



UNIVERSITÀ DEGLI STUDI DI MILANO

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea in Scienze e Tecnologie dell'Informazione

TECNICHE DI ANALISI STATICA
PER LA SICUREZZA DI APPLICAZIONI WEB:
PROBLEMATICHE ED IMPLEMENTAZIONI

Dario Battista Ghilardi

753708

Relatore:

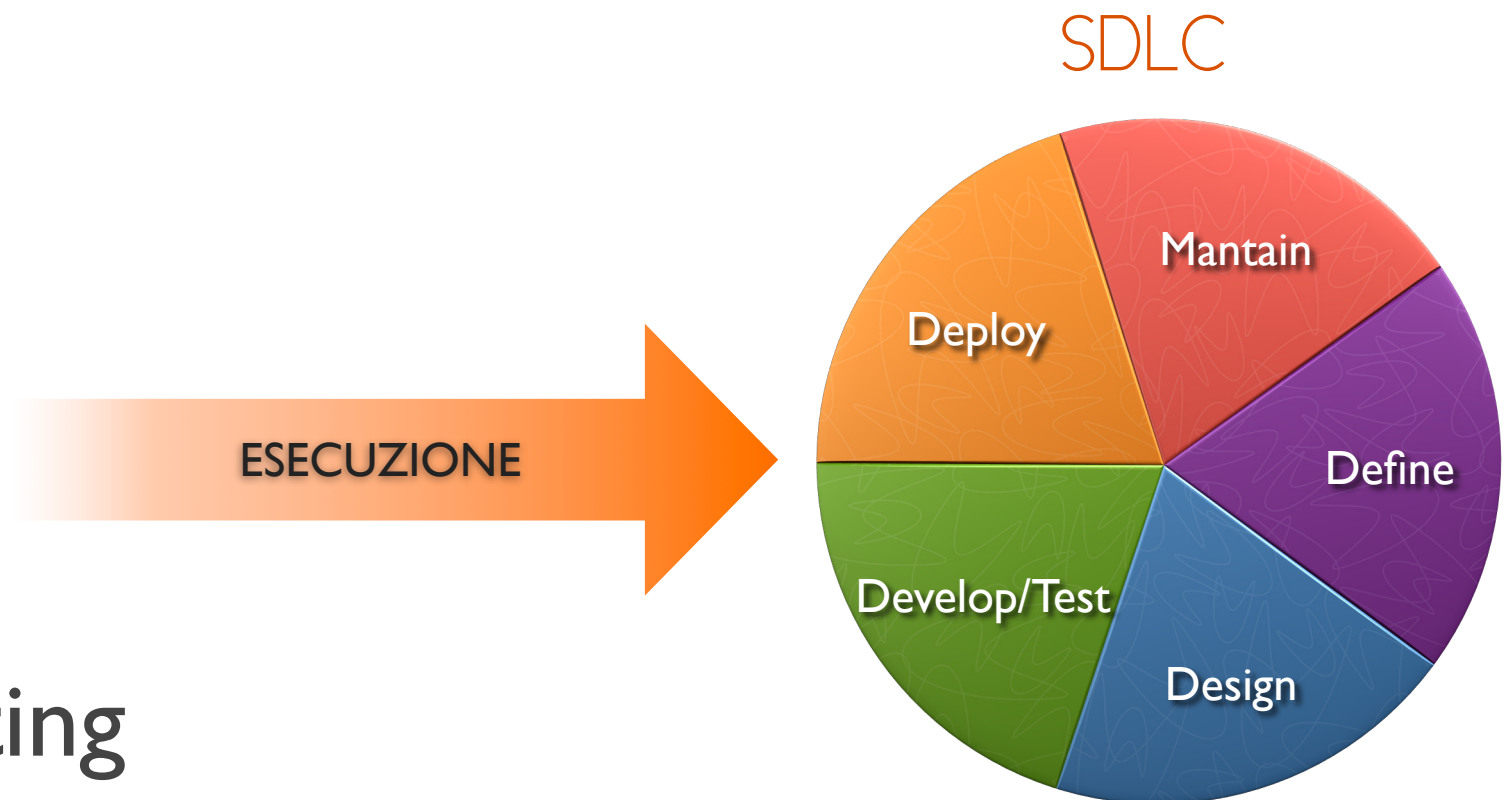
Prof. Marco Cremonini

LA DIFFUSIONE DELLE APPLICAZIONI WEB E' IN CONTINUO AUMENTO



Le applicazioni necessitano di sicurezza

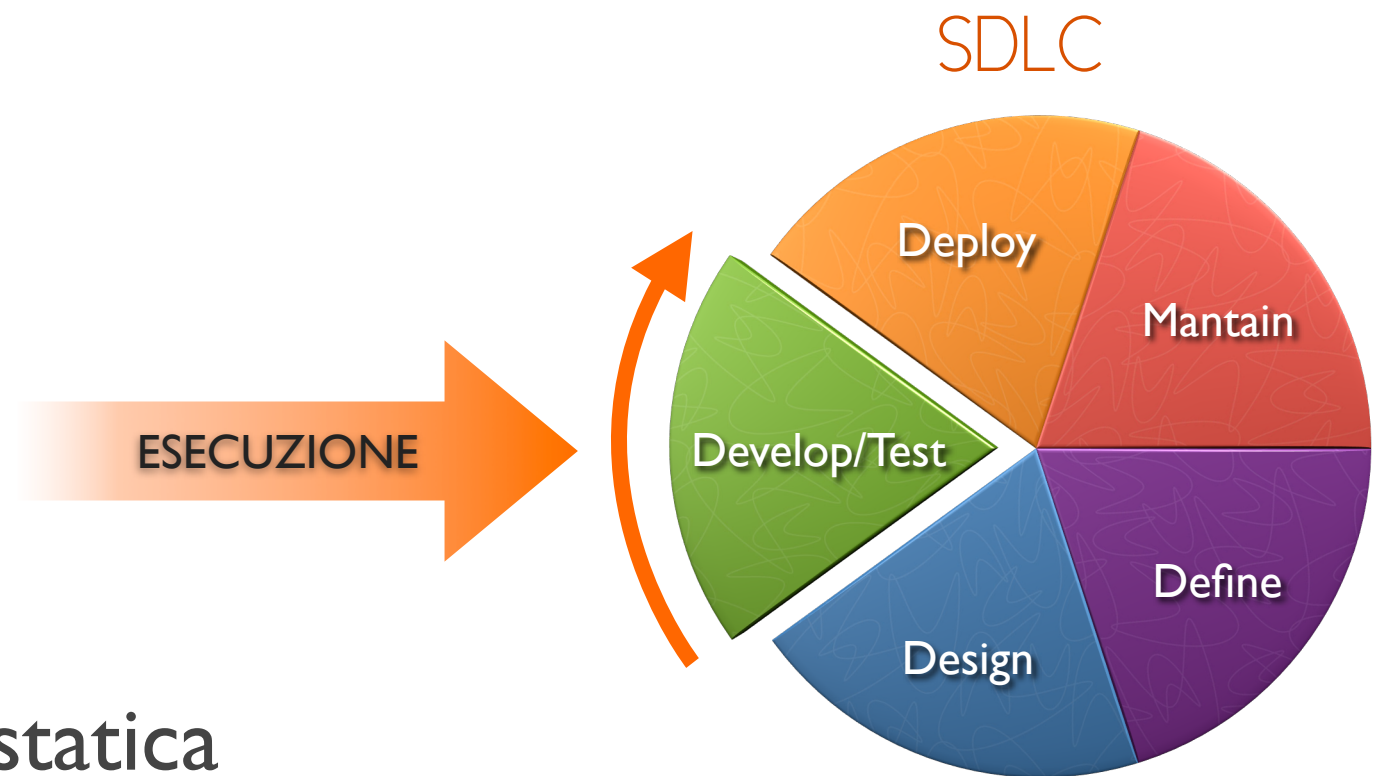
- Analisi statica
- Analisi dinamica
- Code Review
- Penetration testing



MODIFICHE SUCCESSIVE A SVILUPPO E TESTS
COMPORTANO COSTI ELEVATI

In questa tesi:

- Proposta di una soluzione al problema attraverso l'uso dell'analisi statica
- Stato dell'arte dell'analisi statica
- Tecniche ed implementazioni nella ricerca di vulnerabilità
- Proposta e progettazione di un nuovo tool



Obiettivo: rendere la sicurezza un processo

Perchè l'analisi statica?

Analisi statica: Ispezione automatica del codice sorgente di un software senza eseguirlo.

- + **RAPIDA** Può essere usata durante lo sviluppo
- + **SUFFICIENTEMENTE EFFICACE** Statisticamente la maggioranza delle vulnerabilità deriva da errori comuni degli sviluppatori (sicurezza non codificata nello sviluppo)
- **NON CONOSCE I DATI NEL FLUSSO**
- **NECESSITA DI APPROSSIMAZIONE** Valutazione di codice run-time, inclusione run-time di files esterni, tipizzazione dinamica...

E perchè non altre tipologie di analisi?

Code review: Ispezioni manuale del codice sorgente



LENTA E COMPLESSA

Analisi dinamica: Osservazione del comportamento del software durante la sua esecuzione.



VALIDA SOLO PER L'INPUT CORRENTE

Durante lo sviluppo: vantaggi

- Anticipo i controlli di sicurezza
- La correzione delle problematiche costa meno
- Lo sviluppatore diventa consapevole
- Riduco la possibilità che esista una finestra di esposizione

FOCUS SU PHP:

- Estremamente diffuso
- Curva di apprendimento facile
- Numerosi applicativi open-source maturi



Come si esegue: sorgente dell'analisi

SUL CODICE SORGENTE:

- Difficile per regole lessicali che governano linguaggio
- *grep* usava questo approccio, ormai abbandonato
- Uso di espressioni regolari

SUI TOKENS, OUTPUT DELLA LEXICAL ANALYSIS:

- Tokens: rappresentazione delle istruzioni indipendente dalle regole lessicali
- Rappresentazione simile al codice sorgente ma già utilizzata per fare analisi statica
- Uso di *tokenizer* o *PHP_TokenStream*

SULL'AST, OUTPUT DELLA SYNTAX ANALYSIS:

- Abstract syntax tree: rappresentazione ad albero del codice sorgente
- Semplificata l'individuazione dei flussi di istruzioni
- Uso di *PHP-Parser*

SUL BYTECODE:

- Output complesso da parsare per il basso livello del codice fornito
- Uso di *Bytekit*

Come si esegue: approccio

TAINT ANALYSIS:

- Ogni variabile modificabile dall'utente corrisponde ad un rischio di sicurezza
- Analisi definisce che tutti i valori di input vengono considerati pericolosi

DEFINIZIONE SENSITIVE SINKS:

- Vengono definiti i punti in cui i valori possono causare danno

DEFINIZIONE SANITIZATION ROUTINES:

- Vengono definite le funzioni che rendono sicure le variabili

RAFFINATEZZA E TEMPO DI ESECUZIONE VARIANO IN BASE A:

FLOW SENSITIVE / INSENSITIVE:

- Tiene conto dell'ordine in cui le istruzioni vengono eseguite

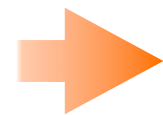
LOCAL / GLOBAL ANALYSIS:

- Traccia le proprietà e lo stato di taint delle variabili in ogni step per ogni funzione

Stato dell'arte su codice PHP:

- Codesecure

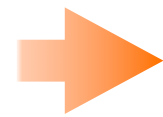
- Fortify 360



Tools commerciali, scarsi dettagli implementativi
noti ma funzione di supporto alla code review

- Pixy

- RIPS



Tools open-source

I tool esistenti sono adatti ad essere usati durante lo sviluppo?

Pixy:

Scritto in Java da Jovanovic (EURECOM) nel 2005

- Analisi flow sensitive e globale
- Usa come input un *parse-tree* del codice
- Non supporta PHP 5 e codice ad oggetti
- Inizialmente sviluppato per XSS, poi SQL Injection e command injection

STATO ATTUALE: Abbandonato dal 2007.

RISULTATI: Accettabili solo su codebase molto datate.

Esteso da **SANER** al fine di determinare il risultato di validazioni custom (analisi dinamica su espressioni regolari).

RIPS:

Scritto in PHP da Johannes Dahse

- Analisi flow sensitive e globale
- Usa come input uno stream di tokens, derivato da *tokenizer*
- Supporta parzialmente PHP 5 e codice ad oggetti
- Rileva XSS, SQL injections, file disclosure, code evaluation, remote command execution e logic flaw
- Codebase poco ingegnerizzata

STATO ATTUALE: In sviluppo. Release stabili.

RISULTATI: Accettabili solo su codebase datate.

Buona velocità.

Ottimo report finale.

Vulture:

Soluzioni esistenti non soddisfacenti, progettato Vulture in collaborazione con EURECOM.

CARATTERISTICHE ATTESE:

- Scritto in PHP su Symfony 2
- Estensibile tramite bundles che definiscono sinks, sanitization functions e valori tainted
- Analisi flow sensitive e globale
- In input l'AST del sorgente da *PHP-Parser*
- Supporto completo a PHP 5
- Rilevazione HTTP Parameter Pollution
- Velocità di scansione elevata

STATO ATTUALE: In sviluppo. Non ancora rilasciato.

Considerazioni finali:

- Tool commerciali orientati alla code review
- Tool open-source non hanno catturato audience nella community
- Sviluppo di tool complesso per via della dinamicità di PHP

QUINDI

Settore quasi inesplorato.

L'esigenza è però molto sentita, l'ambito è promettente per un business.

SVILUPPI FUTURI

- Concludere Vulture e valutarlo concretamente nel SDLC.