vengono individuati e impediti prima che si verifichino.

Queste premesse delineano uno scenario sostanzialmente ottimistico: gli strumenti per lo sviluppo del software sicuro ci sono e altri arriveranno in futuro.

Tuttavia, lo stato della sicurezza dei sistemi e il numero di attacchi di cui si ha notizia non sembrano confermare questa visione.

Il numero di violazioni dei sistemi informatici è in aumento e attaccanti con sufficienti risorse e motivazioni possono minacciare molti sistemi, anche critici per i cittadini.

Anche se spesso ci immaginiamo hacker dotati di conoscenze e abilità fuori dal comune, comunemente gli attacchi sfruttano vulnerabilità note, anche da parecchio tempo, che sono disseminate nel codice sorgente di scarsa qualità.

Questo fenomeno è dovuto a molteplici fattori.

# L'obsolescenza dei processi alla base del software difettoso

Le nuove tecnologie devono essere integrate nei processi per lo sviluppo del software.

Questi processi sono spesso peculiari di ogni azienda e, in alcuni casi sono andati consolidandosi nel tempo.

Integrare o modificare processi di questo tipo ha spesso un costo elevato che può essere (o essere percepito come) superiore al rischio di produrre software difettoso.

Chi sviluppa le metodologie per la verifica del software tipicamente non si pone il problema dell'integrazione nei processi industriali delle aziende che potrebbero trarre i maggiori benefici dalle nuove tecniche.

# Valutare l'obsolescenza dei sistemi per uno sviluppo sicuro

I processi industriali, per quanto delicati, possono essere aggiornati nel tempo.

Ma cosa succede a quei sistemi che, per qualche motivo, si sono consolidati fino a uscire da un vero e proprio processo che ne adegui le caratteristiche allo scenario contemporaneo?

Anche se può sembrare strano questo è un caso piuttosto comune.

Negli scorsi decenni l'informatica e l'elettronica sono penetrate rapidamente in molte catene di produzione.

Oggetti puramente meccanici sono stati integrati con circuiti e relativo firmware per sfruttare la flessibilità e i bassi costi delle nuove tecnologie.

Purtroppo, in alcuni casi questi software sono stati sviluppati rapidamente per soddisfare precisi requisiti funzionali.

Chi poteva prevedere che, in pochi anni, un componente meccanico originariamente isolato sarebbe diventato un dispositivo IoT collegato alla rete ed esposto ad attacchi?

# Verificare sempre la molteplicità delle fonti

Quando si realizza un prodotto che include lo sviluppo di un software è importante chiedersi da dove arriva il codice.

Pensiamo per esempio ad una moderna automobile. Molti dei suoi componenti meccanici includono centraline che eseguono del software, ad esempio per la gestione delle frenate.

Inoltre, l'automobile monta un navigatore satellitare, magari integrato in un sistema di infotainment di bordo.

# Verificare sempre la molteplicità delle fonti

Ognuno di questi oggetti, e quindi il software che ospitano, è riconducibile a uno o più fornitori esterni all'azienda che assembla l'automobile.

Ogni attore di questa catena ramificata assembla codice esistente e ne produce di nuovo senza conoscere nel dettaglio le specifiche degli altri elementi che andranno a comporre l'oggetto finale.

Il risultato è un'automobile (uno smartphone, un frigorifero…) programmata con codice scritto da vari sviluppatori con diverse affiliazioni.

Qualcuno di questi potrebbe anche aver trovato su stack overflow parte del codice che controlla i freni dell'auto.

Gli strumenti basati sui metodi formali hanno spesso dei costi nascosti che vanno comunque tenuti in considerazione.

Per esempio, molte di queste tecniche utilizzano algoritmi computazionalmente complessi che non riescono a scalare adeguatamente su sistemi di complessità reale.

Di conseguenza è possibile procedere a una verifica formale di sottosistemi di dimensione contenuta oppure di astrazioni matematiche dei sistemi reali.

Nel primo caso i risultati della verifica formale devono essere interpretati considerando il contesto in cui opera il sottosistema analizzato.

Nel secondo la verifica si basa su un modello matematico che deve essere opportunamente realizzato per garantirne la fedeltà rispetto all'originale.

In entrambi i casi è necessario investire risorse significative per ottenere risultati concreti.

In assenza di motivazioni adeguate potrebbe essere preferibile investire su tecniche meno sofisticate ma comunque consolidate e spesso efficaci come il testing o l'ispezione manuale del codice sorgente.

Alcuni dei sistemi da cui dipendiamo sono controllati da programmi scritti in modo scorretto e insicuro.

Il codice di scarsa qualità, però, non sempre è il risultato di una mancanza di competenza o attenzione. In alcuni casi il contesto all'interno del quale viene prodotto un sistema favorisce la diffusione delle vulnerabilità.

Le tecniche di analisi formale del software possono dare un contributo decisivo per contrastare questo fenomeno, ma alcuni ostacoli ne rallentano o impediscono l'applicazione.

Il crescente costo degli attacchi potrebbe in futuro fornire l'incentivo decisivo per superare questo scalino tecnologico