- Convalida del modello architetturale;
- Verifica dell'esistenza di una contromisura per ogni potenziale minaccia identificata.

I tool a supporto di questa fase sono stati identificati nel paragrafo 6.4.2.

9.3 Implementazione di applicazioni sicure

Le azioni di sicurezza che devono essere intraprese in questa fase possono essere così sintetizzate:

- **Data Validation:** verificare la presenza di vulnerabilità che possono riguardare eventuali dati corrotti in ingresso e che possono portare a un comportamento anomalo dell'applicazione;
- **Control Flow**: verificare i rischi collegati all'assenza di specifiche sequenze di operazioni che, se non eseguite nel corretto ordine, possono portare a violazioni sulla memoria o sull'uso scorretto di determinati componenti;
- **Analisi Semantica**: rilevare eventuali problematiche legate all'uso pericoloso di determinate funzioni o API (es. funzioni deprecate);
- **Controllo Configurazioni**: verificare i parametri intrinseci di configurazione dell'applicazione;
- **Buffer Validation**: verificare la presenza di buffer overflow sfruttabile attraverso la scrittura o la lettura di un numero di dati superiore alla reale capacità del buffer stesso.

L'esame del codice sorgente applicativo deve portare alla produzione, mediante la Static Analysis, delle seguenti tipologie di documenti:

- **Report delle Vulnerabilità riscontrate**: report di dettaglio delle vulnerabilità riscontrate nella fase di analisi statica del codice tramite gli strumenti a supporto;
- **Remediation Plan**: report che analizza i falsi positivi ed indirizza la risoluzione delle problematiche riscontrate nell'analisi stessa.

La figura che segue sintetizza gli elementi in input e l'output prodotto dal processo SAST:

Figura 20 - Input e Output della fase Static Analysis

9.3.1 Identificazione degli strumenti a supporto

Nel paragrafo 6.5.1 sono stati presentati i tool a supporto di questa fase. Si riporta di seguito un estratto:

- closed source
  - IBM App Scan (versione SAST),
  - Checkmarx,
  - CodeDx,
  - HP fortify.

- open source
  - SonarQube,
  - FxCop (.NET),
  - BRAKEMAN (Ruby on Rails),
  - PMD (Java, XML e XSL),
  - PYLINT (Python),
  - CppCheck (C/C++),
  - FindBugs (Java),
  - JSHint (Javascript),
  - OWASP Dependency-Check (Java,.NET, Ruby, Node.js, Python, supporto limitato per C/C++).

L'utilizzo combinato dei tool sopra indicati consente una copertura ad ampio spettro semplificando significativamente la revisione manuale che richiederebbe molto tempo.

In Appendice 2 è fornito un approccio metodologico per la valutazione dei tool. L'approccio è basato su una 'Scheda valutazione' che identifica la baseline per l'analisi. I parametri di valutazione includono, ad esempio:

- Linguaggi di programmazione supportati (C/C++, java, JPS, ..);
- Standard supportati (OWASP, Top 10, SANS 25, ..);
- Le vulnerabilità riconosciute (Sql injection, Cross-site scripting, ..);
- L'incidenza dei falsi positivi;
- La capacità di analizzare le dipendenze da librerie esterne;
- Il supporto alla reportistica,
- Altro.

Ogni elemento viene valutato assegnando uno score (da 0 a 10) adeguatamente motivato. La metodologia è stata applicata, a titolo di esempio, su tre tool (i risultati sono indicativi con finalità di linee guida):

Tools	Categoria	Fase	Report
Checkmarx	SAST	Implementation / Verification	Vedi Appendice 2.a
CodeDx	SAST/DAST	Implementation/Verification	Vedi Appendice 2.b
SonarQube/SonarLint	SAST	Implementation	Vedi Appendice 2.c

1) **Checkmarx**, è un tool per l'analisi statica del codice, posizionato da Gartner nel quadrante Challengers nell'ambito dell'Application Security Testing (AST). Supporta numerosi linguaggi (vedi scheda nella tabella di cui sopra). Può essere integrato a vari livelli nell'ambito della fase di Implementation: IDE, build server, bug tracking tools.