modificata in base ai risultati dell'analisi del rischio. In fase di progettazione, si raccomanda l'uso di [6]UMLsec[35]. Per la fase di implementazione, si suggerisce di scegliere un linguaggio di programmazione che meglio soddisfa gli obiettivi di sicurezza. Inoltre, particolare attenzione deve essere posta su come evitare: (i) buffer overflow, (ii) format string vulnerabilities. Essi sottolineano di utilizzare per la crittografia algoritmi già verificati. Per la fase di security assurance vengono indicate le seguenti attività: static vulnerability code scanning, code reviews, ad-hoc unit e system security testing, fuzz testing.

8.2.6 Secure Software Development Model (SecSDM)

SecSDM[36] utilizza l'analisi dei rischi nella fase di specifica dei requisiti al fine di dare priorità alla modellazione delle minacce. Gli obiettivi di sicurezza di alto livello quali la riservatezza, l'integrità e la disponibilità sono poi identificati sulla base delle minacce rilevate [7].

In fase di progettazione, vengono identificate e selezionate le funzionalità di sicurezza per mitigare le minacce e raggiungere gli obiettivi di sicurezza. SecSDM propone di seguire standard di secure coding durante la fase di implementazione.

8.2.7 Software Security Assessment Instrument (SSAI)

SSAI[3738] raggruppa un insieme di attività che utilizzano determinate risorse e strumenti per lo sviluppo di software sicuro. [8] [9] La prima risorsa che SSAI fornisce è un database online[39] che contiene informazioni sulle varie vulnerabilità e le indicazioni per la loro mitigazione. [10] La seconda risorsa SSAI è una security checklist che può essere sviluppata e utilizzata come guida per lo sviluppo sicuro. Sono forniti i dettagli di come redigere una checklist e quali sono gli elementi potenziali che possono essere inclusi[40]. La terza risorsa è un elenco di strumenti accessibili pubblicamente, per la scansione statica del codice. SSAI fornisce anche Flexible Modeling Framework (FMF), uno strumento di modellazione e il property-based testing tool (PBT), che utilizza le proprietà di sicurezza specificate nella security checklist o nel FMF come base dei test per il software.

8.2.8 Hadawi's Set of Secure Development Activities

Hadawi[41] identifica 25 vulnerabilità (common vulnerabilities) da evitare durante lo sviluppo [11]. Egli propone anche una serie di requisiti di sicurezza per le fasi di progettazione e implementazione che, se adottati, aiuterebbero ad evitare queste vulnerabilità [12].

---

[35] J. Juerjens, Secure Systems Development with UML, Springer, 2005.

[36] L. Futcher and R.v. Solms, "SecSDM: A Model for Integrating Security into the Software Development Life Cycle," In IFIP International Federation for Information Processing, Volume 237, Proc. of the 5th World Conference on Information Security Education, Springer, 2007, pp. 41-48.

[37] D.P. Gilliam, T.L. Wolfe, J.S. Sherif, and M. Bishop, "Software Security Checklist for the Software Life Cycle," In Proc. of the 12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03), Linz, Austria, IEEE CS Press, 2003, pp. 243-248.

[38] D. Gilliam, J. Powell, E. Haugh, and M. Bishop, "Addressing Software Security Risk and Mitigations in the Life Cycle," In Proc. of the 28th Annual NASA Goddard Software Engineering Workshop (SEW'03), Greenbelt, Maryland, USA, 2003, pp. 201-206.

[39] DOVES: Database of Vulnerabilities, Exploits, and Signatures, http://seclab.cs.ucdavis.edu/projects/DOVES/. Last Accessed March 2009.

[40] D.P. Gilliam, T.L. Wolfe, J.S. Sherif, and M. Bishop, "Software Security Checklist for the Software Life Cycle," In Proc. of the 12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03), Linz, Austria, IEEE CS Press, 2003, pp. 243-248.

[41] M.A. Hadawi, "Vulnerability Prevention in Software Development Process," In Proc. of the 10th International Conference on Computer & Information Technology (ICCIT'07), Dhaka, Bangladesh, 2007,