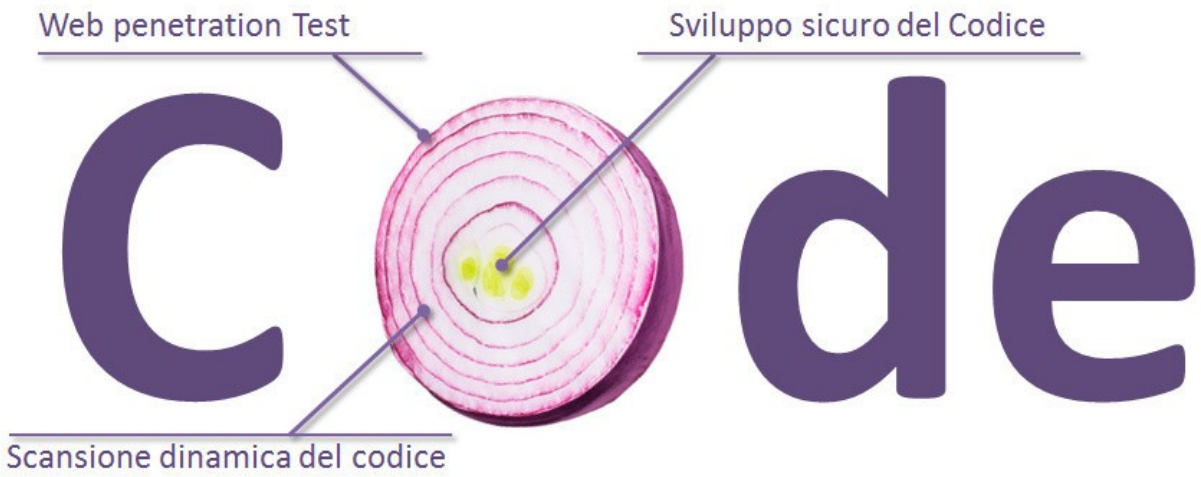
Sviluppo sicuro del codice... 4 semplici domande a 3 strati di una cipolla

Author : Massimiliano Brolli

Date : 19 novembre 2018

Sempre più spesso, in grandi organizzazioni, si parla di Sviluppo Sicuro del Codice (di seguito SSC) ovvero strumenti automatici a supporto dei programmatori che consentono di analizzare il software prodotto alla ricerca di vulnerabilità (tipicamente Owasp TOP10 o SANS25) da sanare prima del rilascio per rendere un software robusto a prova di intrusione.

Ma questo approccio premia davvero?



Esistono differenti modi di effettuare test di sicurezza applicativi, tutti basati nel rilevare errori di programmazione e di implementazione delle logiche applicative, ma la pratica dello sviluppo sicuro risulta sempre lo strato più delicato ed “insicuro” per produrre applicazioni "rock-solid".

Occupandomi di verifiche di sicurezza, molto spesso mi sono imbattuto nel rilevare negli assessment impatti derivanti l’abuso di vulnerabilità non sanate nello SSC e molto spesso di tali vulnerabilità non vi era traccia nei report prodotti dagli strumenti di scansione statica. Ma

prima di darvi una visione di dettaglio delle mie considerazioni dobbiamo porci una domanda: "*che cosa è una scansione statica del codice?*" Se per voi la risposta è "*un tool che consente di identificare tutte le problematiche di sicurezza applicative che possono mettere a rischio la nostra applicazione"*, siete completamente fuori strada.

Infatti gli strumenti di scansione statica (ma come vedremo successivamente anche dinamica, tralasciando quanto dichiarato dai fornitori dei tool) consentono di rilevare, con buona confidenza, vulnerabilità tipiche di depurazione dell'input quali XSS ed *Injection.

La tabella in calce (Fonte Owasp con qualche arricchimento) pone attenzione su questo problema, ma lascio a voi le dovute considerazioni.

ID Owasp	Vulnerabilità	Sfruttamento	Diffusione	Individuazione con strumenti automatici	Impatto
A1	Injection	**Facile**	**Comune**	**Medio**	**Severo**
A2	Broken Authentication	**Facile**	**Comune**	**Difficile**	**Severo**
A3	Sensitive Data Exposure	**Medio**	**Diffuso**	**Medio**	**Severo**
A4	XML External Entities (XXE)	**Medio**	**Comune**	**Facile**	**Severo**
A5	Broken Access Control	**Medio**	**Comune**	**Difficile**	**Severo**
A6	Security Misconfiguration	**Facile**	**Diffuso**	**Facile**	**Medio**
A7	Cross-Site Scripting (XSS)	**Facile**	**Diffuso**	**Facile**	**Medio**
A8	Insecure Deserialization	**Difficile**	**Comune**	**Medio**	**Severo**
A9	Using Components with Known Vulnerabilities	**Medio**	**Diffuso**	**Medio**	**Medio**
A10	Insufficent Logging & Monitoring	**Medio**	**Diffuso**	**Difficile**	**Medio**

Il problema principale, analizzando nel dettaglio le TOP10 Owasp, è che esistono vulnerabilità con impatti molto severi che non vengono rilevate dagli strumenti automatici e tali vulnerabilità sono comportamenti “applicativi” o errori (spesso orrori) di progettazione che non possono essere rilevati dagli strumenti di oggi.

Ad esempio *Broken Authentication* e *Broken Access Control* sono vettori di attacco formidabili che conducono ad impatti molto severi senza pensare ad una SQL Injection difficile da rilevare, quindi la scansione statica porta con se dei limiti in quanto:

- Non mette al riparo dalla presenza di errori nella logica applicativa, proprio gli ambiti di sfruttamento più critico come ad esempio carenze nello sviluppo della profilazione e nel principio del *need to know*, ovvero consentire l'accesso ai contenuti sui quali si è autorizzati;
- Introduce molti Falsi positivi (oltre il 50% delle rilevazioni sono tali) e ancora più rischiosi e fuori controllo sono i Falsi negativi, ovvero vulnerabilità reali non identificate spesso SQL injection;
- La bontà della scansione risulta dipendente dall'istruzione degli scanner e dell'ambiente di scansione (ad esempio dipendenze tra librerie, dipendenze tra moduli applicativi, framework di terze parti utilizzati e warning di scansione non gestiti).

Ma il "cattivo" non lavora in sviluppo ma in produzione e magari dalla Cina o dalla Russia, quindi potrebbe venire lecita la seconda domanda: "...ma se sposto il controllo in produzione utilizzando scanner dinamici potrei avere maggiori margini di confidenza?" La risposta è la stessa di quando si confronta un uomo e una macchina. Ma al netto della risposta la scansione dinamica (ad esempio gli strumenti di web scanning) sono utili come tutti i Vulnerability Assessment perché spostano il controllo più vicino all'attività illecita ma anche in questo caso sono presenti una serie di limitazioni da considerare:

- Gli scanner web non mettono al riparo dalla presenza di errori applicativi di sviluppo del software (come visto in precedenza nelle scansioni statiche) anche se hanno un ventaglio maggiore di copertura proprio perché eseguite in campo.
- Uno Scanner dinamico è molto intrusivo, pertanto è buona pratica farlo in collaudo per salvaguardare l'esercizio, ma facendolo in collaudo si introducono possibili elementi di debolezza come ambienti disomogenei, versioni software differenti, patch differenti, ambienti di collaudi non disponibili, ecc...
- L'intrusività delle policy è direttamente proporzionale alle vulnerabilità rilevate. È importante avere esperienza in tal senso per ottenere il giusto compromesso.
- A differenza delle tecnologie e dalle policy definite può introdurre Falsi positivi e i più rischiosi Falsi negativi.

Quindi siamo arrivati alla terza domanda: "... ma allora, forse è meglio lasciare perdere tutto per avviare dei Penetration Test Applicativi?". Potrebbe essere una soluzione, ma anche in questo caso ci sono dei punti di attenzione da prendere in considerazione in quanto:

- Un PT web si fa direttamente in esercizio e da un buon grado di confidenza del rischio presente nell'applicazione e risulta poco invasivo in quanto svolto da un tecnico specializzato e l'efficacia è direttamente proporzionale agli skill del personale specializzato;
- In applicazioni complesse non può essere praticabile su tutte le "foglie" dell'applicazione per i limiti temporali di gestione di un controllo (un PT non è un Ethical-APT a livello temporale);

- Consente di identificare gli “Impatti reali” ed è efficace nel rilevare carenze dove risulta impossibile l’identificazione con scansioni statiche/dinamiche;
- Fornisce un quadro completo (anche abbinandolo ad una scansione web) del reale grado di rischio dell'applicazione;
- Ha dei costi superiori alle consuete attività automatizzate.

Quindi ci posizioniamo sulla quarta domanda …“e allora? ... quale è la soluzione migliore da adottare?”, la risposta come al solito è “dipende” in quanto:

1. Lo Sviluppo Sicuro del Codice è demandato alle fabbriche, serve a "rastrellare" tramite scansione statica le falle di sicurezza più semplici e deve essere fatto. Inoltre l’utilizzo di questi scanner (inteso come tool di sviluppo allo stesso livello di un Eclipse o un VisualStudio) crea valore indiretto introducendo “Consapevolezza” nei Team di sviluppo, molti di questi ancora oggi con grosse lacune in termini di sviluppo sicuro del software;
2. La scansione dinamica (come tutti i vulnerability assessment) è il “setaccio a grana grossa”, è utile, porta benefici e vulnerabilità spesso reali ma occorre prestare molta attenzione sui rischi in produzione, sull'invasività dei prodotti di scansione scelti, sulle policy utilizzate rispetto a quanto visto in precedenza;
3. Il Penetration test Applicativo è la soluzione ideale per avere una buona confidenza sui rischi derivanti un abuso della nostra applicazione in quanto riesce a coprire tutte le TOP10 Owasp, ma un PT su una applicazione complessa potrebbe avere costi proibitivi oltre ad essere a livello di tempistiche poco sostenibile.

Concludendo, tutte e tre le tecniche (scansione statica, dinamica e PT Applicativo) sono tecniche che forniscono raramente risultati sovrapponibili, sarà la nostra esperienza metterle a fattor comune per individuare il livello di sovrapposizione e i delta delle vulnerabilità rilevate.

Il rischio Zero non esiste (pensiamo all’incidente del 28/09/2019 di Facebook denominato “view as” che ha portato al reset di 100mln di token) ma esiste il rischio "tollerabile". Tutte e tre le tecniche dovranno quindi essere misurate per comprendere dove una sia migliore dell'altra, dove dovranno essere scelte tecniche miste, anche in funzione dell'appetibilità dei dati contenuti all'interno delle nostre applicazioni. Su applicazioni con dati critici, personalmente utilizzo tutte e tre le tecniche mentre su applicazioni con dati non critici, anche la sola scansione statica può risultare una scelta sensata.

E’ chiaro che chi sviluppa software non può limitarsi all'utilizzo di uno strumento di scansione ma deve essere guidato dall'esperienza e dall'applicazione delle migliori best practices conosciute in termini di sviluppo sicuro del codice. Per chi usa e abusa di forniture esterne, risulta di focale importanza l’aspetto contrattuale articolato su deliverable differenti a seconda della criticità dei sistemi.

Effettuando un paragone, mi viene da pensare ai guasti difficili da identificare in una macchina di oggi e i meccanici che collegano la centralina al computer e fanno solo quello che lui dice nelle attività di riparazione. Altri meccanici invece che conoscono a fondo tutte le componenti del veicolo comprendono che quanto fornito dallo scanner potrebbe essere una indicazione di

un difetto presente in un'altra parte non chiaramente identificata, identificabile univocamente dalla loro esperienza.

Articolo a cura di **Massimiliano Brolli**