ISACA®
Servire i professionisti dell'IT Governance
Rome Chapter

Ciclo di vita del software: strumenti e procedure per migliorarne la sicurezza

Roberto Ugolini

roberto.ugolini@postecom.it

INDICE DELLA PRESENTAZIONE :

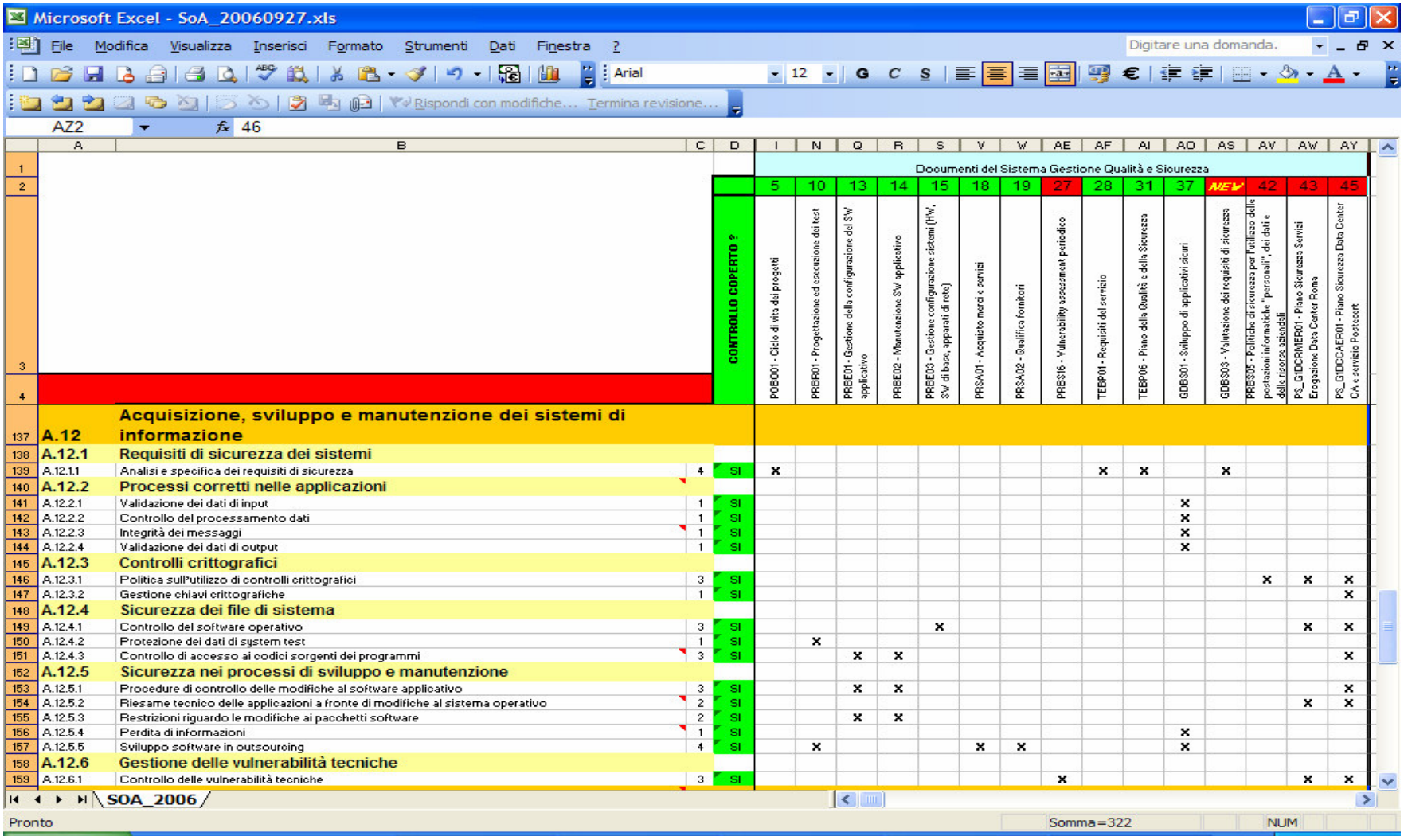
1. Il processo di sviluppo del codice
2. ISO/IEC 27001 e Statement of Applicability
3. Scrivere software (in)sicuro
4. Penetration testing
5. E quindi? SDLC

Il processo di sviluppo del codice

Il processo di sviluppo sicuro del codice (SDLC) è composto da un certo numero di azioni, alcune delle quali supportate da strumenti tecnologici, altre da procedure organizzative.

Il processo di SDLC non può essere pensato come una semplice aggiunta della sicurezza alla fine dello sviluppo del prodotto, subito prima del rilascio.

ISO/IEC 27001 e Statement of Applicability (1/2)



Microsoft Excel - SoA_20060927.xls

File Modifica Visualizza Inserisci Formato Strumenti Dati Finestra ?

Digitare una domanda.

Arial 12

Rispondi con modifiche... Termina revisione...

AZ2 fx 46

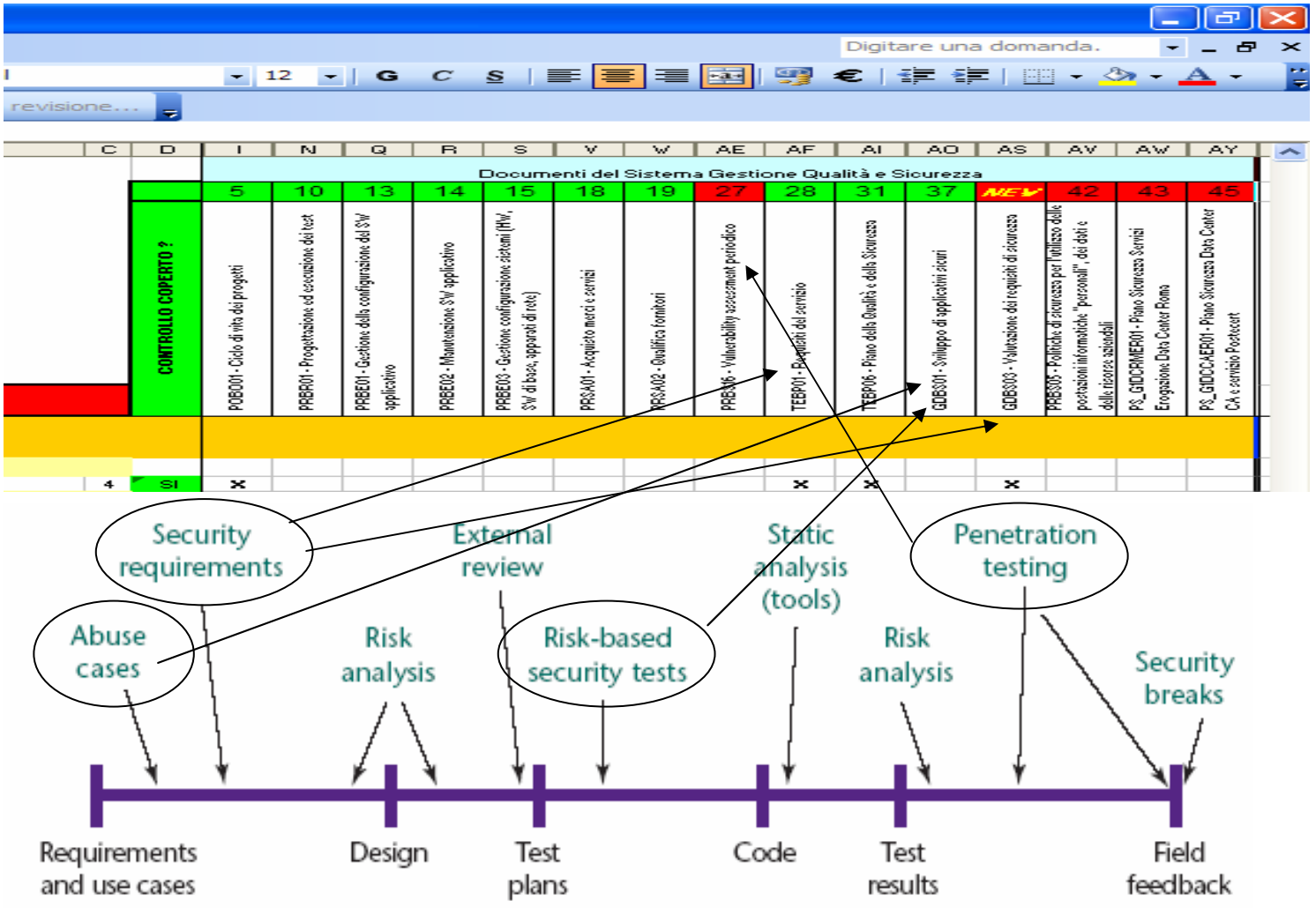
	A	B	C	D	I	N	Q	R	S	V	W	AE	AF	AI	AO	AS	AV	AW	AY
1					Documenti del Sistema Gestione Qualità e Sicurezza														
2					5	10	13	14	15	18	19	27	28	31	37	NEW	42	43	45
3				CONTROLLO COPERTO ?	PO8O01 - Ciclo di vita dei progetti	PRBR01 - Progettazione ed esecuzione dei test	PRBE01 - Gestione della configurazione del SW applicativo	PRBE02 - Manutenzione SW applicativo	PRBE03 - Gestione configurazione sistemi (HW, SW di base, apparati di rete)	PRSA01 - Acquisto merci e servizi	PRSA02 - Qualifica fornitori	PRBS16 - Vulnerability assessment periodico	TEBP01 - Requisiti del servizio	TEBP06 - Piano della Qualità e della Sicurezza	GDBS01 - Sviluppo di applicativi sicuri	GDBS03 - Valutazione dei requisiti di sicurezza	PRBS05 - Politiche di sicurezza per l'utilizzo delle postazioni informatiche "personali", dei dati e delle risorse aziendali	PS_GIDCRIMER01 - Piano Sicurezza Servizi Erogazione Data Center Roma	PS_GIDCCAER01 - Piano Sicurezza Data Center CA e servizio Postecert
4																			
137	A.12	Acquisizione, sviluppo e manutenzione dei sistemi di informazione																	
138	A.12.1	Requisiti di sicurezza dei sistemi																	
139	A.12.1.1	Analisi e specifica dei requisiti di sicurezza	4	SI	×								×	×		×			
140	A.12.2	Processi corretti nelle applicazioni																	
141	A.12.2.1	Validazione dei dati di input	1	SI											×				
142	A.12.2.2	Controllo del processamento dati	1	SI											×				
143	A.12.2.3	Integrità dei messaggi	1	SI											×				
144	A.12.2.4	Validazione dei dati di output	1	SI											×				
145	A.12.3	Controlli crittografici																	
146	A.12.3.1	Politica sull'utilizzo di controlli crittografici	3	SI													×	×	×
147	A.12.3.2	Gestione chiavi crittografiche	1	SI															×
148	A.12.4	Sicurezza dei file di sistema																	
149	A.12.4.1	Controllo del software operativo	3	SI					×									×	×
150	A.12.4.2	Protezione dei dati di system test	1	SI		×													
151	A.12.4.3	Controllo di accesso ai codici sorgenti dei programmi	3	SI			×	×											×
152	A.12.5	Sicurezza nei processi di sviluppo e manutenzione																	
153	A.12.5.1	Procedure di controllo delle modifiche al software applicativo	3	SI			×	×											×
154	A.12.5.2	Riesame tecnico delle applicazioni a fronte di modifiche al sistema operativo	2	SI														×	×
155	A.12.5.3	Restrizioni riguardo le modifiche ai pacchetti software	2	SI			×	×											
156	A.12.5.4	Perdita di informazioni	1	SI											×				
157	A.12.5.5	Sviluppo software in outsourcing	4	SI		×				×	×				×				
158	A.12.6	Gestione delle vulnerabilità tecniche																	
159	A.12.6.1	Controllo delle vulnerabilità tecniche	3	SI								×						×	×

SOA_2006

Pronto Somma=322 NUM

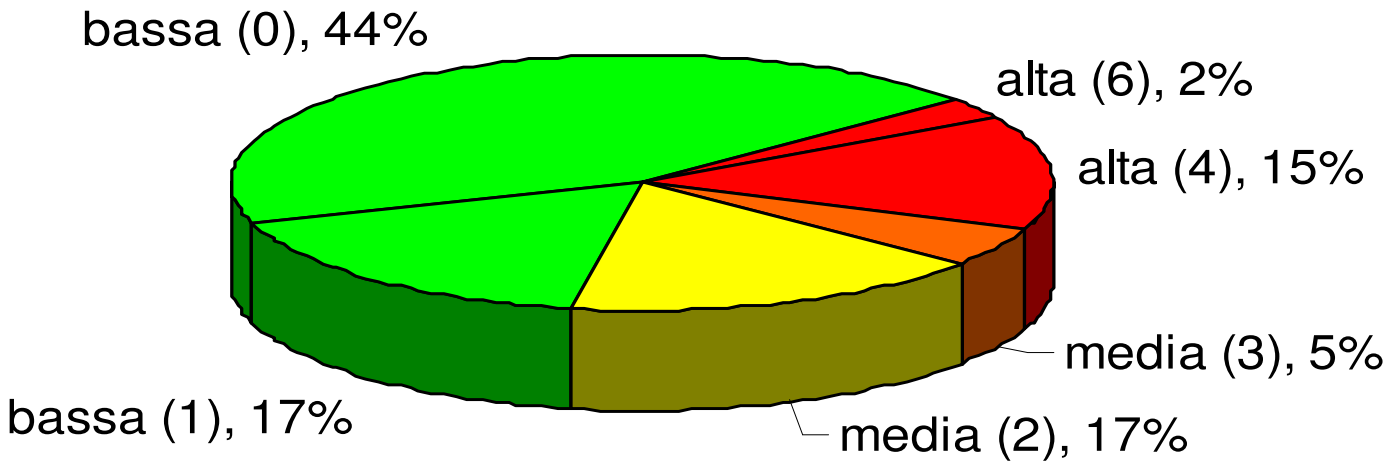
roberto.ugolini@postecom.it

ISO/IEC 27001 e Statement of Applicability (2/2)



Risultati del vulnerability assessment applicativo su servizio web "Critico" effettuato nel secondo semestre 2006

Diffusione vulnerabilità per criticità effettiva (livelli da 0 a 6)



Indipendentemente dal problema della SDLC, dotarsi di un processo di sviluppo del codice che minimizzi gli errori per KLOC (Kilo Line Of Code) permette di ridurre l'incidenza di questo tipo di errori, errori non solo legati alla sicurezza del codice.

Recenti stime stabiliscono nell'ordine di 1 a 7 il numero di difetti per centinaia di linee di nuovo codice, coinvolgendo quindi grandi quantità di codice per progetti complessi, scrivere software insicuro è quindi una certezza!

Questa conclusione comporta che le organizzazioni debbano ridurre di un fattore di grandezza 2 o 3 il numero di difetti nelle varie fasi (specifiche, design, implementazione) per sperare di produrre software sicuro.

La SCD come parte della *software engineering* pratica

- **Team Software Process (TSP) sviluppato dall'Istituto di Software Engineering della Carnegie Mellon University e TSP – Secure**
- **Modelli possibili di design e sviluppo:**
  - **Rational Unified Process (RUP)**
  - **Correctness by Construction sviluppato dalla Praxis Critical Systems**
  - **Metodologia Open Web Application Security Project**
- **Modelli di processo: Capability Maturity Model**
  - **Systems Security Engineering CMM (ISO/IEC 21827:2002)**
- **Linguaggi di programmazione**
- **Pattern di attacco**
- **Software di terze parti**

Principi di Secure Software Design

Economia dei meccanismi. **Il design dovrebbe essere il più semplice e piccolo possibile.**

Impostazioni default a prova di errore. **Le politiche di accesso dovrebbero essere basate sui permessi (deny) piuttosto che sulle esclusioni (allow).**

Mediare ogni accesso. **Ogni accesso a qualsiasi elemento dovrebbe essere controllato da una autorità (modulo software di autorizzazione ad esempio).**

Open design. **Il design non dovrebbe essere un segreto.**

Separazione dei privilegi. **Ove possibile occorre considerare che i sistemi di protezione che fanno uso di più condizioni (es. chiavi), sono più robusti e flessibili di quelli che fanno uso di una sola condizione (es. non solo chiavi crittografiche ma in generale chiavi di accesso ed autorizzazione).**

Privilegi minimi. **Qualsiasi utente del sistema dovrebbe operare avendo il minimo livello di privilegi necessario per compiere il suo lavoro.**

Ridurre i meccanismi di gestione comune delle autorizzazioni. **Il numero di cross correlazioni e dipendenze nelle autorizzazioni fra differenti utenti dovrebbe essere ridotto al minimo.**

Accettabilità Psicologica. **L'interfaccia utente del sistema dovrebbe essere studiata per essere facile da usare e non complicare le operazioni di gestione della sicurezza, in modo da favorire la loro corretta applicazione da parte degli utenti.**

Modellazione delle minacce

Le metodologie di modellazione delle minacce possono essere utilizzate per identificare i rischi e guidare le conseguenti scelte progettuali, sviluppo e testing.
La metodologià viene principalmente usata nelle prime fasi di vita del progetto, usando specifiche semi formali, viste del sistema, flussi di dati, activity diagrams ecc, ma nulla vieta di utilizzarla anche nelle successive fasi, anche a partire dal codice sviluppato. La metodologia è particolarmente importante per l'evoluzione del sistema: al primo sviluppo dell'applicazione permette di indirizzare l'analisi del rischio risolvendo i problemi maggiori, nelle successive evoluzioni permette di indirizzare gli aggiornamenti nella risoluzione delle peggiori situazioni (ad alto rischio).

In generale la modellazione delle minacce permette di suddividere e modellare un attacco al sistema e di scomporlo nelle parti coinvolte.

Strumenti (1/2)

Tool di analisi automatica del codice

- **Tipo di servizio: se client side, servizio centralizzato a cui si invia il codice, analisi statica locale alla macchina, oppure analisi statica di tutto il sistema (considerando anche sistemi distribuiti). Ad esempio alcuni servizi hanno una controparte web che ospita il *reasoner* del codice.**

- **Classe di problemi: tipo di problemi in grado di essere individuati. Le classi di problemi del codice che possono essere individuate staticamente non sono molte, e spesso sono legate al tipo di linguaggio utilizzato (C, C++, Java, J2EE ecc.). Occorre selezionare il tool adatto in base a criteri di aggiornamento e di supporto dei linguaggi maggiormente usati in azienda. Spesso i sistemi più avanzati offrono supporto per differenti linguaggi di programmazione.**

- **Piattaforma OS (Operating System)**

- **Livelli di controllo selezionabili da utente: tipo di severità dell'analisi svolta, selezionabile dall'utente. Questo parametro risulta importante in quanto indice del controllo che l'utente ha sulle analisi svolte, per meglio partizionare i problemi riscontrati, anche in base a considerazioni di sulla riduzione del rischio.**

- **Compilatori supportati: che tipo di integrazione offre con i compilatori esistenti. Questi strumenti si integrano con differenti tecnologie negli ambienti di programmazione ed a differenti livelli. Nel caso in cui vi sia una maggiore integrazione con l'ambiente di programmazione, viene aggiunta una modalità di compilazione che crea file intermedi oggetto (.o) compilati appositamente per poter svolgere i test. In altri contesti invece l'ambiente di esecuzione dei test è proprietario, separato dalle IDE disponibili in commercio.**

Test

standard functionality testing

risk-based security testing

system, user and acceptance testing

penetration testing

Penetration testing (1/2)

Il Penetration Testing è la metodologia oggigiorno più applicata per quanto riguarda la verifica della sicurezza dei prodotti, in parte anche in quanto ha l'interessante (... e pericolosa!) caratteristica di poter essere svolta al termine del ciclo di sviluppo.

Sfortunatamente questo tipo di test però viene svolto da esperti esterni per una durata limitata di tempo e può identificare solamente le vulnerabilità evidenti e superficiali.

Inoltre è troppo legato a fattori aleatori, quali la capacità del tester e la assoluta mancanza di metodologie di test universalmente accreditate.

In generale questo tipo di testing viene di fatto svolto troppo tardi nel ciclo di sviluppo e non da che risultati superficiali: spesso il risultato di questi test è quello di creare una certificazione superficiale del prodotto.

Penetration Testing: manuali o automatizzati?

Possono (e devono) coesistere in un'attività condotta in modo esaustivo

Gli strumenti per Web Application sono ideali nell'identificare vulnerabilità tecniche

L'esperienza e la capacità umane sono invece fondamentali per trovare vulnerabilità logiche e creare "mis-use case"

Strumenti per penetration testing: open source o commerciali?

E' necessaria una combinazione equilibrata di personale esperto e formato e strumenti automatizzati, sia commerciali sia opensource

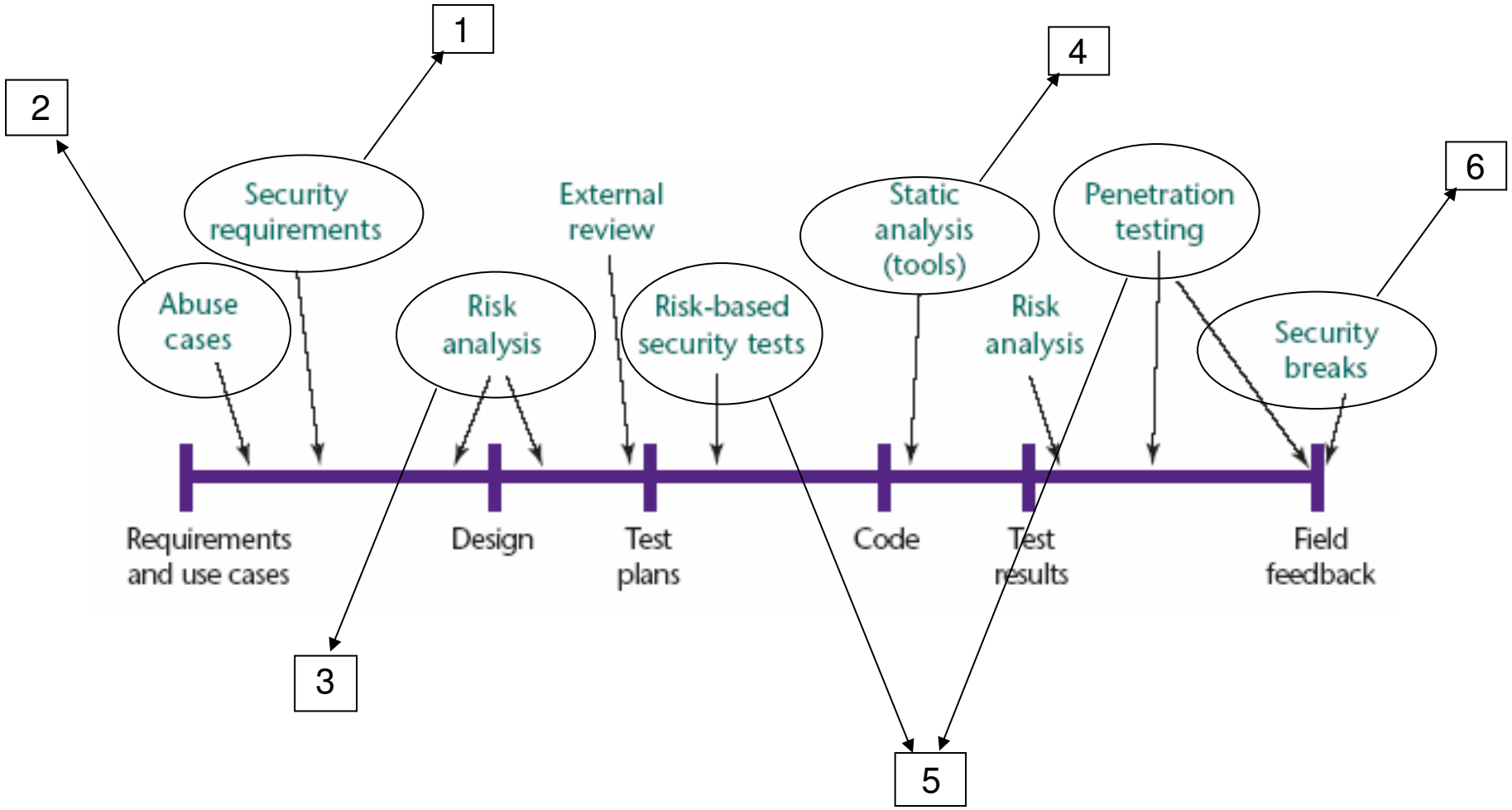
I tool automatizzati aiutano un security tester ad effettuare in modo più rapido e maggiormente completo l'assessment di una web application

Il tool richiede la conoscenza dei suoi limiti e di come padroneggiarlo per essere proficuo

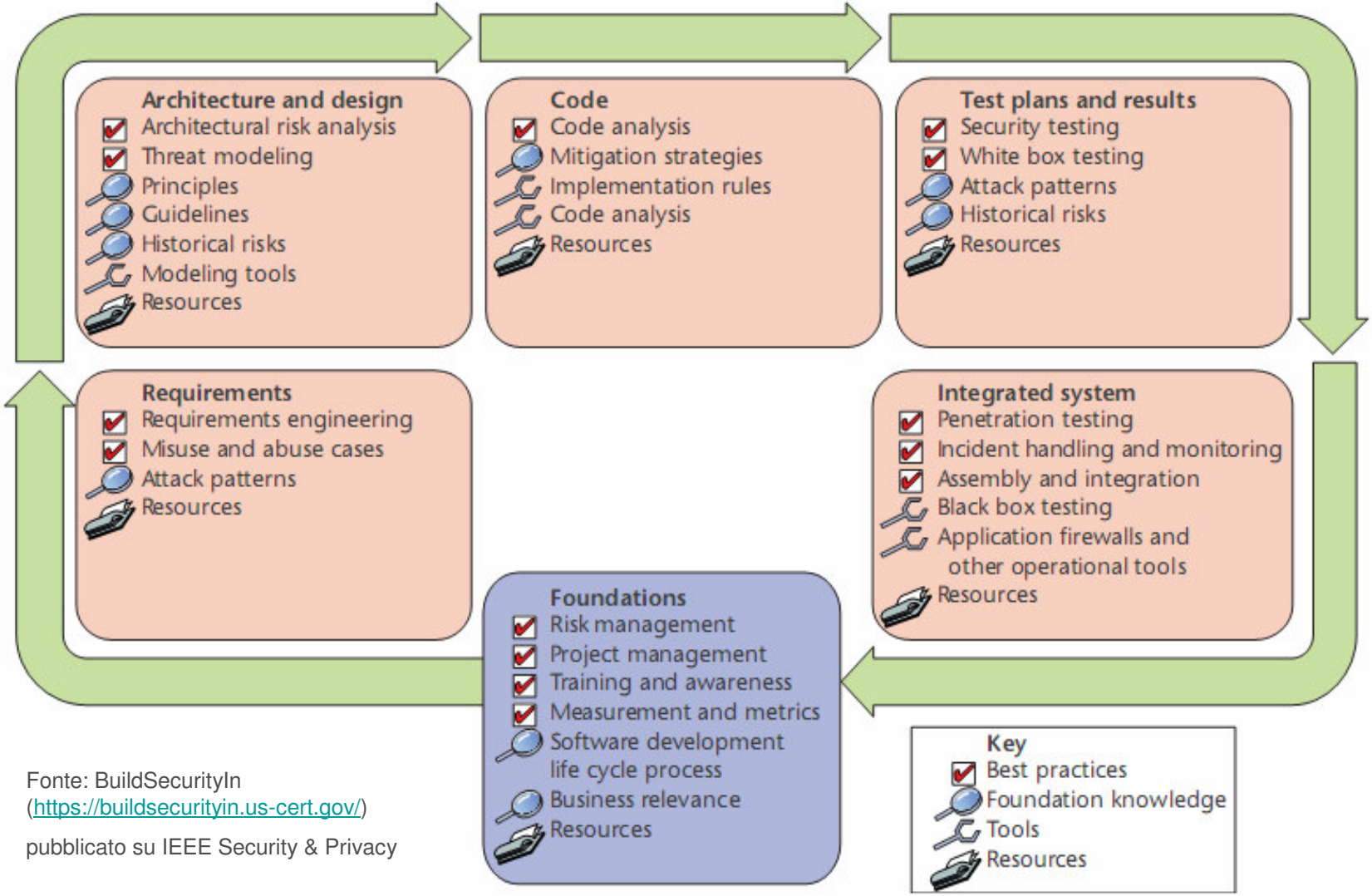
Non sostituiscono l'attività manuale e possono per ciò essere combinati con strumenti opensource

In questo modo è possibile affrontare vulnerabilità tecniche o logiche, ridurre i falsi positivi/negativi e gestire l'assessment di applicativi di grosse dimensioni

E quindi? (1/2)


1
2
3
4
5
6
Security requirements
Abuse cases
Risk analysis
External review
Risk-based security tests
Static analysis (tools)
Risk analysis
Penetration testing
Security breaks
Requirements and use cases
Design
Test plans
Code
Test results
Field feedback

E quindi? (2(2)



Fonte: BuildSecurityIn
(https://buildsecurityin.us-cert.gov/)

pubblicato su IEEE Security & Privacy