Choosing System Security-Engineering Practices : evaluation criteria and a selected survey

Technical Report

Jim Yuill, et al.

2008

Abstract: This technical report is intended to aid computer-system developers in choosing system security-engineering (SSE) practices. The report focuses on codified SSE practices, e.g., Common Criteria and Microsoft's Security Development Lifecycle. Criteria for evaluating SSE practices are presented, including a taxonomy of SSE practices. This report also contains a selected survey of prominent SSE practices, from government and commercial sectors. In the survey, the SSE practices are described using the present report's evaluation criteria.

Evaluating and choosing SSE practices is challenging: there are many SSE practices, and they serve different purposes (e.g., government procurement and product development). Also, SSE practices are made for specific types of systems (e.g., stand-alone products vs. IT systems), and for specific types of security undertakings (e.g., security development vs. security assurance). Further, based on our survey, it is not uncommon for SSE practices to have significant problems and limitations, and those problems and limitations may not be immediately apparent. Using an SSE practice will often require modifying and enhancing it, to make it compatible with a system's purpose and environment, and to work-around the practice's problems and limitations.

This report seeks to address these challenges in evaluating, choosing, and using SSE practices. The objective is to aid computer-system developers in: 1) identifying SSE practices that will be most effective for their systems, 2) determining how an SSE practice will need to be modified and enhanced for their systems, and 3) avoiding SSE practices that will be ineffective or problematic.

Table of Contents:

1 Forward (2019)

This report is from a research project at North Carolina State University (NCSU) in 2008. System security-engineering (SSE) practices were surveyed, to identify practices that could be useful for a campus-wide cloud-computing system. That system was operational at the time, and it was very large and innovative [ABP07]. The system's leaders were interested in identifying SSE practices that would be helpful for further developing the system, for administering it, and for providing security assurance.

This research project also developed principles for evaluating SSE practices. The purpose was to help the cloud-computing system's developers and operators determine which SSE practices would be best for them. The research-project's objective was not to choose SSE practices for the cloud-computing system, but to enable stakeholders to make those choices.

In 2019, now eleven years later, the survey is dated. However, the principles for categorizing and evaluating SSE practices should still be largely relevant, as they are based on the underlying nature of SSE practices.

2 Introduction

This technical report is intended to aid computer-system developers in choosing system security-engineering (SSE) practices. Criteria for evaluating SSE practices are presented, including a taxonomy of SSE practices. Also, a selected survey is presented for prominent SSE practices, from government and commercial sectors. The survey is made using the present paper's evaluation criteria.

As used here, *systems security-engineering* (SSE) is concerned with providing security, and security-assurance, for a computer system. *Security assurance* provides evidence that a system is adequately secure. Two ways of providing assurance are evaluation and certification. SSE spans the entire system life-cycle, including conceptualization, development, production, use, and support. *SSE practices* have to do with the way SSE is carried out, and they include such things as methods, processes, practices, techniques and guidelines. *Codified SSE practices* are SSE practices that have been recorded for others to use. This report is concerned with codified SSE practices, and they are referred to as simply *SSE practices* when the meaning is clear. Examples of SSE practices, from those surveyed in the present paper, include: Microsoft's Security Development Lifecycle (SDL), NSA's System Configuration Guides, the ISO 2700 IT-security standards, and Common Criteria.

Evaluating and choosing SSE practices is challenging. There are many SSE practices. SSE practices are made to serve specific purposes (e.g., government procurement and product development). Also, they are made for specific types of systems (e.g., stand-alone products vs. IT systems), and for specific types of security undertakings (e.g., security system-development vs. security assurance).

In addition, SSE practices tend to change over time. For many SSE practices, corrections and improvements are made periodically, as well as updates for keeping-up with advances in security and technology. A good example is Microsoft's SDL, which is updated frequently on Microsoft's website. (SDL is described in section 8.1 on page 30.) When SSE practices are not updated, their usefulness tends to diminish over time, and they become obsolete. For an SSE

practice that is regularly updated, it can be difficult to find published evaluations for recent versions of the practice. Further, published evaluations themselves can quickly become outdated. Evaluating SSE practices one's self is often necessary, and it can require time-consuming and skilled research.

A further challenge in evaluating SSE practices is that, based on our survey, they often have significant problems and limitations which are not readily apparent. Those who develop an SSE practice may overstate its usefulness, or present it in its best context. Also, the authors of an SSE practice may simply be unaware of assumptions they have made about the practice's purpose and environment, and not realize how the SSE practice would be incompatible with other purposes or environments. For instance, Common Criteria is presented as a general-purpose approach to security assurance, but in practice it's used primarily, perhaps exclusively, in government procurement. Many experts think it is not well-suited for commercial markets. (Common Criteria is described in section 8.2 on page 45.)

A challenge in using codified SSE practices is that they often require modification and enhancement. There is wide variation in systems' security problems and environments. An SSE practice is created for particular purposes and environments, so an SSE practice may need to be modified to make it compatible with the system where it is being used. Also, it may be necessary to work around the SSE practice's problems and limitations. Microsoft's SDL provides a good example of an SSE practice's environmental requirements. Parts of SDL require a relatively large security budget, such as Microsoft's. Consequently, SDL may need to be modified when using it on systems with smaller budgets.

This paper seeks to address these challenges in evaluating, choosing, and using SSE practices. The intention is to aid developers in: 1) identifying SSE practices that will be most effective for their systems, 2) determining how an SSE practice will need to be modified and enhanced for their systems, and 3) avoiding SSE practices that will be ineffective or problematic.

In the following sections, the first two sections address evaluating and choosing SSE practices, respectively. Next, the survey of SSE practices is presented. A final section concludes. The appendix contains detailed reviews of individual SSE practices.

3 Understanding and evaluating SSE practices

This section presents a taxonomy of SSE practices. The taxonomy is intended to provide an overall framework for understanding and evaluating SSE practices. In addition, this section describes attributes of SSE practices that influence their usefulness for a particular system. These attributes are key criteria for evaluating SSE practices.

This section also presents general types and attributes of SSE practices. These generalizations are drawn from our survey of SSE practices. A limitation of these generalizations is that they are based on an incomplete survey. Our survey is extensive, but many practices were omitted in the survey, for lack of time. A broader survey is left for future research, and it would provide a basis for expanding and improving these generalizations.

This section's generalizations are illustrated by examples of actual SSE practices. When an SSE practice is referenced, a bibliographic citation is not given if the practice is described in this report. Instead, a cross-reference is given to the section where the practice is described.

The following sections present, respectively: a taxonomy of SSE practices, SSE practices' purposes and environments, how SSE practices are created, and limitations of SSE practices. A final section discusses SSE practices related to government computer-security mandates.

3.1 A taxonomy of SSE practices

Not only are there many different codified SSE practices, but there are many different *types* of codified SSE practices. This section describes types of SSE practices. A taxonomy is presented, and the categories are derived from our survey of SSE practices. Overall, this taxonomy is an approximation, based on our survey. A more complete and thorough categorization is left for future research. The taxonomy is shown in Figure 1, and it is described in the upcoming sections.

3.1.1 The primary categories of SSE practices

This section describes the primary types (categories) of SSE practices that we found in our survey. The taxonomy's top-most categories of SSE practices are:

- **Government mandates for specific computer-security services:** e.g., HIPAA's privacy requirements [HHS08]
- **Computer-systems security:** codified practices for computer-systems security, e.g., Microsoft's Security Development Lifecycle (SDL) (sec. 8.1), Common Criteria (CC) (sec. 8.2) and ISO 27K (sec. 8.5)

Those two categories are shown in the diagram, in the second level (from the top).

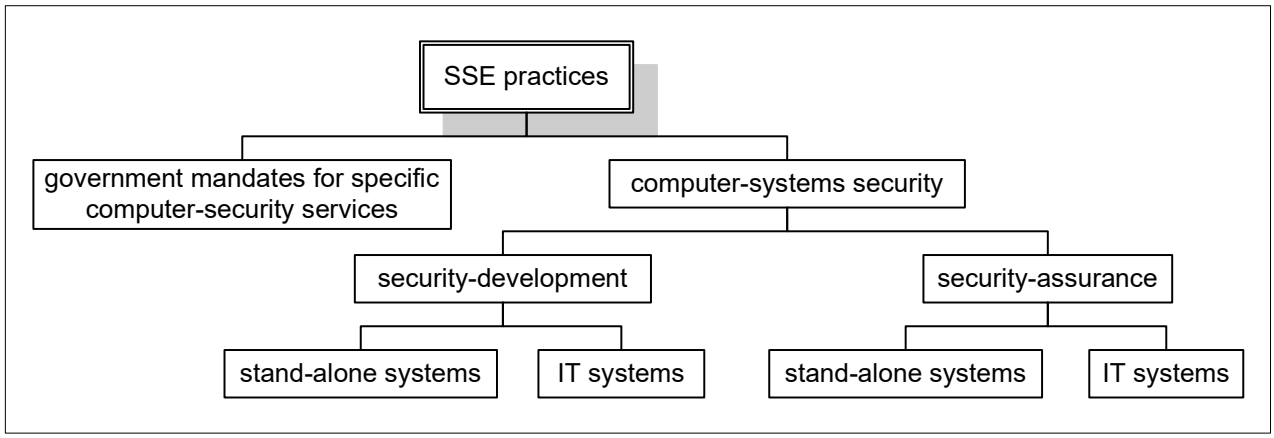

Figure 1 : Taxonomy of SSE practices

The category *computer-systems security* can be sub-divided into two categories of SSE practices:

- **Security-development:** SSE practices for developing systems and system-operations, e.g., SDL
- **Security-assurance:** SSE practices for evaluating and certifying a system's security e.g., CC's security certification

Those two categories are shown in the diagram, in the third level (from the top).

The category *security-development* can be sub-divided into two categories of SSE practices:

- **Stand-alone systems:** SSE practices for developing stand-alone systems, such as operating systems and firewalls. An example of such practices is SDL.
- **IT systems**: SSE practices for developing IT systems, such as a government agency's entire IT system. An example of such practices is ISO 27002 (sec. 8.5).

Those two categories are shown in the diagram, on the left side of the bottom level.

The category *security-assurance* can be sub-divided into two categories of SSE practices:

- **Stand-alone systems:** SSE practices that provide security assurance for stand-alone systems. An example of such practices is CC.
- **IT systems**: SSE practices that provide security assurance for IT systems. An example of such practices is ISO 27001 (sec. 8.5).

Those two categories are shown in the diagram, on the right side of the bottom level.

An SSE practice may address more than one of the bottom-level categories (i.e., leaves) in the taxonomy. However, based on our survey, an SSE practice's primary purpose will typically address just one of the bottom-level categories. For example, CC is primarily concerned with security assurance, but it prescribes some security practices that are also part of security development. In addition, there are collections of SSE practices that address both security development and security assurance, e.g., ISO 27K (sec. 8.5) and NIST's computer-security standards (sec. 8.3). However, based on our survey, these collections typically have separate SSE practices for security development and security assurance. For example, with ISO 27K, ISO 27001 is the standard for certification, and ISO 27002 focuses on system development.

3.1.2 Security-development practices

In the taxonomy of SSE practices, there is a category for security-development practices, e.g., Microsoft's SDL. These practices provide guidance for developing systems and system-operations. In the taxonomy, these practices were further categorized by the type of system to which they apply: a) stand-alone systems, and b) IT systems. Another way to categorize, and understand, the security-development practices is by the scope of the development process that they address. Among the development practices we surveyed, there are five such categories:

- **Security-development methods:**

There are security-development methods for specific components of the system lifecycle. For example, there are a number of risk analysis methods, e.g., OCTAVE (sec. 5.1.3). Also, both SDL and NIST's IT security-standards include security design methods. The methods prescribe development processes, and they address both what is to be done and how.

- **Security-development techniques:**

Security-development techniques prescribe ways to carry out specific development tasks. Examples are secure coding and testing techniques, such as SDL's. In this report, security-development techniques differ from security-development methods in that techniques focus on detailed tasks and procedures, and the methods addresses broader problem-solving processes.

- **Security-controls guidance:**

Security controls are the safeguards or countermeasures used for an information system, e.g., firewalls [Kis06]. They protect the confidentiality, integrity, and availability of the system and its information. Security controls can be management, operational, or technical. Security-controls guidance describes "best practices" for using security controls, e.g., configuring firewalls. NIST and NSA (sec. 5.3) both provide extensive collections of security configuration guides.

- **Security-development lifecycle methodologies:**

There are security-development methodologies that address all, or most, of the system lifecycle. These methodologies are made-up of a comprehensive collection of security-development methods and techniques, e.g., for security design, implementation and testing. Two lifecycle methodologies are Microsoft's Security Development Lifecycle (SDL) and NIST's IT security standards. SDL is used for producing secure products. The NIST standards prescribe the development of IT systems and IT operations.

- **Security-development frameworks for IT systems:**

There are security-development "frameworks" for IT systems. They provide a comprehensive outline of the things that need to be done to secure IT systems, e.g., access control, acquisition, and asset management. These security-development frameworks focus more on what needs to be done rather than on how. (In contrast, the security-development methods address both what needs to be done and how.) Two security-development frameworks for IT systems are ISO 27K and ISF's Standard of Good Practice for Information Security (sec. 5.1.2).

3.1.3 SSE assurance practices

In the taxonomy of SSE practices, there is a category for security-assurance practices. An example of such a practice is CC's security certification. There are a variety of security-assurance practices, and the taxonomy further categorized them by the type of system to which they apply: a) stand-alone systems, and b) IT systems. There are two additional categories of security-assurance practices; however, they are beyond the scope of this research, and were not included in the taxonomy:

- **SSE development processes:**

There is a system for evaluating SSE development processes. It is an extension of the

Capability Maturity Model (CMM), and it is called SSE-CMM [SSE03].

- **Professional certifications:**

There are a number of professional certifications for security engineers, e.g., the Certified Information Systems Security Professional (CISSP).

Ultimately, security-assurance practices provide assurance for specific purposes. Some of these purposes include:

a) Government procurement of computer products, e.g., CC
b) Government IT operations, e.g., NIST's standards
c) Hiring SSE professionals, e.g., professional certifications

3.2 SSE practices' uses and environment

In evaluating an SSE practice for use in a particular application, it is essential to have an accurate understanding of what the practice can be used for, and what environments it can be used in. SSE practices are developed for particular uses, and for particular environments. An SSE practice may not be appropriate for other uses, or in other environments. Alternatively, an SSE practice may need to be modified for other uses, or other environments. Further, in our survey, we found that some SSE practices do not provide accurate and complete statements regarding the uses they can serve, or the environment in which they can be used. For instance, the authors of an SSE practice may have been overly optimistic about the uses of the practice (e.g., as with CC), or they may have built the practice for their own distinctive environment and not fully explained (or realized) the ways in which the practice will be incompatible with other environments (e.g., as with SDL).

This section discusses how SSE practices are built for specific uses and environments, and it discusses ways to assess and evaluate their uses and environments.

a) An SSE practice's uses:

SSE practices typically state their intended uses. Due to the problems described above, it may be necessary to further identify the practice's actual uses. Two ways of doing this are: a) to examine how the SSE practice was created and what problems it was originally intended to solve, and b) to examine how the practice is actually used. Some examples follow.

It appears that CC's intended uses are different than what it can actually be used for. CC is presented as a general-purpose process for security assurance. However, its actual use and environment is government procurement. The CC standard is a government creation, and it is only used when mandated for government procurement. Many experts think it is not suitable for use outside of government.

Microsoft's SDL was created to make Microsoft's products secure, and it was originally made to reduce the high incidence of security problems in those products. SDL may not be useful, or as useful, for other purposes, e.g., making a different company's products secure.

Some SSE practices serve multiple purposes, but these purposes may conflict with each other and reduce the practice's effectiveness. An example is NIST's asset-valuation process

(FIPS 199) [NIS04]. It's primary purpose is asset valuation. However, it appears that the process is also designed to be performed in a relatively simple and objective way that is amenable to evaluation by external auditors (i.e., for security assurance) [Ros07]. However, asset valuation is inherently complex and subjective. There are credible reports that NIST's asset-valuation process is convoluted and not every effective [KS07, Spi07, Tar]. One reason could be its attempt to make asset-valuation objective. In reality, asset-valuation is inherently subjective, by nature (this is a basic tenet of economics, termed the *subjective theory of value*) [Mis49]. NIST's asset-valuation process would not be the most effective in applications that do not have NIST's requirements for objective evaluation by external auditors.

b) The SSE practice's environment:

An SSE practice is developed for particular environments, and it may not be appropriate in other environments. The SSE practice's intended environment will have its own requirements, and they will affect how the SSE practice works. Three distinct environments are:

- Government operations:

By nature, government operations must be managed largely by using objective standards, rather than subjective standards [Mis44]. This is because, in government management, subjective evaluation is vulnerable to abuse. For many security problems, their solutions inherently require subjective evaluation, e.g., determining how much security is enough. Problems that require subjective evaluation often cannot be solved well by government, due to its requirement for objective standards. In contrast, private businesses typically have greater recourse to using subjective standards, which can allow them to operate more effectively than government [Mis44].

SSE practices designed for government-use will be influenced by the government's requirement for objective standards. Consequently, government SSE practices with highly objective evaluation standards may not be appropriate for private businesses, when they can use subjective evaluation standards that are more effective. An example of a problem with inherently subjective solutions is risk assessment. By its nature, risk assessment requires subjective evaluation of assets (i.e., asset-valuation). As described earlier, NIST's asset-valuation process is designed for government use, and the process's problems appear to stem from its attempt to use objective standards for asset-valuation. Further, risk assessment also requires subjective evaluation of threats and vulnerabilities, due to incomplete information and inherent uncertainties.

- Government-mandated SSE practices:

There are several government-mandated SSE practices, such as NIST's standards and CC. Such practices are designed to compel compliance by users, some of whom are not cooperative nor trustworthy. For instance, CC certification must guard against unscrupulous vendors who attempt to obtain certification using the least amount of effort. For a government-mandated SSE practice, guarding against such abuse can create substantial overhead and make the SSE practice very expensive to implement. This overhead can limit the usefulness of government mandated practices and prevent their adoption outside of government mandates. For

instance, such overhead could be one of the reasons CC is only used when mandated for government procurement.

- Corporations:

SSE practices for corporate use will be designed for corporate environments and corporate requirements. For instance, Microsoft's SDL was developed for increasing the security of its products. Much of SDL solves security problems that are specific to Microsoft and its environment. For example, SDL is made for improving security in existing products that are large, complex, and relatively old, and which had substantial security problems. Also, the SDL process is led by a group of full-time security experts who work directly with development teams. SDL may not be appropriate in different environments, e.g., with different types of systems, or in organizations that do not have full-time security experts.

3.3 How SSE practices are created

An SSE practice's usefulness, for a specific application, is also determined by how the SSE practice is created, including:

- How the SSE practice is funded,
- How the SSE practice is developed, and
- Who develops the SSE practice.

An SSE practice's funding affects its usefulness in several ways. The development of SSE practices can be very expensive. Also, SSE practices must be updated on a regular basis, to keep them from becoming outdated. Using an SSE practice can involve a substantial long-term investment in training, adapting it for use, and also in using it over the system's lifecycle. For such an SSE practice, its usefulness will depend upon its future funding and development.

An SSE practice's development, and usefulness, can be affected by the source of its funding and the accountability for its funding. For example, government-mandated SSE practices, such as CC, are funded by government fiat, and used by government fiat. Consequently, they are subject to the standard problems encountered in government management and bureaucracy. For instance, government-mandated SSE practices can be ineffective yet still be widely used simply due to the force of government. This may be the case with CC, judging by the criticism it has received, as described in section 8.2. Another example of how funding sources affect SSE practices is corporate-funded practices, such as SDL. A corporation would only fund development of an SSE practice, and make it publicly available, if that was expected to increase profits (directly or indirectly). Microsoft's motivations for publicizing SDL are discussed in section 8.1.

In our survey, we observed several different sources of funding for the research and development of SSE practices, including:

- Government funding for its own SSE practices, e.g., NIST standards and CC
- Government funding for academic SSE research, e.g., SSE-CMM may be funded this way
- Corporate consortiums, e.g., ISF's Standard of Good Practice for Information Security is funded by member organizations, many of which are corporations

- Corporate funding of its own SSE practices, e.g., Microsoft's SDL. Also, SSE practices used at Cigital are described in a book by Cigital's CTO, Gary McGraw [McG06].
- Individual authors, e.g., authors of books, papers, and blogs, who just represent themselves

How an SSE practice is developed also has a large influence on the practice's usefulness. This includes how the SSE practice is tested, and how it is corrected, improved, and updated. An SSE practice needs to be tested on actual projects. The SSE practice will need to be revised regularly, to make corrections and improvements, and also to keep-up with advances in technology. In our survey we did not find information from the authors of the NIST and CC standards regarding how they tested or used the standards on actual projects, before publishing the standards. Also, we did not find information on how those SSE practices are updated. Such information may well be available, but if so, it does not seem conspicuous. In contrast, the Microsoft SDL team is very open about how SDL has been used, corrected and improved. This information is intertwined with their explanations of SDL. Microsoft very actively corrects and improves its SDL process, and it publicizes these updates.

An SSE practice provides SSE capabilities. The extent of those capabilities are determined by the skill and experience of the practice's authors. To assess an SSE practice, it is very helpful to know the identity, skills and accomplishments of the people who developed the practice. Those who develop SSE practices must be skilled both with SSE and with codifying SSE practices. In our survey we did not encounter descriptive information about the teams that made the NIST and CC standards. (Though, again, that information may be available.) It can reasonably be assumed that those teams are made-up of government employees or contractors. Being unknown to us, their skills are also unknown to us. In contrast, the Microsoft SDL team is well known via its public web-sites, articles and books. The accomplishments of the team and its members are described. Their accomplishments are with systems that are well-known and highly successful.

3.4 Limitations of SSE practices

The SSE practices that we surveyed often had major problems. One reason is that there are inherent difficulties and limitations in codifying SSE practices. For example, there are aspects of SSE that are not amenable to standardization. This section discusses some of the inherent limitations of SSE practices.

One of the challenges in creating an SSE practice is the wide variety of problem-domains encountered. There is variation in:

- The types of computer systems to be secured,
- The types of organizations in which the computer systems are used,
- The threats faced,
- The value of the assets to be secured,
- Security requirements, including risk requirements,
- Security resources available, including budgets, time and technical skills.

An SSE practice's usefulness depends on how widely it can be used. However, given the variation among problem-domains, an SSE practice will typically have to be limited to particular

problem-domains, e.g., government, private-sector, etc.

Also, an SSE practice's problem-domain changes over time. Examples include technological advances and the emergence of new threats and assets. A 20 year-old SSE practice may involve technology that is now obsolete. Economic changes can also affect SSE practices, such as changes in security budgets (e.g., increased government funding) and changes in labor costs (e.g., reduced costs from off-shore labor). Over time, the changes in an SSE practice's problem-domain are likely to reduce the practice's effectiveness, making on-going updates necessary.

Further, an SSE practice may depend on particular aspects of a problem-domain, and in ways the authors did not realize or state. For instance, in reading about Microsoft's SDL process, we found parts of it very difficult to grasp until we realized it was built around specific aspects of the Microsoft development process. As an example, it appears the Microsoft development teams are not expected to have extensive security expertise, and they are aided by members of a security team that serve as consultants.

Another challenge in creating SSE practices is that some security-development processes are not amenable to standardization, e.g., processes that cannot be made into *standard operating procedures* (SOPs). Security development often involves subjective analysis and problem solving, which can be difficult to standardize. As describe earlier, risk analysis tends to be very subjective due to the subjective nature of asset valuation, threat analysis, and vulnerability analysis. Also, SSE can involve multivariable optimization, which typically requires subjective judgement. An example is risk minimization. Further, solutions that require creativity are also not amenable to standardization, as the mental process of creativity is not well understood.

SSE is a relatively new discipline, so for some security tasks, such as security design, well-developed methods may not yet be available [KS07].

As a whole, SSE is concerned with solving complex problems that involve much uncertainty. Codified SSE practices can help with this. However, SSE solutions often require more than codified practices. They also require security aptitude, insight, creativity, experience and wisdom. We expect that SSE practices will be helpful in acquiring basic and intermediate-level skills, but in general, SSE practices alone won't make someone highly skilled.

Expecting too much from SSE practices will likely lead to problems and even failure. Such problems are illustrated by a word of caution from a critic of the Capability Maturity Model (CMM):

> *Still, it has become a lot clearer to me why the CMM philosophy is so much more popular than it deserves to be. It gives hope, and an illusion of control, to management. Faced with the depressing reality that software development success is contingent upon so many subtle and dynamic factors and judgments, the CMM provides a step by step plan to do something unsubtle and create something solid. The sad part is that this step-by-step plan usually becomes a substitute for genuine education in engineering management, and genuine process improvement.* [Bac99]

We expect that SSE practices can be very useful, and even necessary, in developing SSE capabilities. However, choosing SSE practices requires wisdom and an understanding of their limitations. Also, an SSE practice may need to be modified for use in a particular domain, due to the domain's specific requirements and environment.

3.5 Government-mandated SSE practices

There are a variety of government-mandated SSE practices, and this section presents a taxonomy for these practices. The taxonomy is based on our survey of SSE practices. Since our survey was limited, the taxonomy is presented as an approximation—a more complete and thorough categorization is left for future research.

For the U.S., the primary political units are the federal government and the fifty states. Each political unit can mandate SSE practices, within its jurisdiction. The taxonomy of government-mandated SSE practices is shown in Figure 2. In the top level, the SSE practices are categorized by the political units that mandated them.

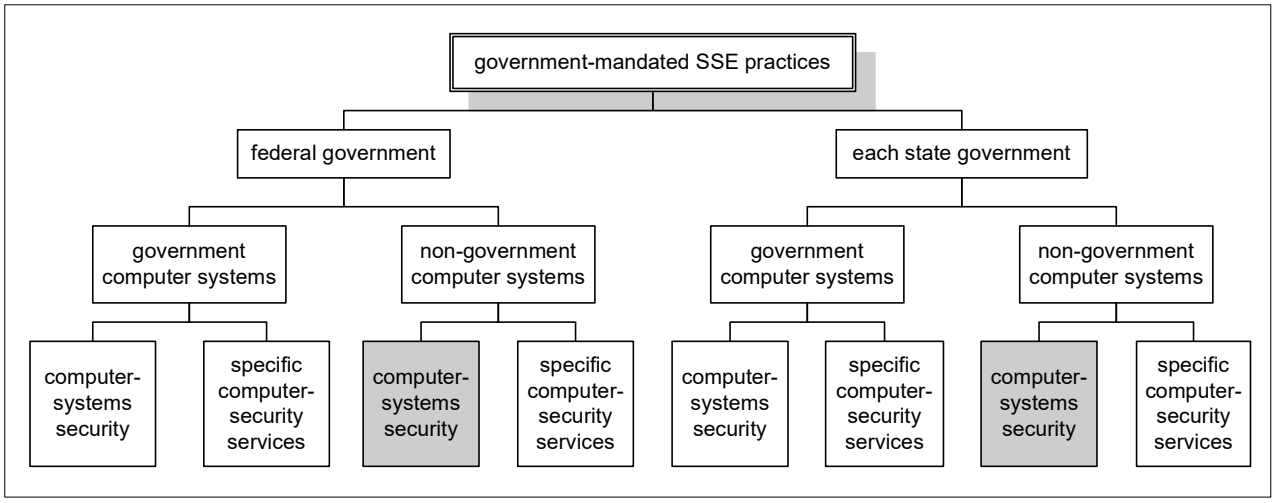

Figure 2 : Taxonomy of government-mandated SSE practices

Overall, government mandates for computer security tend to be quite different for:

- Government computer systems, and
- Non-government computer systems.

These two categories are shown in the taxonomy (Figure 2), in the second level from the top. Mandates for government computer systems may also apply to government contractors' computer systems.

There are two types of SSE practices that are mandated:

- SSE practices for computer-systems security, e.g., CC, NIST's computer-security standards, and ISO 27K
- Government mandates for specific computer-security services, e.g., HIPAA's privacy requirements.

These two categories are shown in the taxonomy (Figure 2), in the bottom-most level. (These two categories are also in the taxonomy of SSE practices, shown earlier in Figure 1 on page 5.)

From our survey, it appears that some of the categories in the taxonomy do not presently

exist. We are not aware of any mandated SSE practices for computer-systems security, for non-government computer systems. In the diagram, this category is shaded in grey, for both federal and state governments. There is an upcoming section on government SSE practices, and it describes the specific government-mandated SSE practices that we found in our survey (section 5.3 on page 20).

4 Choosing SSE practices

This section presents an approach, and principles, for choosing SSE practices for use with a particular system. The first section presents principles for choosing SSE practices, and a second section addresses security planning. A system's security plan provides a framework for choosing SSE practices.

4.1 Principles for choosing SSE practices

Evaluating and choosing SSE practices occurs over a system's lifespan. A system's security plan provides the context needed for choosing appropriate SSE practices. It frames the system's security-problems and solutions. Typical security-plan topics include: security objectives, security requirements, risk analysis, and specification of available resources, including budgets. An overall security plan is outlined in the next section (section 4.2 on page 15). It is based largely on the ISO 27K standards' security-planning guidance [ISO05a].

In choosing SSE practices, their purpose and environment must be compatible with the target system. An SSE practice is developed for a specific purpose and environment, as described earlier (section 3.2, page 8). Codified SSE practices may have nuances in their purpose and environment that are difficult to discern and that limit the practice's use in particular applications. Also, it is likely that SSE practices will need some modification to adapt them for use in the target system. For example, SSE practices made for commercial or government environments would likely need to be modified for use in open-source systems. We strongly recommend that SSE practices be carefully analyzed to ensure their purpose and environment are compatible with the target system.

Codified SSE practices have inherent limitations, as described earlier (section 3.4, page 11). It is important to have realistic expectations of SSE practices. Generally, they will be useful for solving many problems, but not all. They provide useful guidance, but personal skill and wisdom are still needed.

We recommend caution in using government-mandated SSE practices, e.g., CC and NIST's standards, especially when they are not required. These practices will have overhead for protecting against abuse and for meeting bureaucratic requirements for objectivity, as described earlier (section 3.2 on page 8). Also, the practices are funded by government fiat, and used by government fiat, so they are subject to the typical causes of government ineffectiveness. In contrast, SSE practices developed for commercial use will have had to meet a market test for effectiveness, e.g., Microsoft's SDL. Often, these practices are developed by organizations and people that are well-known, accomplished and credible. In general, we prefer commercially based SSE practices, for these reasons.

SSE practices are implemented by people and organizations. The SSE practices chosen for a system must be compatible with the implementers' development capabilities and development culture. Also, SSE capabilities are developed over time, and require training and

experience.

Almost all systems are developed within an IT system. For a system to be secure, the system needs to be developed within a secure IT system. SSE practices are needed to have a secure IT system. Further, there are systems that consist of both stand-alone systems and IT systems. An example is NCSU's Virtual Computing Lab (VCL). Such systems will likely need different SSE practices for their stand-alone systems and for their IT systems.

4.2 Security planning

Effective planning is critical for system security. This section outlines primary issues to be addressed in an overall security plan. Also outlined are primary issues to be addressed in security-requirements analysis and in risk analysis. The focus is on issues that determine what SSE practices can be used for a system, and how they can be used. Further information on developing security plans can be found in the SSE practices surveyed in this report.

4.2.1 Developing an overall security plan

An overall system security-plan includes the topics: a) managing security development, b) the security of the system itself, c) security evaluation and assurance, and d) the SSE practices that are used. For each of these topics, primary issues that affect SSE practices are outlined below. Most of these issues are taken from the ISO 27K standards, and in particular, its section on critical success factors for information security [ISO05a].

- **Managing security development:**
  - Leadership:
    - Management must support and be committed to the security plan.
    - The security plan must be effectively marketed to all of the system's stakeholders.
  - Direction:
    - Security objectives and activities should support the system's overall objectives.
    - Security-requirements analysis and risk analysis are foundational for security planning.
  - Project management:
    - Budgets and schedules are needed.
    - For long-term planning, an estimate is needed for the system's future development.

- **Security of the system itself:**
  - The system's security should be based on security-requirements analysis and risk analysis.
  - On-going processes are needed for incident management and for fixing vulnerabilities.
  - Over time, the system's security will need to be updated for system modifications (e.g., enhancements) and for changes in system risks, i.e., changes in assets, threats, and vulnerabilities.

- **Security evaluation and assurance:**
    - The system's security-assurance requirements must be determined. Security assurance is needed by the system's stakeholders, including those who produce and develop the system, and those who will deploy and use the system.

- **SSE practices:**
    - A set of SSE practices are needed for the system's development lifecycle. These practices include implementing, maintaining, monitoring, and improving security.
    - SSE practices also need to be selected for security assurance. Security assurance can be provided by evaluating the system's security. Evaluation includes activities such as testing, monitoring and analysis. Security assurance can also be provided by using good SSE practices, and by demonstrating they were used.
    - A plan is needed for developing and improving the system's SSE practices, both in the short and long term.
    - The system's SSE practices must be compatible with the organization that develops the system, including the organization's management, development culture and development capabilities.
    - SSE practices are performed by people, so developing SSE practices includes recruiting, training and education. SSE skills and capabilities are developed over time.

4.2.2 Security requirements and risk analysis

Security-requirements analysis and risk-analysis are critical parts of security planning. This section provides an outline of the primary issues addressed in that analysis. Most of this outline is taken from the ISO 27K standards, and in particular, its introduction to these topics [ISO05a]. Further guidance in security-requirements analysis and risk-analysis can be found in the SSE practices recommended in this report.

- **Security requirements:**

An essential part of security planning is developing the system's security requirements [ISO05a]. Four of the primary sources of security requirements are:

- Risk analysis,
- The objectives and requirements for the system itself,
- Customer requirements for security and security-assurance,
- Any legal, statutory, regulatory, or contractual requirements that need to be satisfied by the system's stakeholders, including those who develop the system and those who use the system.

- **Risk analysis:**

Risk analysis identifies risks to system assets. Risks for an asset are calculated as a function of three things: 1) the asset's value, 2) its vulnerabilities, and 3) the threats it faces. For example, when a very valuable asset has essentially no vulnerabilities and no significant threats, the risk is low. When a very valuable asset has very little vulnerability, but is earnestly

attacked, the risk is high. Risks are identified for the system's stakeholders, including the people who develop the system and the people who use the system. For each of the system stakeholders, risk analysis takes into account their overall objectives with the system.

Two of the primary uses of risk analysis are:

- Security requirements are identified by a methodical analysis of security risks.
- The risks to an asset determine what is an appropriate amount of security for the asset. Risk-analysis results are used to allocate finite security resources in a way that best reduces overall risk.

During a system's lifetime, risk analysis should be repeated periodically to address any changes in assets, vulnerabilities and threats. Also, risk analysis is categorically different when developing general-purpose systems (e.g., Windows or the Apache web server) and when developing a system for a particular end-use (e.g., NCSU's payroll system). A general-purpose system is used in a variety of ways. Among these uses, there is relatively wide variation in the system's assets, vulnerabilities and threats. Risks must be assessed for the variety of ways the general-purpose system will be used. In contrast, systems developed for a particular end-use have much more specific assets, vulnerabilities and threats.

In summary, in choosing effective SSE practices, it is necessary to understand such things as the practice's use and purpose, how it was created, and its limitations. Also, the SSE practice must be compatible with the implementer's development capabilities and development culture. Security planning is critical for choosing SSE practices, as it provides the framework for understanding a system's security problems and solutions.

5 Survey of SSE practices

This section summarizes our survey of SSE practices. Summaries are given for major categories in the taxonomy of SSE practices. The taxonomy figure is repeated here, for reference (Figure 3). In our survey, we encountered many SSE practices, and examined three of them in depth: Common Criteria (CC), NIST's IT security standards, and Microsoft's Security Development Lifecycle (SDL). The other practices were investigated to varying degrees of depth.

The practices that appeared to be the best developed and most useful, from our survey, are:

- Microsoft's SDL is a comprehensive set of security-development practices for the system lifecycle, for stand-alone systems. We thought it was the best-developed set of SSE practices that we found, though it does have some weaknesses and limitations, which we describe.
- ISF's "The Standard of Good Practice for Information Security" provides an outline for IT-systems security. It is very well developed. It is also well funded, and sponsored by a consortium of organizations, from commercial and government sectors.
- NIST's IT security standards provide an extensive set of security-controls guidance that may be very useful. NIST also has security-development practices for IT-systems, but some of these practices appear to have significant shortcomings.

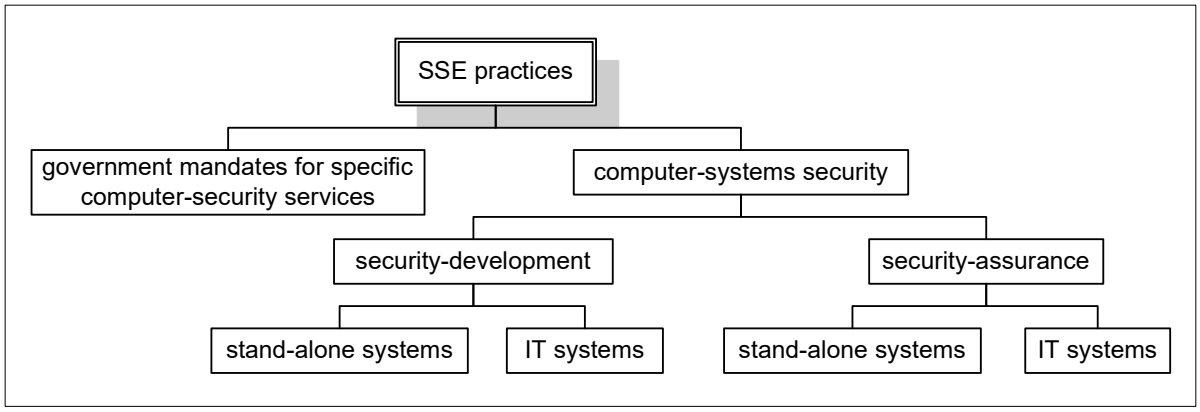

Figure 3 : Taxonomy of SSE practices (from section 3.1 on page 5)

The survey is summarized in the following sections. SSE practices for security-development are presented first, followed by SSE practices for security assurance. A section on government SSE practices addresses both mandated and non-mandated practices.

5.1 SSE practices for security-development

This section discusses SSE practices used for security-development. The first two sub-sections are on SSE practices for stand-alone and IT systems, respectively. They are followed by a sub-section on risk-analysis practices. Risk analysis is used for both stand-alone and IT systems, though there are some differences in the way risk-analysis is performed for each of them.

5.1.1 SSE practices for stand-alone systems

Microsoft has implemented a security-development process called the *Security Development Lifecycle* (SDL). It is a comprehensive collection of SSE practices for the whole development lifecycle. It is the best-developed set of SSE practices we have found. SDL was created for developing Microsoft's products, but it appears to be well-suited for typical stand-alone systems in which security is important. An extensive description of SDL appears in the appendix (section 8.1 on page 30), and much of that section addresses adapting SDL for use outside of Microsoft.

The book *Software Security*, by Gary McGraw, presents a set of SSE practices for most of the development lifecycle [McG06]. We have read some of this book, and its practices may be very helpful. Currently, there are a number of books on SSE practices for specific parts of the system development lifecycle, such as secure coding and testing techniques. Investigation of these books is left for future research. Such books can readily be found on Amazon.com.

5.1.2 SSE practices for IT systems

There are quite a few comprehensive SSE practices for IT systems, and they address the whole IT-system lifecycle. Of the practices we reviewed, most are government SSE practices: ISO 27K, NIST's IT security standards, DoD's DIACAP standard, and NSA's IATF program. They are described in the upcoming section on government SSE practices (section 5.3 on page 20). We reviewed one non-government standard: ISF's, and it is described below. Apparently

Microsoft has an SSE practice for IT called SDL IT. However, it appears that very little has been published about it.

In summary, our primary findings regarding SSE practices for IT systems are:

- ISF's "Standard of Good Practice for Information Security" provides a thorough outline of the issues that need to be addressed for IT security. ISF is a consortium of IT organizations, from business and government. So, it has the advantage of being privately funded and subject to market demands. This is ISF's primary standard, and it is freely available. Based on our limited investigation, this standard may be a better choice than the ISO 27K standards, if ISO 27K certification is not needed. Further information on ISF's standard follows, below.
- The ISO 27K standards appear to be well developed and have been used a lot. They are not as extensive as the NIST standards, but provide a thorough outline of the issues that need to be addressed for IT security. The ISO 27K standards must be purchased, and the two major standards are about $30 each. The ISO 27K standards are further described in the upcoming section on government SSE practices.
- NIST's IT security standards provide an extensive security-development methodology for the IT lifecycle. They also provide a large collection of security-controls guidance. NIST's standards are the most extensively developed set of SSE practices we found for IT security. Overall, its practices for risk analysis and security design appear to have significant problems, but its extensive security controls can be very useful. These standards are free. NIST's IT standards are further described in the upcoming section on government SSE practices.

Further information on ISF's "Standard of Good Practice for Information Security" follows. The Information Security Forum (ISF) is an international non-profit consortium of over 300 organizations, including over 50 of the Fortune 100 companies [ISF08a]. Its members fund and cooperate in developing practical information-security research, including best-practices, methodologies and tools. The cost of membership is somewhere around £16K a year [ISF].

ISF's primary SSE practice is its "Standard of Good Practice for Information Security" (hereafter referred to as *ISF's standard*) [ISF08b]. ISF has spent $100M developing it, over 18 years. It is a comprehensive set of information security-specific controls. They also include controls for complying with legal and regulatory requirements, such as the Sarbanes-Oxley Act. In addition, it covers the main security controls in other major standards, such as ISO 27002. ISF's standard is used by many organizations as the basis for their internal information-security standards and guidelines. They also use it for meeting compliance obligations. For example, Sun Microsystems's IT security is based on the ISF standards [Dix06]. A credible website on security standards says that ISF's standard "has long been well regarded as a broadly-scoped pragmatic standard for information security." [Ise08a].

ISF's standard can be freely downloaded. It is 375 pages. We perused the standard, and it appears to be very useful by itself, without additional material that would only be available to ISF members. It has a section entitled, "Information for Non-ISF Members", that indicates the standard is intended to be of use to non-members.

5.1.3 Risk analysis

Risk analysis is one of the primary elements of security planning, as described earlier (section 4.2.2 on page 16). Our survey of risk analysis practices was very limited, but promising

sources are recorded here for future research:

The well-known security researcher Peter Neumann wrote a seminal paper on risk analysis that is still very relevant and useful today [Neu78]. It provides a cogent and very effective framework for understanding and performing risk analysis. These principles can be applied to both stand-alone and IT systems.

The ISO 27001 Security website has a lengthy annotated list of risk analysis methods [Ise08b]. Regarding this list, they state, "it is difficult to recommend a particular method or tool without knowing more about your organization in terms of its experience with risk analysis and information security management, size/complexity, industry and so on."

One of the more visible risk analysis methods is OCTAVE (Operationally Critical Threat, Asset and Vulnerability Evaluation) from Carnegie Mellon University's Software Engineering Institute [OCT08].

BSI is the German Federal Office for Information Security. Its document "IT Security Guidelines" offers an alternative approach to risk assessments [BSI07]. They argue that "an all-embracing risk assessment is not always necessary. . ." They observe that risk assessments are very expensive, and that "experts with appropriate know-how are needed. The relevant input variables, such as the probability or extent of damage, are very difficult to ascertain and at best can only be calculated roughly." They offer an alternative solution: "BSI's IT Baseline Protection Manual. . . builds on the fact that a majority of the IT systems and applications used in practice are run in a similar fashion and in comparable operational environments."

5.2 SSE practices for security-assurance

In general, when an SSE system-development practice is government-mandated, then a security-assurance practice must also be mandated, i.e., for evaluation and certification. We looked at three government-mandated security-development practices for IT systems, and they all address both security development and security assurance. They are: NIST's standards for IT security, ISO 27K, and the DoD's DIACAP standard. They are described in section 5.3 (page 20).

Common Criteria is a government-mandated security assurance practice for stand-alone systems. It too is described in section 5.3 (page 20).

The DoD sponsored the development of the Systems Security Engineering Capability Maturity Model (SSE-CMM). It is a security-assurance practice that focuses on assuring the quality of the security-development process. SSE-CMM is based on CMM. CMM is a software assurance practice, and it focuses on evaluating the quality of software-engineering practices. In our initial investigation of CMM, we found three persuasive papers which argue that there are fundamental flaws in CMM [Bac99, BM91, CF02]. Further research of CMM and SSE-CMM is needed to investigate these flaws and their possible impact on SSE-CMM.

5.3 Government SSE practices

This section describes current government SSE practices. Both mandated, and non-mandated practices are described, but the primary focus is on mandated practices. An example of a mandated practice is CC, which is required for government procurement, and an example of a non-mandated practice is NSA's security configuration guides, which appear to be just

recommendations. A taxonomy of government-mandated SSE practices was presented earlier (section 3.5 on page 13). The taxonomy describes general types of mandated practices, and it is used here to provide a context for describing currently-mandated practices. (In the present section, when the context is clear, government computer-security mandates will be referred to as *computer-security mandates,* or simply, *mandates*.)

When developing a system that may be regulated by government computer-security mandates, it is necessary to know which government mandates, if any, are applicable. This can be difficult as there are disparate sources of such government mandates, and the mandates change over time. Also, given the current political-processes, government mandates tend to grow in number, scope and complexity. Consequently, books and papers on government computer-security mandates are quickly outdated. The best source we've found for tracking government computer-security mandates is the Unified Compliance Framework (UCF) [UCF08a]. UCF is produced by a consulting company, but much of the UCF is freely accessible. The UCF is a compilation of computer-security mandates and standards from around the world. It is the most comprehensive and up-to-date source on mandates that we have found. The UCF includes a list of the mandates and standards that it supports, and this list alone is very useful for knowing what mandates and standards presently exist [UCF08b].

Another useful source is a survey of USFG computer security mandates, from the Congressional Research Service (CRS) [CRS04]. However, this survey was written in 2004, so parts of it may now be outdated. The survey is summarized here, using the taxonomy of government computer-security mandates from section 3.5 (page 13). In 2004, the USFG had mandated SSE practices for both computer-systems security (e.g., CC and NIST's standards) and for specific computer-security services (e.g., HIPAA's privacy requirements). The mandated SSE practices for computer-systems security only applied to government computer-systems (and to government-contractors' systems). For non-government computer-systems, the USFG had not mandated SSE practices for computer-systems security. On the other hand, the USFG does have mandates for specific security services, for both government and non-government computer systems. The CRS survey reports that,

> *There are currently no general federal requirements for private entities other than federal contractors operating systems for the federal government to secure their computer systems. However, there are requirements for entities who hold or process certain types of personal information to ensure the confidentiality of that information. To date, this includes financial information and medical information . There is also a federal requirement that certain firms that register with the Security and Exchange Commission (SEC) must include in the financial reports an assessment of their internal financial controls. To the extent that each of these types of information is held and or processed electronically, the security of some private computer systems come under federal regulation* [CRS04].

The National Defense Industrial Association (NDIA) has developed a guidebook for systems assurance, and it is dated January 2007 [NDI07]. It has a chapter on government computer-security mandates, including those from the DoD services (Army, Navy, etc.). That chapter contains an extensive collection of mandates, with descriptions of each.

In our survey of SSE practices, we investigated several government SSE practices, including mandated practices, and they are briefly summarized here. For most of them,

additional information is provided in the appendix.

- **NIST's SSE standards:**

The National Institute of Standards and Technology (NIST) issues computer-security mandates for USFG IT systems. These mandates apply to all USFG civilian agencies. The mandates prescribe an extensive collection of SSE practices for both security development and security assurance. The security-development practices include risk analysis and security design, as well as guidance for a large number of security controls (e.g., firewalls). The security-assurance practices include evaluation and certification. From our survey, it appears NIST's SSE practices for risk analysis and security design have significant problems, but its extensive security controls can be very useful. The NIST SSE standards are further described in the appendix (section 8.3 on page 45).

- **Common Criteria (CC):**

CC is an SSE practice for security assurance, for stand-alone systems. It is a standard that some governments require for procurement, e.g., the US DoD. To sell information-security products in these markets, CC certification is required. Much has been published about problems with CC, and there is extensive criticism of it. CC is further described in the appendix (section 8.2 on page 45).

- **ISO 27000 standards (ISO 27K):**

These are a series of ISO standards for IT-systems security. They provide SSE practices for security development and for security assurance. Over half of the ISO 27K certifications are in Japan, where the government mandates its use. In other countries where certifications have been issued, each country has relatively few certifications. For instance, 2733 certifications are held by Japanese organizations, but only 74 certifications are held by U.S. organizations. ISO 27K is further described in the appendix (section 8.5 on page 47).

- **U.S. DoD:**

The DoD has an extensive set of information-assurance (IA) policies and guidance [DIS08a]. We were not able to find a clear and comprehensive presentation of them, and we suspect that this is not available at present. (In contrast, a clear and comprehensive presentation of the NIST standards is readily available.). The primary source for DoD IA information consists of a large collection of documents that are organized in a rudimentary way [DIS08b]. The documents are made complex by copious acronyms and cross references to other documents. At least one of the key policies is still in development, i.e., the "Information Assurance Strategic Plan" is an interim plan [DoD08a]. In addition, each service (Army, Navy, etc.) has its own standards.

The DoD's primary SSE practices appear to be DIACAP and Common Criteria. DIACAP is a new standard for DoD IT systems. It went into effect in November 2007, and it includes SSE practices for security development and security assurance [DoD07, WS07]. In our research, further investigation is needed for the DoD policies and guidance.

- **NSA:**

The NSA's Information Assurance Directorate (IAD) has developed some SSE practices for the USFG and DoD [IAD08b, IAD08c]. The practices that are applicable to typical systems include: an extensive set of security configuration guides, a comprehensive set of SSE practices for the system lifecycle, and vulnerability-detection processes for vendors who provide that service to the USFG. Among all of these SSE practices, none appear to be mandatory. In addition, NSA oversees the U.S. Common Criteria program. The SSE practices developed by NSA are further described in the appendix (section 8.4 on page 46).

- **DHS:**

As we were completing this report, we discovered that the Department of Homeland Security (DHS) has a website with an extensive collection of SSE practices [DHS08]. The website is for a DHS program called "Build Security In". The SSE practices do not appear to be mandatory.

6 Conclusion

Evaluating and choosing SSE practices is challenging: there are many SSE practices, they serve different purposes, and they often have significant problems and limitations, which may not be immediately apparent. To address these challenges in evaluating, choosing, and using SSE practices, this report presented four aspects of SSE practices:

- A taxonomy of SSE practices, which provides an overall framework for understanding and evaluating SSE practices,
- Evaluation criteria for SSE practices, which focuses on attributes of SSE practices that influence their usefulness for a particular system,
- Principles for choosing SSE practices, which are based on the evaluation criteria and security planning, and
- A selected survey of SSE practices, which is presented in the context of the taxonomy of SSE practices.

The highlights from our survey are:

- Among the existing SSE practices, many have been made to implement government mandates for computer security. Such practices are built to meet government requirements that typically do not exist outside of government. This can significantly reduce the usefulness of these SSE practices outside of government.
- SSE practices are built for specific purposes, uses and environments. Often, these constraints are not fully and accurately described. When using an SSE practice on a particular system, it is important to assess the practice's limitations and constraints, as it is likely the practice will need to be modified and adapted for use.
- From our survey, the SSE practices that stood-out as being most useful are: Microsoft's SDL, ISF's "The Standard of Good Practice for Information Security", and NIST's security-controls guidance.

- From our survey, the SSE practices that stood-out as needing improvement are: Common Criteria, and NIST's SSE practices for risk analysis and security design.

A limitation of this research is that we have not used these SSE practices. The research results are based on reports from others and our own analysis. Also, our survey was extensive, but far from being exhaustive, both in breadth and depth. Consequently, parts of this research report are likely to be naive or incomplete. Our intent was to document our research results, should they be useful to others who are working on similar problems.

7 Bibliography

[ABP07] Averitt, S., M. Bugaev, A. Peeler, H. Shaffer, E. Sills, S. Stein, J. Thompson, M. Vouk, "Virtual Computing Laboratory (VCL)", *Proceedings of the International Conference on Virtual Computing Initiative*, pp. 1-16, Research Triangle Park, NC, May 2007.

[Bac99] Bach, J. "The Immaturity of CMM", (includes text of original version published in *American Programmer*, September 1994), Satisfice web site, 1999. Accessed on June 19, 2008 <http://www.satisfice.com/articles/cmm.shtml>.

[Bej07] Bejtlich, R. "Someone Please Explain Threats to Microsoft", author's blog "TaoSecurity", October 1, 2007, Accessed on June 25, 2008 at <http://taosecurity.blogspot.com/2007/10/someone-please-explain-threats-to.html>.

[BK05] Bidstrup, E. and E. Kowalczyk, "Security Development Lifecycle: Changing the Software Development Process to Build in Security from the Start", presentation slides, Microsoft Security Summit West 2005, Redmond, WA, 2005. Accessed on July 10, 2008 at http://download.microsoft.com/download/0/1/a/01a053e8-3e18-4f73-b8e7-68d53a8232da/Bidstrup-Kowalczyk_SSW-2005.ppt>.

[BM91] Bollinger, T. and C. McGowan. "A Critical Look at Software Capability Evaluations", *IEEE Software*, 8(4):25-41, July 1991.

[BSI07] German Federal Office for Information Security (BSI). "IT Security Guidelines", BSI website, 2007. accessed August 12, 2008 <http://www.bsi.de/english/gshb/guidelines/guidelines.pdf>.

[CF02] Conradi, R. and A. Fuggetta. "Improving Software Process Improvement", *IEEE Software*, July/August 2002.

[CRS04] Moteff, J. "Computer Security: A Summary of Selected Federal Laws, Executive Orders, and Presidential Directives", Congressional Research Service, Order Code RL32357, April 16, 2004. Accessed on June 10, 2008 at <http://opencrs.com>.

[Dal08] Dallman, J. "'Crawling' Toward SDL", Security Development Lifecycle blog, March 06, 2008. Accessed on July 2, 2008 at <http://blogs.msdn.com/sdl/archive/2008/03/06/crawling-toward-sdl.aspx>.

[DHS08] Department of Homeland Security (DHS). "Build Security In" website. DHS National Cyber Security Division. Accessed on August 18, 2008 at <https://buildsecurityin.us-cert.gov/daisy/bsi/home.html>.

[DIS08a] Defense Information Systems Agency (DISA). *Information Assurance Support Environment : Your One-Stop-Shop for IA Information* [web site], U.S. DoD DISA. Accessed on August 6, 2008 at <http://iase.disa.mil/index2.html>.

[DIS08b] Defense Information Systems Agency (DISA). "Policy and Guidance", web page at [DIS08a]. Accessed on August 6, 2008 at <http://iase.disa.mil/policy-guidance/index.html>.

[Dix06] Dixon, M. "SunSAR - Security Adequacy Review", Sun Microsystem's employee blog site, author's weblog, December 2, 2006. Accessed on August 12, 2008 at <http://blogs.sun.com/identity/entry/sunsar_-_security_adequacy_review>.

[DoD07] ASD(NII)/DoD CIO. Department of Defense Instruction Number 8510.01, "SUBJECT: DoD Information Assurance Certification and Accreditation Process (DIACAP)", U.S. DoD, November 28, 2007. Accessed on August 6, 2008 at <http://www.dtic.mil/whs/directives/corres/pdf/851001p.pdf>.

[DoD08a] U.S. Department of Defense, "The Department of Defense Information Assurance Strategic Plan", March 2008. Accessed on August 6, 2008 at <http://iase.disa.mil/policy-guidance/dod_interim_ia_strategic_plan_mar_08.pdf>.

[HHS08] U.S. Department of Health & Human Services. Health Insurance Portability and Accountability Act (HIPAA) website. Accessed on August 8, 2008 at <http://www.hhs.gov/ocr/hipaa/>.

[HL03] Howard, M. and D. LeBlanc. *Writing Secure Code*, second edition, Microsoft Press, 2003.

[HL06] Howard, M. and S. Lipner. *The Security Development Lifecycle*, Microsoft Press, 2006.

[HLO06] Hernan, S., S. Lambert, T. Ostwald, and A. Shostack. "Uncover Security Design Flaws Using The STRIDE Approach", *MSDN Magazine*, November, 2006. Accessed on July 10, 2008 at <http://msdn.microsoft.com/en-us/magazine/cc163519.aspx>.

[How04] Howard, M. "Mitigate Security Risks by Minimizing the Code You Expose to Untrusted Users", *MSDN Magazine*, November, 2004. Accessed on July 10, 2008 at <http://msdn.microsoft.com/en-us/magazine/cc163882.aspx>.

[How05] Howard, M. "A Look Inside the Security Development Lifecycle at Microsoft", *MSDN Magazine*, November, 2005. Accessed on July 10, 2008 at <http://msdn.microsoft.com/en-us/magazine/cc163705.aspx>.

[IAD08a] NSA Information Assurance Directorate website. Accessed on August 7, 2008 at <http://www.nsa.gov/ia/index.cfm>.

[IAD08b] NSA Information Assurance Directorate. "Products", web page at [IAD08a]. Accessed on August 7, 2008 at <http://www.nsa.gov/ia/government/gover00001.cfm>.

[IAD08c] NSA Information Assurance Directorate. "Services", web page at [IAD08a]. Accessed on August 7, 2008 at <http://www.nsa.gov/ia/government/gover00002.cfm>.

[IAD08d] NSA Information Assurance Directorate. "Security Configuration Guides", web page at [IAD08a]. Accessed on August 7, 2008 at <http://www.nsa.gov/snac/index.cfm?MenuID=scg10.3.1>.

[IAD08e] NSA Information Assurance Directorate. "Information Systems Security Engineering", web page at [IAD08a]. Accessed on August 7, 2008 at <http://www.nsa.gov/ia/government/isse.cfm?MenuID=10.3.2>.

[IAD08f] NSA Information Assurance Directorate. "Information Security (INFOSEC) Assurance Training & Rating Program", web page at [IAD08a]. Accessed on August 7, 2008 at <http://www.nsa.gov/ia/industry/education/iatrp.cfm?MenuID=10.3.2>.

[IAS00] NSA Information Assurance Solutions. "Information Assurance Technical Framework (IATF)", Release 3.0, National Security Agency, 2000. Accessed on August 7, 2008 at <http://handle.dtic.mil/100.2/ADA393328>.

[Ise08a] IsecT Ltd. "ISO 27001 Security" web site, IsecT Ltd., New Zeland, 2008. Accessed on August 12, 2008 at <http://www.iso27001security.com/index.html>.

[Ise08b] IsecT Ltd. "Information security risk analysis/risk assessment", web page at [Ise08a], 2008. Accessed on August 12, 2008 at <http://www.iso27001security.com/html/faq.html#RiskAnalysis>.

[ISF] Information Security Forum (ISF). "The Information Security Forum", presentation slides, ND. Accessed on August 12, 2008 at <http://www2.devoteam.at/expert.talk/20050623/Expert-Talk_20050623_Certified-Security_ISF.pdf>.

[ISF08a] Information Security Forum (ISF) web site. Accessed on August 13, 2008 at <https://www.securityforum.org/index.htm>.

[ISF08b] Information Security Forum (ISF). "The Standard of Good Practice for Information Security", The Standard of Good Practice website, 2008. Accessed on August 13, 2008 at <https://www.isfsecuritystandard.com/SOGP07/index.htm>.

[ISO05a] International Standards Organization (ISO). "Information technology — Security techniques — Code of practice for information security management", International Standard ISO/IEC 17799, Second edition, June 15, 2005.

[ISO05b] International Standards Organization (ISO). "Information technology - Security techniques - Information security management systems - Requirements", ISO/IEC 27001-2005, 2005.

[ISO08] The ISO 27000 website. Accessed on August 15, 2008 at <http://www.27000.org/>.

[Kis06] Kissel, R. "Glossary of Key Information Security Terms", NIST IR 7298, National Institute of Standards and Technology, April 2006.

[KS07] Keblawi, F. and D. Sullivan. "The Case for Flexible NIST Security Standards", *Computer*, 40(6):19-26, June 2007.

[Lin08] Lindstrom, P. "Is Microsoft's SDL Working?", Security and Risk Management Strategies Blog, May 16, 2008. Accessed on July 18, 2008 at <http://srmsblog.burtongroup.com/2008/05/is-microsofts-s.html>.

[McG06] McGraw, G. *Software Security : Building Security In*, Addison Wesley, 2006.

[Mic06] Microsoft. "Risk Assessment Document", Version 3.1, April 7th, 2006. Contained on the CD that is distributed with [HL06].

[Mic08a] Microsoft. "Microsoft Security Development Lifecycle (SDL)", Version 3.2, Microsoft Corporation, 2008. Accessed on July 7, 2008 at <http://www.microsoft.com/DOWNLOADS>.

[Mic08b] *Microsoft Security Development Lifecycle (SDL)* [website]. Microsoft. Accessed on July 14, 2008 at <http://msdn.microsoft.com/en-us/security/cc448177.aspx>.

[Mic08c] *The Security Development Lifecycle* [blog]. Microsoft. Accessed on July 14, 2008 at <http://blogs.msdn.com/sdl/default.aspx>.

[Mic08d] *MSDN Magazine Homepage*. Microsoft. Accessed on July 14, 2008 at <http://msdn.microsoft.com/en-us/magazine/default.aspx>.

[Mis44] Von Mises, L. *Bureaucracy*, Yale University Press, 1944.

[Mis49] Von Mises, L., *Human Action*, Ludwig von Mises Institute, 1949.

[NDI07] NDIA Systems Assurance Committee. "Systems Assurance – Delivering Mission Success in the Face of Developing Threats", V0.10, National Defense Industrial Association, January 29, 2007. Accessed on August 13, 2008 at <http://www.ndia.org/Content/ContentGroups/Divisions1/Systems_Engineering/PDFs18/Systems_Assurance_Committee_PDFs/Sys_Assur_NDIA_Guidebook_29JAN_2007.doc>.

[Neu78] Neumann, P. "Computer security evaluation", in AFIPS Conference Proceedings, pp. 1087-1095. AFIPS Press, January 1978. Reprinted in Rein Turn (ed.), *Advances in Computer System Security*, Artech House, 1981.

[NIS04] National Institute of Standards and Technology (NIST). "FIPS PUB 199: Standards for Security Categorization of Federal Information and Information Systems", NIST, Information Technology Laboratory (ITL), February 2004. Accessed on August 15, 2008 at <http://csrc.nist.gov/publications/PubsFIPS.html>.

[NIS06a] National Institute of Standards and Technology (NIST). "Minimum Security Requirements for Federal Information and Information Systems", *ITL Bulletin*, NIST, Information Technology Laboratory (ITL), March 2006. Accessed on August 15, 2008 at <http://csrc.nist.gov/publications/PubsITLSB.html>.

[NIS06b] National Institute of Standards and Technology (NIST). "FIPS PUB 200: Minimum Security Requirements for Federal Information and Information Systems", NIST, Information Technology Laboratory (ITL), March 2006. Accessed on August 15, 2008 at <http://csrc.nist.gov/publications/PubsFIPS.html>.

[NIS06c] National Institute of Standards and Technology (NIST). "Special Publication 800-53, Revision 1: Recommended Security Controls for Federal Information Systems", NIST, Information Technology Laboratory (ITL), December 2006. Accessed on August 15, 2008 at <http://csrc.nist.gov/publications/PubsSPs.html>.

[NIS08a] National Institute of Standards and Technology (NIST), Computer Security Resource Center website, NIST, Computer Security Division. Accessed on August 15, 2008 at <http://csrc.nist.gov/index.html>.

[NIS08b] National Institute of Standards and Technology (NIST), Special Publications (800 Series) web-page, at [NIS08a]. Accessed on August 15, 2008 at <http://csrc.nist.gov/publications/PubsSPs.html>.

[OCT08] OCTAVE web site. Accessed on August 13, 2008 at <http://www.cert.org/octave>.

[Ros07] Ross, R. "Managing Enterprise Security Risk with NIST Standards", *Computer*, 40(8): 88-91, August 2007.

[Sho07a] Shostack, A. "The Trouble with Threat Modeling", Security Development Lifecycle blog, September 26, 2007. Accessed on June 27, 2008 <http://blogs.msdn.com/sdl/archive/2007/09/26/the-trouble-with-threat-modeling-2.aspx>.

[Sho07b] Shostack, A. "The New Threat Modeling Process", Security Development Lifecycle blog, October 1, 2007. Accessed on June 27, 2008 <http://blogs.msdn.com/sdl/archive/2007/10/01/the-new-threat-modeling-process.aspx>.

[Sho07c] Shostack, A. "Getting into the Flow With Threat Modeling", Security Development Lifecycle blog, October 11, 2007. Accessed on June 27, 2008 <http://blogs.msdn.com/sdl/archive/2007/10/11/getting-into-the-flow-with-threat-modeling.aspx>.

[Sho07d] Shostack, A. "Making Threat Modeling Work Better", Security Development Lifecycle blog, October 16, 2007. Accessed on June 27, 2008 <http://blogs.msdn.com/sdl/archive/2007/10/16/making-threat-modeling-work-better.aspx>.

[Sho07e] Shostack, A. "Threat Modeling Self Checks and Rules of Thumb", Security Development Lifecycle blog, October 22, 2007. Accessed on June 27, 2008 <http://blogs.msdn.com/sdl/archive/2007/10/22/threat-modeling-self-checks-and-rules-of-thumb.aspx>.

[Sho07f] Shostack, A. "The STRIDE per Element Chart", Security Development Lifecycle blog, October 29, 2007. Accessed on June 27, 2008 <http://blogs.msdn.com/sdl/archive/2007/09/26/the-trouble-with-threat-modeling-2.aspx>.

[Spi07] Spitzner, L. "FISMA vs. ISO 27000", author's blog, November 28, 2007. Accessed on June 18, 2008 <http://lspitzner.blogspot.com/2007/11/fisma-vs-iso-27000.html>.

[SSE03] SSE-CMM Project. "Systems Security Engineering Capability Maturity Model (SSE-CMM) Model Description Document', Version 3.0, SSE-CMM Project, June 15, 2003.

[SS04] Swiderski, F. and W. Snyder. *Threat Modeling*, Microsoft Press, 2004.

[Tar] Tarbet, G. "Categorizing a System: Why must this be so hard?", The Compliance Authority web site, n.d. Accessed on June 18, 2008 <http://www.thecomplianceauthority.com/articles.shtm>.

[UCF08a] Unified Compliance Framework (UCF) [web site]. Accessed on June 12, 2008 <http://www.unifiedcompliance.com>.

[UCF08b] Unified Compliance Framework. "The list of currently tracked authority documents", web page from [UCF08a]. Accessed on August 7, 2008 at <http://www.unifiedcompliance.com/forms/tracked_documents.php>.

[Whi07] Whittaker, J. "Testing in the SDL", Security Development Lifecycle blog, May 24, 2007. Accessed on July 10, 2008 at <http://blogs.msdn.com/sdl/archive/2007/05/24/testing-in-the-sdl.aspx>.

[Whi08] Whittaker, J. *How to Break Software: A Practical Guide to Testing*, Addison Wesley, 2008.

[WS07] Williams, P. and T. Steward. "DoD's Information Assurance Certification & Accreditation Process", *Defense AT&L*, September-October 2007. Accessed on August 6, 2008 at <http://www.dau.mil/pubs/damtoc.asp>.

8 Appendix

A survey of SSE practices was presented in section 5. Additional information about specific practices is provided here in the appendix. These sections are referenced from the survey.

8.1 Microsoft's SSE practices

Microsoft has implemented a comprehensive set of SSE practices called the *Security Development Lifecycle* (SDL). SDL can be characterized using the taxonomy of SSE practices presented earlier (section 3.1 on page 5). From the perspective of the taxonomy, SDL is a security-development practice for stand-alone systems. SDL is the best-developed set of SSE practices we have found, and it appears to be well-suited for many types of systems for which security is important. This section describes SDL, how it can be useful outside of Microsoft, and how it can be adapted for use outside of Microsoft.

SDL is further described in the following sections:

8.1.1 SDL's history, uses, and benefits

8.1.2 How SDL works

8.1.3 SDL's limitations and problems

8.1.4 Using SDL outside of Microsoft

8.1.5 Reference information for using SDL

8.1.1 SDL's history, uses, and benefits

In 2002 Microsoft started an intensive program to improve the security for all of its products. Named the *Trustworthy Computing initiative* (TwC), the program was a response to mounting security problems that were not only affecting Microsoft's customers, but also Microsoft's reputation. The TwC led to the development of Microsoft's codified Security Development Lifecycle (SDL). Microsoft has made a very large investment both in developing and using SDL. Further, SDL is a required process for all Microsoft products that have security requirements. Microsoft reports that using SDL has substantially improved its products' security. For example, there have been large reductions in the number of publicly-reported vulnerabilities for its products.

Microsoft makes SDL publicly available through books, papers, seminars, and conferences. There appear to be at least two motives for disclosing and disseminating SDL. Microsoft provides computing systems such as operating systems, servers, network infrastructures, and development systems. Helping application-developers with security development will improve the overall security of its customers' computing environments. Secondly, the TwC includes a policy of public transparency regarding Microsoft's security, which seems to be aimed at gaining public trust and at providing security assurance. Publicizing SDL provides security assurance for its products by demonstrating that Microsoft is serious

about security and that Microsoft not only has good SSE capabilities, but its capabilities are better than its competitors', e.g., open-source. (The SDL book has a section that discusses weaknesses in the open-source security-development process, and SDL's advantages over it [HL06].)

SDL provides a comprehensive set of SSE practices that cover the development lifecycle, including design, implementation and testing. SDL is compatible with most development processes, and SDL's practices can be adopted incrementally, typically without radical changes to existing development processes. We expect that SDL can provide most of the SSE practices needed for developing typical stand-alone systems. SDL is designed for the Microsoft environment, so typically, it will need to be adapted for use in other environments. Also, SDL has some limitations and problems which will have to be worked-around. Further, SDL can be enhanced by augmenting it with other SSE practices.

One of SDL's primary strengths is that it is actively used to develop very complex and successful products. Those who develop SDL are well known in the security community. They work with SDL and actively improve and correct it. We've been impressed by how candid the SDL developers have been about problems with SDL, and the improvements they have made. Another strength of SDL is Microsoft's huge investment in it and their apparent long-term commitment to it. This includes extensive documentation that is actively updated. Certainly, Microsoft products and SDL have weaknesses and problems. However, we have not found a set of SSE practices that have such strengths and benefits. Also, SDL is amenable to being customized and augmented with additional SSE practices.

8.1.2 How SDL works

This section provides an overview of SDL, and it was compiled from several Microsoft sources. SDL is based on three elements of secure software-development [Mic08a]:

- Best practices,
- Process improvements, and
- Metrics.

SDL is made-up of a set of SSE practices. SDL's goals are to [How05, Mic08a]:

- Minimize security-related vulnerabilities in the design, code, and documentation,
- Detect and eliminate vulnerabilities as early as possible in the development lifecycle, and
- Reduce the severity of the security-related defects in released products.

Microsoft's overall security-engineering framework is called *SD3+C*: Secure by Design, Secure by Default, Secure in Deployment, and Communication [Mic08a]. SDL implements much of SD3+C, and especially Secure by Design and Secure by Default [How05].

- *Secure by Design* includes best-practices for security architecture, design, and structure.

- *Secure by Default* is a set of design principles that include: least privilege (running with the fewest possible permissions), defense in depth (avoiding single points of security failure), conservative default settings, minimizing the attack surface area (i.e., parts of the system directly exposed to untrusted users), and avoiding risky changes to the host system's security

settings (e.g., not opening firewall ports). Also, services that are not commonly used are turned off by default.

- *Secure in Deployment* focuses on ensuring end-users deploy the product in a secure manner. It includes security deployment guides, analysis and management tools, and patch deployment tools.

- *Communications* is concerned with keeping end-users informed about security problems and solutions. It includes transparent communication about security vulnerabilities and updates.

There is an analogous concept to SD3+C for privacy, and it is *PD3+C*: Privacy by Design, Privacy by Default, Privacy in Deployment, and Communication [Mic08a].

The SDL practices are outlined in Figure 4 and Figure 5. They are shown in parallel with the Microsoft development lifecycle (which is much like most organizations' development lifecycles). SDL can be used with a variety of development processes, tools, and languages. An overview of SDL's primary practices follows.

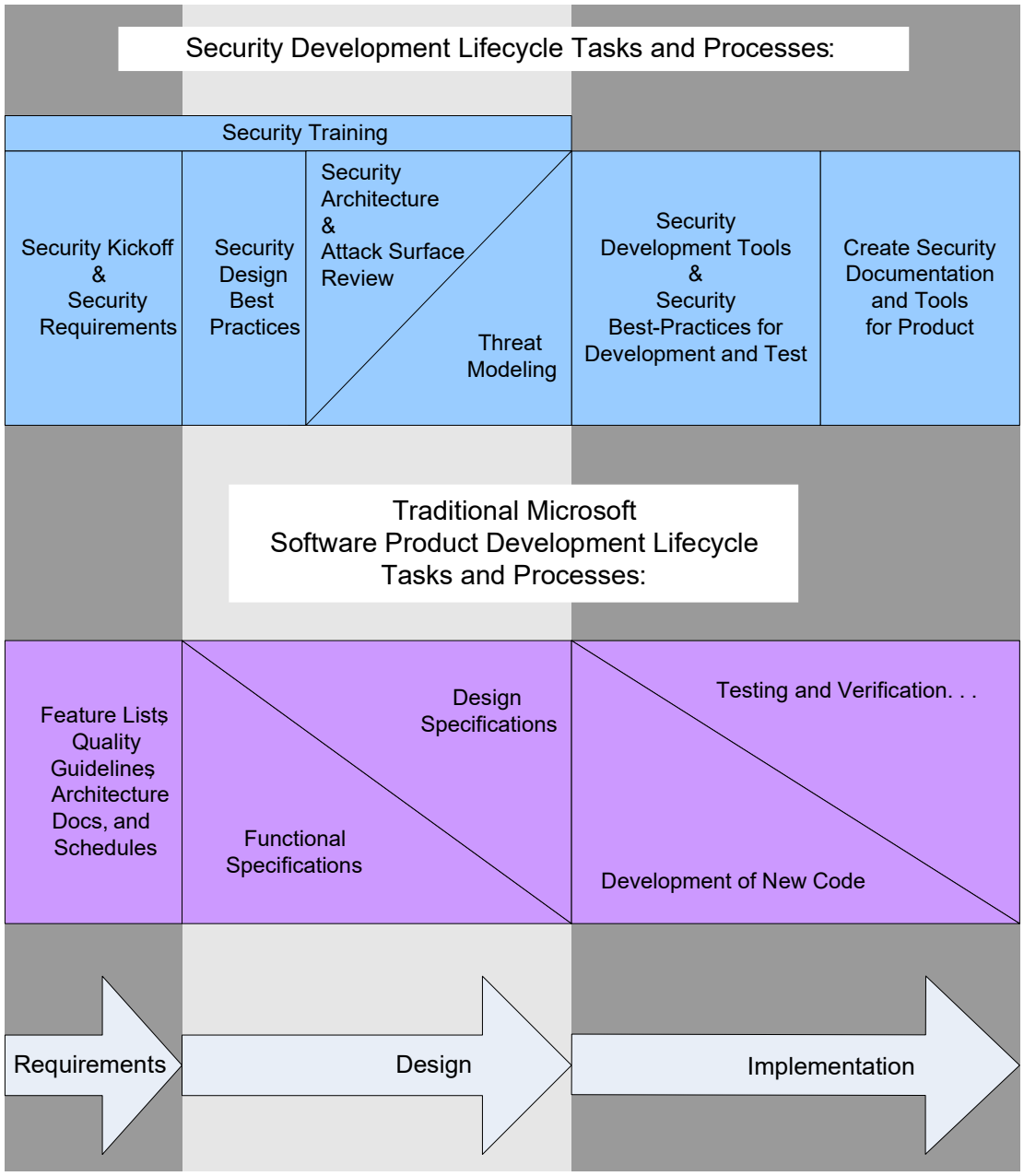

Figure 4 : Outline of SDL practices, part 1 [BK05]

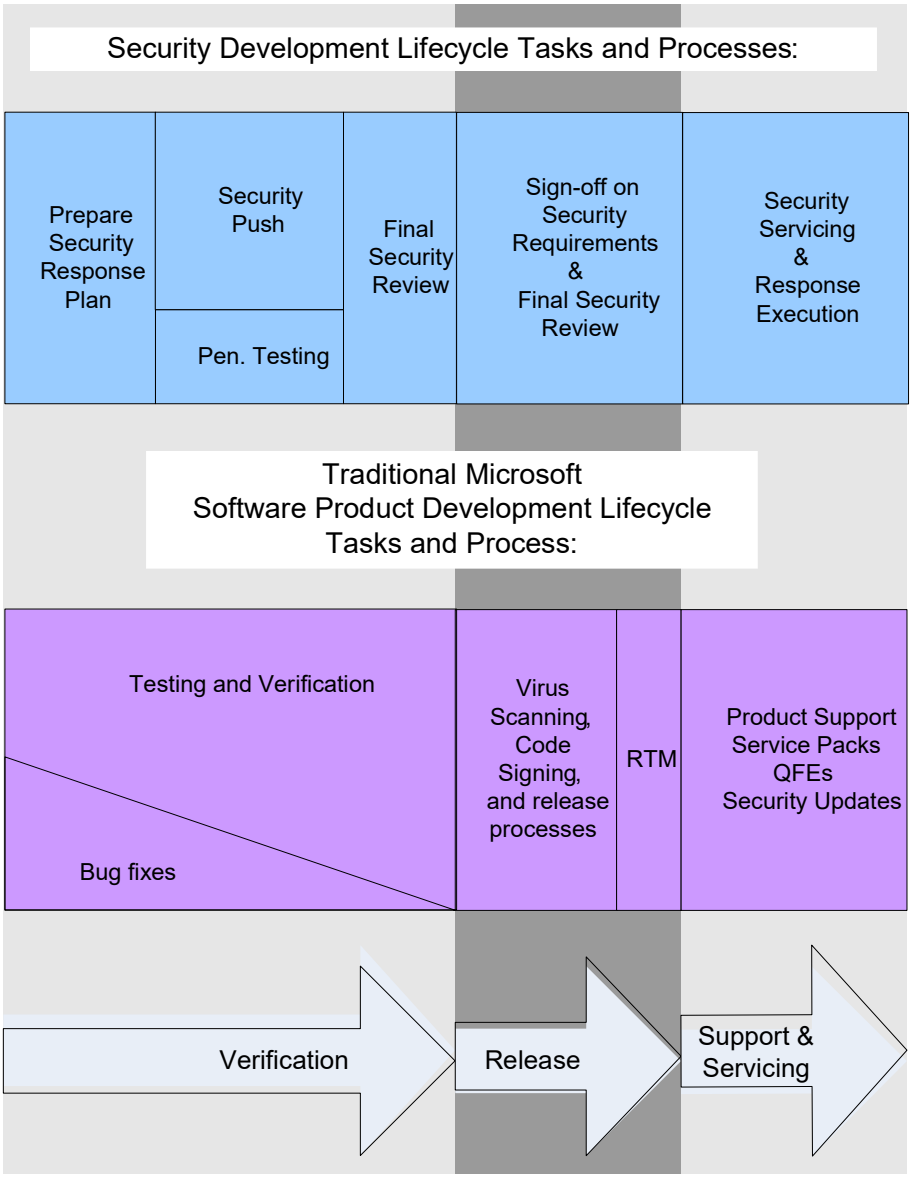

Figure 5 : Outline of SDL practices, part 2 [BK05]

- **Security design:**

SDL's security-design process is made-up of a set of design principles and techniques. One of the primary design techniques is reducing the system's *attack surface area* [How04]. This includes a set of techniques for minimizing the parts of the system that are exposed to untrusted users. The techniques include: a) reducing the services that are turned-on by default, b) reducing entry points available to untrusted users, and c) reducing privileges of processes and users. Other design principles and techniques include defense in depth, the principle of least privilege, and secure defaults.

There is some overlap between the categories of Secure by Design and Secure by Default. Fundamentally, Secure by Default is a set of design principles. The overlap in

categories is not necessarily a problem, however, the overlap does not seem to be explained clearly. Such analytical weaknesses in SDL concepts are not uncommon. However, they are not major problems, just shortcomings to work-around.

- **Threat modeling:**

Threat modeling (TM) is one of SDL's primary processes [HLO06]. It is used to analyze the parts of a system that are potentially vulnerable to attack. (The name *Threat Modeling* is a misnomer as TM focuses on potential vulnerabilities and not on threats, such as hackers. This terminology problem is further described in section 8.1.5.2 on page 42.) TM provides a systematic way of analyzing the system's security design. TM is applied to a well-developed design, and it is used to identify design flaws and to improve the design. TM models the system design, from the perspective of security, and data flow diagrams (DFDs) are used to represent the design. The DFD notation includes *trust boundaries*, and they represent the border between trusted and untrusted elements of the system. A simple DFD example is shown in Figure 6.

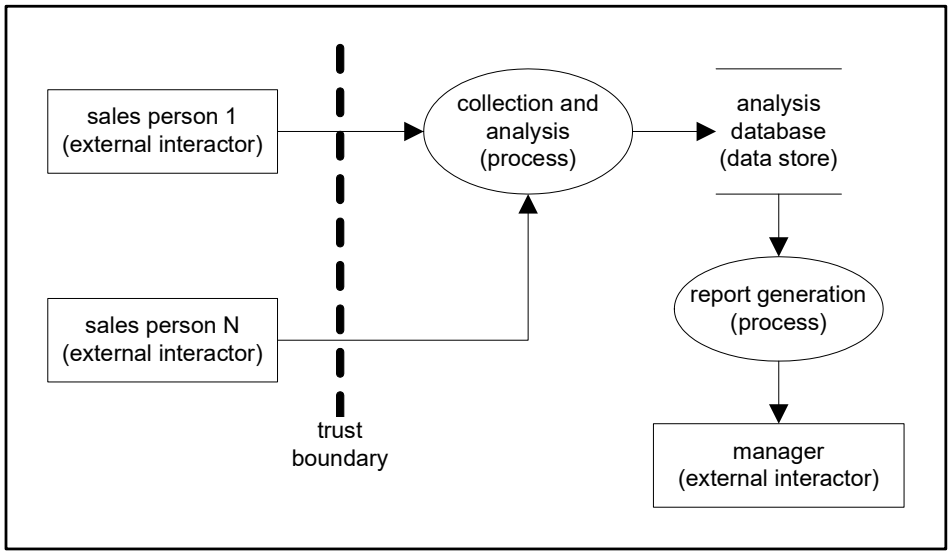

Figure 6 : Threat Model DFD example [HLO06]

In the TM DFD, one of the primary places for potential vulnerabilities is where data flows across trust boundaries. TM categorizes potential vulnerabilities according to the types of attacks that exploit the vulnerability. The attack types are: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. These attack types are referred to by their acronym STRIDE. Each of these types of attacks is against one of the basic security properties, as shown in Table 1.

Table 1 : Vulnerability types and related security properties

Vulnerabilities (STRIDE)	Security Properties
Spoofing	Authentication
Tampering	Integrity

Repudiation	Non-repudiation
Information disclosure	Confidentiality
Denial of service	Availability
Elevation of privilege	Authorization

- **Security implementation and testing:**

SDL includes several best-practices for secure coding. The largest collection is in the Microsoft book *Writing Secure Code*, second edition [HL03]. Also, compiler support is used to identify and prevent security problems. Static code-analysis tools are also used. Microsoft has identified a set of C and C++ APIs that are prone to causing security problems, and a header file is available to prevent these APIs from being used [Mic08a]. In the Microsoft development organization, these are "banned APIs".

SDL also includes best practices for security testing. The overall testing framework focuses on three areas: a) attacks against vulnerabilities due to unpredicted conditions in the application's environment (e.g., in file system configurations, or runtime libraries), b) direct attacks against the application itself, and c) attacks against vulnerabilities caused by the way the application is used (e.g., misconfiguration) [Whi07]. Security testing techniques include file fuzzing (generating invalid data files, e.g., for jpeg files), penetration testing, and the use of vulnerability regression tests [Mic08a].

As a whole, SDL is very well documented. There is extensive documentation for the overall SDL process and for its individual development practices. A guide to the SDL documentation, and to learning about SDL, is provided in section 8.1.5.1 (page 41).

8.1.3 SDL's limitations and problems

This section discusses SDL's limitations and problems. The analysis is from the perspective of using SDL outside of Microsoft. The following sub-sections present, respectively: ways that SDL is designed for the Microsoft environment, limitations of the SDL documentation, and lastly, problems with SDL. These limitations and problems are things to work around. As a whole, we expect SDL will be very effective for developing typical systems.

8.1.3.1 SDL is designed for the Microsoft environment

Parts of SDL are designed for Microsoft's environment, including Microsoft's development processes, and its personnel, management, and products. In our experience, there are parts of SDL that can be puzzling unless one understands how they are motivated by Microsoft's environment. Also, there are parts of SDL that may not be appropriate, or optimal, for other environments and systems.

The SDL documentation does mention that SDL may need to be customized when it is used outside of Microsoft.[1] However, the documentation provides little guidance for identifying the parts of SDL that are designed for distinctive aspects of Microsoft's environment. Also, there is not much guidance for modifying SDL for use in other environments. This section discusses ways that SDL is designed for distinctive aspects of Microsoft's environment. Later

[1] For instance, this is mentioned in an article on Threat Modeling (TM) [Sho07f] and in Microsoft's SDL specification [Mic08a].

sections discuss ways to adapt SDL for use outside of Microsoft.

Parts of SDL are designed for the way Microsoft divides security responsibilities between its product-development groups and its security-development group. We assume Microsoft's product-development groups typically work on a single product (e.g., Windows) or a product subsystem (e.g., Windows' printing subsystem). For Microsoft's product-developers, it appears they can have limited security skills due to a lack of experience or even aptitude, and they may also lack time to focus on security. A separate security-development group develops the SDL process, enforces its use, and helps product-development teams implement SDL. This division between product developers and security experts is a distinctive part of Microsoft's SSE practices, though, it is probably not uncommon in other development organizations.

In learning about SDL, we found its Threat Modeling (TM) puzzling and difficult to understand until we realized that it accommodates Microsoft's separate product-development and security-development groups. This is further discussed in section 8.1.5.4 (page 44).

Parts of SDL are designed for distinctive aspects of Microsoft's products. Most of Microsoft's products are general-purpose computing-systems and tools. The products are used in a wide variety of ways, so users' security needs and threats vary widely. Security for Microsoft products must be built for a wide range of assets and threats (e.g., types of hackers), and this influences how SDL works. For instance, SDL does not seem to place a lot of emphasis on analyzing system assets. The reason may be that such analysis is not very useful for general-purpose systems and tools. (SDL's lack of emphasis on system assets is discussed in section 8.1.5.3 on page 43.) For systems that are not as general-purpose as Microsoft's products, a more detailed analysis of assets and threats can be possible, and even necessary. A Microsoft article recommends that TM be customized in this way [Sho07f].

Another distinctive attribute of Microsoft's products is their large volume of sales and high profits. Microsoft products have an economy of scale that is unique, or at least rare. It allows Microsoft to have very large security-development budgets, relative to what is available for other systems. Some parts of SDL may not be affordable or cost effective for other systems. Further, SDL does not seem to place a lot of emphasis on cost-and-benefit analysis for security development. Cost-and-benefit analysis would be much more important for systems with relatively small security-development budgets.

Microsoft's motives in developing SDL are also distinctive. SDL grew out of Microsoft's efforts to reduce the high rate of vulnerabilities in its products. Microsoft products' security problems were not only hurting Microsoft customers, but they were also harming Microsoft's reputation. Given Microsoft's motive of improving its reputation, it is within the realm of possibility that parts of SDL are designed more to improve security image rather than overall security strength.

As an example, one of the main ways Microsoft demonstrates security improvements is by counting publicly-reported vulnerabilities. Would it be possible that parts of SDL are more concerned with minimizing the number of publicly-reported vulnerabilities rather than minimizing the total losses from security problems? For instance, could this be one reason SDL places so much emphasis on TM and so little emphasis on asset analysis? SDL's emphasis on TM is discussed in section 8.1.5.4 (page 44). (In a similar vein, a blog writer has raised questions about the validity of Microsoft's use of vulnerability metrics in its claims about SDL's usefulness [Lin08].) A focus on reducing publicly-reported vulnerabilities is not necessarily

incorrect. However, as a Microsoft security objective, it is likely to influence SDL's design.

In summary, SDL is designed for some distinctive aspects of Microsoft's development environment and products. The distinctive aspects that were presented are: Microsoft's development organization (separate product-development and security-development groups), Microsoft's products (general-purpose products, and large security-development budgets), and Microsoft's motivations in developing SDL (the objective of reducing publicly-reported vulnerabilities).

8.1.3.2 Limitations of SDL documentation

Given the complexity of SSE, the Microsoft SDL documentation cannot be a complete nor entirely accurate representation of what Microsoft does to secure its products. Using SDL will require adapting it for use outside of Microsoft, and also, users of SDL must have security skills in addition to SDL. Some of the limitations of Microsoft's SDL documentation are described here.

Microsoft's SDL specification is careful to qualify the limitations of its completeness, accuracy, and applicability [Mic08a]. It states:

> *This documentation is not an exhaustive reference on the SDL process as practiced at Microsoft. Additional assurance work may be performed by product teams (but not necessarily documented) at their discretion. As a result, this example should not be considered as the exact process that Microsoft follows to secure all products.*
>
> *. . . This document outlines the SDL process used by Microsoft product groups for application development. It has been modified slightly to remove references to internal Microsoft resources and to minimize Microsoft-specific jargon. We make no guarantees as to its applicability for all types of application development or for all development environments. Implementers should use common sense in choosing the portions of the SDL that make sense given existing resources and management support.*

There are several ways that the SDL documentation could differ from how SDL actually works at Microsoft. An inherent limitation of documenting any SSE practice is that SSE can only be partially codified. Microsoft's SDL documentation could not possibly embody all of the processes, company culture, and skills that go into making Microsoft's products secure. Another way the Microsoft SDL documentation will differ from Microsoft's practice is that the SDL process undergoes regular improvement and correction. Microsoft's SDL specification states that SDL recommendations may be published at anytime, and that SDL requirements for Microsoft developers will be updated about every six months [Mic08a].

There is a possibility of bias in Microsoft's documentation of its SDL process. The SDL documentation may be an ideal view of what Microsoft does, i.e., it may document what they would like to be doing. It's possible that, in some ways, the actual practice falls short of what they describe. Also, the documentation is produced by the Microsoft security team. It is possible that the Microsoft product developers could have a different view, or a less favorable view, of the SDL process. We searched for candid "insider" reports about SDL use at Microsoft, such as forum posts by Microsoft product developers. However, we could not find any. We hope Microsoft has represented SDL accurately, but the possibility of bias warrants critical

reading.

8.1.3.3 Problems with SDL

Microsoft improves SDL over time, and some of these improvements are corrections for problems they encountered in using SDL [Sho07a]. Microsoft's candor about SDL's problems and the published corrections are one of SDL's greatest strengths. In using SDL outside of Microsoft, it should be expected that problems will be encountered in SDL, and solutions will be needed for working around them.

Of the problems we encountered with SDL, most have to do with Microsoft's environment. These problems were described earlier in section 8.1.3.1 (page 36). Additional problems related to Microsoft's environment are described later, including problems with understanding TM (section 8.1.5.4 on page 44), and problems with SDL under-emphasizing assets (section 8.1.5.3 on page 43).

The primary problem we encountered with SDL itself is in its terminology. SDL redefines some standard computer-security terms, and on occasion, it uses these terms in inconsistent and inaccurate ways. These problems can make the SDL documentation confusing and difficult to understand. This problem is further described in section 8.1.5.2 (page 42). A similar problem is that some SDL concepts need further refinement and analysis, to express them more clearly and to use them in more consistent ways. One example is the overlap between the concepts of Secure by Design and Secure by Default, as described earlier in section 8.1.2 (page 31). Overall, we think that SDL has much merit, and that these problems are just things to be worked around.

8.1.4 Using SDL outside of Microsoft

This section outlines an initial plan for the use of SDL outside of Microsoft. The topics covered are, respectively: process planning, using SDL, and customizing SDL.

- **Process planning:**

A plan will need to be developed for using SDL outside of Microsoft. This plan should be within the framework of an overall system security-plan, as discussed in section 4.2 (page 15). It will take time to develop SDL skills and to adapt SDL for use with a particular system and its environment. There is merit in starting with the parts of SDL that provide quick benefits and that involve low cost and low risk. Adopting SDL is an engineering project, and it will require experimentation, adaptation, resourcefulness and perseverance.

- **Using SDL:**

Microsoft has provided extensive guidance for organizations to adopt SDL use. Much of the guidance focuses on Microsoft's use of SDL, but some of it is for those outside of Microsoft. The SDL book is the primary source for adopting SDL [HL06]. It is written largely for development managers, including development-process managers. The Microsoft SDL specification is another useful source [Mic08a]. It asserts that SDL can be adopted incrementally and without radical changes in the development process. The initial stage of SDL is "Stage 0: Education and Awareness", and it is described in both of those sources.

A Microsoft security-team member has written a helpful article on using SDL in organizations other than Microsoft [Dal08]. The approach he recommends for adopting SDL is "crawl, walk, run", which involves starting with basic parts of SDL and working-up to use of the full process. When starting out, he recommends implementing some of SDL's core principles, but in ways that are simple, low-cost and effective. He outlines a plan for initial SDL use, and it includes: 1) performing a detailed analysis of the application's architecture and its attack surface (via TM), 2) using tools that will perform security analysis on the application (compiler defenses, static analysis tools, and testing tools), and 3) evaluating the application's security problems to determine how the development process has improved security (this includes use of an incident-response process to diagnose, fix and track bugs).

- **Customizing SDL:**

SDL will need to be customized for use outside of Microsoft. The primary reasons for this include:

  - There are aspects of SDL that are specific to Microsoft and not applicable to, or not optimal for, other products or environments (as described in section 8.1.3.1 on page 36).
  - There are some problems with SDL itself, that need to be worked-around (as described in section 8.1.3.3 on page 39).
  - SDL can be augmented with additional SSE practices.

Within Microsoft, SDL is used by its product-development groups and its security-development group, as described in section 8.1.3.1 (page 36). A similar structure could be used for smaller development organizations, however, the security-development group may just be one person, and security development may not be a full-time responsibility for him.

In summary, SDL provides a comprehensive set of SSE practices that cover the development lifecycle, including design, implementation and testing. We expect that SDL can provide most of the SSE practices needed for typical systems. SDL is designed for the Microsoft environment, so the SDL process will need to be adapted for use in other environments. Also, SDL has some limitations and problems which will have to be worked-around. One of SDL's primary strengths is that it is actively used to develop very popular and successful products. Those who develop SDL use it, and they actively improve and correct it. Certainly, Microsoft products and SDL have weaknesses and problems. However, we have not found a set of SSE practices that have its extensive strengths and benefits. A limitation of our investigation is that we have not actually used SDL, so our assertions about its usefulness are untested hypotheses.

8.1.5 Reference information for using SDL

This section contains reference information for using Microsoft's SDL outside of Microsoft. The first section is a guide to sources for learning about SDL. The remaining sections discuss problems and limitations that SDL users will need to work around. Those sections address, respectively: problems in SDL's computer-security terminology, how SDL and TM underemphasize assets, and lastly, challenges in understanding threat modeling.

8.1.5.1 Learning about SDL

As a whole, SDL is very well documented. There is extensive documentation for the overall SDL process and for its individual development practices. Also, the documentation is actively maintained and updated. There are several sources of SDL documentation, and one challenge in learning about SDL is that information on specific topics can be spread out among these sources. This section describes the SDL documentation, and also, where information can be found on specific topics.

The SDL books provide the most extensive documentation:

- *The Security Development Lifecycle* [HL06]
- *Threat Modeling* [SS04]
- *Writing Secure Code*, second edition [HL03]

Improvements to SDL are published in papers and articles, but they tend to build upon the latest edition of the SDL books.

Microsoft specifies the SDL process that its developers are required to follow, and a public version of this document is available [Mic08a]. Apparently, the public version is very close to the internal Microsoft version . Understandably, references to Microsoft's proprietary technologies and internal resources are omitted. This SDL specification is updated regularly. The current version is 3.2, and it is dated 2008. It is not a tutorial, but each section contains recommendations for tutorial books and papers that are publicly available.

Microsoft has an ongoing process for improving and correcting SDL. We have been impressed by how Microsoft is candid about SDL's problems, and how it is improved over time. Improvements and corrections to SDL seem to be published primarily in two places: Microsoft's SDL blog [Mic08c], and Microsoft's *MSDN Magazine* [Mic08d]. These articles typically build upon the information in the SDL books. The blog and magazine can also be useful sources for information on specific parts of SDL, e.g., testing.

Microsoft just started a new SDL website, and it is intended to be the primary online source for SDL [Mic08b]. The web site was started when we were completing this paper, so we have not reviewed it yet.

There are some weaknesses in the SDL documentation that can make learning SDL challenging. SDL's improvements and corrections are impressive, but following these updates can add to the challenge of learning SDL. Also, we have not found a single place for the SDL sources, nor a comprehensive reading-list for them. (However, Microsoft's new SDL website may provide this.) Other challenges we encountered in learning SDL were from problems in SDL itself. For example, there are problems in SDL's security terminology that can make it hard to follow. Such problems can be worked-around, and they are described in section 8.1.3.3 (page 39).

The SDL sources that we found to be the most useful are described here, and they are organized by topic. The Microsoft SDL-specification also lists sources for each SDL step [Mic08a].

- **SDL:** The two primary sources appear to be the SDL book [HL06] and Microsoft's official SDL specification [Mic08a]. The SDL specification has very useful references to current sources, for each SDL step.
- **Threat Modeling (TM):** There is a book devoted to TM [SS04]. Also, TM is the largest section in the SDL book [HL06]. The SDL book was published two years after the TM book, so the SDL book probably presents an improved version of TM. In the SDL blog, Adam Shostack recently wrote six articles on TM [Sho07a, Sho07b, Sho07c, Sho07d, Sho07e, Sho07f]. He describes Microsoft's current TM processes, and he is candid about problems with its prior TM processes. The *MSDN Magazine* has an article on TM that provides a good introduction to its use of data-flow-diagrams and vulnerability analysis.
- **Security testing:** A Microsoft security-test manager has written a short overview of their approach to testing [Whi07]. It provides a useful framework for testing. However, the concepts do not seem fully developed, and they could be further refined. This author also has written a book on testing [Whi08], and the Microsoft SDL specification lists the book as a testing reference [Mic08a]. The SDL book also has a chapter on testing [HL06].
- **Using SDL:** There is a very useful blog article on adapting SDL for use in development environments outside of Microsoft [Dal08].

8.1.5.2 Problems in SDL's computer-security terminology

There are problems in SDL's computer-security terminology. SDL redefines some standard computer-security terms, and on occasion, it uses these terms in inconsistent and inaccurate ways. These problems can make the SDL books confusing, and difficult to understand. This section describes these terminology problems.

One of SDL's key concepts and terms is *threat*, as in Threat Modeling (TM). Unfortunately, SDL does not use the standard definition for this common computer-security term, but it has redefined the term [Bej07, McG06]. In addition, the SDL documentation uses the term *threat* in different ways, and some of those uses are unclear [Bej07]. Further, having multiple meanings for *threat* prevents readers from just using simple substitution for the redefined term.

The TM book defines *threats* as being the "adversary's attack goals for a specific system" (pg. 26) [SS04]. Similarly, another Microsoft source says that *threats* refer to "which attacks are you trying to stop?" [Sho07a]. Both sources recognize that other people use the term *threat* to refer to hackers, or their agents. That is the conventional meaning of *threat* [Bej07, McG06].

Richard Bejtlich describes how Microsoft security texts use the term *threat* in multiple ways [Bej07]. He shows passages from the book *Writing Secure Code* [HL03], where *threat* means different things, including vulnerability, attack, and risk. He also shows an instance in the SDL book where *threat* is used in its conventional meaning. Gary McGraw describes how SDL's *threat analysis* is really a form of risk analysis [McG06].

We have noted additional uses of the term *threat* in the SDL book [HL06]. There, threat often seems to mean "potential vulnerability", e.g., as in "Identify Threats to the System" on page 116. In general, SDL might be clearer if the authors explicitly stated whether *threats* refer to potential or actual vulnerabilities. For example, is the threat modeling process used to identify potential or actual vulnerabilities? At times, the term *threat* seems to make most sense as being "ways software can potentially be attacked", or "ways hackers could attempt to attack software",

e.g., the STRIDE threat types as described on page 114.

Richard Bejtlich also notes misuse of the term *risk* in the SDL documentation. He shows a reference from the SDL book where its use of the term *risk assessment* really refers to vulnerability assessment.

In their criticism of Microsoft's terminology problems, Richard Bejtlich and Gary McGraw both make a point of saying the SDL process has much merit. The terminology problems are just something that SDL users have to work around.

8.1.5.3 SDL and TM underemphasize assets

Microsoft's Security Development Lifecycle (SDL) and Threat Modeling (TM) processes appear to underemphasize the role of system assets in computer security. As a general principle, the overall goal of computer security is to protect assets. Also, a key component of security planning is risk analysis, and assets are the central focus of risk analysis.[2] However, in SDL and TM, assets are not given such a central and prominent role. This section shows how SDL and TM underemphasize system assets, and some possible explanations for Microsoft's approach are given. Suggestions are made for the role of assets in adapting SDL and TM for use outside of Microsoft.

The SDL book has little mention of assets [HL06]. For instance, they are not mentioned in the table of contents and there is very little reference to them in the index. DFD elements are called system assets, but we only found a parenthetical reference to this (pg. 116). Chapter 3 of the SDL book recommends a Microsoft template for risk assessment [Mic06]. This template makes no mention of the word "asset".

The TM book does address assets [SS04]. However, threats are the focus of this book, not assets. Also, TM's threat analysis focuses on assets of interest to an adversary. Often, the system owner's assessment of assets is quite different than the adversaries' assessments. For instance, script kiddies may highly value defacing a company's web site, but have little interest in the company's business operations.

We think that assets should be a central focus of SSE, especially in security planning and design. SDL and TM do not seem to have assets in such a central role. The reason for this may be revealed in an article about TM by Adam Shostack, who is a member of Microsoft's security team [Sho07a]. He describes how "asset enumeration" had been one of eleven steps in the TM process. However, he said that developers who were not security experts generally lacked the skill needed to perform asset enumeration. Apparently, the asset enumeration step was removed from TM for this reason.

Another aspect of SDL is that it is built for developing Microsoft's products, which are tools for general purpose use, e.g., Windows and Office. Mr. Shostack describes how systems that serve more specific uses can be analyzed in more specific ways than Microsoft's general-purpose systems [Sho07f]. So, for Microsoft's general-purpose products, it may not be as useful for SDL to focus on assets, as their valuation would vary so much among customers' uses of the products. As an example, a Windows workstation is used in countless ways, and the value of the data it protects ranges from almost nothing to incalculably large amounts.

---

[2] The conventional meaning of *risk analysis* is being used here. SDL's use of this term is different, as described in section 8.1.5.2 (page 42).

In adapting SDL and TM for use with systems that are not as general-purpose as Microsoft's, we recommend that assets be given a more central role. More information on assets can be found in the risk analysis practices we recommend in section 5.1.3. Microsoft's experience in asset enumeration indicates that asset analysis may need to be performed by security experts.

8.1.5.4 Understanding Threat Modeling

We found Threat Modeling (TM) to be somewhat puzzling and confusing at first. What helped in understanding TM was realizing how it accommodates the Microsoft development environment. This section explains how TM accommodates the Microsoft development environment, and how that affects the use of TM outside of Microsoft.

In learning about SDL, it was hard to understand why TM is given much more emphasis than security design. In the TM book's security-lifecycle model, security-design is the second step, and TM is the third step. However, TM is the largest section in the SDL book [HL06]. Also, there is an entire TM book [SS04], but there is no security design book for SDL. TM focuses on finding security problems in a system whose design has largely been worked-out. TM is largely a form of testing, used to find security problems in the design. It was hard to understand why TM would get more attention than security design. Ideally, it seems the emphasis should be on correctly putting security in the system when it is designed. We found it puzzling and confusing for TM to be given more attention than security design.

Another thing that was confusing about TM is the way it overlaps with SDL's security-design step: the security-design step involves TM activities, and TM involves security-design activities. For instance, during the security-design step, attacks should be considered. However, considering attacks is what the TM process does. Further, the TM process involves *mitigation of threats*. However, this mitigation of threats is really security design. We found it puzzling and confusing for TM and security-design activities to be mixed between the two steps. Why not have a single security-design step? It would compose a design based on assets to be protected, system vulnerabilities, potential attacks (e.g., STRIDE), and available defensive countermeasures (e.g., that provide CIA).

What helped in making sense of SDL's security-design and TM steps was considering Microsoft's environment. This explanation is speculative, but it seems plausible:

1) In general, Microsoft's developers are not security experts [HL06, Sho07a]. Apparently, during the system-design stage, Microsoft developers do the best job they can on security design. Having a separate TM stage provides a way for the security department's experts to review the security design and find problems. If Microsoft had a single security-design step, then all the system designers would need to have security expertise, and this is not feasible.

In general, it's not practical to expect that most developers be security experts. Security engineering is a unique skill, just as technical writing is a unique skill. It's well known that programmers typically dislike technical writing and many lack writing aptitude or skill. Similarly, security engineering requires a certain savvy, and not all developers have aptitude for security engineering. Further, acquiring security skills takes time and experience.

2) SDL grew out of Microsoft's efforts to reduce the high incidence of security bugs in its products [HL06]. TM is largely a form of testing, used to find security problems in the design. SDL's focus on TM may have been motivated by Microsoft's goal of dramatically reducing security bugs in the short term.

Another reason for separate security-design and TM steps could be complexity. Security design is complex, and although it involves consideration of attacks, it could be helpful to have an additional step that focuses exclusively on attacks.

In summary, we found it puzzling and confusing for TM to be given much more emphasis than security design. Also, it was puzzling why SDL's security-design and TM steps overlapped. We found SDL and TM made much more sense when we considered them in the context of Microsoft's development and security teams and Microsoft's security objectives.

SDL's focus on TM may work well outside of Microsoft when the system developers do not have security expertise, and they are advised by a security expert. If the system developers have security expertise, it would probably be best to put more emphasis on security design, and less on TM.

8.2 Common Criteria

Common Criteria (CC) can be characterized using the taxonomy of SSE practices presented earlier in section 3.1 (page 5). From the perspective of the taxonomy, CC is an SSE practice for security assurance, for stand-alone systems. It is a standard that some governments require for procurement, e.g., the US DoD. To sell information-security products in these markets, CC certification is required.

Much has been published about problems with CC, and there is extensive criticism of it. In our research of CC we came across so much criticism of it that we wrote a lengthy paper that surveys CC's problems and criticisms.

A significant amount of funding has been provided for CC development, in the range of millions of dollars. CC standards have been developed, papers published, and international conferences held. These accomplishments give the appearance that CC is a useful SSE practice. However, in our investigation we encountered relatively few positive reports about CC. The positive reports were often by people who were benefiting from CC, e.g., their jobs were dependent upon CC. We encountered much more criticism than positive reports about CC. In fairness though, we must note that our investigation was not exhaustive, and it focused more on critical reports than positive reports.

Since there is so much credible criticism of CC, we strongly recommend against using it. For those compelled to use it, our survey of CC problems may be helpful.

8.3 NIST's security standards and guidelines

The National Institute of Standards and Technology (NIST) issues computer-security mandates for USFG IT systems. These mandates apply to all USFG civilian agencies. The mandates prescribe an extensive collection of SSE practices for both security development and security assurance. The security-development practices include risk analysis and security design, as well as guidance for a large number of security controls (e.g., firewalls). The security-assurance practices include evaluation and certification. From our survey, it appears NIST's

SSE practices for risk analysis and security design have significant problems. However, its extensive security controls can be very useful [Spi07]. Hereafter, NIST's SSE practices will simply be referred to as the NIST standards. The remainder of this section describes the NIST standards and the criticism of its risk analysis and security design processes.

NIST has well over 100 standards documents, though not all of them are required. They are freely available at its website [NIS08a]. A good overview of the mandated NIST standards is provided by the NIST bulletin, "Minimum Security Requirements for Federal Information and Information Systems" [NIS06a]. As it describes, the primary standard is "FIPS 200: Minimum Security Requirements for Federal Information and Information Systems" [NIS06b]. FIPS 200 calls for agencies to categorize the potential impacts from worst-case security compromises of their information systems. The categories used for potential impact are: low, medium and high.

The standard for categorizing potential impact is "FIPS PUB 199: Standards for Security Categorization of Federal Information and Information Systems" [NIS04]. FIPS 200 also calls for agencies to implement security controls for the potential impacts. The standard for implementing security controls is "Special Publication 800-53, Revision 1: Recommended Security Controls for Federal Information Systems" [NIS06c]. NIST has an extensive set of standards for specific security controls, perhaps over 100 of them, though not all are required [NIS08b].

NIST's impact-categorization process is a foundational part of its overall security-design process. Several credible sources have provided well-reasoned criticism of NIST's impact categorization process [KS07, Spi07, Tar]. Some of it includes:

- The impact analysis is not a full risk assessment [KS07, Spi07]. It neglects analysis of threats and vulnerabilities, which is needed for understanding risks to assets and for determining which security controls are necessary and appropriate [KS07].
- Neither FIPS 199 nor FIPS 200 addresses issues such as cost, schedule, size of project, and the challenges of securing legacy systems. NIST has not published any estimates of the cost of implementing the standards, nor a plan for achieving them [KS07].
- The approach is designed for existing systems, but it would not work well with new systems, as their design is composed over the system's initial lifecycle [KS07].
- Categorization is very difficult because the categories are not well defined [Tar].
- In general, NIST's standards are too fixed, and inflexible, given the nature of security design [KS07, Spi07]. Security design relies on human judgment, so an effective design process must be flexible to accommodate subjective solutions. Flexibility is also needed for the wide variation among systems and their security needs [KS07].

8.4 NSA's SSE practices

The NSA's Information Assurance Directorate (IAD) has developed some SSE practices for the USFG and DoD [IAD08b, IAD08c]. Those that appear to be applicable to typical systems include:

- **Security Configuration Guides:** The IAD provides an extensive set of "Security Configuration Guides" for various types of systems (e.g., firewalls), and for specific

commercial and open-source systems (e.g., Windows and Linux) [IAD08d]. The IAD reports that the guides are being used throughout the USFG as security baselines. Use of the guides does not appear to be mandatory.

- **Information Assurance Technical Framework (IATF):** The IATF is a comprehensive set of SSE practices for the system lifecycle, and it focuses on IT systems [IAD08e, IAS00]. It provides tutorial guidance and is not prescriptive. We were not able to investigate IATF as its website was unavailable, and the IATF document we found is dated, i.e., from 2000 [IAS00]. Further investigation is warranted, when the IATF website is available.

- **Information Security Assurance Training & Rating Program (IATRP):** This program provides SSE practices for vulnerability detection and analysis [IAD08f]. The program's focus is on working with vendors that provide these services to the USFG.

In addition, the NSA oversees the U.S. Common Criteria program. It is described in section 8.2 (page 45).

8.5 ISO 27K

The ISO 27000 standards (ISO 27K) are a series of ISO standards for IT-systems security. They provide SSE practices for security development and for security assurance. Over half of the ISO 27K certifications are in Japan, where the government mandates its use. In other countries where certifications have been issued, each country has relatively few certifications. For instance, 2733 certifications are held by Japanese organizations, but only 74 certifications are held by U.S. organizations.

There is an official ISO 27K website [ISO08]. ISO sells the documents for the ISO 27K standards, but they can be purchased from ANSI for much less [Ise08a]. The two primary standards are ISO 27001 [ISO05b] and ISO 27002 [ISO05a]. ISO 27002 was formerly the ISO 17799 standard, which itself was formerly known as BS7799-1 (from the British Standards Institute). The ISO 27001 Security website is a very useful source of information on ISO 27K [Ise08a]. Lance Spitzner has written a useful review of ISO 27K [Spi07].

About the author: Jim Yuill has over 20 years of experience in computer systems R&D. He has a PhD in computer science, with a thesis in computer security, from North Carolina State University (2006). https://jimyuill.com/