Reducing the Attack Surface for Sensitive Data

George O. M. Yee
Computer Research Lab, Aptusinnova Inc., Ottawa, Canada
Dept. of Systems and Computer Engineering, Carleton University, Ottawa, Canada
e-mail: george@aptusinnova.com, gmyee@sce.carleton.ca

Abstract—**Breaches of sensitive data have been occurring at an alarming rate to the embarrassment and expense of companies. It would appear that in each breach, the attack surface for the data has been sufficiently large to attract attackers. Reducing this attack surface is a way to lessen the likelihood of breaches. This paper presents methods for reducing the attack surface of the data held in the online computer systems of organizations. The methods are applied to a software system's architecture early in the design process, as an approach for designing-in security. This work first defines the attack surface and then uses this definition to obtain methods for reducing the attack surface. The definition also leads to a formula for calculating the size of the attack surface. The formula incorporates the fact that vulnerabilities differ within the architecture. This paper further gives recommendations on how to apply the methods effectively and illustrates this application using two examples. Reducing the attack surface may not prevent breaches, but it will make them less likely to occur.**

Keywords-sensitive data; private data; breaches; attack surface identification; attack surface reduction.

I. INTRODUCTION

This work extends Yee [1] by a) extending the application domain to all sensitive data, not just private data, b) improving the calculation of the size of the attack surface to account for the fact that some parts of the software architecture are more likely to be attacked than others, c) improving the explanations throughout the paper as well as updating the examples of breaches in Section I, d) adding a second application example, and e) increasing the number of references.

Breaches of sensitive data held by companies and other types of organizations have been occurring at an alarming rate. In recent years, each year has been accompanied by its assortment of data breaches. Consider the following sampling of breaches in 2020 [2], the year of this work:

- July 20, 2020: Ancestry.com, an unsecured server exposed sensitive data belonging to 60,000 clients of this family history search company. The lost data include email addresses, geolocation information, IP addresses, system user IDs, support messages and technical support details.
- June 22, 2020: BlueLeaks, over 296 GB of data was leaked from US law enforcement agencies and fusion centers, and posted on an online searchable portal called BlueLeaks. The leaked information contained more than one million files, including scanned documents, videos, emails, audio files, some of which had sensitive and personal data, such as names, bank account numbers, and phone numbers.
- April 20, 2020: Beaumont Health, the personal and medical data of more than 112,000 employees and patients of Beaumont Health was accessed by a hacker after compromising employee email accounts using a phishing attack. The lost data included names, birth dates, social security numbers, driver's license numbers, medical condition information, and bank account data.
- February 20, 2020: MGM Resorts, the personal information of over 10.6 million hotel guests who had stayed at MGM Resorts was found posted on a hacking forum. The information included names, home addresses, phone numbers, emails, and dates of birth. On July 15, 2020, researchers found 142 million personal records of formers guests of MGM Resorts hotels for sale on the Dark Web, suggesting that the original breach was larger than previously announced.

Apparently, the attack surface for the data that was breached, or the number of ways that the data could be accessed and stolen, was sufficiently large and attractive to the attackers.

Given the rate of recent data breaches, it is clear that more needs to be done to reduce the probability of a data breach occurring. The objective of this work is to derive methods for reducing the attack surface of sensitive data held in online (i.e., connected to the Internet) computer systems of organizations. The methods are obtained from consideration of the definition of the attack surface, which in turn is based on how an attack happens. This definition also leads to a straightforward formula for calculating the size of the attack surface, which can be used to verify that use of the methods does indeed reduce the attack surface. The methods focus on reducing the attack surface by altering the system architecture, rather than the deployment of add-on security appliances, such as firewalls and intrusion detection systems. The methods are meant to be applied at the early stages of design within a software development cycle, as part of the Design for Security toolset.

This paper is organized as follows. Section II explains sensitive data, attacks, and attack surface. Section III derives methods for reducing the attack surface. Section IV illustrates the methods using two application examples. Section V describes related work. Section VI discusses some potential issues. Finally, Section VII presents conclusions and future work.

II. SENSITIVE DATA, ATTACKS, AND ATTACK SURFACE

This section explains sensitive data, attacks, attack surface, and how to calculate the size of the attack surface.

A. Sensitive Data, Attacks, and Attack Surface

Sensitive data is data that needs protection and must not fall into the wrong hands. It includes private or personal data. Sensitive data also includes non-private information that may compromise the competitiveness of the organization if divulged, such as trade secrets or proprietary algorithms and formulas. For government organizations, non-personal sensitive data may include information that is vital for the security of the country for which the government organization is responsible.

Private data, also known as personal data, is data about an individual, can identify that individual, and is owned by that individual [3]. For example, an individual's driver license number, passport number, or credit card number can each be used to identify the individual and are therefore considered as private data. The individual's privacy then refers to his/her ability to control the collection (what personal data and collected by which party), purpose of collection, retention, and disclosure of that data, as stated in the individual's privacy preferences [3].

DEFINITION 1: *Sensitive data* (SD) is information that must be protected from unauthorized access in order to safeguard the privacy of an individual, the well-being or expected operation of an organization, or the well-being or expected functioning of an entity for which the organization has responsibility.

DEFINITION 2: An *attack* is any action carried out against an organization's computer system that, if successful, results in the system being compromised.

This work focuses on attacks that compromise the SD held in the online systems of organizations. The attacker who launches an attack may be internal (inside attacker) or external (outside attacker) to the organization. This work applies to both types of attackers. An internal attacker usually has easier access to the targets of his/her attack and he/she may hide his/her attacks in the guise of normal duty.

Salter et al. [4] give an interesting insight into what enables a successful attack: "Any successful attack has three steps: One, diagnose the system to identify some attack. Two, gain the necessary access. And three, execute the attack. To protect a system, only one of these three steps needs to be blocked." Thus, an attack surface must contain a target that the attacker deems worthy of attack (suit his/her purpose for the attack) and that target must be accessible to the attacker. For this work, the target that is potentially worthy of attack is the SD that is accessible to attackers. In a computer system, this SD is either moving (travelling from one location to another), at rest (stored), or being used (by some process). This leads to the following definition of attack surface:

DEFINITION 3: The *attack surface* for sensitive data, also called the *data attack surface*, contained in an online computer system is the set of all locations in the system that contain attacker accessible SD in the clear, where the SD is moving, at rest, or being processed.

In Definition 3, "attacker accessible SD" means that the attacker is able to exfiltrate the SD using some agent of attack, such as malware against stored SD and SD being processed, or a man-in-the-middle attack against a link containing moving SD. Also, we assume that attackers would attack SD that is in the clear rather than SD that is encrypted. In the rest of this paper, by "attack surface" we mean the data attack surface, unless otherwise indicated. Figure 1 shows an example data attack surface.

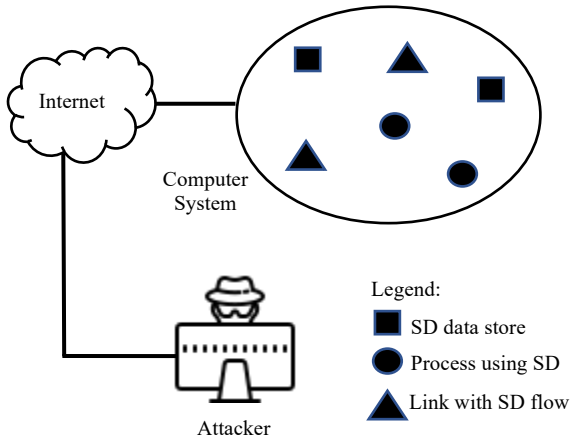

Figure 1. Example data attack surface consisting of the set of all 6 attacker accessible locations in the system that contain SD in the clear.

An alternative definition of attack surface for SD contained in a computer system is the set of ways the attacker has to exfiltrate the SD. However, given the complexity of computer systems and the fact that the tools available to the attacker to use in his/her attacks are unknown to us, it is next to impossible to determine this set. On the other hand, locations that contain attacker accessible SD are easier to identify. Since an exfiltration must be from a location that contains SD, the set of such exfiltrations depends on the set of such locations. The larger the set of locations, the larger the set of exfiltrations. The smaller the set of locations, the smaller the set of exfiltrations. Therefore, Definition 3 in a sense includes this alternative definition, but in addition, is more easily applied.

As mentioned above, in the first step of a successful attack, the attacker diagnoses the system to identify the attack [4]. A smaller attack surface will make this step more difficult for the attacker. Therefore, a smaller attack surface corresponds to higher security, which is why we wish to reduce the attack surface. Definition 3 also gives rise to this conclusion: a smaller attack surface means a smaller number

of locations that contain SD, which in turn means fewer opportunities for exfiltration of the SD, or in other words, higher security.

Definition 3 is consistent with the intuitive understanding of an attack surface (the usual meaning), which is "the set of ways in which an adversary can enter the system and potentially cause damage" [5]. Each "way" corresponds to a location in Definition 3 that in turn corresponds to methods for exfiltrating SD from the location.

B. Calculating the Size of the Attack Surface

It would be useful to have a numerical value for the size of the attack surface, since then we could a) compare attack surfaces at different stages of development to see if the system's security is getting better or worse, b) compare attack surfaces of different systems when choosing a system for purchase, and c) easily see if actions taken to reduce the attack surface have indeed reduced it.

As mentioned above, sensitive data held in a computer system can be in the following three states: moving, at rest, or being processed. These states correspond respectively, within a computer system, to SD that is moving along a link, SD that is stored in a data store, and SD that is being processed. Thus, the locations in Definition 3 refer to links, datastores, and processes that contain attacker accessible SD. Definition 3 then leads naturally to the following formula for calculating the size of the attack surface for sensitive data.

Let N be an *estimate* of the size of the attack surface for SD. Let m, n, and k be the number of links, data stores, and processes, respectively, that contain attacker accessible SD in the clear. Then

$$N = m + n + k \tag{1}$$

Equation (1) says that N is found by adding up the number of attacker accessible locations in the system that contain SD, namely: the number m of links, the number n of data stores, and the number k of processes, all of which contain attacker accessible SD. This equation follows directly from Definition 3, by simply replacing "attack surface" with "size of the attack surface" and "set" with "size of the set" in that definition. We call N an estimate because it is impossible to know all the vulnerabilities in a system, and hence it is impossible to have an accurate value of the size of the attack surface. As well, an estimate suffices for the three benefits mentioned at the beginning of this Section. In the following, unless stated otherwise, we use "size" to mean "estimated size".

Equation (1) is the minimum size of the attack surface because it counts the number of links, data stores, and processes that form the attack surface. It can only be smaller if one or more of these components are not part of the attack surface, which would contradict Definition 3. Further, (1) makes no distinction between links, data stores, or processes as locations that contain attacker accessible SD. Yet, given a choice between attacking a link, a data store, or a process in the same computer system, where the difficulty level is the same for all locations, the attacker will probably choose to attack a data store of SD as it is more likely to give the attacker what he/she wants. This preference for data stores is seen in the large number of data store breaches that have occurred. Thus, a data store should contribute more to the size of the attack surface than a link or a process. A datastore makes the system more vulnerable to attack and this should be reflected in the size of the attack surface, which is also a measure of the system's vulnerability to attack. We can reflect this in (1) by making the contribution of data stores to the size of the attack surface as $M_d \bullet n$ where M_d is a positive integer multiplier. In fact, we can have positive integer multipliers M_l for links and M_p for processes, and corresponding contributions $M_l \bullet m$ and $M_p \bullet k$ to the size of the attack surface. Getting back to reflecting the greater contribution of data stores to the size, we can set $M_l = M_p = 1$ and $M_d = 2$, which states that each data store contributes twice as much to the size of the attack surface as either a link or a process. Thus, the equation for the size of the attack surface with multipliers is

$$N = M_l m + M_d n + M_p k \tag{2}$$

and with the above values of the multipliers to reflect the increased contribution to size of data stores, (2) becomes

$$N = m + 2n + k \tag{3}$$

We will use (3) rather than (1), since it suits our goal and is closer to reality. Applying (1) to Figure 1 gives an attack surface of size $N = m + n + k = 2 + 2 + 2 = 6$. Applying (3) gives 8, reflecting the greater vulnerability of the system due to its data stores.

We have used $M_l = M_p = 1$ and $M_d = 2$ with the rationale that attackers are more attracted to data stores than to links and processes, plus the observation that there has been many breaches of data stores. Note that $M_l = M_p = 2$ and $M_d = 3$ would have worked as well. The effect of these values in reducing the attack surface is that eliminating a data store from the attack surface gives a greater reduction than removing a link or a process. This means that if a data store can be removed, it should be. However, the values can be anything so long as they relatively reflect what the contributions to size should be according to past history or other sources of information, such as the system architecture. Since we don't have access to other values, we have used the ones above since they reflect our conviction regarding datastores. Perhaps in the future, these values can be refined based on studies. Note that whatever values are used for M_l, M_d, and M_p, our attack surface reduction techniques will always show N decreasing after applying each technique. Note also that (2) treats all links equally, all data stores equally, and all processes equally in terms of their contributions to the size of the attack surface. To allow each location to have its own specific contribution would be to attempt a level of accuracy that is unwarranted in light of our lack of knowledge of attacker behaviour as well as all the vulnerabilities in the system.

III. Reducing the Attack Surface

This section derives methods for reducing the attack surface based on (3).

A. Methods for Reducing the Attack Surface

Equation (3) implies that the attack surface will decrease if and only if any or all of the quantities m, n, or k decrease. Therefore, the attack surface may be reduced by the following methods, where each method decreases m, n, or k.

a) Make a SD location useless to the attacker.
b) Combine two or more SD locations into a single SD location.
c) Deny the attacker access to a SD location.
d) Remove a SD location from the system.

The following explains these methods in greater detail and describes how they may be carried out.

a) *Make* a *SD Location Useless to the Attacker*

As mentioned above, in the first step of a successful attack, the attacker diagnoses the system to identify an attack, or in our case, the SD target for the attack. In this diagnosis, it is reasonable to assume that the attacker will ignore any target that he/she finds useless for his/her purposes. Such targets may be removed from the attack surface. Some ways to make a SD target useless to an attacker are:

- Obfuscate (e.g., encrypt) the SD at the location. The attacker will not want to exfiltrate SD that cannot be read. The computer system will need to be able to de-obfuscate the data securely for its own purposes.
- Anonymize the SD at the location. Again, the attacker will not want SD that cannot be linked to individuals, since it is this linking that adds value to the data, e.g., for advertising purposes. The computer system will need to be able to de-anonymize the data securely for its own purposes.

Note that this method does not affect any other location, whereas the following methods do.

To illustrate, obfuscating one data store and one process in Figure 1 results in Figure 2, where the obfuscated data store and the obfuscated process have been removed from the attack surface. It can be seen that the attack surface in Figure 2 is reduced (size 5) relative to the attack surface of Figure 1 (size 8).

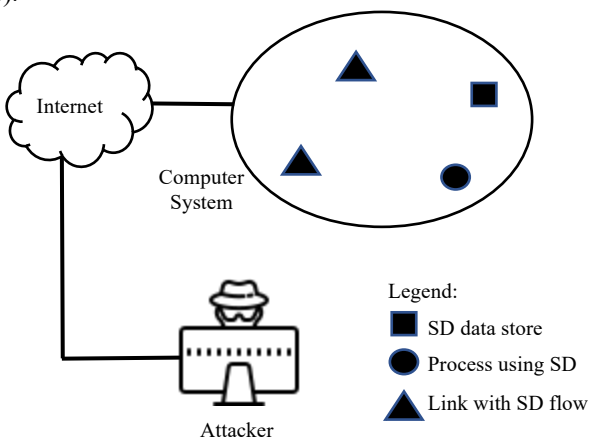

Figure 2. Resulting reduced data attack surface of size 5 after obfuscating locations in Figure 1.

b) *Combine Two or More SD Locations into a Single SD Location*

This method will decrease the number of SD locations and reduce the size of the attack surface per (3). Links carrying SD to/from the locations that were combined may need to be moved to the combined location, or they may be merged if they extend from the locations that were combined to a common endpoint. Merged SD carrying links correspond to a reduction of SD carrying links from the attacks surface. In addition, changes to the software logic may be needed for data stores or processes that were combined to accomplish reading or storing the data in the combined location (for combined data stores), or new processing of data in the combined location (for combined processes). As we have seen above, this method can remove SD carrying links from the attack surface when such links can be merged.

To illustrate, suppose in Figure 1 that we combine two data stores and two processes into one data store and one process. Suppose also that combining the data stores allowed the merging of two links into one, but combining the processes did not change the number of links. Figure 3 shows the result, obtained by removing 1 data store, 1 process, and 1 link from Figure 1 due to combining locations. We see that the attack surface has been reduced from size 8 (Figure 1) to size 4 (Figure 3).

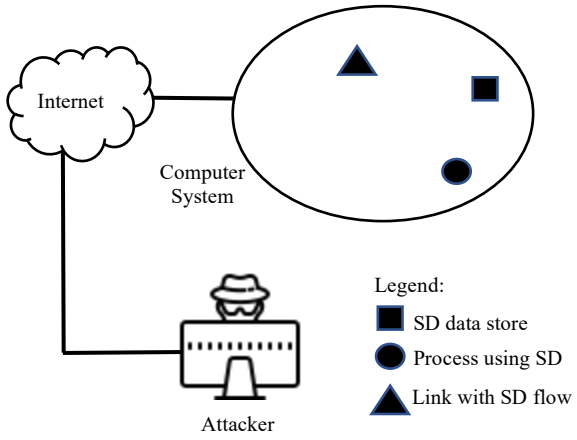

Figure 3. Resulting reduced data attack surface of size 4 after combining (method b) or removing (method d) locations in Figure 1.

c) *Deny the Attacker Access to a SD Location*

It may be possible to have some SD locations offline, thus denying the attacker access to these locations. For example, this may be possible for certain self-contained processing, such as analytics, that can be done using SD that is offline. In this case, all data stores, processes, and data links involved solely in the processing to be moved offline may be removed from the attack surface of the system and re-constituted into the offline system. It may be necessary to update the offline SD data stores periodically using data from the system that is online. This update will need to be done in a secure fashion, perhaps by transferring the data manually using disks, after making sure that no malware can infect the offline system via this transfer. Although the destination locations are offline, it

may still be possible for transferred malware to exfiltrate the offline data, e.g., hiding the data in the disks that are used for transfer and then transmitting the data once the disks are on the online part of the system. The locations moved offline are still vulnerable to inside attack, so they would have to be secured against such attack. Defending against inside attacks has been extensively researched, e.g., [6], [7].

Another way to deny the attacker access to a SD location applies to SD links. Here, SD links are implemented on a hardware platform along with other components of the system. An example of such a link is the communication channel between the CPU and the GPU implemented on a computer motherboard. An attacker would find it very difficult to access such links for an attack such as man-in-the-middle. Given other targets that are easier to access, the attacker will not attack such links and they can be removed from the attack surface.

As an illustration of moving some SD locations offline, Figure 4 shows a data store, a process, and a link taken out of Figure 1 and assembled into an offline system. The attack surface of the computer system has been reduced from size 8 (Figure 1) to size 4 (subtracting the contributions to the attack surface of the moved locations). However, the offline system would need to be secured against inside attack.

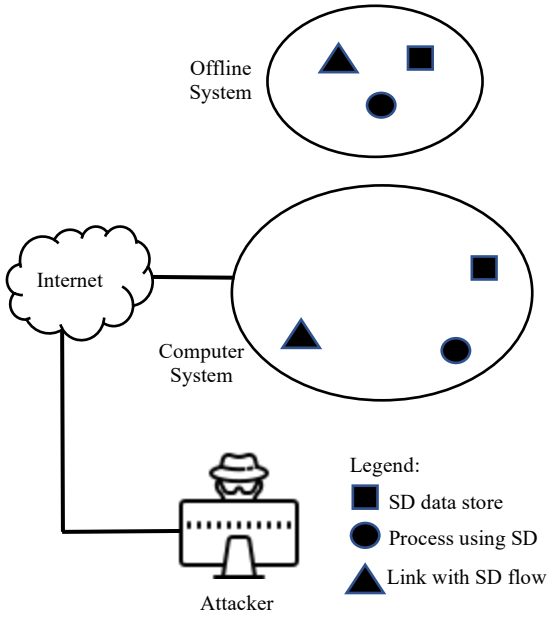

Figure 4. Resulting reduced data attack surface of size 4 for the computer system after moving some locations in Figure 1 offline.

d) Remove a SD Location from the System

Another way to reduce the attack surface is to remove a SD location from the system by deciding that the SD in the location is no longer required. For example, a company that stores the credit card information of its customers for their convenience may decide to stop storing this information, and instead, ask the customer for their credit card information every time the customer goes through checkout. This is in general a good decision, to avoid storing SD that may get compromised, at the cost of a little inconvenience. In this case, the associated credit card SD datastore would no longer be needed, and would be removed from the attack surface. Another example is the removal of a process that periodically sends customers the status of their order. The process uses SD consisting of the customer's name and email address to send the status. Suppose that this process is no longer necessary because the customer can now use a new Web interface to check order status. Removal of this process from the system removes it from the attack surface. Interestingly, removal of a SD location can also result in removing other SD locations that are connected to the location that is removed. For example, the removal of a SD data store or a process that uses SD can result in also removing connected SD locations, such as the links that carry SD, or a SD data store that the removed process was exclusively using. Thus, removing a SD location not only removes that location from the attack surface but can also lead to removing other SD locations further reducing the attack surface.

To illustrate, suppose it is decided that one of the processes in Figure 1 is no longer needed. Removing this process means that a data store and a link that were used only by this process are also no longer needed. Thus, the attack surface of the computer system in Figure 1 is reduced from size 8 to size 4, and the reduced system is shown in Figure 3.

B. Applying the Methods

Since the above methods operate on attacker accessible SD locations, it is recommended that they be applied in the second phase of two phases, where the attacker accessible SD locations are identified in the first phase. These phases are carried out on an architectural representation of the online system, such as a Data Flow Diagram (DFD) [8] (see the application examples in Section IV). The phases are as follows.

- Phase 1: Identify SD locations by tracing the flow of sensitive data in the online computer system, looking for where SD enters the system, where SD flows (links), where it is stored (data stores), and where it is used (processes). Identifying the SD locations by tracing the flow of SD in the system implies that there are paths to the SD that an attacker can use to exfiltrate the SD. We therefore conclude that all SD locations found in this manner in an online system are attacker accessible SD locations. Given the ingenuity of attackers (the exfiltration could even be aided by an insider of the organization that owns the computer system, through social engineering), this conclusion is valid.
- Phase 2: Apply the above methods to the attacker accessible SD locations found in Phase 1, where possible, while considering the potential negative effects on the following aspects of the system:
  - Performance
  - Reliability and dependability
  - Ease of maintenance
  - Implementation cost

For example, encryption or anonymization incurs extra overhead, combining data stores may introduce a performance bottleneck since the newly combined data store will now need to additionally support data accesses that were originally shared among the data stores that were combined. Combining SD locations in general may reduce modularity and lead to extra effort needed to maintain the system.

Three guiding rules for applying the methods are:

1. Look for opportunities to apply a method to a data store since according to (3), removing a data store from the attack surface gives a greater reduction than removing either a link or a process.
2. Look for opportunities to apply the methods where the potential negative effects mentioned above are minimal.
3. It may be more efficient to consider method a) last, since the other methods can add/delete links that are candidates for method a).

Carrying out the above phases clearly requires knowledge of the computer system in terms of identifying the SD locations. Some basic knowledge of security would also be advantageous. These skills should be found within the software development team responsible for developing the system, perhaps with a little security training if needed.

IV. Application Examples

This section illustrates how to apply the methods for reducing the attack surface for sensitive data using two example online computer systems: one for selling merchandise and the other for airline reservation.

A. Online Seller of Merchandise

Suppose that the system of the online seller of merchandise (e.g., Amazon.com) is at the beginning stages of development and that the development team has produced a DFD showing how sensitive data will flow, be stored, and used in the system. This DFD is shown in Figure 5.

The system in Figure 5 allows the customer to enter his/her "name", "address", "email", "item selected" for purchase, and "credit card info" for payment. These comprise the SD for this example. Five processes cooperate to provide the functionality for the system. One datastore stores the customers' sensitive data; another datastore contains inventory data, i.e., what items are in stock. The system is an online system since it is for an online seller. The SD locations will be found by tracing the flow of SD in the system (described below). All SD locations in the system are attacker accessible SD locations, as noted above in the description of Phase 1 (Section III-B).

Applying Phase 1 in Section III-B, we trace the flow of sensitive data from the point where the data enters the system at process 1. From there, the SD passes through process 1 and is stored in the customer datastore. After this datastore, the SD is split up with the "credit card info" going to process 4 to be used, and the "name", "address", "email", and "item selected" going to process 2, where the "item selected" datum is used, and "name" and "address" are passed to process 5 to print the shipping label, whereas "email" is passed to process 3 to send the customer the shipping status. Thus, we can identify the SD locations as links, datastores, and processes through which the SD passes, is stored, and used. Note that inventory data is not considered SD in this example. These attacker accessible SD locations are shown in Table I.

Table I shows that there are 6 attacker accessible SD link locations, 1 attacker accessible SD datastore, and 5 attacker accessible SD processes. For Figure 5, prior to the application of the above methods, (3) gives the size N of the attack surface for sensitive data as $N = m + 2n + k = 6 + 2 + 5 = 13$.

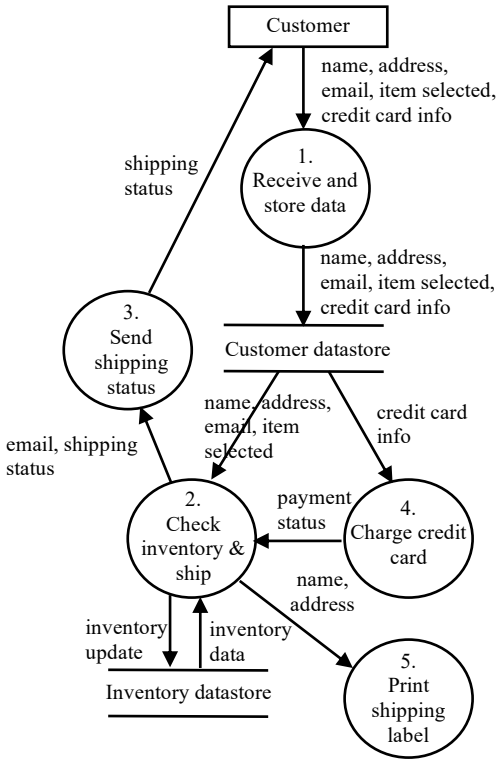

Figure 5. DFD for online seller system, showing how data flows, are stored, and used.

TABLE I. Attacker accessible SD locations in Figure 5

	Links	Datastores	Processes
1	link into process 1	customer datastore	process 1
2	link out of process 1		process 2
3	link from customer datastore to process 2		process 3
4	link from customer datastore to process 4		process 4
5	link into process 5		process 5
6	link into process 3		

Applying Phase 2 in Section III-B, we first use the above methods on the attacker accessible locations in Table I, as follows:

- Using method d), remove the customer datastore from the system; it was decided that storing customer SD was not needed (customer purchase history can be stored securely on the customer's device by the seller's website and later retrieved by that same website). Note that removing this data store also caused the removal of a SD link (the link between process 1 and this data store). This change was seen as acceptable, and not significantly impacting performance or system maintainability.
- Using method b), combine process 3 with process 2; this was seen to have negligible impact on performance and an acceptable reduction in modularity.
- Using method b), combine process 5 with process 2; this was also seen to have negligible impact on performance and an acceptable reduction in modularity.

These changes result in the DFD shown in Figure 6. Table II gives the attacker accessible SD locations corresponding to Figure 6.

Table II shows that there are 3 attacker accessible SD link locations and 3 attacker accessible SD processes. For Figure 6, (3) gives the size N of the attack surface for sensitive data as $N = m + 2n + k = 3 + 0 + 3 = 6$. Thus, the application of methods d) and b) have reduced the attack surface from 13 to 6.

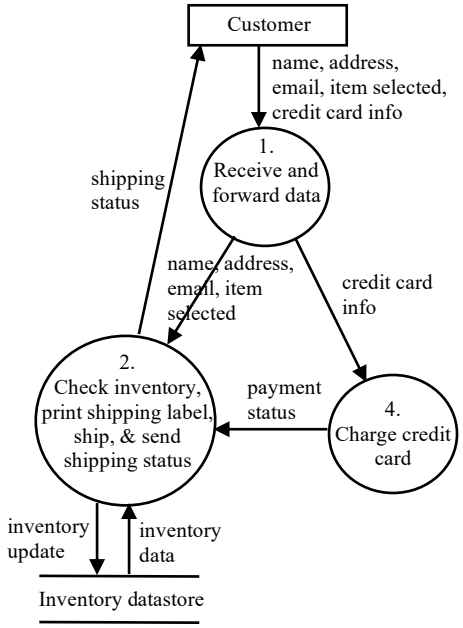

Figure 6. DFD for online seller system after combining processes and removing the customer data store.

We can further reduce the attack surface as follows:

- Using method a), obfuscate (encrypt) the links in Table II; the impact on performance due to the extra over head is deemed acceptable.
- Using method a), obfuscate (encrypt) the SD in the processes shown in Table II; here, the impact on performance and the cost involved for extra code to handle encryption/decryption were considered unacceptable, and this reduction method was not applied. This reduction may have been feasible if these processes could use encrypted SD, but these processes require SD in the clear.

TABLE II. ATTACKER ACCESSIBLE SD LOCATIONS IN FIGURE 6

	Links	Datastores	Processes
1	link into process 1		process 1
2	link from process 1 to process 2		process 2
3	link from process 1 to process 4		process 4

Table III shows the remaining attacker accessible SD locations after applying method a) to the links in Table II. The new attack surface is of size $N = m + 2n + k = 0 + 0 + 3 = 3$. The application of the methods in Section III-A has improved the security of sensitive data in the system by reducing the size of the attack surface from 13 to 3.

TABLE III. REMAINING ATTACKER ACCESSIBLE SD LOCATIONS IN FIGURE 6 AFTER OBFUSCATING THE LINKS IN TABLE II

	Links	Datastores	Processes
1			process 1
2			process 2
3			process 4

Comparing Figure 6 to Figure 5, reducing the attack surface required the following architectural changes to the system: i) eliminating the customer database, ii) reducing the number of processes from 5 to 3 by eliminating processes 3 and 5, and iii) changing the functionality of processes 1 and 2. As noted above, the implications of these changes were accepted by the development team.

The size of the attack surface obtained by applying the above methods may depend on which methods were applied and the order in which they were applied. In particular, it may depend on the available opportunities for applying methods d) and b). For example, by using only method a) (obfuscation) on the link and datastore locations in Table I, assuming that it is not advisable to apply method a) to the processes due to unacceptable impacts on performance and costs, we obtain an attack surface of size 5 (for the remaining 5 processes since the obfuscated links and datastore would have been removed from the attack surface), which is larger than the attack surface of size 3 obtained above by opportunistically applying methods d) and b) before method a). This is the rationale behind guiding rule 3 in the description of Phase 2 above, that it may be more efficient to apply method a) last.

B. Online Airline Reservation System

ACCURES is an online airline reservation system that has been awarded to a software company for development. The system is to consist of 3 modules: MAIN, MOD-CAN, and MOD-EU. MAIN and MOD-CAN operate in Canada,

whereas MOD-EU operates in Germany. MAIN communicates with MOD-CAN and MOD-EU to receive flight request details, and in turn, sends them flights assigned details corresponding to the requests. MOD-CAN and MOD-EU process customer transactions, where each transaction consists of receiving customer identification details, flight request details, and payment information (e.g., credit card details), storing the customer information in a data store, processing the payment, and sending the customer his/her travel itinerary, which marks the end of the transaction. The development team creates the DFD for ACCURES shown in Figure 7. Note that in this DFD, all data are SD, and all locations are attacker accessible SD locations.

Applying Phase 1 in Section III-B, we trace the flow of sensitive data from the point where sensitive customer data enters the system at processes 4 and 7. Sensitive data also enters the system in the form of "flight availability updates" but these updates are stored in the Flights data store and go no farther. Referring to Figure 7, we see that the customer SD flows though all locations of MOD-CAN and MOD-EU. We also see that the sensitive data items "flight details requested" and "flight details assigned" flow through all locations of MAIN. Thus, we can conclude that all locations in ACCURES are attacker accessible SD locations, as shown in Table IV.

TABLE IV. ATTACKER ACCESSIBLE SD LOCATIONS IN FIGURE 7

	Links	Datastores	Processes
1	link into Flights data store	Flights	process 1
2	link between Flights data store and process 1	Customer data 1	process 2
3	link between process 1 and process 2	Customer data 2	process 3
4	link between process 2 and process 3		process 4
5	link between process 2 and process 6		process 5
6	link between Customer and process 4		process 6
7	link between process 4 and Customer data 1		process 7
8	link between process 3 and process 4		process 8
9	link between process 3 and Customer data 1		
10	link between Customer data 1 and process 5		
11	link between Customer and process 7		
12	link between process 7 and Customer data 2		
13	link between process 6 and process 7		
14	link between process 6 and Customer data 2		
15	link between Customer data 2 and process 8		

Table IV shows that there are 15 attacker accessible SD link locations, 3 attacker accessible SD datastores, and 8 attacker accessible SD processes. For Figure 7, prior to the application of the above methods, (3) gives the size N of the attack surface for sensitive data as $N = m + 2n + k = 15 + 6 + 8 = 29$.

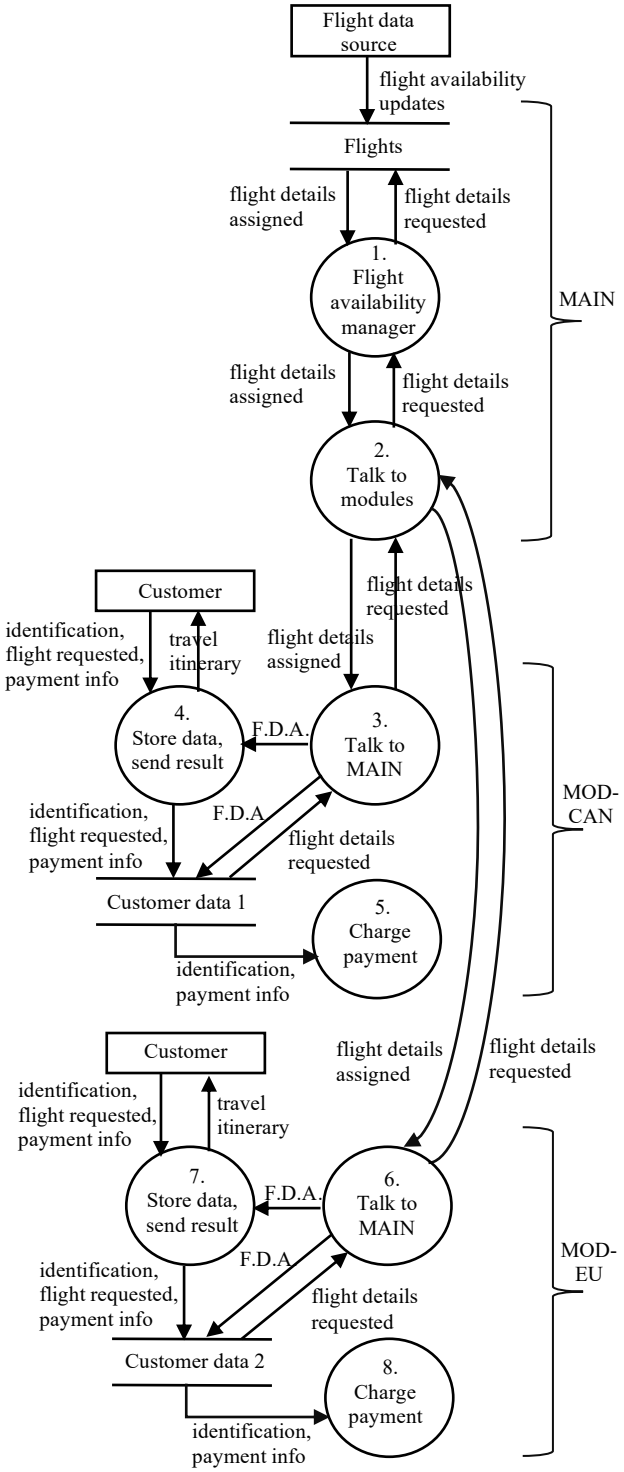

Figure 7. DFD for ACCURES, "F.D.A." stands for "Flight details assigned" – used due to lack of space. All data are SD.

Note that we count a 2-way communication (two arrows in opposite directions) as one link. An example is the 2-way communication between processes 1 and 2. This is because such a communication is considered implemented on a single physical cable, so it is really a single link that will be vulnerable to attack.

Applying Phase 2 in Section III-B, we first use methods b) and d) on the attacker accessible locations in Table IV, as follows:

- Using method b), combine process 5 with process 4 so that process 4 will now take on the additional function of charging the payment. Similarly, combine process 8 with process 7 so that process 7 additionally charges the payment. Note that these applications of method b) also eliminate the link from data store "Customer data 1" to process 5 and the link from data store "Customer data 2" to process 8. These changes were not seen as impacting performance, reliability, or maintenance and were accepted.
- Using method d), remove the two datastores "Customer data 1" and "Customer data 2" from the system; the software company's client decided that storing the customers' sensitive private information in these datastores results in excessive risks that the data could be compromised by attackers. The system would still be able to function according to expectation without these data stores. Note that removing these datastores also caused the removal of 4 SD links that were connecting to the data stores (two links per data store). These changes were also considered feasible and accepted.

These changes result in the DFD shown in Figure 8. Table V gives the attacker accessible SD locations corresponding to Figure 8.

TABLE V. ATTACKER ACCESSIBLE SD LOCATIONS IN FIGURE 8

	Links	Datastores	Processes
1	link into Flights data store	Flights	process 1
2	link between Flights data store and process 1		process 2
3	link between process 1 and process 2		process 3
4	link between process 2 and process 3		process 4
5	link between process 2 and process 6		process 6
6	link between Customer and process 4		process 7
7	link between process 3 and process 4		
8	link between Customer and process 7		
9	link between process 6 and process 7		

Table V shows that there are 9 attacker accessible SD link locations, 1 attacker accessible SD datastore, and 6 attacker accessible SD processes. For Figure 8, (3) gives the size N of the attack surface for sensitive data as $N = m + 2n + k = 9 + 2 + 6 = 17$. Thus, the application of methods b) and d) have reduced the attack surface from 29 to 17.

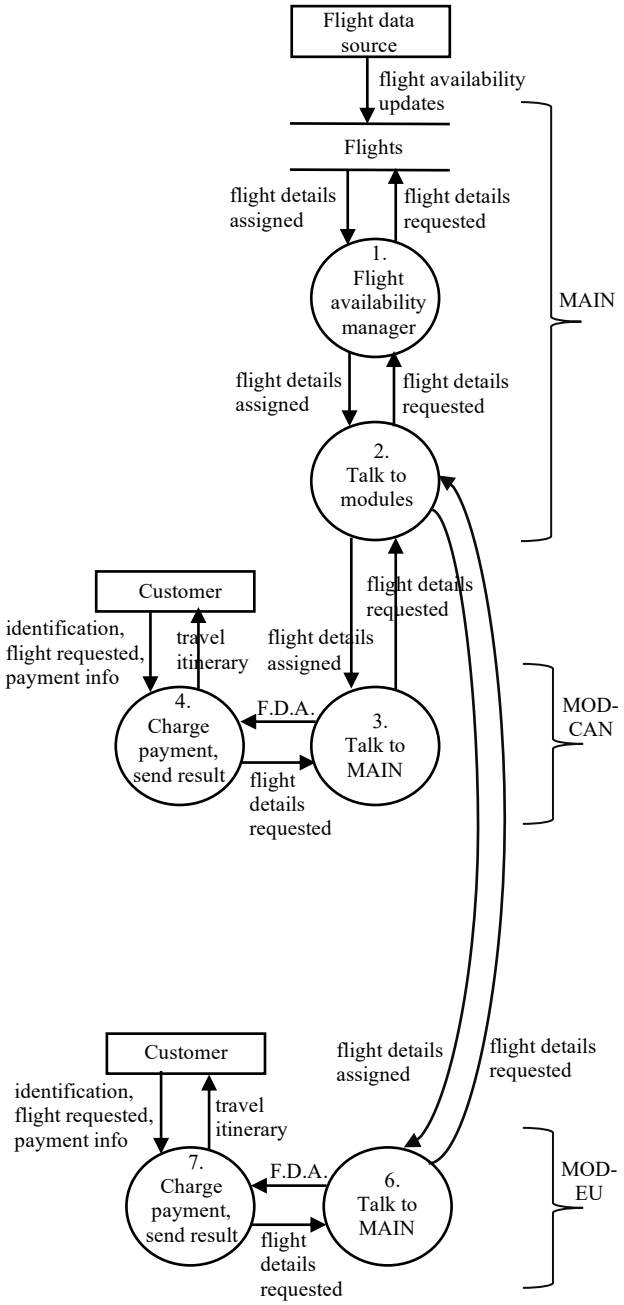

Figure 8. DFD for ACCURES after changes, "F.D.A." stands for "Flight details assigned" – used due to lack of space. All data are SD.

It is possible to further reduce the attack surface of Figure 8, as follows:

- Using method c), deny an attacker access to the SD link between process 3 and 4 by implementing MOD-CAN on one physical platform (see explanation for method c) above). Similarly, deny access to the SD link between processes 6 and 7 by implementing MOD-EU on one physical platform. Finally, deny attacker access to the SD link between the Flights datastore and process 1 and the SD link between processes 1 and 2 by implementing MAIN on one physical platform. These changes were not seen as impacting performance or maintenance and were accepted.
- Using method a), obfuscate the Flights data store. Next, obfuscate the 5 SD links that extend outside the modules. These links are: between "Flight data source" and the Flights datastore, between process 2 and process 3, between process 2 and process 6, between "Customer" and process 4, and between "Customer" and process 7. These changes were deemed feasible. Consideration was also given to obfuscating or combining some of the processes, but this was not done as it would significantly impact performance.

These changes do not alter the DFD in Figure 8. Table VI gives the remaining attacker accessible SD locations. The new attack surface is of size $N = m + 2n + k = 0 + 0 + 6 = 6$. The application of the methods in Section III-A has improved the security of sensitive data in the system by reducing the size of the attack surface from 29 to 6.

TABLE VI. REMAINING ATTACKER ACCESSIBLE SD LOCATIONS

	Links	Datastores	Processes
1			process 1
2			process 2
3			process 3
4			process 4
5			process 6
6			Process 7

V. RELATED WORK

In this section, we refer to "attack surface" in the general sense.

Most closely related to this work is this author's previous work on reducing the attack surface [9][10]. However, this previous work differs from the current work in at least the following two ways: a) the previous work proposes a graphical model with which to identify the attack surface whereas the current work does not require any such model, and b) the previous work reduces the attack surface by requiring the developer to learn and modify the graphical model whereas the current work has no such requirement.

Some of the following related works deal with attack surface identification and reduction at the code or binary levels, whereas this work deals with it at the architectural level. A few of these works reduce the attack surface by removing unnecessary code or features similar to the removal of SD locations in this work. A. Kurmus et al. [11] look at reducing the attack surface of commodity OS kernels by identifying code that is not used and removing it or preventing it from executing. T. Kroes et al. [12] investigate reducing the attack surface through dynamic binary lifting, removal of unnecessary features, and recompilation. R. Ando [13] presents work on attack surface reduction through call graph enumeration in which attackable call graphs are removed. S. N. Bukhari et al. [14] propose reducing the attack surface corresponding to cross-site scripting by employing secure coding practices. G.V. Neville-Neil [15] writes that "the best way to reduce the attack surface of a piece of software is to remove any unnecessary code". Obermeier et al. [16] propose to reduce the attack surface of next-generation industrial control systems through the use of a dynamic security system that adapts the parameters of network and security controls according to underlying changes in the control system environment. This results in the security controls only allowing data transfer that is required by the control system, thus reducing the attack surface.

The following works look only at identifying the attack surface. M. Sherman [17] investigates attack surfaces for mobile devices. This author claims that mobile devices exhibit attack surfaces in capabilities, such as communication, computation, and sensors, that are generally not considered in current secure coding recommendations. C. Theisen et al. [18] propose the use of risk-based attack surface approximation (RASA) which uses crash dump stack traces to predict what code may contain attackable vulnerabilities. Their goal is to help software developers prioritize their security efforts by providing them with an attack surface approximation. P. K. Manadhata and J. M. Wing [5] provide a much more detailed metric of attack surface than is defined in this work. Their metric may be considered as a measure of the size of the attack surface for all possible threats. Our definition of data attack surface may be considered as a subset of their definition. It is not possible to compare these two definitions in terms of accuracy or usefulness since they were defined with different purposes in mind. The same applies to any attempt to compare our definition with any other definition of attack surface.

Software attack surface identification and reduction is closely related to software vulnerability analysis, where the greater part of research also appears to be at the code level. Perl et al. [19] use an SVM classifier to find vulnerabilities in code repositories. Li et al. [20] describe VulPecker, a tool that can find a specific vulnerability in source code. Pang et al. [21] describe predicting vulnerable software components using a method built on a deep neural network. Anand et al. [22] suggest a way of classifying security patterns based on the type of vulnerability they treat. Also in this vulnerabilities category but working at the architectural level is this author's work, Yee [23], which deals with using a graphical model to identify and remove vulnerabilities during design. Yee [23] differs mainly from the current work in that it focuses on vulnerabilities found through risk analysis whereas the current work focuses on the data attack surface found by counting the attacker accessible SD locations in a model of the system such as its DFD.

Some works propose to increase security through attack surface expansion rather than attack surface reduction. For cloud services, T. Al-Salah et al. [24] propose three attack surface expansion approaches that use decoy virtual machines co-existing with the real virtual machines in the same physical host. They claim that simulation shows that adding the decoy virtual machines can significantly reduce the attackers' success rate. For enterprise networks, K. Sun and S. Jajodia [25] propose a new mechanism that expands the attack surface, so that attackers have difficulty in identifying the real attack surface from the much larger expanded attack surface. Note that these works do not contradict reducing the attack surface to improve security, since the attack surface is not really expanded but only appears to be expanded due to the addition of decoys.

VI. Discussion

It has been suggested that combining multiple SD locations into a single SD location using attack surface reduction method b) will result in a greater loss should the attacker target this combined location. However, recall that all SD locations on the data attack surface are attacker accessible. Therefore, the multiple locations existing prior to combining were all attacker accessible. But because of the greater attack surface prior to combining, it is more likely that all of those locations will be attacked than the single combined location. Note that prior to combining, the attacker will attack all the locations rather just a few, since there is no reason for him/her to stop until he/she gets all the SD. Thus, the likelihood of the single combined location being attacked after combining is less than the likelihood of the multiple locations being attacked prior to combining, and the potential data loss is the same both after combining and prior to combining. This shows the advantage of a smaller attack surface. In addition, there is always the possibility of applying reduction method a) (make a SD location useless to the attacker) to the combined location, removing it from the data attack surface.

Definition 3 describes the attacker accessible SD locations that make up the data attack surface as containing SD that is in the clear. It has been suggested that these locations can also contain encrypted SD since some attackers may still attack such locations, in order to obtain encrypted SD which they would then decrypt. While this is possible, our view in this work is that attackers would not attack such locations, since there are many other attacker accessible SD locations that contain SD in the clear. However, if we were to allow locations to contain encrypted SD, our attack surface reduction approach would be impacted as follows: reduction method a) would not be applicable to such locations, and such locations would remain part of the data attack surface, unless eliminated by methods b), c), or d). The overall impact would be slightly fewer opportunities to reduce the data attack surface.

We admit that our choice of $M_l = M_p = 1$ and $M_d = 2$ in Section II-B may not be accurate, but accuracy is not needed to show that the data attack surface is smaller after each reduction method is used. Nevertheless, these multiplier values do represent a reasonable approximation to the real values, which can only be ascertained with further research studies. Furthermore, perhaps the real values are not so important, since the true benefit of knowing the size of the data attack surface is to be able to use it for comparison and measurement purposes as a result of some improvement effort, i.e., the capabilities listed in the first paragraph of Section II-B, and our values already provide this benefit.

VII. Conclusions and Future Work

This work has presented an easy way to identify and calculate the size of the attack surface for sensitive data held within an online computer system, based on finding attacker reachable SD locations in the system. This work has also introduced methods for reducing the attack surface that are to be applied at the architectural level early in the development cycle, prior to coding, as part of the Design for Security toolset.

Applying the methods does not require developers to learn a new model or a new coding language. Apart from the methods themselves, which are straightforward, a minimal level of security knowledge is needed, in order to understand the concept of data attack surface, the purpose of the methods, and how they work. Knowledge of the computer system is the major requirement, but developers already have this knowledge. Although the methods themselves are straightforward, applying them can be challenging in terms of their impact on performance, ease of maintenance, and other factors, as mentioned above. However, the goal of applying the methods is not to obtain the smallest attack surface possible, but rather to reduce the attack surface while balancing the needs of performance, reliability, maintainability, and so on. Thus, it is quite acceptable not to have attained the smallest attack surface possible, so long as those other needs are satisfied. We expect the methods to be acceptable to developers and their management because of their practicality and ease of application.

Future work includes improving the identification of the attack surface and the calculation of its size. In addition, we hope to refine the methods for reducing the attack surface from developer feedback, obtained perhaps through workshops and trials. Other future work consists of investigating new methods for reducing the attack surface and looking at tools that could indicate a method's impact on such aspects as performance, reliability, ease of maintenance, and implementation costs.

References

[1] G. Yee, "Reducing the Attack Surface for Private Data," Proc. Thirteenth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2019), October 2019, pp. 28-34.

[2] Identity Force, "2020 Data Breaches – The Worst So Far," [retrieved: December, 2020] https://www.identityforce.com/blog/2020-data-breaches

[3] G. Yee, "Visualization and Prioritization of Privacy Risks in Software Systems," International Journal on Advances in Security, issn 1942-2636, vol. 10, no. 1&2, pp. 14-25, 2017, [retrieved: Dec., 2020] http://www.iariajournals.org/security/

[4] C. Salter, O. Sami Saydjari, B. Schneier, and J. Wallner, "Towards a Secure System Engineering Methodology," Proceedings of New Security Paradigms Workshop, Sept. 1998, pp. 2-10.

[5] P. K. Manadhata and J. M. Wing, "An Attack Surface Metric," IEEE Transactions on Software Engineering, vol. 37, no. 3, pp. 371-386, May/June, 2011.

[6] M. L. Collins, M. C. Theis, R. F. Trzeciak, J. R. Strozer, J. W. Clark, D. L. Costa, T. Cassidy, M. J. Albrethsen, and A. P. Moore, "Common Sense Guide to Mitigating Insider Threats, Fifth Edition," Software Engineering Institute, Report Number CMU/SEI-2016-TR-015, December 2016.

[7] D. M. Cappelli, A. P. Moore, and R. F. Trzeciak, The CERT Guide to Insider Threats, Addison-Wesley Professional, ISBN 9780321812575, January 2012.

[8] T. DeMarco, Structured Analysis and System Specification, Prentice Hall, May, 1979.

[9] G. Yee, "Modeling and Reducing the Attack Surface in Software Systems," Proceedings, 11th Workshop on Modelling in Software Engineering (MiSE'2019), May 2019, pp. 55-62.

[10] G. Yee, "Attack Surface Identification and Reduction Model Applied in Scrum," Proceedings, 2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), June 2019, pp. 1-8.

[11] A. Kurmus, A. Sorniotti, and R. Kapitza, "Attack Surface Reduction for Commodity OS Kernels: Trimmed Garden Plants May Attract Less Bugs," Proceedings of the Fourth European Workshop on System Security (EUROSEC '11), April 2011, article no. 6 (no page number available).

[12] T. Kroes, A. Altinay, J. Nash, Y. Na, and S. Volckaert, "BinRec: Attack Surface Reduction Through Dynamic Binary Recovery," Proceedings of the 2018 Workshop on Forming an Ecosystem Around Software Transformation (FEAST '18), October 2018, pp. 8-13.

[13] R. Ando, "Automated Reduction of Attack Surface Using Call Graph Enumeration," Proceedings of the 2018 2nd International Conference on Management Engineering, Software Engineering and Service Sciences (ICMSS 2018), January 2018, pp. 118-121.

[14] S. N. Bukhari, M. A. Dar, and U. Iqbal, "Reducing Attack Surface Corresponding to Type 1 Cross-Site Scripting Attacks Using Secure Development Life Cycle Practices," Proceedings of the 4th International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB-18), February 2018, pp. 1-4.

[15] G. V. Neville-Neil, "Reducing the Attack Surface," Communications of the ACM, vol. 61, issue 2, pp. 27-28, February 2018.

[16] S. Obermeier, M. Wahler, T. Sivanthi, R. Schlegel, and A. Monot, "Automatic Attack Surface Reduction in Next-Generation Industrial Control Systems," Proceedings of the 2014 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), December 2014, pp. 1-8.

[17] M. Sherman, "Attack Surfaces for Mobile Devices," Proceedings of the 2nd International Workshop on Software Development Lifecycle for Mobile (DeMobile 2014), November 2014, pp. 5-8.

[18] C. Theisen, B. Murphy, K. Herzig, and L. Williams, "Risk-Based Attack Surface Approximation: How Much Data is Enough?," Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP '17), May 2017, pp. 273-282.

[19] H. Perl, S. Dechand, M. Smith, D. Arp, F. Yamaguchi, K. Rieck, S. Fahl, and Y. Acar, "VCCFinder: Finding Potential Vulnerabilities in Open-Source Projects to Assist Code Audits," Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS'15), October 2015, pp. 426-437.

[20] Z. Li, D. Zou, S. Xu, H. Jin, H. Qi, and J. Hu, "VulPecker: An Automated Vulnerability Detection System Based on Code Similarity Analysis," Proceedings of the 32nd Annual Conference on Computer Security Applications (ACSAC'16), December 2016, pp. 201-213.

[21] Y. Pang, X. Xue, and H. Wang, "Predicting Vulnerable Software Components through Deep Neural Network," Proceedings of the 2017 International Conference on Deep Learning Technologies (ICDLT'17), June 2017, pp. 6-10.

[22] P. Anand, J. Ryoo, and R. Kazman, "Vulnerability-based Security Pattern Categorization in Search of Missing Patterns," Proceedings of the 2014 Ninth International Conference on Availability, Reliability and Security (ARES), September 2014, pp. 476-483.

[23] G. Yee, "Removing Software Vulnerabilities During Design," Proceedings of the 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), vol. 2, July 2018, pp. 504-509.

[24] T. Al-Salah, L. Hong, and S. Shetty, "Attack Surface Expansion Using Decoys to Protect Virtualized Infrastructure," Proceedings of the 2017 IEEE International Conference on Edge Computing (EDGE), June 2017, pp. 216-219.

[25] K. Sun and S. Jajodia, "Protecting Enterprise Networks through Attack Surface Expansion," Proceedings of the 2014 Workshop on Cyber Security Analytics, Intelligence and Automation (SafeConfig '14), November 2014, pp. 29-32.