Exploring How to Apply Secure Software Design Principles

SHOUKI A. EBAD
Department of Computer Science, Faculty of Science, Northern Border University, Arar 91431, Saudi Arabia
e-mail: shouki.abbad@nbu.edu.sa

ABSTRACT
Secure design principles (SDPs) are employed to be a solution against many types of attacks. However, it has been shown that software designers are not familiar with the notion of SDPs or do not understand how to implement them in the design stage. This paper tries to bridge this gap by applying SDPs to a real-world software project, electronic promotion system (ePS), and commenting on the contribution of each SDP. Saltzer and Schroeder’s eight principles, along with three additional principles proposed by others, are chosen to be applied to ePS. The results show that most of the SPDs identified here were instrumental and applied in the ePS’s design. Most of the eleven SDPs, economy of mechanism, fail-safe defaults, least privilege, least common mechanisms, sound authentication, defense in depth, and input validation were implemented on ePS to a great extent. Others, namely separation of privileges and psychological acceptability, were applied to a limited extent. The remaining two principles, complete mediation and open design, did not play a vital role, as ePS by itself satisfies these two principles. Some contradictions and interrelations among the SDPs when they were applied were also debated. Taking into account the integration of ePS with other enterprise systems in the same organization, it was felt placing SDPs in a general context would be beneficial and sufficient. This work is expected to bridge the gap between software developers and state-of-the-art research on software SDPs.

INDEX TERMS
Software security, secure software engineering, secure design principles, software engineering, Saltzer and Schroeder’s principles.

I. INTRODUCTION

Even though security is a part of software quality assurance for all types of software [1], users are still struggling to maintain their information. Regrettably, thousands of software engineers are less experienced in security skills as these were not part of their curriculum at university [2]. Different security techniques have been proposed to be applied in each software development life cycle (SDLC) phase, comprising requirements, design, coding, and testing. However, 32% of research work in the existing literature focused on techniques solely in the design stage [2]. This is because the vulnerabilities of design are the major source of security risk in software systems, and 50% of software errors are often identified in the design stage [3]. Several specialists advise adding resources to improve the system design rather than fixing vulnerabilities in operational systems. This is because reducing the risks during the design phase may minimize the efforts in other phases [4]. Accordingly, a software designer must check the security affairs in this phase and make the required modifications in order to improve the overall security. To this end, the IEEE Computer Society Center for Secure Design was launched in 2014 to place more emphasis on design flaws than code bugs [5].

Software designers are expected to comply secure design principles (SDPs) that recommend best practices in the implementation of security metrics in a software [6]. Because they aim to improve security and reduce threats, SDPs are employed to be a solution to security problems of all system types: Software systems, hardware systems, network systems, or Internet of Things (IoT) systems. It is worth noting that the focus of this paper is on application-level SDPs (hereafter just SDPs). Therefore, it is not surprising that the first focus of MSSDL\(^1\) and CLASP\(^2\) is secure by design through implementing SDPs beyond coding practices. Practically, if the software developers implement these SDPs, their systems are expected to be secured against different

---
\(^{1}\)Microsoft secure development life cycle.
\(^{2}\)Comprehensive, lightweight application security process.
security threats. Unfortunately, there is a knowledge gap in SDPs; software developers and designers do not know how to apply and adhere SDPs in the software design phase [6]. The developers and designers rely on searching the internet and browsing e-books whenever they face security issues [7]. Trying to address this deficit early on, researchers have recommended including SDPs in software security-related education; rather than focusing on particular technologies, education should focus on understanding principles that can be abstracted and applied to several situations across various systems [2]. This paper attempts to fill this gap by learning by example, explaining how developers can apply SDPs via a real-world software project—electronic promotion system (ePS)—with the aim of improving software engineers’ knowledge to stop security violations in their systems. With no compatibility between research and the commercial environment of developers and their existing skills, research would not have a significant impact.

This article is structured as follows: Section 2 reviews the existing related works. The research approach is presented in Section 3. In Section 4, detailed modeling of the security requirements of ePS, an ongoing web-based academic promotion system, is described. Section 5 presents the results of our implementation of SDPs on ePS. In section 6, limitations of the study and threats to its validity are discussed. Finally, Section 6 concludes the paper with a recommendation for further research.

II. RELATED WORK
Arce et al. [5] identified some of the most important design flaws that have led to security problems over the past years. The list of concerns was presented by software security professionals who had contributed to both real-world data and expertise. To investigate their familiarity and working knowledge of SDPs, Almousa et al. [6] surveyed people who had experience in designing or developing software, exploring if the demographic variables were related to their knowledge of SDPs. They also found misconceptions about SDPs and collected participants’ opinions on the ways to implement SDPs. Alshammari [8] identified a set of security metrics to measure the security of object-oriented programs during the design and coding stages of development. He also developed a tool to quantify the security metrics of UML designs and Java programs. This program fixed a lot of bugs while adding functionalities. Smith [9] outlined a series of design principles for secure systems that have been in use since 1975. Some of these principles, including least privilege, open design, fail-safe defaults, separation of privilege, and least common mechanism have become staples of information security practice. In contrast, other principles such as economy of mechanism, complete mediation, and psychological acceptability were considered less important. Fernandes et al. [10] analyzed SmartThings apps (SmartApps) and several of device handlers. Although SmartThings implements a privilege separation principle, over 55% of SmartApps were over-privileged, owing to the capabilities being too coarse grained. Moreover, the SmartThings event subsystem, which uses devices to communicate asynchronously with SmartApps via events, does not sufficiently protect events that carry sensitive information, as in the case of lock codes. Shrobe et al. [11] reviewed and interpreted design principles that can advance total-system trustworthiness and a high assurance of computer systems. Here, they examined the extent to which the CHERI (Capability Hardware Enhanced RISC Instructions) system hardware-software co-design effort has successfully applied those principles (either intentionally or serendipitously).

Mosteiro-Sanchez et al. [12] analyzed the most relevant security strategies in Industry 4.0, focusing on the defense-in-depth principle with generic layers: Physical, perimeter, network, host, application, and data. They proposed an encryption algorithm called attribute-based encryption to achieve an end-to-end security approach. Mathas [13] proposed secure best practices for Java, PHP, and.NET and automated/manual code review. The automated review is made using the SonarQube and Reshift static tools to analyze Apache Unomi and dotCMS. The findings showed why static analysis becomes an integral part of SDLC while providing insights into the most cutting edge of static tools. They also demonstrated why it is not the only security measures that must be implemented. Alenezi et al. [14] synthesized security best practices presented by security models for linear and agile development lifecycles. This synthesis is geared toward providing a single unified source of knowledge about security best practices for software engineers. It is made available through the MediaWiki platform to assist practitioners in defending against security attacks and dealing with security errors in an appropriate manner. White [15] identified the need for a holistic approach to IoT security and identified causes for IoT insecurity such as secure SDLC adoption (including SDPs), organizational/human psychology, and security challenges due to the nature of IoT.

Table 1 summarizes the focus of all the studies mentioned above. In conclusion, it is clear the research community has given comparatively little attention to describing how to apply SDPs to a real-world software project. From Table 1, two studies focused on applying SDPs to a real-life system [10], [12]. However, these studies have two important limitations in comparison to the present study: (a) they did not focus on all SDPs but rather one or two principles, while this study addresses most if not all SDPs; (b) their studies were applied not on a software level but on a hardware or network level. This paper aims to fill this gap by applying SDPs to a real-world example in an effort to improve the understanding of software designers and developers of SDPs, though the understanding itself is another challenge [6].

III. RESEARCH APPROACH
A case study is a research strategy used to investigate an entity or phenomenon in its real-world situation [16]. It is used in various domains including sociology, medicine, and psychology. Since it has been shown that case study
methodology is suitable for industrial assessment of software techniques and tools, the term “case study” appears in the titles or abstracts of software engineering research articles [17]. Herein, this methodology is used to evaluate how certain principles apply to a real-life context.

### A. THE STUDY’S OBJECTIVE

The objective of this study can be expressed as follows:

“The purpose of this study is to apply the SDPs through a real-world software system from the point of view of software designers in the context of the industry.”

ePS, an ongoing software project at an educational institution, is used as a representative state-of-the-art practice. Using such an application from a public environment highlights the negative implications of introducing SDPs, both for other systems and for the designers and developers involved in the project. Beyond this, working with real-world software systems is often challenging when considerable computing research is required, especially in less-developed countries where management and organizational issues may include poor planning, resistance to change, government regulations, weak business process reengineering, and lack of awareness [18]. This paper attempts to present application of SDPs on an ongoing web-based application from a public university. The results are analyzed and discussed later in this article. At the time of writing, this ongoing system, ePS, is not yet implemented and remains in the design phase. For confidentiality, the university and system are labeled “UNV” and “ePS”, respectively. The premise behind ePS is a system that manages documents and other artifacts related to the academic promotion of faculty members at universities. This system has requirements that specify when those artifacts can be submitted to the system and who is authorized to submit and view them.

### B. RESEARCH QUESTIONS (RQs)

To accomplish this, the following three research questions (RQs) should be answered:

- **RQ1**: How should SDPs be applied to a real-world software application?
- **RQ2**: Which SDPs do or do not play a role in the protection of that software application?
- **RQ3**: Are there any contradictions or interrelations among SDPs?

---

### C. ePS DESCRIPTION

ePS is a web-based application that replaces a paper work when applying for academic promotion in UNV. It gives a platform to processing academic promotion for members who satisfy the suitable requirements. Promotion is an academic reward that gives faculty members some tangible and intangible features, such as raising salaries, reducing teaching-load hours, and evaluating the research of others. An applicant can be promoted via two levels, starting at the assistant professor level and progressing through associate professor and full professor. The entire process takes half an academic year and has several steps and categories of users. The steps obey a timeline that links users to multiple permissions at a variety of steps of the process that requires handling different data types of input/output shown in Table 2.

High-level requirements for ePS indicate that users must perform the tasks required of them to generate the artifacts for their given role(s). The portfolio has three components:

- University and community-serving activity such as teaching load, participating in committee’s work, and student’s evaluation.
- Research activity such as authoring books and publishing papers in journals/conferences.
- Submitting all forms required by S.C. such as curriculum vita (CV) and plagiarism statement.

Some of the above artifacts are instantly generated by other enterprise systems, such as enterprise resource planning (ERP) and student information system (SIS).

### D. PROMOTION PROCESS

The S.C. administration has a clear vision about what should be achieved by using its information technology. The primary objective behind designing and implementing ePS was twofold: first, to integrate the major functions and processes into a technological platform, effectively supporting the completion of all tasks within these functions. Second, to provide the S.C. with complete data sets in
deciding on promotions. A step toward this objective was the re-engineering of the paper-based work. This meant that the faculty members had to throw out the paper-based system and switch to a web-based system. The development team communicated with the S.C secretary and conducted 6 to 7 milestone face-to-face meetings.

IV. ePS MODELING

A. SECURITY REQUIREMENTS

ePS’s requirements for step-based functionality produced three security challenges for the software designers. The first required tracking the different steps of the whole process. Therefore, the timeline mechanism was proposed in order to test the current date against the holidays when all personnel are not working. The second required identifying a user’s permissions: a concern that was met by assigning each user to a role. The third required tracking the completeness of the documents associated with single promotion cases. Figure 1 is a data-driven model of ePS. Data-driven models show the whole sequence of actions that happen from an initial input being processed to the corresponding output, i.e., end-to-end processing in ePS [1]. In this type of model, the activities are represented by rounded rectangles, and the data flowing between these activities are represented by rectangles. Each process relates to role information and that process’s required actions.

The request with the portfolio is sent or received from or by several people at different levels in UNV as follows: First, the request for promotion is submitted to the head of department (HoD) along with the whole portfolio. Second, if HoD makes no comment, the case is presented to the department council to gain approval from its members, after which the case, along with the department minutes of meeting (MoM), goes to the college council (higher level) to gain approval as well. This transformation takes 2–4 weeks in order to allow the members of both councils to look at the entire transaction. The college dean then sends the complete transaction with the college MoM to the university rector for approval. With the three approvals (Department, College, and Rector), the transaction is sent to the S.C. Secretary who checks briefly if there is any missing or incomplete artifact. If no comment is given by the S.C. Secretary, the transaction moves to an internal committee for detailed evaluation. Finally, the whole case is summarized in a memo to be presented to the S.C. in the nearest meeting. With the S.C.’s approval, the case is sent to three external reviewers to evaluate whether the applicant deserves the promotion. If any of these levels does not approve the transaction, it goes back to the previous level along with a justification of the rejection or asking for a modification. As in Figure 1, the external evaluation subsystem starts with the selection of a reviewer by the S.C. (or S.C. Secretary if he/she has had a delegation from S.C.); if the reviewer approves, he/she...
can access ePS to view the research papers, which he/she would then evaluate and submit reports on with a clear recommendation for promotion. The names of reviewers are kept anonymous for all ePS users except the S.C. Secretary. It is difficult if the above workflow relies on paperwork, as it all portfolios tend to accumulate under the S.C. secretariat, creating a large volume of transactions that, in turn, results in delays in the execution of requests. The point is to avoid sending and receiving documents through a traditional e-corresponding system, where the applicant prepares the portfolio manually, making the loss of handwritten forms or the complete applicant’s portfolio and mistakes inevitable. In this scenario, many problems may occur regarding the accuracy and speed of the completion of the process.

At each step, ePS users can invoke actions associated only with that step or role. Other actions include showing the portfolio, and adding, modifying, and deleting (AMD) the related artifacts. To display the artifacts from a past step, all actions involved for that step should have been completed. It includes submitting needed reports, documenting required recommendations, and getting approval for all needed reports by related users. Users of ePS should only view applicants that they have permission to see. These permissions are specified by a membership in the department, college, department council, college council, and the S.C. Secretaries, including the promotion committee. A single administrative role (i.e., super-user) that monitors the system in which he/she can view all applicants has been given to the S.C. Secretary according to the formal paper-based rules. The above functionality-based interaction can be modeled by a use case (UC) diagram; each UC shown as an ellipse represents a discrete task that involves external interaction by actors depicted as stick figures [1]. A UC-based elicitation is used in the requirements engineering process. Figure 2 gives a UC diagram of an overview of ePS’s interaction, but more detail for interaction description, such as constraints on some roles, actors, or interactions, can be added. The UNV’s rector is ignored here because he/she does not use ePS; the transaction is sent or received to or from him/her by the formal e-corresponding system.

**A. ANSWERING TO RQs**

1) **HOW SHOULD SDPs BE APPLIED TO A REAL-WORLD SOFTWARE APPLICATION? (RQ1)**

Saltzer and Schroeder proposed eight design principles to secure computer systems, namely economy of mechanism, fail-safe defaults, complete mediation, open design, separation of privileges, least privilege, least common mechanisms, and psychological acceptability. Although these principles were proposed decades ago, they are as precise now as they were when originally written. They have been used repeatedly and successfully in the design and implementation of different systems [1], [9], [14], [20].

However, some articles and textbooks might use different terms. For instance, some sources characterize separation of privileges as a control rather than a principle. Others present a more specific variant of some principles. For instance, segregation of duties and deny by default are nothing more than separation of privileges and fail-safe defaults, respectively [9]. Moreover, researchers have also put forward other principles to reflect current technology and terminology, such as defense-in-depth, sound authentication,

**B. ePS LIMITED VIEW DESIGN PATTERN**

The Limited View (LV) pattern allows functions that the present user is privileged to access [19]. With LV, protection checks are made before creating the view. It takes the user permissions of the current session and creates an interface checks are made before creating the view. It takes the user permissions of the current session and creates an interface designed for that user. This pattern is ideal for ePS owing to the set of operations it provides, and most ePS users have permission for a limited number of operations. LV provides users with those operations that they may access. A general LV implementation was created as follows:

- Each role was restricted to the functionality exist to this user.
- User functionality was restricted to a subset given by the working system, including uploadPortfolio, showPortfolio, and submitRecommendation, uploadMoM.

Five roles were considered for this exercise: Applicant, College Representative, HoD, Dean, and S.C. Secretary. For instance, if the user were logged as a HoD, the process would enter the “department review” step. The HoD can then make the following functions:

- View the portfolio.
- Enter HoD’s initial evaluation.
- Upload department MoM regarding the promotion request.
- Submit the entire case to the next stage/level (i.e., Dean).
- HoD can AMD any file he/she has uploaded.

The designers considered different security requirements that require particular elements to be properly defined. A role is nothing more than a particular type of user. For instance, the applicant role is linked to faculty members who are applying for promotion, and there are also specific types of files, such as portfolio documents, and evaluation by College Representative/HoD/Dean. “Upload” privileges are specific to the roles. For instance, the applicant role submits applicant artifacts such as portfolio documents, HoD uploads evaluation reports, and the department MoM and external reviewers submit their evaluation reports. ePS uses four classes of artifacts:

- Documents that represent portfolio documents
- Evaluations by College Representative, HoD, Dean, and S.C. Secretary
- MoM by department and college
- Reports by all three external reviewers.

In addition, the timeline entity determines when a role can upload and show files. Figure 3 (part of the class diagram) illustrates the relationship of the timeline with roles.

**V. RESULTS AND DISCUSSION**

1) **SALTZER AND SCHROEDER DESIGN PRINCIPLES**

Saltzer and Schroeder proposed eight design principles to secure computer systems, namely economy of mechanism, fail-safe defaults, complete mediation, open design, separation of privileges, least privilege, least common mechanisms, and psychological acceptability. Although these principles were proposed decades ago, they are as precise now as they were when originally written. They have been used repeatedly and successfully in the design and implementation of different systems [1], [9], [14], [20].

However, some articles and textbooks might use different terms. For instance, some sources characterize separation of privileges as a control rather than a principle. Others present a more specific variant of some principles. For instance, segregation of duties and deny by default are nothing more than separation of privileges and fail-safe defaults, respectively [9]. Moreover, researchers have also put forward other principles to reflect current technology and terminology, such as defense-in-depth, sound authentication,
and input validation [1], [5], [9]. Basically, these principles are summarized here:

1. **Economy of mechanism**: Keep the design as simple and small as possible.
2. **Fail-safe defaults**: “No access” or “deny” unless a user is explicitly provided access.
3. **Complete mediation**: Check every access with no exception.
4. **Open design**: It should not depend on security by obscurity.
5. **Separation of privileges**: Use separate privileges or multi-party authorization (e.g., two keys to unlock).
6. **Least privilege**: Each user or program should operate by using the fewest privileges.
7. **Least common mechanisms**: Minimize function common to more than one user.
8. **Psychological acceptability**: The human interface is created for ease of use. Users routinely apply the protection mechanisms correctly.
9. **Sound authentication**: Validating an entity’s identity.
10. **Defense in depth**: Employ several different techniques to ensure security.
11. **Input validation**: Unexpected input format might cause the system to behave in an unanticipated way.

The above eleven SDPs can drive the answer to RQ1 from an application-level perspective, where ePS is the examined real-world example. Table 3 explains the applicability of each of the above SDPs in meeting the security requirements specific to ePS. From that table, some SDPs (e.g., open design and complete mediation) might not be as important these days, as networking in the current organizations applies such principles in any case. ePS would work under the UNV’s network, where techniques used to secure the network reflect on other enterprise systems in the UNV, including ePS.

2) **WHICH SDPS DO OR DO NOT PLAY A ROLE IN THE PROTECTION OF THAT SOFTWARE APPLICATION? (RQ2)**

The primary observations of the importance of each SDP to ePS are as follows:

**a: EFFECT OF THE UNV’S ENTERPRISE SYSTEMS ON SDPs**
Enterprise systems integrated with ePS include ERP and SIS. These systems clearly play a critical role in ePS. Any incorrect data found in such systems would affect ePS’s behavior and consequently affect the final decision of promotion. Accordingly, the principles of economy of mechanism, fail-safe defaults, and sound authentication are expected to be affected in this regard, since roles in ePS originate from ERP and, consequently, their privileges would be granted or revoked according to those roles. The evaluation related to past teaching load would be performed according to what is displayed in SIS.

**b: ECONOMY OF MECHANISM**
The Economy of mechanism or simplicity principle is applied to ePS to a considerable extent. ePS did not, however, address the principle from all aspects. For example, an evaluation report submitted by all users, including the external reviewers, should be formatted using a word processor so that the service’s features, such as spell check and word count, are readily available. From an evolution attribute perspective, ePS is designed according to ministry regulations, which are updated periodically. The importance of the economy of mechanism or simplicity principle has therefore become questionable in practice despite its importance in principle. Updating according to ministry rules differs from that of any other authority or administration because the ministry often represents the state regime. In the future version of ePS, complexity might be unavoidable; as Albert Einstein once said, “Everything should be made as simple as possible, but no simpler.”

**c: COMPLETE MEDIATION**
The principle of complete mediation does not play an important role in ePS’s design. It may have been important for computer systems in the early part of the 21st century, however, before information assurance had become a key element in most organizations. These days, access request decisions in the UNV’s network are distributed among separate mechanisms, including antivirus programs, web/spam filters, and firewalls. As such, this principle has become impractical and reflects an obsolete picture of security decision making.

**d: OPEN DESIGN**
Similar to the principle of complete mediation, there is no real need for the principle of open design in ePS, which is itself open to design irrespective of its security constraints. Open design is a very general design principle whose ultimate goal is to minimize secrets. However, simply avoiding the “security by obscurity” principle is too vague to be of practical use [1]. Although a good design principle is always good for security [20], design and implementation cannot be left unclear until the late development stages. There is considerable debate about the importance of open design in relation to security, with some people insisting that the principle of “security by obscurity” is reasonable, though design flaws are so apparent as to be discovered from outside, even with no de-obfuscation.3 However, there are situations in which this principle is required, as in the cryptographic approach, where the private keys should be hidden while the algorithm code is public.

**e: SEPARATION OF PRIVILEGES**
The principle of separation of privileges is restrictive because it limits access to system entities [6]. This principle is applied to ePS by prohibiting one user to work with more than one role simultaneously. For example, it is possible for Role 1 also to be Role 2 (e.g., where the Dean is also HoD); this was considered when setting up user permissions. After the user login, he/she must select the role (e.g., as an applicant, College Representative, HoD, Dean, or S.C. Secretary), and when he/she needs to switch to another role, he/she should log out of the present role and log in again with the other role. This principle might be necessary for a particular type of system that requires two keys for unlocking or a two-person control mechanism like safety-critical systems, such as the one in ePS.3

---

3For a detailed review about measuring obfuscation quality, interested readers are advised to consult [21].
| # | SDP | How to apply? |
|---|---|---|
| 1 | Economy of mechanism | - ePS does not require installation on client machines but runs as a service and is accessed via a web browser.  
- Uses the Single sign-on (SSN) mechanism.  
- Large amounts of data are automatically retrieved from other enterprise systems. For example, CVs and applicant’s teaching loads could be automatically retrieved from ERP and SIS, respectively.  
- As with CVs being retrieved from ERP, authorization is given based on the role. For example, updating the user’s position title by ERP is reflected in ePS; if the user became a HoD, all HoD privileges would automatically be granted to him/her. |
| 2 | Fail-safe defaults | - For now, the ePS only works within the internal UNV network; i.e., it does not work outside the campus.  
- If role R with level L generates artifact A for a higher level, then all levels below L have no access to A. For example, what the Dean uploaded to the S.C. Secretary could not be seen by Applicant, College Representative, and HoD, as shown in Figure 4.  
- In addition, all privileges of a user associated with a role R are revoked from him/her (i.e., “no access”) when he/she is no longer with R, based on his/her current state in ERP; updating ERP requires explicit permission. |
| 3 | Complete mediation | At present, decisions of access requests in UNV’s network are distributed among separate mechanisms, including antivirus programs, web/spam filters, and firewalls (see the answer to RQ2). |
| 4 | Open design | ePS by itself is open to design (see the answer to RQ2). |
| 5 | Separation of privileges | - Users must be linked to multiple roles. It is possible for an HoD to also be an applicant. This must be taken when create committees and user permissions.  
- After the user login, he/she must select his/her role (i.e., as an applicant, College Representative, HoD, Dean, S.C. Secretary). |
| 6 | Least privilege | - AMD operation on the artifact is done by specific users according to their roles.  
- Privileges increase from applicant role (low level) up to S.C. Secretary (high level).  
- Modification to the applicant’s portfolio is made by only one role (S.C. Secretary).  
- External reviewers are given temporary accounts to access ePS; the accounts are removed immediately after the evaluation is done. |
| 7 | Least common mechanisms | Sharing the same function with multiple roles is commonly avoided. The need for and use of overly functional and service roles is also minimized, even with the S.C. Secretary role who looks like a super-user (Figure 2). |
| 8 | Psychological acceptability | - SSO has enormous appeal in relation to the principle of psychological acceptability.  
- ePS’s interface is designed according to a pre-existing style that had already been used in UNV. Users would thus be familiar with the security aspects of the interface and its subsystems to a reasonable extent.  
- ePS’s usability (i.e., ease of use) depends on different IT infrastructure entities, such as its technical components, operators, and operating environment. Much of ePS’s capability can be hidden by such entities. |
| 9 | Sound authentication | - Authenticating users works by using the UNV’s centralized authentication service, Active Directory by Microsoft. Other integrated systems, such as ERP and SIS, authenticate themselves to ePS. A single mechanism of this kind can serve as a logical “choke point” that cannot be bypassed [5].  
- In addition, resetting a password requires two-factor authentication. |
| 10 | Defense in depth | Like in other enterprise systems in UNV, generic security layering is used [12]:  
- Physical layer: for example, entry to the datacenter is made using handwritten permission, and USBs are not allowed.  
- Network layer: for example, IDS/IPS monitoring systems, firewalls, and remote access like VPN services are controlled.  
- Application layer: two-factor authentication is applicable for sensitive functions as mentioned in the sound authentication principle. See Figure 5. |
| 11 | Input validation | All input formats are specified correctly. In addition, to reduce the mistakes in this regard, many inputs), such as the country of the external reviewer, and the weight of each research paper, are not entered by keys but by the selection of menu options via a mouse (i.e., a pull-down menu. ePS tries to run using a WIMP interface (windows, icons, mouse, and pull-down menus). |
as nuclear weapons, top-secret crypto materials, and controls in aircrafts. In short, some SDPs (e.g., separation of privilege, complete mediation, and open design) may be useful by themselves in some systems while not in others.

**f: INPUT VALIDATION**

Figure 6 shows a potential weakness that could take place in ePS. The principle of input validation may be violated when the applicant uploads the research paper as an image or scanned document and ePS accepts it because the file extension is in a pdf format. In such a case, the format should not be accepted by ePS, given that the nature and content of the file are not pdf, regardless of the file extension. This problem will appear when the S.C. Secretary tries to check the similarity rate/plagiarism with other materials (papers, documents, websites, etc.). Usually, the plagiarism programs (e.g., iThenticate tool⁴) care for the storage mechanism. Such programs usually fail to generate the similarity report unless the file under evaluation is in true pdf format.

3) **ARE THERE ANY CONTRADICTIONS OR INTERRELATIONS AMONG SDPs? (RQ3)**

Based on ePS example described above, some of the SDPs are contradicted and others are interrelated:

- **a: ECONOMY OF MECHANISM VERSUS LEAST PRIVILEGE**
  
  The principle of least privilege sounds contradictory to what the economy of mechanism principle aims for in ePS, because the former principle works on the need-to-AMD idea that restricts the simplicity feature to some extent. Additionally, although ePS runs as a service and is only accessed via a web browser, the server software is still vulnerable to deployment errors and omissions.

- **b: AUTHENTICATION VERSUS PSYCHOLOGICAL ACCEPTABILITY**
  
  ePS depends on textual passwords in the Active Directory’s authentication system. Such a method fails the psychological acceptability principle. Users have been harangued for decades to pick hard-to-guess passwords, not share them, not use the same one in multiple places, and so on [22]. In reality, several people do not comply those rules [9]. Using another authentication mechanism aside from the password, such as the biometric mechanism, may coincide with psychological acceptability.

- **c: PSYCHOLOGICAL ACCEPTABILITY VERSUS SEPARATION OF PRIVILEGES**
  
  As mentioned in Table 3, ePS’s SSO solution has a large appeal in relation to the principle of psychological acceptability. However, as a consequence of exposure from inside or outside, this solution can produce several security vulnerabilities. Having the simplicity provided by the SSO

---

⁴https://www.ithenticate.com/
VI. THREATS TO VALIDITY

Three concerns that may threaten the validity of this research are presented here, internal validity, construct validity, and external validity [16]:

- Internal validity: No causal references have been made throughout this study, so internal validity is not a concern.
- Construct validity: It is debatable whether using SDPs in a direct way as conducted in this study is a good method to build secure systems. There is a possibility that some errors or mistakes are made during the entire process, i.e., it is easy to misuse some of the SDPs. In contrast, other methods try to apply the SDPs indirectly by embedding them in the system artifacts.
- External validity: The results were obtained by applying SDPs on a web-based application, ePS, for academic promotion in an academic organization. The research findings may be generalizable to a similar type of software in a similar context. However, the validity of this claim requires additional research and analysis, because the RQs herein are not general but based on one case (i.e., specific to the example of ePS).

VII. CONCLUSION AND FUTURE WORK

The literature showed that the vulnerabilities of the system design level are the major source of security risk. Besides the security standards, the software engineers and designers should follow secure design principles (SDPs) that are employed as a solution against many types of attacks. Recent studies indicated that there is a misconception about SDPs and that designers are not familiar with the idea of SDPs. This paper has sought to fill this gap by instructing the developers how to apply SDPs through an ongoing real-world software system called ePS, an electronic promotion system, and through observations about the contribution of each SDP. The paper provided a detailed description of ePS, including the development procedures, non-functional requirements, and data modeling. Saltzer and Schroeder’s eight principles, along with three additional principles proposed by others, were chosen to be applied here. Saltzer and Schroeder’s principles include economy of mechanism, fail-safe defaults, complete mediation, open design, separation of privileges, least privilege, least common mechanisms, and psychological acceptability. The additional three are sound authentication, defense in depth, and input validation. The results show that most of the SDPs enumerated here were instrumental and applied in ePS’s design. Most SDPs were implemented on ePS to a great extent, such as economy of mechanism, fail-safe defaults, least privilege, least common mechanisms, sound authentication, defense in depth, and input validation. Others applied to a limited extent, namely separation of privileges and psychological acceptability. The last two principles, complete mediation and open design, were not relevant to ePS because they were already implemented. Some contradictions and interrelations among the SDPs once they were applied were also debated. Taking into account the integration of ePS with other enterprise systems in the same organization, it is felt that placing SDPs in a general context will be beneficial and sufficient. Besides bridging the gap between software practitioners and state-of-the-art research on software SDPs, this work may serve as a foundation for further studies in the application of SDPs.

What would be desirable for secure systems is that practical theory and theoretical practice sufficiently overlap. What is called “best practice” is usually a common technique that found its way into practice [11]. However, such practices or principles can be misapplied even by highly qualified programmers, and bad programming languages can still be used. Having practical designs is a good place to start; in terms of further research, the pressing need is to find design principles or propose new ones that are appropriate for system development, as this can considerably improve system security and cybersecurity in general. The second opening for future work is to suggest an approach for how developers of ePS or other applications can improve their understanding and knowledge. There are no statistics on the team that was developing ePS as it relates to SDPs. It would be beneficial for the designers of ePS to follow some rubric to provide the
relevant SDPs, after which the researchers can compare those SDPs to theirs. This way, they can say that their approach improved the designers’ understanding of SDPs.

REFERENCES

[1] I. Sommerville, *Software Engineering*, 10th ed. London, U.K.: Pearson, 2015.

[2] S. A. Ebad, “What topics should or should not be included in software security education—Qualitative content analysis,” *Comput. Appl. Eng. Educ.*, vol. 30, no. 6, pp. 1753–1773, Jul. 2022, doi: 10.1002/cae.22554.

[3] R. A. Khan, S. U. Khan, H. U. Khan, and M. Ilyas, “Systematic mapping study on security approaches in secure software engineering,” *IEEE Access*, vol. 9, pp. 19139–19160, 2021, doi: 10.1109/ACCESS.2021.3052311.

[4] V. Maheshwari and M. Prasanna, “Integrating risk assessment and threat modeling within SDLC process,” in *Proc. Int. Conf. Inventive Comput. Technol. (ICICT)*, Coimbatore, India, Aug. 2016, pp. 1–5, doi: 10.1109/INVENTIVE.2016.7823275.

[5] I. Arce, K. Clark-Fisher, N. Dawsani, J. DelGrosso, D. Dhillon, C. Kern, T. Kohno, C. Landwehr, G. McGraw, B. Schoenfeld, M. Seltzer, D. Spinellis, I. Tarandach, and J. West, “Avoiding the top 10 software security design flaws,” *IEEE Comput. Soc.*, Washington, DC, USA, Tech. Rep., 2014. Accessed: Sep. 25, 2022. [Online]. Available: https://ieeecs-media.computer.org/media/technical-activities/CYBSI/docs/Top-10-Flaws.pdf

[6] M. Almousa, M. Keshavarz, and M. Anwar, “Awareness and working knowledge of secure design principles: A user study,” in *HCI for Cybersecurity, Privacy and Trust* (Lecture Notes in Computer Science), vol. 12210, A. Moallim, Eds. Cham, Switzerland: Springer, 2020, doi: https://doi.org/10.1007/978-3-030-50309-3_1.

[7] Y. Acar, C. Stranksy, D. Wermke, C. Weir, M. L. Mazurek, and S. Fahl, “Developers need support, too: A survey of security advice for software developers,” in *Proc. IEEE Cybersec. Develop. (SecDev)*, Cambridge, MA, USA, Sep. 2017, pp. 24–26, doi: 10.1109/SecDev.2017.17.

[8] B. Alshammari, “Quality metrics for assessing security-critical computer programs,” Ph.D. thesis, Dept. Sci. Technol., Queensland Univ. Technol., Brisbane, QLD, Australia, 2011.

[9] R. E. Smith, “A contemporary look at Saltzer and Schroeder’s 1975 design principles,” *IEEE Secur. Privacy*, vol. 10, no. 6, pp. 20–25, Nov./Dec. 2012.

[10] E. Fernandes, J. Jung, and A. Prakash, “Security analysis of emerging smart home applications,” in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Jose, CA, USA, May 2016, pp. 636–654, doi: 10.1109/SP.2016.44.

[11] H. Shrobe, D. L. Shrier, and A. Pentland, “Fundamental trustworthiness principles in CHERI,” in *New Solutions for Cybersecurity*. Cambridge, MA, USA: MIT Press, 2018, pp. 199–236.

[12] A. Mosteiro-Sanchez, M. Barcelo, J. Astorga, and A. Urtiaga, “Securing IoT using defence-in-depth: Towards an end-to-end secure industry 4.0,” *J. Manuf. Syst.*, vol. 57, pp. 367–378, Oct. 2020.

[13] C. Mathas, “Secure coding practices for web applications,” M.S. thesis, School Inf. Commun. Technol., Dept. Digit. Syst., Univ. Piraeus, Piraeus, Greece, 2021.

[14] M. Alenezi, H. A. Basit, M. A. Beg, and M. S. Shaukat, “Synthesizing secure software development activities for linear and agile lifecycle models,” *Softw., Pract. Exp.*, vol. 52, no. 6, pp. 1426–1453, Jun. 2022, doi: 10.1002/pace.3072.

[15] C. A. White, “Root causes of insecure Internet of Things and holistically addressing them,” in *Proc. Int. Conf. Comput. Sci. Comput. Intell.* (CSCI), Las Vegas, NV, USA, Dec. 2020, pp. 1066–1074, doi: 10.1109/CSCIS1800.2020.00198.

[16] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell, and A. Wesslen, *Experimentation in Software Engineering*. Berlin, Germany: Springer, 2012.

[17] C. Wohlin, “Case study research in software engineering—It is a case, and it is a study, but is it a case study?” *Inf. Softw. Technol.*, vol. 133, May 2021, Art. no. 106514, doi: 10.1016/j.infsof.2021.106514.

[18] S. A. Ebad, “An exploratory study of ICT projects failure in emerging markets,” *J. Global Inf. Technol. Manag.*, vol. 21, no. 2, pp. 139–160, Apr. 2018.

[19] J. Yoder and J. Barcalow, “Architectural patterns for enabling application security,” in *Proc. 4th Conf. Patterns Lang. Program. (PLoP)*, Monticello, IL, USA, 1997, p. 30.

[20] C. P. Pfleeger and S. L. Pfleeger, *Security in Computing*, 4th ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2006.

[21] S. A. Ebad, A. A. Darem, and J. H. Abawajy, “Measuring software obfuscation quality—A systematic literature review,” *IEEE Access*, vol. 9, pp. 99024–99038, 2021, doi: 10.1109/ACCESS.2021.3094517.

[22] S. A. Ebad, “Lessons learned from offline assessment of security-critical systems: The case of Microsoft’s active directory,” *Int. J. Syst. Assurance Eng. Manag.*, vol. 13, no. 1, pp. 535–545, Feb. 2022, doi: 10.1007/s13198-021-01236-2.

SHOUKI A. EBAD received the Ph.D. degree in computer science and engineering from the King Fahd University of Petroleum and Minerals, Saudi Arabia, in 2012. He is currently working as an Associate Professor with the Department of Computer Science, Faculty of Science, Northern Border University, Saudi Arabia. Before that, he held several positions, such as a Lecturer, the Head of Department, the Vice-Dean, the Assistant Dean for Technical Affairs at IT Deanship, and the Secretary General of the Scientific Council. He is also a Sun Certified Programmer for the Java 2 Platform. His current research interests include software engineering and software security. He has published a number of articles in these areas.