MythBreaking SCR Tools

Comuni certezze e falsi miti della static code analysis

Giorgio Fedon
Owasp Italy – Technical Director

giorgio.fedon@mindedsecurity.com

Security Summit 2011

The OWASP Foundation
http://www.owasp.org

Automatic secure code review tools

Cos'è un tool di static code analysis?

- Tool per l'Analisi Whitebox del Codice Sorgente
- Categorizzo i tool nel seguente modo:

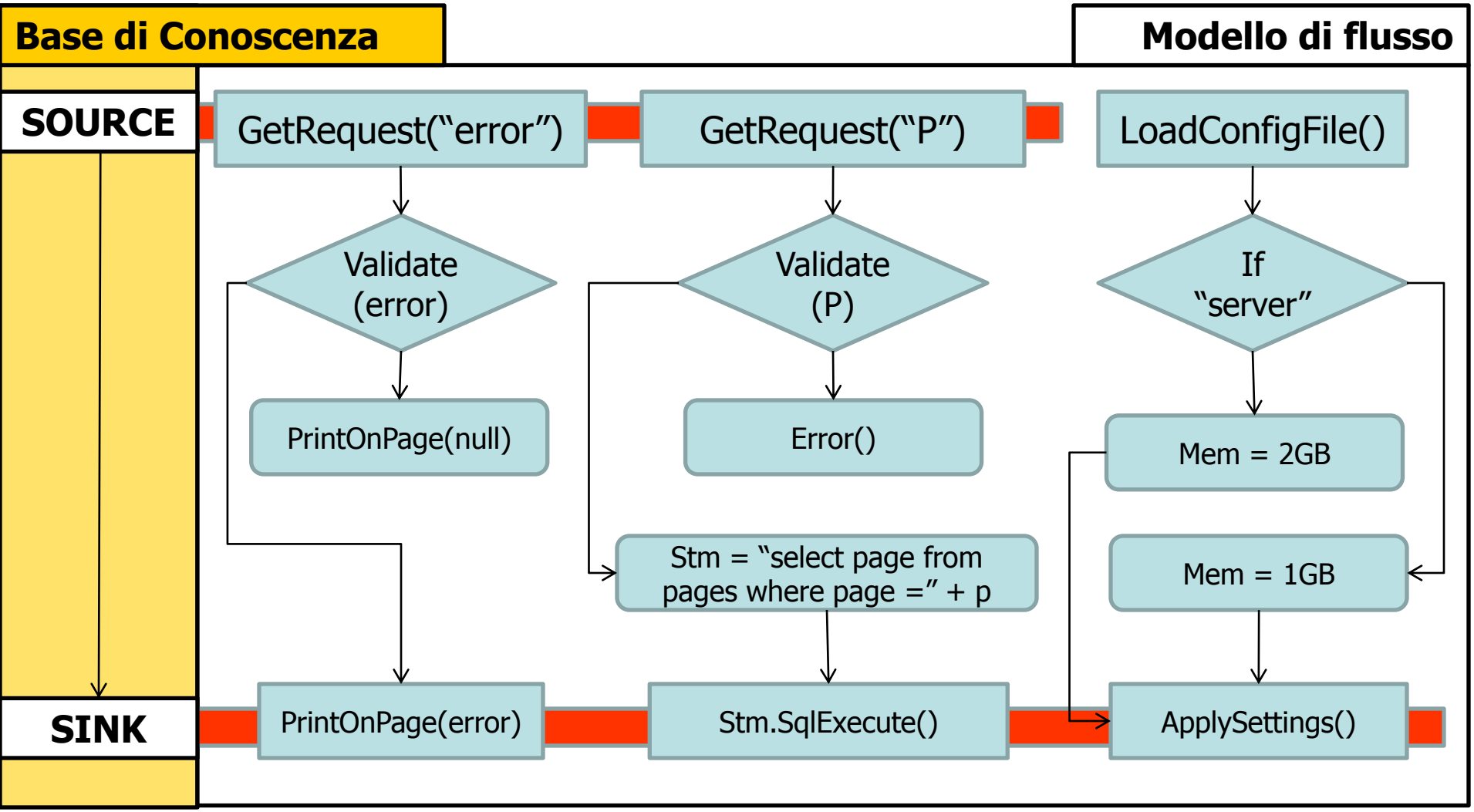
1. Full intermediate Model

2. Optimized Flow Graph

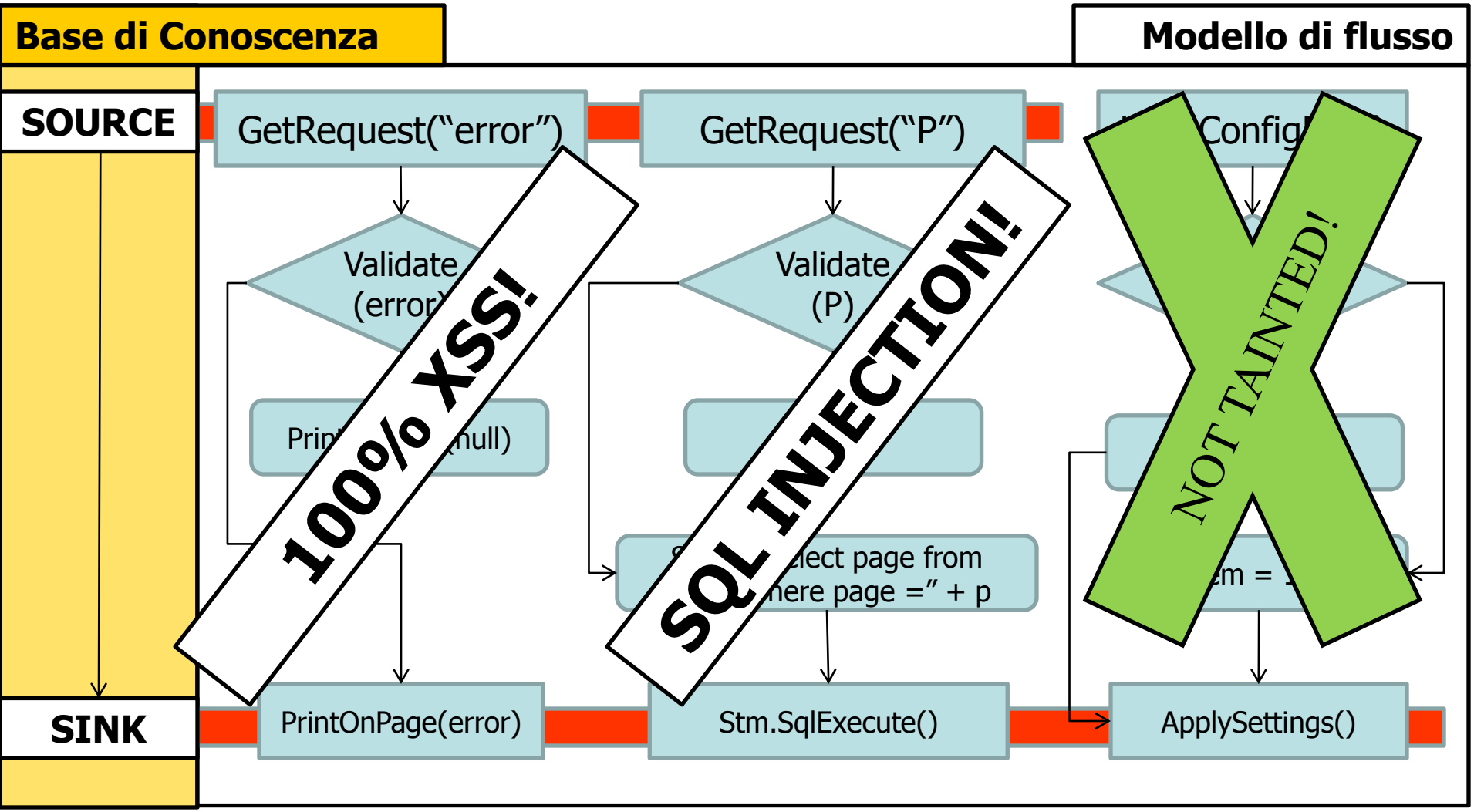
Concetti

- Modello di Flusso
  - Creato da un motore in grado di comprendere il linguaggio che sta analizzando da un punto di vista "*formale*" (assegnazioni, chiamate a funzione, metodi…)
  - Risultato: Grafo di tutti i flussi
- Knowledge Base
  - Solo i flussi controllabili da un attaccante sono di fatto sfruttabili (rischio alto) -> Data Tainting
  - Solo i flussi contenenti funzioni pericolose sono effettivamente "pericolosi"

Source e Sink



Source e Sink



Vendors Marketing Strategy

- Il code review manuale è impensabile perché troppo oneroso
- I tool di code review sono in grado di trovare quasi tutte le vulnerabilità perché "vedono il codice"
- Una volta identificate le problematiche aiutano gli sviluppatori ad effettuare un fixing corretto
- Investire in knowledge base automatica vuol dire risparmiare anche in consulenze esterne
- Sono stati studiati per integrarsi comodamente nei processi aziendali di software security
- Aumentano il controllo sull'operato degli sviluppatori

Speedcubing

- Un individuo sufficientemente allenato è in grado di risolvere un cubo 3x3 in meno di 40 secondi

Manual code review

- Capacità e esperienza permettono ad un consulente di effettuare una code review manuale in un numero contenuto di giorni
- Necessità di numerosi skill
  - Conoscenza approfondita dei linguaggi di programmazione
  - Conoscenza delle problematiche di sicurezza
  - Più ci si allena più si diventa “bravi”
- Come vedremo più avanti è una attività complementare e indispensabile

I tool di static analysis sono in grado di trovare tutte le vulnerabilità?

Il Code Review è l'approccio migliore?

- Code Review (CR)

Creazione del modello dei flussi

↓

Applicazione Knowledge base

↓

Output da analizzare

- Penetration Test (PT)

Interazione con l'applicazione reale

↓

Fault Injection (Abuse Case)

↓

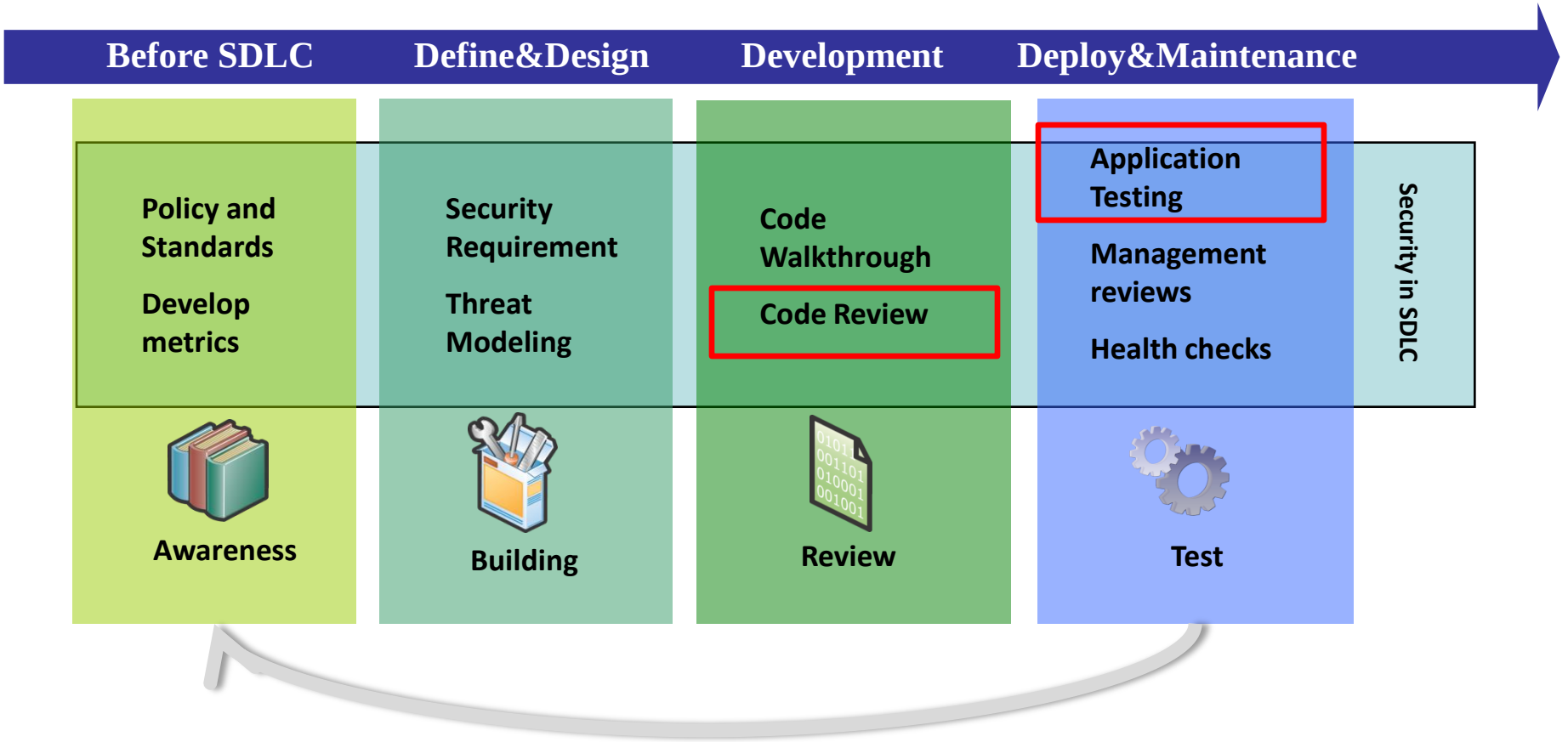
Output da analizzare

Code Review

CR e PT sono approcci complementari

Penetration Test

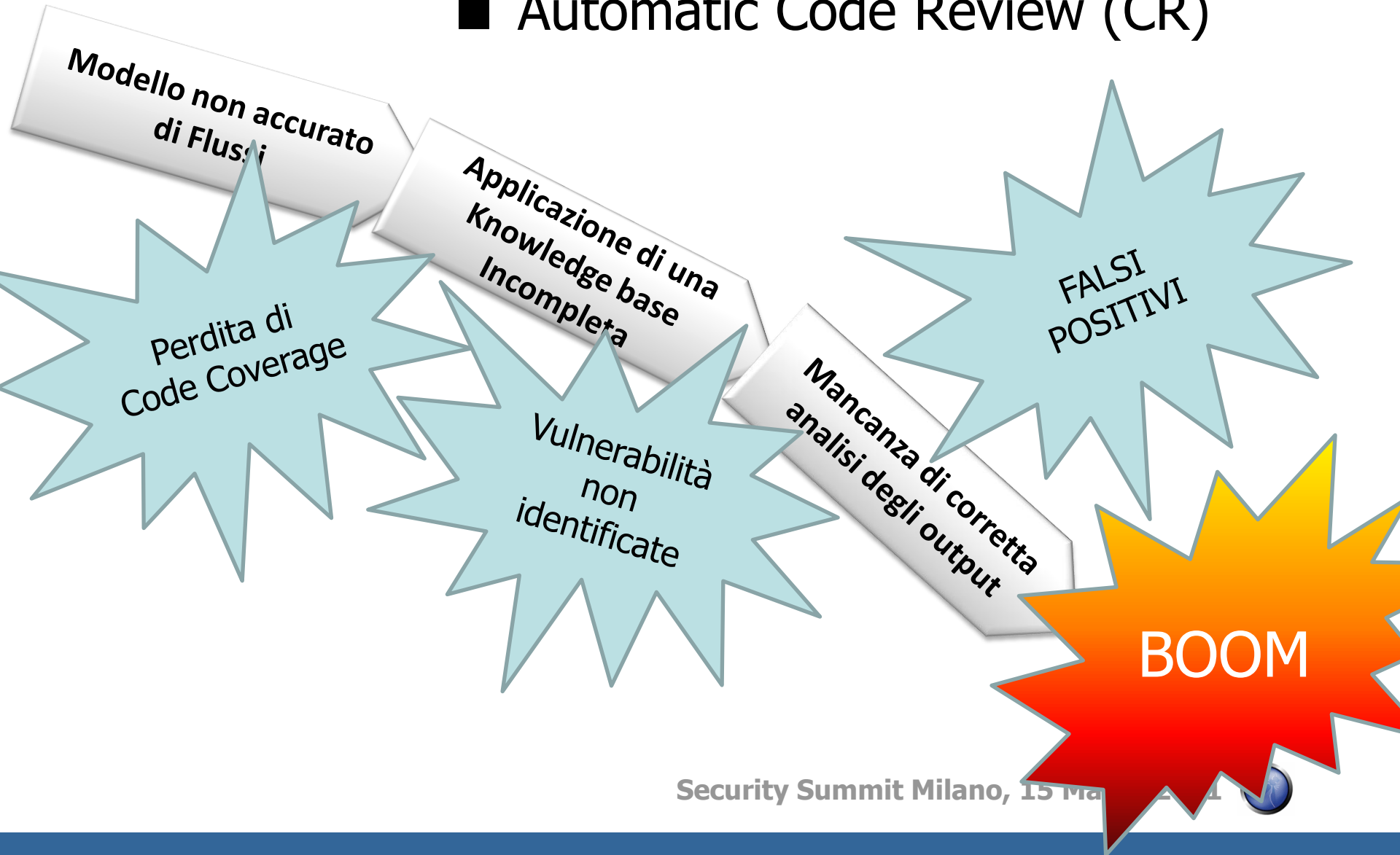
Piano di Sicurezza nello Sviluppo



Il ciclo si ripete su base temporale, in vista delle problematiche riscontrate e dei nuovi sviluppi

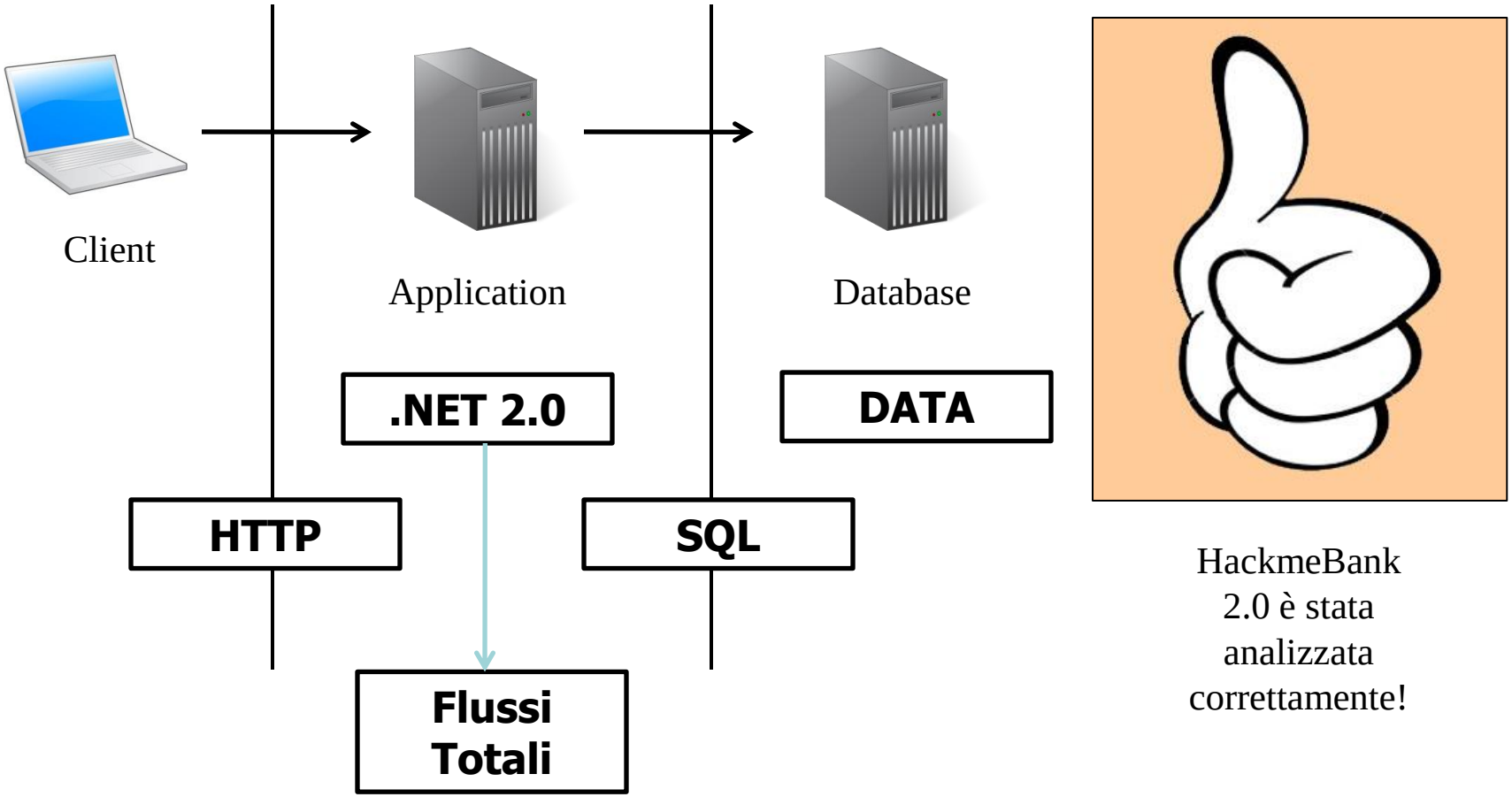
L'approccio automatico ha delle limitazioni

- Automatic Code Review (CR)



Il giorno della Demo

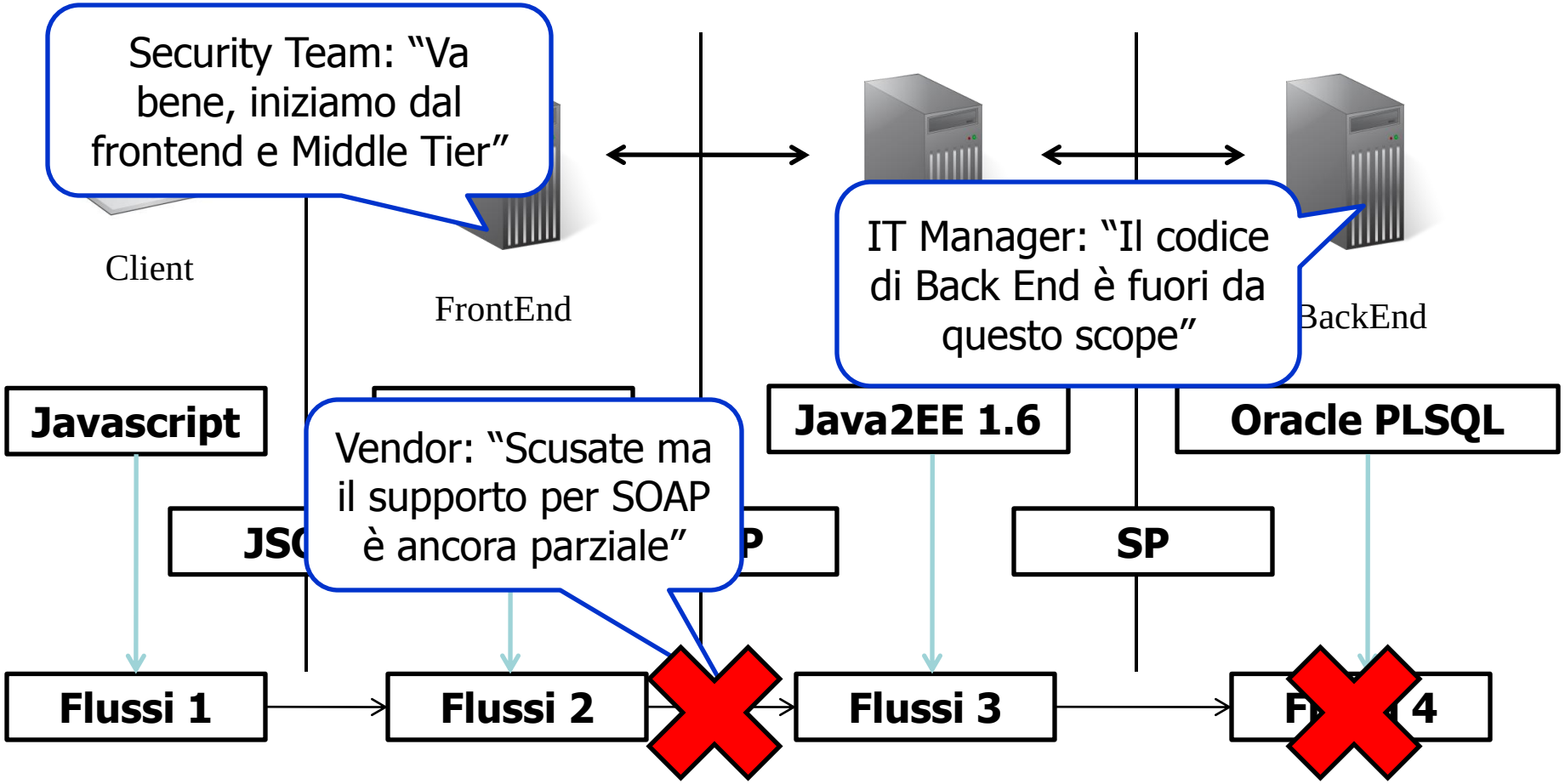
- Modello di flussi esempio “vendor”



HackmeBank 2.0 è stata analizzata correttamente!

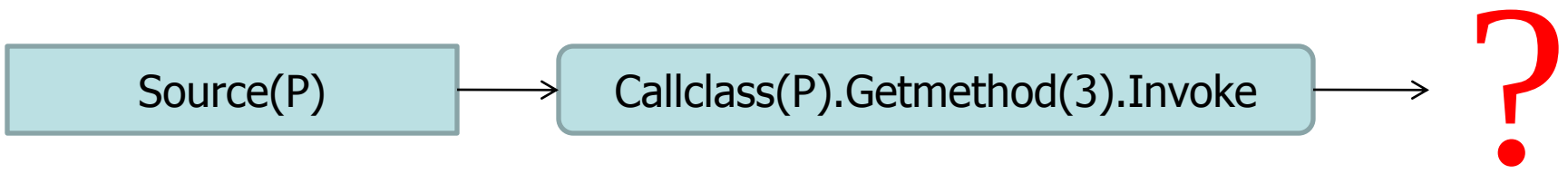
Problemi nella creazione del modello di flusso

- Problemi del Multi-Tier (esempio reale)

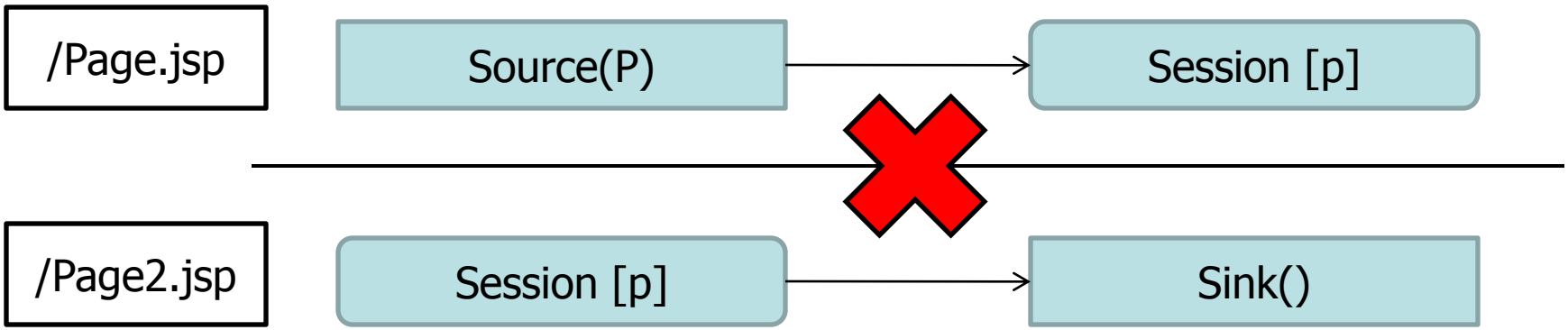


Problemi nella creazione del modello di flusso

- Flussi creati da riferimenti a Runtime
  1. Reflection



  2. Memory Pools (E.g. Session)



Problemi nella creazione del modello di flusso

- Errori dell'interprete
  1. Incapacità di seguire "troppe" assegnazioni

     B = A
     C = B
     D = C ….

  2. Costrutti non supportati

  3. Framework con azioni da file di configurazione

Problemi nella Knowledge Base

- Source Mancanti
  - URL() Completa come input
  - Input da Metodi o Header HTTP
  - Input con definizioni da WSDL per i WebServices
  - Input da canali particolari es. RMI o Socket

- Sink Mancanti
  - Provider verso i Web Services
  - Funzioni custom pericolose di crittografia
  - Riscritture di funzioni pericolose es. Eval() in Java2EE

Attività manuali complementari

- La customizzazione permette di correggere buona parte delle precedenti limitazioni
- Aumentare l'accuratezza del Modello:
  - Creazione di regole custom per i pool ovvero per Session, Viewstate etc.
  - Modifiche al motore di parsing
- Migliorare la Knowledge Base:
  - Prima: Manual Code Review
  - Poi: regole custom per identificare le problematiche viste manualmente

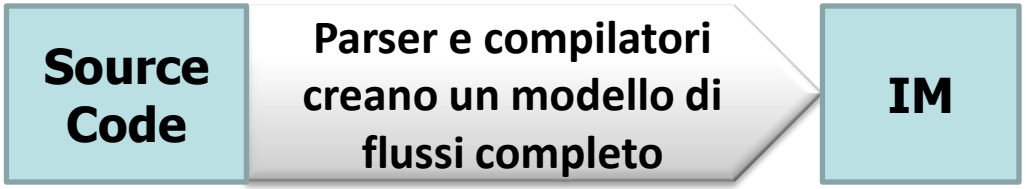
I tool di static analysis sono in grado di trovare tutte le vulnerabilità?

I tool di automatic code review aiutano gli sviluppatori ad effettuare un fixing corretto?

Investire in knowledge base automatica vuol dire risparmiare in consulenze esterne
?

Il tool di SCR e i tempi di esecuzione

- I vendor consigliano di effettuare i check durante lo sviluppo per ottimizzare il processo…



Il tempo per creare un IM completo può raggiungere 2/3 ore

Il tool di SCR e i tempi di esecuzione

- I tool che creano un grafo di flussi sfruttando funzionalità native del framework sono circa 20 volte più veloci

Hanno però diversi svantaggi

- Sono meno flessibili: Supportano di norma un unico linguaggio

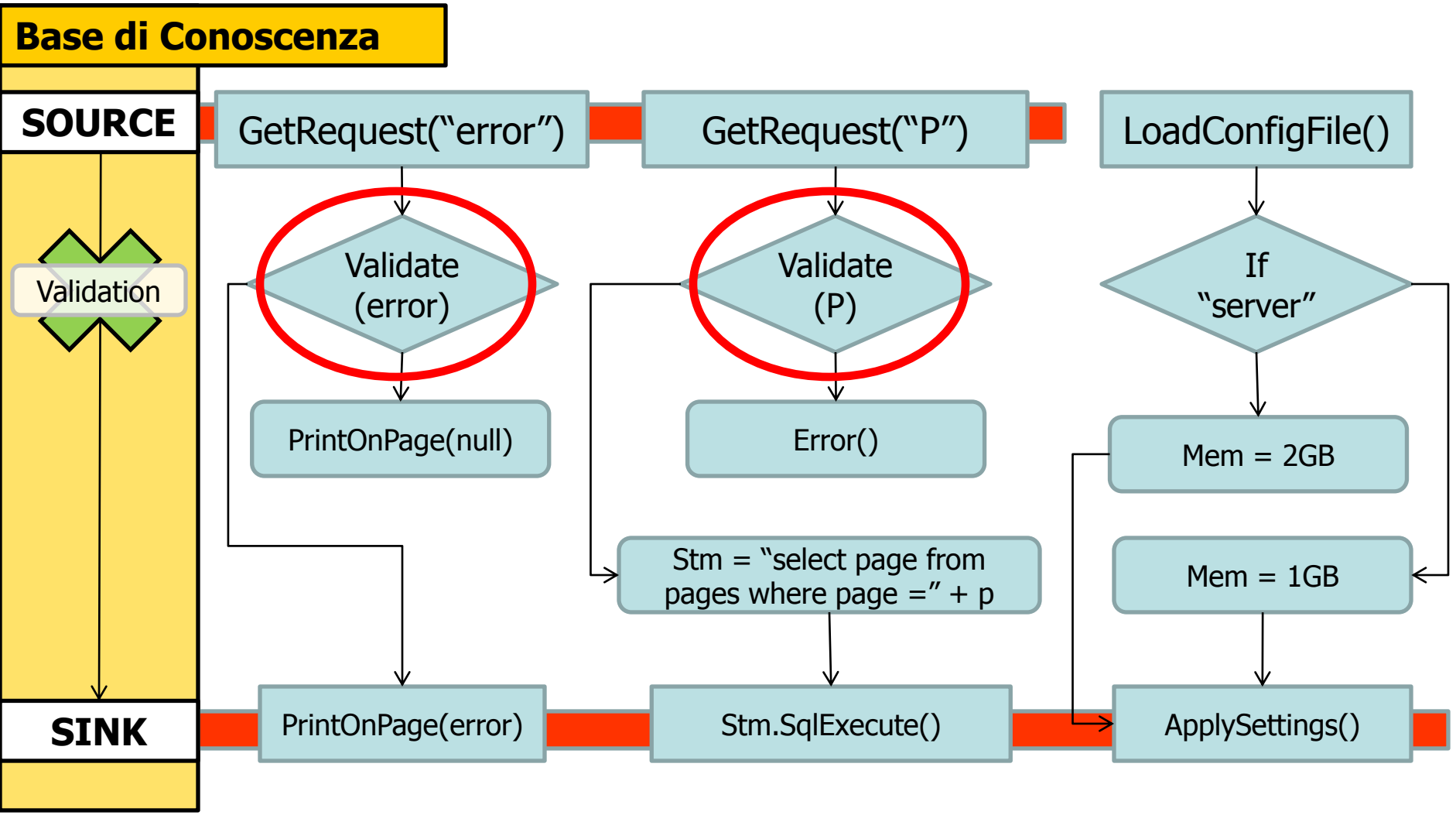
Ma anche un notevole vantaggio

- Alcuni sono freeware! e.g. **CAT.NET + FXCop!**

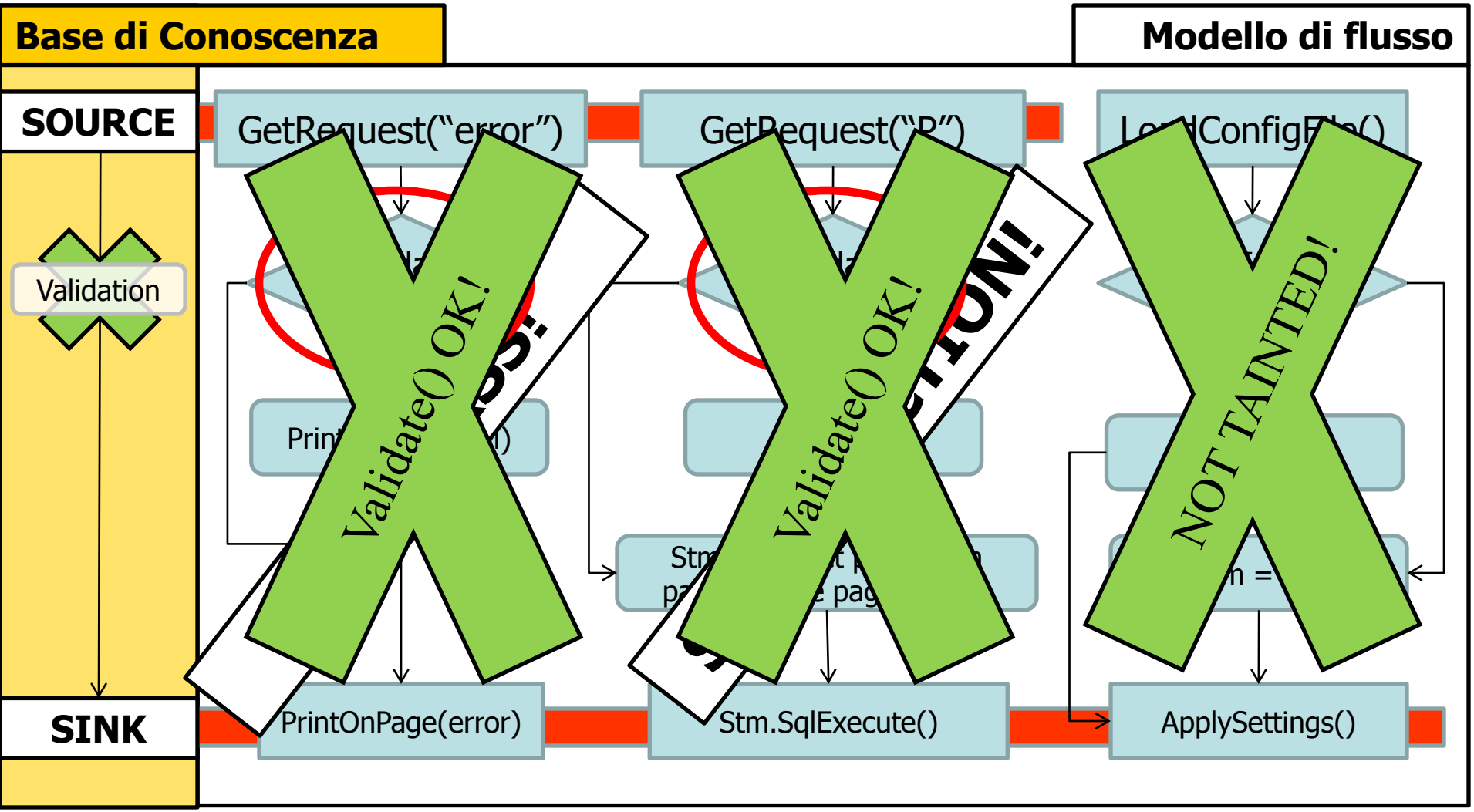
Le regole di validazione

- Le regole di validazione sono “filtri” che considerano il flusso come correttamente validato
- I tool delegano implicitamente al team di sicurezza di creare delle regole di validazione custom e “convenzioni” con il team di sviluppo
- Esempio di regola:
  - Se nel flusso un nodo contiene “*validate*”, quel flusso è sicuro…

No validation points in standard rulesets!



No validation points in standard rulesets!



Le regole di validazione: Il problema

- I tool interpretano unicamente il codice a livello formale, quindi non entrano nel merito del codice di validazione
- Cosa fà "Validate?" In realtà?

```
private void validate(String Input) {
String Output = Input.replaceAll("'", "");
Return Output;
}
```

- E' stato giusto quindi escludere le precedenti vulnerabilità di SQL Injection e XSS?

Attività manuali complementari

- Creazione di regole di Validazione
  - I tool non sono in grado infatti di capire se una vulnerabilità è stata fixata. Il tool interpretano formalmente il codice, non sono in grado (al momento) di emularlo.
- Code Review Manuale delle regole di validazione
  - Attività fra le più importanti!
  - Il tool elimina automaticamente flussi potenzialmente rischiosi basandosi su questi filtri

I tool di automatic code review (da soli) aiutano gli sviluppatori ad effettuare un fixing corretto?

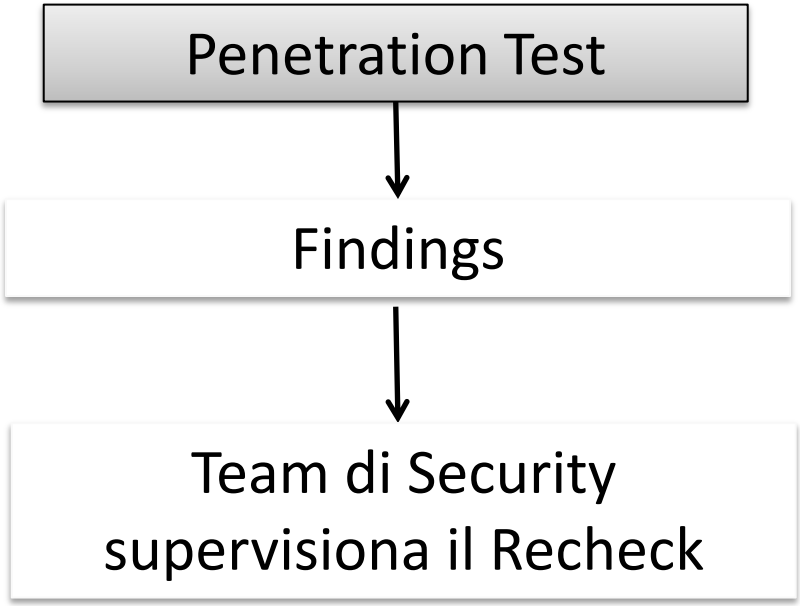
Investire in knowledge base automatica vuol dire risparmiare in consulenze esterne ?

Sono stati studiati per integrarsi comodamente nei processi aziendali di software security ?

Aumentano il controllo sull'operato degli sviluppatori?

Verifica della Remediation e Fixing

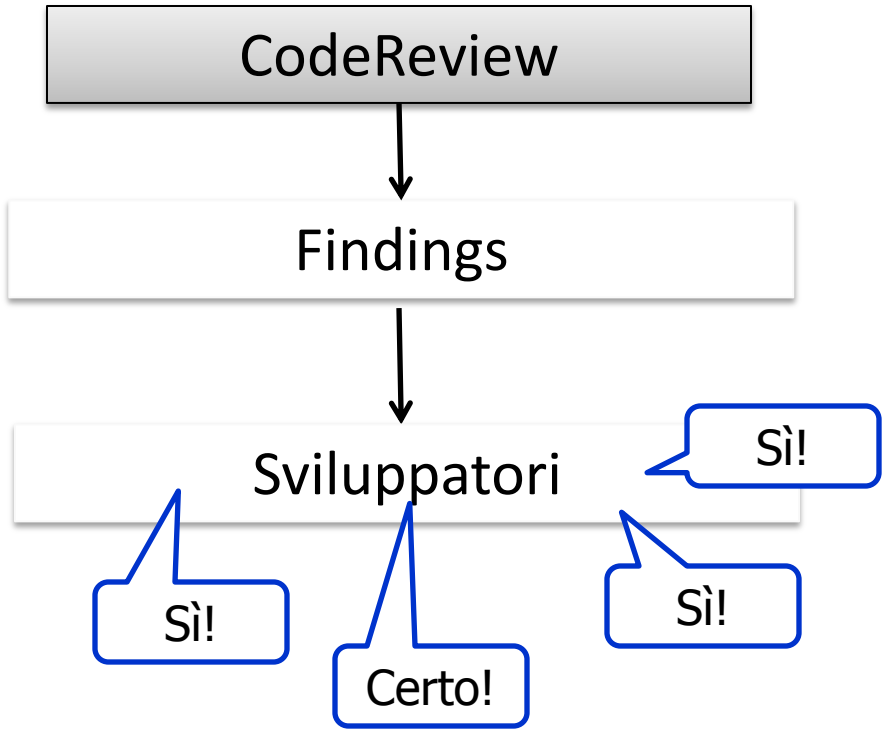
- E' un processo che deve essere supervisionato interamente dal team di security
- Come nel caso del Penetration Test:

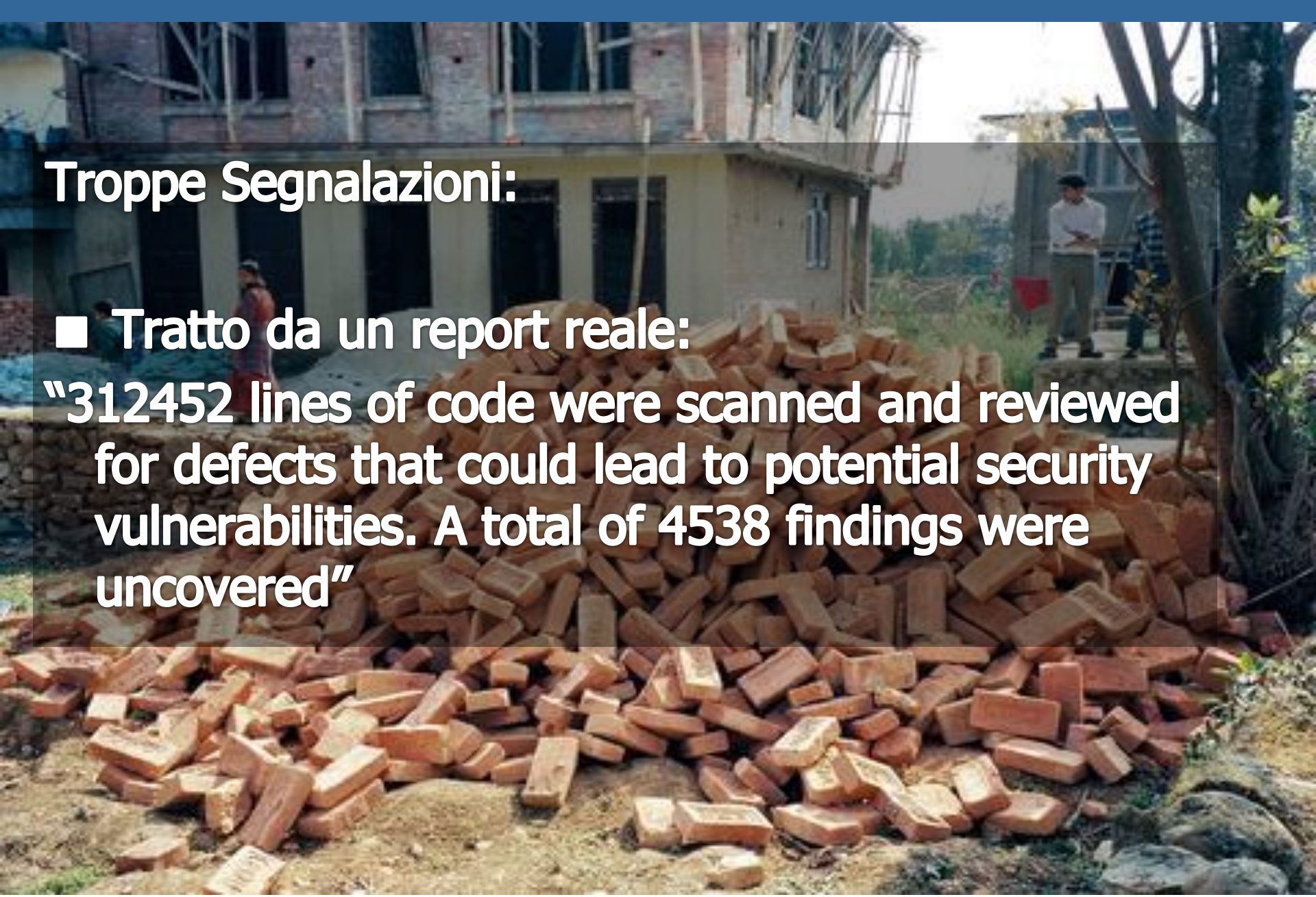


Verifica della Remediation e Fixing

- Nel Code Review è frequente demandato agli sviluppatori l'onere della verifica dei fix

Security Manager





Troppe Segnalazioni:

- Tratto da un report reale:

"312452 lines of code were scanned and reviewed for defects that could lead to potential security vulnerabilities. A total of 4538 findings were uncovered"

Risk Rating Errato

- Il tool dice: Rischio = Impatto * Probabilità
- Corretto? No. La probabilità di accadimento tiene conto degli errori nella generazione del flusso, ovvero che la segnalazione sia un potenziale "falso positivo"

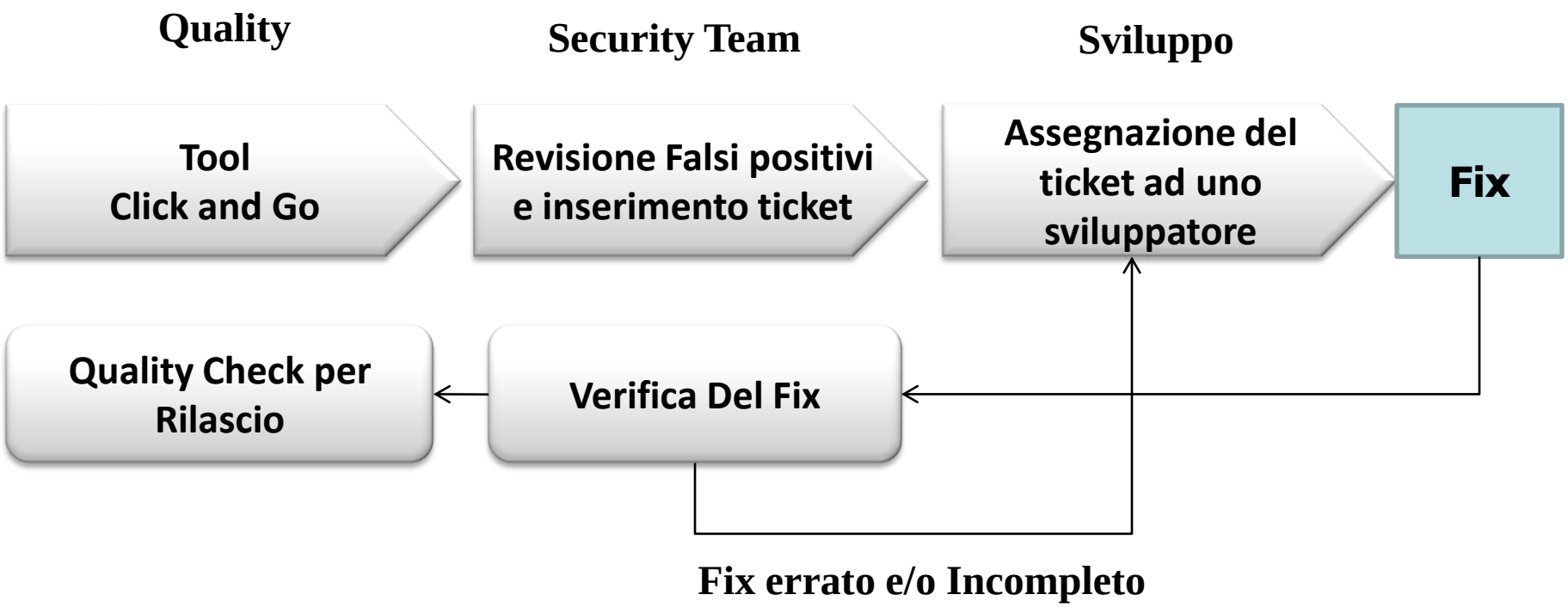
Critical	High	Medium	Low
Redirection To user controlled Site (3)	SQL Injection (4)	Redirection To user controlled Site (2)	Redirection To user controlled Site (1)
Cross Site Scripting (6)		Cross Site Scripting (70)	Cross Site Scripting (20)

Verifica della Remediation e Fixing

- Il team di security spesso non ha accesso al sistema di versioning del codice
- Il sistema di versioning non è integrato col sistema di Bug Tracking
- Ovvero: mancano gli strumenti per verificare che le modifiche siano state effettivamente fatte.

Il processo di verifica nel CodeReview

- La security è un processo che deve essere guidato dal team di security, anche nel code review



Sono stati studiati per integrarsi comodamente nei processi aziendali di software security ?

Aumentano (da soli) il controllo sull'operato degli sviluppatori?

Conclusioni

- Nel caso in cui le competenze non siano presenti internamente si consiglia di esternalizzare le attività seguenti:

1. Verifica della bontà del modello di flusso
2. Creazione di regole custom per i falsi negativi
3. Revisione dei falsi positivi
4. Revisione della robustezza delle remediation
5. Aggiustamento del risk rating dei findings per dare priorità agli interventi

Conclusioni

I tool di code review sono ottimi strumenti,

… ma …

necessitano di personale qualificato che li sappia addomesticare!

Domande

Security Summit 2011