LA SICUREZZA DEL SOFTWARE: UN TASSELLO FONDAMENTALE DELLA SOCIETÀ DIGITALE

Nell'era digitale in cui viviamo, il software è diventato un elemento chiave per la nostra vita, permeando ogni aspetto delle nostre attività personali e professionali. Dallo smartphone al computer, dagli elettrodomestici alle infrastrutture critiche, il software è onnipresente e svolge un ruolo fondamentale nella nostra società

*A cura di Massimo Nannini**

Vista l'importanza che il software riveste, qualunque problema si possa verificare durante il suo funzionamento, può creare notevoli problemi allo svolgimento delle normali attività quotidiane. Elemento questo molto ricercato dai cyber criminali che appunto mirano a creare situazioni critiche atte a minare reversibilmente o irreversibilmente il corretto funzionamento dei sistemi spesso alla ricerca di denaro oppure spinti da convinzioni malevole.

Un software vulnerabile può essere sfruttato da criminali informatici per rubare dati personali, compiere frodi, diffondere malware o compromettere infrastrutture critiche. Il costo

delle violazioni della sicurezza del software è enorme, non solo in termini finanziari, ma anche in danni reputazionali per le organizzazioni coinvolte.
Per affrontare queste sfide occorre fare ricorso ad un approccio metodologico mirato ad inserire le valutazioni di sicurezza in tutte le fasi di sviluppo di una applicazione software e non solo al temine, prima del rilascio al cliente o peggio ancora successivamente a questa fase, mediante un approccio di correzione a posteriori, non appena la condizione anomala viene segnalata attraverso le famose patch di sicurezza.

Secure Software Development Lifecycle (SSDLC)

Il Secure Software Development Life Cycle (SDLC), o Ciclo di Vita dello Sviluppo Software Sicuro, è un approccio metodologico fondamentale per integrare la sicurezza nel processo di sviluppo del software fin dalle prime fasi partendo dalla stesura dei requisiti attraversando tutto il ciclo di vita dello sviluppo. Attraverso questo approccio i problemi di sicurezza possono essere risolti nella pipeline SDLC ben prima della distribuzione in produzione minimizzando così il rischio di trovare vulnerabilità di sicurezza nelle applicazioni rilasciate, ma anche per ridurne al minimo l'impatto quando dovessero venire rilevate successivamente.
Seguendo SSDLC, gli sviluppatori possono creare software robusto e sicuro in grado di resistere meglio alle minacce informatiche e proteggere i dati sensibili.
L'obiettivo di SSDLC non è eliminare completamente i controlli di sicurezza tradizionali, come i penetration test, ma piuttosto includere la sicurezza nell'ambito delle responsabilità degli sviluppatori e consentire loro di creare applicazioni sicure fin dall'inizio.

Da SDLC a Secure SDLC

Software Development Lifecycle descrive come le applicazioni software devono essere costruite attraverso una serie di passaggi (fasi) che sono patrimonio di conoscenza di tutti gli sviluppatori ormai da lungo tempo, faccio riferimento a:

1. Raccolta dei requisiti
2. Analisi dei requisiti
3. Progettazione sulla base dei requisiti
4. Sviluppo, scrittura del codice sulla base dei requisiti
5. Test e verifica
6. Rilascio
7. Mantenimento ed evoluzione

Tutto questo all'interno di un ciclo ripetitivo che tende ad essere sempre più "ristretto" seguendo i dettami delle più moderne teorie per lo sviluppo AGILE e DEVOPS.
Questo modello, considerando anche i più recenti aggiornamenti esiste da circa 60 anni, ma nonostante la sua età, o forse proprio a causa di essa, non comprende attività mirate alla sicurezza.
Ma come possiamo aggiungere sicurezza alla già complessa attività di creazione di software? Come la maggior parte delle cose, tutto ciò che serve è introdurre strategicamente le migliori pratiche per renderlo parte del processo di sviluppo piuttosto che un collo di bottiglia al suo interno.

Prima di capire come incorporare la i temi della sicurezza nell'SDLC stesso, è importante comprendere i tipi di attività che rientrano in questo ambito all'interno di un'organizzazione dedita allo sviluppo di software. Di seguito un elenco non esaustivo di attività che si possono attuare (o pianificare di attuare) per migliorare la sicurezza del software.

Analisi statica
È il processo di scansione automatica del codice sorgente alla ricerca di difetti e vulnerabilità. Si tratta in genere di un processo automatizzato che identifica modelli noti di codice non sicuro all'interno di un progetto software, offrendo ai team di sviluppo l'opportunità di risolvere i problemi di sicurezza in parallelo alle tecniche TDD.

Scansione di sicurezza
Analogamente all'analisi statica, la scansione di sicurezza è un processo generalmente automatizzato che esegue la scansione di un'intera applicazione e della relativa infrastruttura sottostante alla ricerca di vulnerabilità e configurazioni errate. Questo può essere introdotto sotto forma di analisi di scripting crosssite, scansione delle porte o scansione delle vulnerabilità dei container (solo per citarne alcuni).

Revisioni del codice
Sebbene la scansione automatica sia utile, è sempre vantaggioso una ulteriore verifica umana sul codice prima di rilasciarlo in un ambiente di produzione. La maggior parte dei team di sviluppo implementa già le revisioni del codice per aiutare a rilevare difetti e altri errori logici, ma con la giusta ottica sulla sicurezza, le revisioni del codice possono fornire una supervisione utile per garantire che alcune vulnerabilità meno comuni non vengano introdotte.

Test di penetrazione
Il penetration test prevede l'intervento di un professionista della sicurezza informatica per testare la sicurezza dell'infrastruttura di produzione di un'azienda. Un penetration tester fornirà un chiaro rapporto dei diversi problemi che sono sfuggiti a qualsiasi checkpoint dei test di sicurezza.

Ricompense Bug
Si tratta di una pratica più recente, simile ai test di penetrazione. I bug bounty incoraggiano gli utenti a segnalare le vulnerabilità che si trovano a fronte di una ricompensa. I bug bounty sono un ottimo modo per incoraggiare le persone a segnalare i problemi di sicurezza che trovano piuttosto che sfruttarli per il proprio guadagno personale.

Formazione
Il mondo della sicurezza informatica è in continua evoluzione e gran parte dei consigli e delle conoscenze che erano utili un decennio fa non sono più applicabili, proprio come ciò che sappiamo oggi probabilmente non sarà più utile tra qualche anno. La formazione sulla sicurezza può fare molto per mitigare le vulnerabilità alla fonte più comune: l'errore umano.
Le attività sopra descritte rendono evidenza che l'integrazione della sicurezza nel ciclo di vita del software è più che altro da considerare come una integrazione dellf fasi già presenti più che l'inserimento di nuove fasi o attività.
Non esiste una "fase di sicurezza", ma piuttosto un insieme di best practice e strumenti che possono essere inclusi nelle fasi esistenti dell'SDLC (vedi figura 1), trattare dunque la sicurezza come un'evoluzione del processo e non un altro elemento da spuntare nell'elenco delle cose da fare.

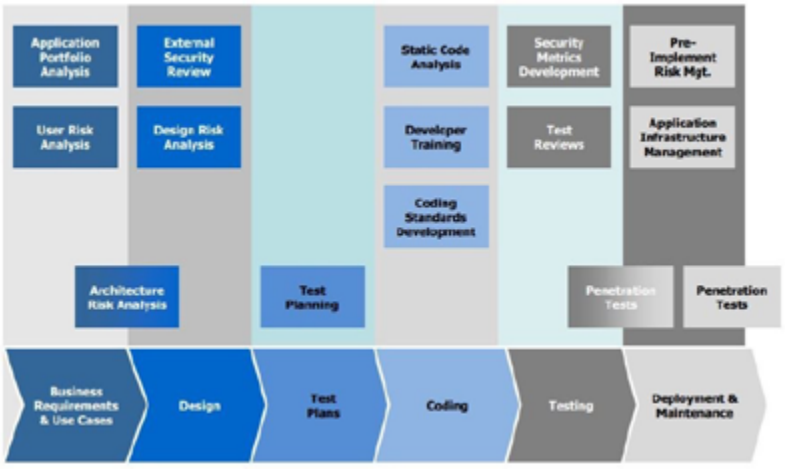

Fig.1 Sicurezza nel ciclo di vita dello sviluppo del software

L'implementazione della sicurezza influisce dunque su ogni fase del processo di sviluppo del software evidenziando così fase per fase, durante l'avanzamento del progetto, le criticità che così possono essere immediatamente corrette rendendo il processo più efficiente ed economico nel suo complesso. I processi sicuri del ciclo di vita dello sviluppo del software incorporano la sicurezza come componente di ogni fase dell'SDLC.

Le fasi del ciclo di vita del software sicuro

Ogni fase del Secure SDLC (SSDLC) deve contribuire alla sicurezza dell'applicazione complessiva, ma avendo ogni singola fase le proprie peculiarità e scopi la considerazione del tema sicurezza assume evidentemente aspetti differenti.
Nel proseguo ci focalizzeremo solo sulle attività riguardanti il tema della sicurezza considerando le altre come elementi noti di un processo di sviluppo software

Mappatura dei requisiti di sicurezza
Il primo passo nel processo SDLC sicuro è la mappatura dei requisiti di sicurezza. Durante questa fase è necessario identificare le minacce alla sicurezza che il sistema software dovrà affrontare valutando anche il potenziale danno che una violazione o un attacco potrebbero causare. Una volta poi identificati i rischi e le minacce per la sicurezza, sarà necessario stabilire obiettivi e traguardi di sicurezza in relazione alle specifiche esigenze specifiche del sistema software e considerare eventuali requisiti legali o normativi.

Progettazione del software
Dopo aver mappato i requisiti di sicurezza e creato una policy di sicurezza, è il momento di iniziare a progettare il sistema. Questa fase traduce i requisiti di sicurezza, ovviamente compenetrati con quelli funzionali dell'applicazione, in una architettura atta a corrispondere ai requisiti esplicitati. Ciò implica garantire che il codice sia strutturato correttamente, che i componenti software siano isolati e che vengano implementati protocolli di sicurezza come l'autenticazione e la crittografia.

Sviluppo del software
Questa è la fase cruciale dove prende forma il progetto software. Gli sviluppatori devono seguire linee guida che permettono di ottenere "codice sicuro" in particolare mi riferisco all'utilizzo dei Design Patterns, delle query parametriche e di sola lettura, la convalida dei dati inseriti dall'utente,

la crittografia. Gli sviluppatori però non possono preoccuparsi solo del codice che scrivono, ma devono anche porre particolare attenzione alle eventuali librerie che utilizzano, che ovviamente non potendo essere direttamente sotto il nostro controllo possono essere facilmente fonte di vulnerabilità. Fortunatamente attraverso strumenti di Software Composition Analysis (SCA) le suddette librerie possono essere "radiografate" rilevando tutti i componenti correlati, le librerie di supporto e le dipendenze dirette e indirette. Possono anche rilevare licenze software, dipendenze obsolete, nonché vulnerabilità e potenziali exploit. Il processo di scansione genera una distinta base (BOM), fornendo un inventario completo delle risorse software di un progetto.

Verifica
La fase di verifica è il momento in cui le applicazioni vengono sottoposte a un accurato ciclo di test per garantire che soddisfino i requisiti di progetto, ma anche i requisiti di sicurezza attraverso test automatici per la ricerca dei percorsi critici, unit test, ecc.

Mantenimento ed evoluzione
Sebbene la fase di manutenzione sia generalmente utilizzata per identificare e correggere i difetti nel codice, è anche il punto in cui verranno scoperte le vulnerabilità. È importante non illudersi pensando che il codice protetto rimarrà sempre protetto. Dai rischi della catena di approvvigionamento agli exploit zeroday, il panorama della sicurezza è in continua evoluzione e disporre di un processo per identificare e rispondere ai problemi non appena si presentano è un passaggio fondamentale nell'implementazione di un SDLC sicuro.

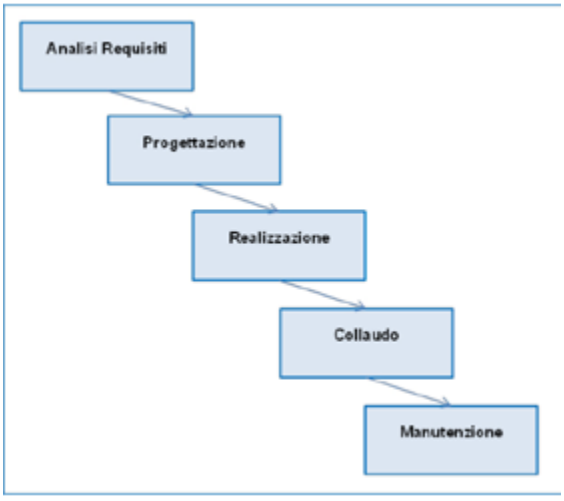

Fig.2 Fasi del ciclo di vita del software sicuro

Da quanto sino a qui illustrato risulta evidente che l'individuazione delle vulnerabilità deve essere svolta durante tutto il ciclo iterativo di sviluppo del prodotto software e questo richiede una adeguata formazione e consapevolezza dei teams di sviluppo in materia di sicurezza e una chiara definizione di linee guida da seguire per la scrittura del codice sicuro. Oltre a questo un posto fondamentale è ricoperto dalle prima di tutte le fasi e cioè la raccolta dei requisiti, quindi concertando in questo momento una analisi dettagliata dei possibili comportamenti di sicurezza dell'applicazione si eviterà di dovere ritornare indietro nell'eventualità che queste vulnerabilità vengano identificate a valle del processo nelle fasi di test, il che comporta molte volte la revisione del progetto con relative problematiche di inefficienze legate al notevole incremento di tempi e costi per porvi rimedio.

Keywords: Secure Software Development Life Cycle, SDLC, Penetration Test, Agile, DevOPS, TDD, bug bounty, Software Composition Analysis, SCA, BOM, Design Pattern, vulnerabilità

(*) Massimo Nannini
Ingegnere elettronico e consulente di impresa, info@gemaxconsulting.it

SOFTWARE SECURITY: A KEY PIECE OF THE DIGITAL SOCIETY

In the digital age in which we live, software has become a key element in our lives, permeating every aspect of our personal and professional activities. From smartphones to computers, from home appliances to critical infrastructure, software is ubiquitous and plays a foundational role in our society.

By Massimo Nannini*

Given the importance of software, any problem that may occur during its operation can create significant problems in the performance of normal daily activities. Element this much sought after by cyber criminals who precisely aim to create critical situations capable of reversibly or irreversibly undermining the proper functioning of systems often in search of money or driven by malicious convictions.
Vulnerable software can be exploited by cybercriminals to steal personal data, commit fraud, spread malware, or compromise critical infrastructure. The cost of software security breaches is enormous, not only in financial terms but also in reputational damage to the organizations involved.
Addressing these challenges requires resorting to a methodological approach aimed at incorporating security assessments at all stages of software application development and not only at the end, before release to the customer or even worse after this stage, through an afterthefact remediation approach, as soon as the anomalous condition is reported through famous security patches.

Secure Software Development Lifecycle (SSDLC)
The Secure Software Development Life Cycle (SDLC), or Secure Software Development Lifecycle, is a fundamental methodological approach to integrating security into the software development process from the earliest stages starting from requirements drafting through the entire development lifecycle. Through this approach, security issues can be resolved in the SDLC pipeline well before deployment to production thus minimizing the risk of finding security vulneabilities in released applications, but also to minimize their impact when they are detected later.
By following SSDLC, developers can create robust and secure software that can better resist cyber threats and protect sensitive data.
The goal of SSDLC is not to completely eliminate traditional security controls, such as penetration testing, but rather to include security as part of developers' responsibilities and enable them to create secure applications from the start.

From SDLC to Secure SDLC
Software Development Lifecycle describes how software applications are to be built through a series of steps (phases) that have been the knowledge of all developers for a long time now, I refer to:

Requirements gathering
Requirements analysis
Design based on requirements
Development, writing code based on requirements
Testing and verification
Release
Maintenance and evolution

All this within a repetitive cycle that tends to be increasingly "narrowed" following the dictates of the most modern theories for AGILE and DEVOPS development.
This model, considering even the most recent updates, has existed for about 60 years, but nonodespite its age, or perhaps because of it, it does not include safetyfocused activities.
But how can we add security to the already complex business of creating software? Colike most things, all that is needed is to strategically introduce best practices to make it part of the development process rather than a bottleneck within it.

Before understanding how to incorporate security issues into the SDLC itself, it is important to comprehend the types of activities that fall under this umbrella within an organization dedicated to software development. Below is a nonexhaustive list of activities that can be implemented (or planned to be implemented) to improve software security.

Static analysis
This is the process of automatically scanning source code for flaws and vulnerabilities. It is typically an automated process that identifies known patterns of insecure code within a

software project, providing development teams with an opportunity to address security issues in parallel with TDD techniques.

Security scanning
Similar to static analysis, security scanning is a generally automated process that scans an entire application and its underlying infrastructure for vulnerabilities and misconfigurations. This can be introduced in the form of crosssite scripting analysis, port scanning, or container vulnerability scanning (just to citate a few).

Code reviews.
Although automated scanning is useful, additional human verification on code before releasing it into a production environment is always beneficial. Most development teams already implement code reviews to help detect defects and other logical errors, but with the right security focus, code reviews can provide useful oversight to ensure that some less common vulnerabilities are not introduced.

Penetration testing
Penetration testing involves bringing in an IT security professional to test the security of a company's production infrastructure. A penetration tester will provide a clear rapport of the various problems that have escaped any security testing checkpoints.

Bug rewards.
This is a newer practice, similar to penetration testing. Bug bounties encourage users to report vulnerabilities they find for a reward. Bug bounties are a great way to encourage people to report security problems they find rather than exploiting them for personal gain.

Training
The world of cybersecurity is constantly evolving, and much of the advice and conoscience that was useful a decade ago is no longer applicable, just as what we know today probably won't be useful in a few years. Security training can do much to mitigate vulnerabilities at the most common source: human error.
The activities described above make it clear that the integration of security into the software life cycle is more to be seen as an integration off steps already in place rather than the insertion of new steps or activities.

There is no "security phase," but rather a set of best practices and tools that canbe included in existing phases of the SDLC (see Figure 1), thus treating security as an evolution of the process and not another item to be checked off the to-do list.

The implementation of security thus affects each phase of the software development process by highlighting stage by stage, as the project progresses, critical issues that can thus be immediately corrected, making the process more efficient and economical overall. Secure software development lifecycle processes incorporate security as a component of each phase of the SDLC.

The phases of the secure software lifecycle
Each phase of the Secure SDLC (SSDLC) must contribute to the security of the complexve application, but since each individual phase has its own peculiarities and purposes the consideration of the security issue evidently takes on different aspects.
In the following we will focus only on the activities concerning the security issue considering the others as known elements of a software development process.

Security Requirements Mapping.
The first step in the secure SDLC process is the mapping of security requirements. During this step, it is necessary to identify the security threats that the software system will face valuing also the potential damage that a breach or attack could cause. Once security risks and threats have then been identified, it will be necessary to establish security goals and objectives in relation to the specific needs specific to the software system and consider any legal or regulatory requirements.

Software design.
After mapping the security requirements and creating a security policy, it is time to begin designing the system. This phase translates the security requirements, obviously interpenetrated with the functional requirements of the application, into an architecture suitable to match the explicit requirements. This involves ensuring that the code is structured correctly, that software components are isolated, and that security protocols such as authentication and encryption are implemented.

Software development.
This is the crucial phase where the software

project takes shape. Developers must follow guidelines to achieve "secure code" specifically I am referring to the use of Design Patterns, parametric and readonly queries, validation of user input, and encryption. Developers, however, cannot only be concerned about the code they write; they must also pay special attention to any libraries they use, which, of course, since they cannot be directly under our control, can easily be a source of vulnerabilities. Fortunately, through Software Composition Analysis (SCA) tools, these libraries can be "xrayed" by detecting all related components, supporting libraries, and direct and indirect dipendencies. They can also detect software licenses, obsolete dependencies, as well as vulnerabilities and potential exploits. The scanning process generates a bill of materials (BOM), forning a complete inventory of a project's software assets.

Verification
The verification phase is when applications undergo a thorough testing cycle to ensure that they meet not only design requirements, but also security requirements through automated critical path testing, unit testing, etc.

Maintenance and evolution
Although the maintenance phase is generally used to identify and correct flaws in the code, it is also the point at which vulnerabilities will be discovered. It is important not to delude oneself by thinking that protected code will always remain protected. From supply chain risks to zeroday exploits, the security landscape is constantly evolving, and having a process to identify and respond to problems as they arise is a critical step in implementing a secure SDLC.

It is evident from what has been illustrated up to now that vulnerability identification must be carried out throughout the iterative cycle of software product development, and this requires adequate security training and awareness of development teams and a clear definition of guidelines to be followed in writing secure code. In addition to this, a fundamental place is held by the first of all phases and that is the requirements gathering, so concerting in this moment a detailed analysis of the possible security behaviors of the application will avoid having to go back in case these vulnerabilities are identified downstream of the process in the testing phases, which many times results in the revision of the project with related problems-matters of inefficiencies related to the significant increase in time and cost to remedy them.

Keywords: Secure Software Development Life Cycle, SDLC, Penetration Test, Agile, DevOPS, TDD, bug bounty, Software Composition Analysis, SCA, BOM, Design Pattern, vulnerability

(*) Massimo Nannini, IT engineer and business consultant, info@gemaxconsulting.it