Experiments and Proofs in Web-service Security

Dawood Sheniar*, Nabeel Hadaad*, David Martin*, Ron Addie* and Shahab Abdullah†

*Faculty of HES, University of Southern Queensland, Australia, {dawoodsallemhussian.sheniar, NabeelMahdy.Hadaad, u1047621, ron.addie}@usq.edu.au

†Open Access College, University of Southern Queensland, Australia, shahab.abdulla@usq.edu.au

***Abstract*—Many web services have a subsystem for allowing users to register, authenticate, reset their password, and change personal details. It is important that such subsystems cannot be abused by attackers to gain access to the accounts of other users. We study a system which was initially prone to such attacks. Specific attacks are demonstrated and the system is then modified to prevent such attacks in future. The design achieved in this way is then analysed to show that it can't be broken in future unless users allow their email to be intercepted. This is achieved by formulating the requirement as a statement of the user's expectations of the system and then analysing the source code of the system to prove that it meets these requirements. The process of attack, correction, and formulation of security rules, and proof that rules hold, is proposed as a methodical security design philosophy.**

***Keywords*—web service security, security design, password reset, security rules, stakeholder analysis.**

I. Introduction

VULNERABILITIES of web systems take a great variety of forms and new ones appear to emerge regularly, so it can seem an endless process to manage and maintain web site or web service security [1]. An approach used regularly is to assign the task of attacking a site to an individual or a team and then to address the discovered weaknesses. We call this a *security audit*. The approach of searching for weaknesses and fixing them is so widely used that it might reasonably be regarded as a design philosophy.

This approach has been used in this paper, and the results of both the attacks and the resulting defences are reported. However, this approach is somewhat pessimistic in that it *assumes* that a more methodical security design philosophy which can guarantee secure design is not available. The main result of this paper is to demonstrate such a methodical design philosophy. We call this *stakeholder security analysis*.

Stakeholder security analysis proceeds as follows:

1. Identify the key stakeholders.
2. For each stakeholder, identify a set of rules *required* by these stakeholders. Note: the collected rules required by all stakeholders must be consistent.
3. Implement procedures which ensure that all rules are enforced.

Both a security audit and a stakeholder security analysis are applied in this paper to a specific subsystem of a web service system being developed and managed by the authors. Note: it is not suggested that stakeholder security analysis obviates the need for a security audit.

A. The Netml System

The Netml system provides services for analysis, design, and implementation of networks [2]. These services are provided by means of a web site. No software is installed on users' computers (except in the form of cached javascript). This system is used in teaching, by computer science students, and in research into network analysis and design.

A key requirement of this system is that users can readily create their own accounts and can reset their password if necessary when they have forgotten it. Users can share networks that they create or load with each other, but by default users cannot access the data of other users.

B. Password Reset Service

Many web systems include a service for allowing users to register, authenticate, reset their password, and change personal details. Password re-setting is one of the most common tasks faced by an enterprise information help desk. Previous studies have shown that password re-setting accounts for about one in four help desk requests [3]. The human involvement in the password re-setting process is costly to support. Thus, it is desirable to have an automated way to authenticate a user for the purpose of password resetting in an enterprise environment [4]. Reset mechanisms are therefore essential at almost every password-protected site to handle forgotten passwords.

However, a password reset system is complex to implement, and can easily include errors which open the system to attack [5].

Typically, password reset systems make use of another system for checking the credentials of a user to confirm the user's identity, for example, the user's email system, or their phone. Any defect in this second system (for example, if the user has revealed their email password to another user, or has allowed others' free access to their phone) will compromise the password reset system which relies on it, so the second system must be one with significantly greater importance, for the user, for this type of design to be satisfactory.

Section II surveys current approaches to security design, including a description of three specific approaches. In Section III the original design of the Netml password reset system is described; in Section IV, a series of experiments in which this system is attacked and then improved (a security audit), is described; in Section V, a stakeholder security analysis is conducted and used to revise the design; in Section VI, the revised design is *proved* to be correct (in respect of two of the rules). Conclusions are presented in Section VII.

II. Web Service Security Design

Because web services (including services provided via apps on mobile phones) are a recent development and continue to evolve in both details and fundamentals, principles of secure design of these services is also a new and evolving area of research and development [6].

This section reviews three different approaches for securing web sites/services. Each of these approaches is usually expressed as a completely independent philosophy for achieving good security. We shall see that these approaches are actually complementary, and to achieve rigorous security all three approaches are needed. Note that although we describe a design philosophy which is able, formally, to prove, i.e. guarantee, security, because no logical system can claim certainty in an absolute sense (in mathematical logic, this fact is expressed in Gödel's incompleteness theorem), the strategy of attacking the system remains useful, even after it has been methodically proved to be correct.

The present paper does not apportion equal emphasis on all approaches because the original contribution of this paper is in the third of them, together with the way the second approach joins with the third to form a more comprehensive whole. The second approach is the one summarised in Subsection II-B and applied in Section IV. The third approach is summarised in Subsection II-C and applied to the Netml password reset system in Sections V and VI.

A. Good Security Design Practice

Good design takes security, ease of access, and usability into account, striking a balance between protecting the system and ease of use. Good practice has evolved a number of practical approaches like minimizing attack surface area [7], establish secure defaults [8], using the principle of defence in depth [9], not trusting services [10], keeping security simple [11], and fixing security issues correctly [12].These approaches are used for maintaining and improving security which are so natural and important that they should be adopted as a first layer of protection as a matter of standard practice, even when more sophisticated approaches are also in use [13].

B. Security Auditing

Strategies for breaking into web systems or services are under continuous development by government and non-government organisations and individuals, both those with friendly intentions and those who wish to exploit security weaknesses for their own advantage. When a new exploit is discovered, if it is discovered first by those with friendly intentions, defences against the exploit are usually developed quickly and published. Exploits discovered by attackers with ill intent can, of course, be deployed before web managers have the opportunity to defend against them. Also, in the period of time immediately after the defence against a new exploit has been published, there is still an opportunity to attack web sites which have not deployed the newly developed defences. This time can be somewhat extended due to the limitated expertise of web-site owners and because the sequence of steps required to address a weakness in a high-level framework can be quite lengthy.

A widely used strategy for improving web site or web service security is to attempt to attack the site by using the strategies which are currently known to be effective one-by-one, or simultaneously, to discover if the site is vulnerable to any of these strategies. Since all of the strategies tried are known, the defences against all of them are also almost certainly known, and hence can be adopted by the web site.

Experiments of this type, and the resulting web service design improvements, are described in Section IV.

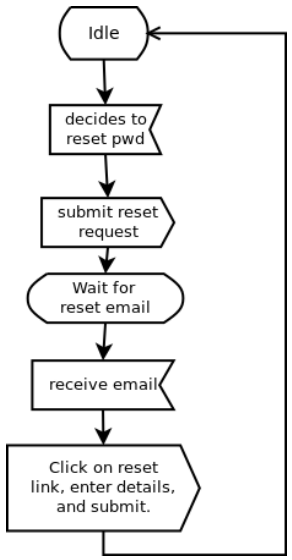

Fig. 1. SDL diagram for User behaviour when resetting their password

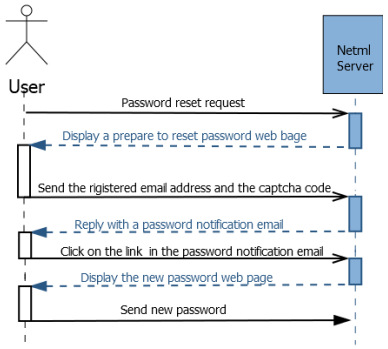

Fig. 2. Sequence diagram for the Netml password reset system

C. Stakeholder Security Analysis

A more fundamental strategy which is not well-developed at present is to seek to develop provably secure protocols and software for all aspects of a web service [14], [15], [16].

The first step in this approach, which is developed further in Section V, is to consider the point of view of all legitimate stakeholders in relation to the service, and to enumerate a complete set of rules required by each of these stakeholders, sufficient to ensure that they will agree to actively participate in the service.

III. Original Design of Password Reset Service

The original design of the password reset service of the Netml system is described in Figures 1 and 2.

There is nothing exceptional about this design. However, in Section IV we discover that there are mistakes in the implementation of this design which make it insecure.

IV. Break-in experiments

A series of experiments in which the system is attacked, successfully, and then modified, are described in this section. These experiments took the form described in Subsection II-B, i.e. a series of attacks were undertaken in accordance with known vulnerabilities and observed behaviour of the system.

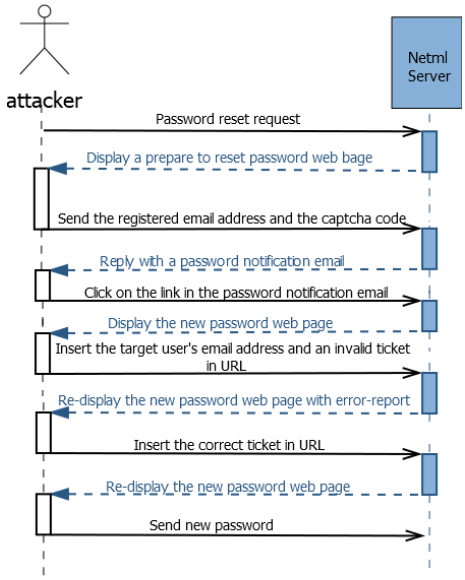

Fig. 3. Sequence diagram for the Netml password reset system under attack

A. First Experiment

A detail not mentioned in Section III is that when an attempt to update a user's password is made, with an invalid ticket, the web page which is prepared to report an error includes both the invalid ticket, and the value that the ticket should have taken. This is a fatal mistake (from the point of view of good security). There is no need to report to an ordinary user what the value of the ticket should have been. The reason this was included in the error-report web page was to enable the software to be readily tested and debugged. However, this information should have been removed from the error page in the working system.

The procedure used to change the password of a different user was as described in Figure 3. First, the password reset system was used to reset the password of the attacker (a perfectly valid procedure, with no ill intent or consequence). During this process, the URLs used at each stage were observed. In particular, the URL which was used to complete the reset of the password of user `addie11` was of the form shown in Listing 1. A URL

```
https://netml.usq.edu.au/netml4_6/registration.jsp?
resetpwd=true&ticket=12345&uname=u1047621&fullName=fake&
email=u1047621@umail.usq.edu.au&organisation=USQ
```

Listing 1: Form of URL which causes the password change web page to be displayed

of this form was then used with the target user's email address and an invalid ticket. This generated an error report page which revealed the value the ticket should have had. Finally, the correct ticket was inserted into the URL and it was sent to the server. This URL caused the web page for supplying a new password to be displayed, which was then used to change the target user's password.

B. Second Experiment

After this defect was corrected, by removing the display of the valid ticket in the error report web page, another experiment was carried out. In some cases the algorithm used to generate tickets is known by attackers because the source code for the server may be available or the attackers may be able to guess the ticket algorithm [17]. In the revised Netml system, tickets were calculated as:

```
hexsha(username+Calendar.getInstance()
.get(Calendar.DAY_OF_YEAR))
```

In the present case this algorithm was deliberately revealed to the attacker.

In response to this attack, the ticket algorithm was changed to that shown in Listing 3. No attacks of the sort described in this section were then able to succeed. A proof that these type of attacks can't succeed is given in Section VI.

V. Stakeholder Security Analysis

Stakeholder analysis includes all those who have influence on decisions, events, or outcomes related to the system [18], [19]. There are several ways to classify stakeholders [20], [21], [22], some of which depend on their threat potential [23], [24]. Stakeholder analysis helps in determining: who would be an attacker, and how would a successful attack be carried out? Also, how should security rules be changed to make the system safer [25]?

However, the most important reason for a careful stakeholder analysis, from the point of view of this paper, is that if we are able to identify a sufficient set of rules that ensure the willing participation of each stakeholder, and if we can enforce these rules, then the system is self-evidently secure. This does not rule out the possibility that through experience stakeholders may, during the lifetime of a system, discover that there are rules which were not initially obvious and which needed to be added to their required rule set. If sufficient care is taken with the stakeholder analysis, such events should be rare.

A. Stakeholder Roles

The role refers to a synonym for the stakeholder type. Two main questions need to be considered: a) which stakeholder roles impact on password reset system safety; and b) which stakeholders are involved in those roles. The resulted four stakeholder roles and their stakeholders are listed in the first and second columns, respectively in table I.

Stakeholder role	Stakeholder
User	Teacher or Student
IT Admin	the System Administrator, IT Department personnel
Netml Admin	an administrator of the Netml System
Guest	Non impacted stakeholder
Attacker	External attacker, Internal attacker(Administration member)

TABLE I. STAKEHOLDER ROLES

1) User: Anyone can become a user. Users are permitted to access and change information they have stored including information about their identity, such as their email address and password. In this study, the role of users includes those who use the password reset system to reset their account passwords. Members of this role can cause a successful attack on their accounts. For example, sharing access information to an inbox that contains password notification email can result in a successful attack on the password reset system.

2) Admin: Admin users are able to access all the same services and information as ordinary users and, in addition, are able to see a variety of reports which are not available to ordinary users. There are different *levels* of admin access, and at a sufficiently high admin access level, data stored by other users is visible. This is a feature which is convenient for users when they need help with their use of the system, although it may be desirable under some circumstances to allow users to have privacy from admin users as well as from other users. Administration role in this system consists of Individuals that can play the role of attackers. For example, an administration's member who has an authorized access to the user's inbox that contains the password notification email.

3) Guest: Guest users are able to access, and run algorithms on, networks which are publicly accessible.

4) Attackers: Although attacker's have no inherent rights, it is useful to consider the objectives and motivations of attackers to better understand the strategies most effective in thwarting them. In particular, this role includes who can achieve a successful attack on the password reset system. As mentioned previously, the attacker may be an administrative member. The others, For example, who can eavesdrop on the password notification email if it is an encrypted email, or who can discover the contents of the password notification email when the inbox of the account user is not password protected.

B. Stakeholder rules

In this subsection, a subset of the rules for two of the key stakeholders have been identified and are tabulated in Table as shown in tables II and III. Rules are classified as mandatory (M) or optional (O). A complete set of rules is beyond the scope of this paper.

Rule number	Users can	Rule type
U1	create a new user identity and password associated with a specific email address	M
U2	reset the password associated with their email address without having to remember their existing password	M
U3	access the services associated with the user account by providing their password	M
U4	not change the password of a user other than themselves	M
U5	not disseminate the supplied email address that contains the Password notification email	O
U6	a user cannot obtain another user's password	M
U7	a user should not reveal their own Netml password to anyone else, or store it unsafely	O
U8	tickets sent to users who are resetting their password are valid for at least 30 minutes and at most 31 minutes	M

TABLE II. USER RULES

Rule number	Admins can	Rule type
A1	ensure that password problems are only resolved after adequate user authentication	M
A2	enable users to access the right resources at the right times and for the right reasons	M
A3	allow users who to authenticate with an an alternate factor	M
A4	not access to the inbox of the account user	M

TABLE III. ADMINISTRATION RULES

Attackers might be considered as stakeholders in their own right, and it is important to consider them in this way, however in the security design the objectives or rules which stakeholders adopt are not ones which we seek to enforce or support. In fact we seek precisely the opposite – to thwart these objectives. Furthermore, the rules of the other stakeholders will normally encompass these requirements. Hence, in the present context it is not necessary to include rules expressing the point of view of attackers.

VI. PROOFS OF CORRECTNESS

In this subsection we describe, in detail, how to enforce rules U1–U4. The procedure for implementing other rules (e.g. A1–A4) follows the same pattern.

Proving correctness of some aspects of a system is very straightforward and all that is necessary is to review the normal procedures adopted by software developers and remark that these are adequate. This applies to rules U1–U3. The reason these rules are easy to prove is that, by their nature, these rules can be demonstrated by a single example. Testing, which is the normal procedure adopted by software developers, is therefore adequate to *prove* that these rules are correctly enforced.

The distinction between rules which are easy to prove and those which are difficult to prove is *logical*, i.e. by their logical nature some rules can be proved by a single instance, while others require a series of deductive steps.

Rules which are difficult to prove are often difficult to implement correctly, also. While attempting to prove the security rules for the password reset subsystem, mistakes in its implementation were discovered which had to be corrected before the system could be proved to be correct. This occurred in other aspects of the system (which were analysed in a similar manner, but for which details have not been included in the paper), also. This supports the practical utility of the methodology investigated in this paper.

The complete set of rules for this system is a superset of those listed in Subsection V-B and so a complete security design and implementation will need to include specification of those rules and their proofs to be complete. In this section, we show that Rules U1–U3 hold and Rule U4 holds, i.e. it is not possible to break into the system.

```
<!-- this is the case of updating details of an
 ↪  existing user -->
<c:if test="${(! empty param.resetpwd) and
 ↪  (param.ticket==target_ticket)}">
   <sql:update dataSource="jdbc/MySQLDB">
      UPDATE USERS set name=?, user_name=?,
 ↪  organisation=?, email=?, password='',
 ↪  user_pass=?
            where user_name= ?
      <sql:param value="${param.fullName}"/>
      <sql:param value="${username}"/>
      <sql:param
 ↪  value="${param.organisation}"/>
      <sql:param value="${param.useremail}"/>
      <sql:param value="${hexshapass}"/>
      <sql:param value="${username}"/>
   </sql:update>
                ...
</c:if>
```

Listing 2: Code for changing passwords

If all users are treated identically, if Rules U1–U3 are verified for one user, they must be valid for all other users. If there are, in fact, distinct classes of user which are treated differently, it will, more generally, be sufficient to check these rules for users in each class. Even in this case, however, it will usually be straightforward to check each of these rules for each class of user.

```
  /**
   * Returns a random salt to be used to hash a
↪   password.
   *
   * @return a 32 bytes random salt
   */
  private static final Random RANDOM = new
↪   SecureRandom();
  public static byte[] nextSalt() {
    byte[] salt = new byte[32];
    RANDOM.nextBytes(salt);
    return salt;
  }
   /**
     *
     * @return a random Hex string of length 64
↪   bytes
     */
  public static String ticket() {
      StringBuilder sb = new StringBuilder();
      byte[] salt = nextSalt();
      for (int k=0; k<salt.length; k++) {

↪   sb.append(String.format("%02x",salt[k]));
      }
      return sb.toString();
  }
```

Listing 3: Algorithm for tickets

A. *Proof of U4*

Proposition 1: Rule U4 is enforced so long as the following two conditions are true:

(a) The password of user U will not be changed by the Netml server unless a ticket is supplied which is valid for changing the password of user U; and,

(b) a ticket which is valid for changing the password of user U can only be received or calculated by user U.

Proof: By assumption (a), a user's password can only be changed if a ticket valid for changing that user's password is provided. By assumption (b), such a ticket can only be received or calculated by the User for whom it is valid. Hence, the only user who can change a password is the user themselves. ■

To apply this proposition we need to verify that the assumptions hold in the Netml system. This is where the "real work" has to be done. Assumption (a) requires checking the code for the Netml system. The relevant code is shown in Listing 2. Observe that if the ticket provided by the user is not valid, the password will not be updated. In addition, the only other place where user passwords are updated cannot be accessed if the email address provided already exists in the database. Hence Assumption (a) holds. Note that proof of this assumption relies on an inherently complex observation that there is only one possible location where passwords can be changed as a consequence of a password reset request.

Assumption (b) also requires checking the source code of the system. There are two aspects of this functionality which need to be checked: (i) that tickets valid for changing the password of user U can only be received by user U; and (ii) tickets valid for user U can't be calculated or guessed (in a feasible time), by a user other than U. The algorithm for calculating tickets is shown in Listing 3. This algorithm uses the Java class `SecureRandom` as the source of random numbers used to generate tickets. This class provides a cryptographically strong random number generator,which means that it produces a *non-deterministic* sequence of random numbers.

This addresses (ii). Requirement (i) relies, in turn, on the assumption that user U manages their email in such a way that they, and only they, can receive email sent to their email address. The Netml software sends tickets valid for a certain email address only to that email address, but if a user allows someone else to know their email account password, another user use this to gain access to their Netml account, by requesting a password change, and then accessing the email which enables the password to the changed.

If U uses un-encrypted email, it is also possible for anyone with physical access to the network by means of which the email is transferred to gain access to the ticket and thereby to change the password of a user who is attempting to reset their password.

This method of re-setting an account password should not be used if the sensitivity of the information protected is greater than that of an email account.

B. *Proof of U8*

The code which checks the timeliness with which a ticket provided in an email is used to reset a password is shown in Listing 4. It can be seen from this code that if the ticket used is not the same as the one generated and stored in the system, when the user requested to reset their password, or if it is used on a different day, or at a time more than 30 minutes after it was generated, the script which is used to reset a users password will, instead, display a login screen. Thus, it is only possible to use a password reset ticket within 31 minutes of it being issued. Proof that tickets are valid for at least 30 minutes is not included here because this is best undertaken by testing.

```
   <!-- this is the case of updating password
↪   of an existing user -->
   <c:if test="${! empty param.resetpwd &&
↪   (param.ticket!=target_ticket
                or
↪   param.minutes>(ticket_minutes+30) or
↪   param.date!=ticket_date)}">
       <jsp:forward page="login.jsp" >
           <jsp:param name="errorMsg"
             value="An invalid or out-of-date
↪   ticket was used to reset a password. Please
↪   try again." />
       </jsp:forward>
   </c:if>
```

Listing 4: Algorithm for checking ticket timeliness

VII. Conclusion

In Section II, three complementary approaches for achieving rigorous security were reviewed. The first of these is a pragmatic list of good practices which help to minimise the effort required to create and maintain good security. If these practices are not followed, the other two approaches, which are more specific in addressing security requirements, will require too much effort to be put into practice. The second approach is to attempt a series of attacks by methods which are known to be currently active, and to use the known (and usually published) techniques to address them.These methods are used by attackers as they are easy to get the worked attacked vector and they know they will work as uptake of fixes and patches is slow.

The third approach is a more systematic and rigorous complement to the second. Instead of enumerating and employing a series of exploits, the threats are systematically classified in the form of a list of rules stated as the requirements of each of the stakeholders. This allows us to logically address the important question of *completeness*. Whereas a list of currently active exploits is inherently evolving and therefore incomplete, it is reasonable to ask each stakeholder (or their representative): "is this list of rules sufficient, if guaranteed, for you to agree to actively participate in this service?" If all stakeholders are satisfied in this way, the web service is secure. Despite an agreement of this sort being established, naturally, if a new approach which undermines security in a way which one or more stakeholders were not able to anticipate is discovered, the rules which stakeholders require will need to change. At least, when this happens, we have an explanation: our fundamental understanding of the nature of the service has progressed.

In addition, the third approach defines how to rigorously review the steps which are taken to address new exploits: these steps should be formulated sufficiently clearly, and in such a way, that the stakeholder rules are provable. This makes a critical connection between the code implementing a web service and the security rules it is expected to conform to. It is not necessary to adopt a special purpose programming language, or methodology, in order to achieve this level of rigour (although adopting special methods may make this task easier).

The key new ideas in this paper are that: (a) there is a simple and logical security design philosophy, which is to identify the rules required by all stakeholders, and ensure they are guaranteed; (b) we can ensure some of these are satisfied by proving they must hold from some of the rules which are guaranteed. Note: proving that *all* rules hold, even for a small system like the one considered in this paper, is not feasible in a paper of this length. For systems which need a security guarantee, it may be feasible to prove that all stakeholder rules hold. However, many systems rely on correct behaviour by the stakeholders, which will limit what level of security can be achieved.

REFERENCES

[1] A. Jøsang and S. Pope, "User centric identity management," in *AusCERT Asia Pacific Information Technology Security Conference*. Citeseer, 2005, p. 77.

[2] R. G. Addie, Y. Peng, and M. Zukerman, "Netml: networking networks," in *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*. IEEE, 2011, pp. 1055–1060.

[3] D. V. Bailey, M. Dürmuth, and C. Paar, "Statistics on password re-use and adaptive strength for financial accounts," in *International Conference on Security and Cryptography for Networks*. Springer, 2014, pp. 218–235.

[4] K. Nimmy and M. Sethumadhavan, "Novel mutual authentication protocol for cloud computing using secret sharing and steganography," in *Applications of Digital Information and Web Technologies (ICADIWT), 2014 Fifth International Conference on the*. IEEE, 2014, pp. 101–106.

[5] C. Routh, B. DeCrescenzo, and S. Roy, "Attacks and vulnerability analysis of e-mail as a password reset point," in *Mobile and Secure Services (MobiSecServ), 2018 Fourth International Conference on*. IEEE, 2018, pp. 1–5.

[6] R. G. Addie, S. Moffatt, S. Dekeyser, and A. Colman, "Five examples of web-services for illustrating requirements for security architecture," in *Proceedings of 2nd International Conference on Data and Knowledge Engineering*, 2011.

[7] A. Bhardwaj and S. Goundar, "Reducing the threat surface to minimise the impact of cyber-attacks," *Network Security*, vol. 2018, no. 4, pp. 15–19, 2018.

[8] H. Lai, J. S.-C. Hsu, and M.-X. Wu, "The impact s of requested permission on mobile app adoption: The insights based on an experiment in taiwan," in *Proceedings of the 51st Hawaii International Conference on System Sciences*, 2018.

[9] E. Toch, C. Bettini, E. Shmueli, L. Radaelli, A. Lanzi, D. Riboni, and B. Lepri, "The privacy implications of cyber security systems: A technological survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 2, p. 36, 2018.

[10] K. Ghirardello, C. Maple, D. Ng, and P. Kearney, "Cyber security of smart homes: Development of a reference architecture for attack surface analysis," in *Living in the Internet of Things: Cybersecurity of the IoT-2018*. IET, 2018, pp. 1–10.

[11] D. Thomsen and E. Bertino, "Network policy enforcement using transactions: The neutron approach," in *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies*. ACM, 2018, pp. 129–136.

[12] M. Tabassum, S. Watson, B. Chu, and H. R. Lipford, "Evaluating two methods for integrating secure programming education," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, 2018, pp. 390–395.

[13] R. S. Ross, M. McEvilley, and J. C. Oren, "Systems security engineering: Considerations for a multidisciplinary approach in the engineering of trustworthy secure systems [including updates as of 1-03-2018]," Tech. Rep., 2018.

[14] M. E. Whitman and H. J. Mattord, *Principles of information security*. Cengage Learning, 2011.

[15] L. O. Mailloux, P. M. Beach, and M. T. Span, "Examination of security design principles from nist sp 800-160," in *Systems Conference (SysCon), 2018 Annual IEEE International*. IEEE, 2018, pp. 1–8.

[16] M. A. Bishop, *Introduction to computer security*. Addison-Wesley Boston, 2005, vol. 50.

[17] K. C. Wang and M. K. Reiter, "How to end password reuse on the web," *arXiv preprint arXiv:1805.00566*, 2018.

[18] P. M. Institute, "A guide to the project management body of knowledge (pmbok® guide)-(simplified chinese)." Project Management Institute, 2018.

[19] K. H. Rose, "A guide to the project management body of knowledge (pmbok® guide)fifth edition," *Project management journal*, vol. 44, no. 3, pp. e1–e1, 2013.

[20] B. Maguire, J. Potts, and S. Fletcher, "The role of stakeholders in the marine planning processstakeholder analysis within the solent, united kingdom," *Marine Policy*, vol. 36, no. 1, pp. 246–257, 2012.

[21] S. Maynard, A. Ruighaver, and A. Ahmad, "Stakeholders in security policy development," in *9th Australian Information Security Management Conference*. Citeseer, 2011, p. 182.

[22] L. Bourne, *Stakeholder relationship management: a maturity model for organisational implementation*. Routledge, 2016.

[23] M. Almorsy, J. Grundy, and I. Müller, "An analysis of the cloud computing security problem," *arXiv preprint arXiv:1609.01107*, 2016.

[24] G. T. Savage, T. W. Nix, C. J. Whitehead, and J. D. Blair, "Strategies for assessing and managing organizational stakeholders," *Academy of management perspectives*, vol. 5, no. 2, pp. 61–75, 1991.

[25] S. Diver, "Information security policy–a development guide for large and small companies," *Sans Institute*, pp. 1–37, 2007.