# Comparison of SDL and Touchpoints

**Karl Tiirik**

Just as quality cannot be tested into software, software security cannot be achieved by adding security features onto code. Software security must be built in continuously during the application development process. As the regular development processes used to build software offers little in terms of software security, specific processes are needed.

The purpose of this essay is to introduce and compare two high-profile processes for the development of secure software: Security Development Lifecycle (SDL) and Touchpoints. This essay gives a brief overview of both processes and also identifies similarities and key differences between the two.

## 1 SDL

SDL is a software development security assurance process that was developed by Microsoft and designed as an integral part of the software development process at Microsoft. SDL was established as a mandatory policy in Microsoft in 2004 [1]. The goal of SDL is "to minimize security-related vulnerabilities in the design, code, and documentation and to detect and eliminate vulnerabilities as early as possible in the development life cycle" [2].

SDL is based on three core concepts—education, continuous process improvement, and accountability. The continuous education and training of people within a software development group is essential and investment in transfer of knowledge helps to react according to changes in technology and the threat landscape. SDL places heavy emphasis on understanding the cause and effect of security vulnerabilities. In response to new technology advancements or new threats regular evaluation of SDL processes and changes is necessary. Data is collected to assess training effectiveness, in-process metrics are used to confirm process compliance and post-release metrics are used to guide future changes. [3]

Microsoft SDL is a collection of mandatory security activities, presented in the order they should occur and grouped by the phases of the traditional software development life cycle (see Figure 1). The activities can provide security benefit if implemented on a standalone basis, but practical experience at Microsoft has shown that security activities executed as part of a software development process lead to greater security gains [3].

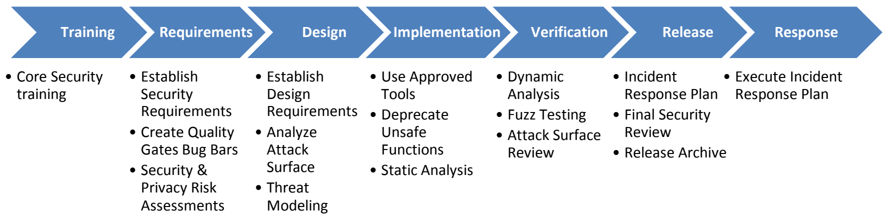

**Figure 1.** The simplified Microsoft Security Development Lifecycle, adopted from [3]

SDL includes general criteria and job descriptions for security and privacy roles. These roles are assigned during the Requirements Phase of the SDL process. These roles are consultative in nature, and provide the organizational structure necessary to identify, group, and mitigate security/privacy issues [3].

As an essential concept, an organization should focus on the quality and integrity of the output produced at each phase. So the exact way the output is produced does not matter. The SDL process does benefit from investments in effective tools and automation, but the real value lies in comprehensive and accurate results [3].

A development team must successfully complete sixteen mandatory security activities to comply with the Microsoft SDL process (Figure 2). Development teams can specify other security activities as necessary, but the sixteen practices must be included [3].

## 2 Touchpoints

Software security touchpoints is a set of best practices. These best practices first appeared as a set in 2004 in IEEE Security & Privacy magazine. Since then, they have been adopted by the U.S. government in the National Cyber Security Task Force report, by Cigital, by the U.S. Department of Homeland Security, and by Ernst and Young [4].

According to McGraw [4] the three pillars of software security are applied risk management, software security touchpoints, and knowledge. He suggests that applying the three pillars in a gradual, evolutionary manner and in equal measure, a reasonable, cost-effective software security program can result.

McGraw described a manageably small set of touchpoints or best practices (Figure 2) that have been formulated over the years out of the extensive industrial experience of its proposer. These touchpoints are based around the software artifacts already produced in a traditional software development process. Although in Figure 2 the artifacts are laid out according to a traditional waterfall model, most organizations follow an iterative approach today, which means that best practices will be cycled through more than once as the software evolves.

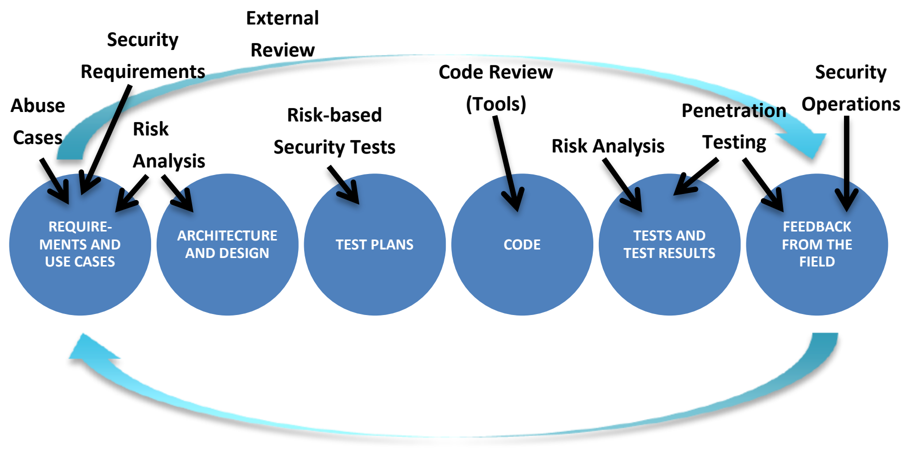

**Figure 2.** Software security touchpoints, adopted from [5]. Circles represent artifacts and arrows indicate the best practices used.

The touchpoints provide a mix of black-hat and white-hat activities, both of which are necessary to get effective results. Black-hat activities are attacks, exploits and breaking software (e.g. penetration testing).White-hat activities are more constructive in nature and cover design, controls and functionality (e.g. code review) [5].

Not all touchpoints have to be adopted to begin to build security in (but is highly recommended). Some touchpoints are by their nature more effective than others, and the most effective should be adopted ones first. Touchpoints in order of effectiveness are [5]: Code review, Architectural risk analysis, Penetration testing, Risk-based security tests, Abuse cases, Security requirements an Security operations.

There is also an additional touchpoint: External analysis which is outside the design team and often a must-have when it comes to security. Software security touchpoints are best applied by people not involved in the original design and implementation of the system.

## 3 Comparison of SDLC and Touchpoints

A summary of comparison between SDL and touchpoints [6] is presented in Table 1. The table includes general comparison and comparison according to development phases. Only the most important aspects are presented (not all similarities differences).

**Table 1.** Comparison between SDL and Touchpoints

| PROPERTY | SDL | TOUCHPOINTS |
|---|---|---|
| **General** | | |
| *Applied to* | Software life cycle phases | Software development artifacts |
| *Adoption* | Standalone or all (recommended) security activities, sixteen mandatory to comply with SDL | Standalone or all touchpoints (recommended) |
| *Process* | Heavyweight and rigid | Quite lightweight and flexible |
| *Team education* | Fundamental part of SDL | Little attention is given to the topic |
| **According to development phases** | | |
| *Project inception* | The personnel involved is organized and roles assigned, security adviser is most important. Determine the type of security bugs that will be addressed. Address the logistics aspects (e.g. tools). Collect and evaluate metrics. | Collect and evaluate metrics, improve measurement strategy. |
| *Analysis and requirements* | Risk analysis based on use scenarios | Anti-requirements and abuse cases, architectural risk analysis, extra security requirements can be identified. |
| *Architectural and detailed design* | Rigorous and thorough threat modeling. At detailed design level: assessing the impact of the project on user privacy and the reduction of the attack surface | The main focus is on threat modeling: threat identification and risk assessment |
| *Implementation and testing* | Apart from secure coding guidelines, SDL lacks real implementation activities. Focus on security testing, mostly black-hat. | Emphasizes the importance of security testing —three out of seven touch points deal with security testing. Both white-hat and black-hat testing. |
| *Release, deployment and support* | Focuses on the response plan defining what to do when a new vulnerability is discovered. Communication with customer important | Touchpoints has very limited support in this phase, only, contribute by fine tuning access controls and configuring the monitoring and logging |

While SDL addresses software life cycle phases, Touchpoints are applied to software development artifacts. Both processes can be applied on a standalone basis (only selected activities/best practices), but it is recommended to adopt them all together [3] [4]. SDL has a comprehensive set of activities and a well-defined process and is in general more rigid, while Touchpoints are very lightweight and can be tailored to the software development process already in use.

In SDL education is fundamental: every team member should have overall education in software security and target members should have advanced education. Education is not part of any touchpoints so little attention is given to it, although it is recognized that people should be sufficiently trained [6].

At the beginning of project inception SDL suggests to make a go/no go decision about the applicability of the methodology to the project at hand. If decision is go, SDL describes in detail how to organize the personnel involved in a secure software project. Most importantly a security advisor is assigned to the project. He helps the developers with security related issues. Other important roles include the development team security contact, the company-wide security team and the security leadership team. Also SDL includes specific activities to address the logistics aspects (e.g. tools) and to determine the type of security bugs that will (or will not) be addressed. Metrics are collected and evaluated during entire life-cycle. Touchpoints has little in place for this phase: emphasize is on the creation, and continuous execution, of an improvement program, so the measurement strategy is constantly improved [6].

In analysis and requirements phase SDL has very few activities. Use scenarios are used for the first step of (architecture-level) threat modeling process. In general Touchpoints covers the broader spectrum of activities. Abuse cases are specified as the combination of a threat agent, an anti-requirement and an attack pattern, together with a mitigating scenario. In addition to the abuse cases, extra security requirements can be identified based on three different sources: laws and regulations, commercial considerations and contractual obligations [6].

In architectural and detailed design phase both processes have a common trait regarding the analysis of threats at architectural level. Both processes suggest the usage of misuse cases to document possible threats. However there are differences in the methodologies proposed: SDL is more precise and thorough than Touchpoints. Both SDL and Touchpoints focus on the security expert that has to scrutinize the architecture [6].

In implementation and testing phase both processes state the importance of security testing and provide good guidance in related activities. SDL focuses mainly on black-hat testing. SDL includes two additional testing activities: the security push and the final security review. Both perform extra testing but have a different focus. The use of automated tools, manual code inspection is encouraged. Touchpoints emphasizes also the importance of security testing, but the focus is a little different: the importance of risk based security testing is stressed. Comparing to SDL Touchpoints uses white-hat activities in addition to black-hat activities [6].

In release, deployment and support phase both processes concentrate on support, rather than deployment. Touchpoints has an exception in this respect, as it contains an activity dealing with the configuration of access control, logging, and monitoring [6].

## References

[1] Evolution of the Microsoft SDL, *Microsoft* http://www.microsoft.com/security/sdl/resources/evolution.aspx

[2] Microsoft Security Development Lifecycle (SDL) Process Guidance - Version 5.0, *Microsoft*, May 16, 2013 www.microsoft.com/download/en/details.aspx?displaylang=en&id=12285

[3] Simplified Implementation of the Microsoft SDL, *Microsoft,* November 4, 2010 http://www.microsoft.com/en-us/download/details.aspx?id=12379

[4] McGraw, G., Software Security: Building Security In, Addison Wesley, 2006.

[5] McGraw, G., Seven Touchpoints for Software Security, 2006 http://www.swsec.com/resources/touchpoints/

[6] Win, B. D., Scandariato, R., Buyens, K., Gr´egoire, J., Joosen, W., On the Secure Software Development Process: CLASP, SDL and Touchpoints Compared, Information and Software Technology archive, Volume 51 Issue 7, July, 2009, Pages 1152-1171 http://dl.acm.org/citation.cfm?id=1539610