Kant, Cybernetics, and Cybersecurity: Integration and Secure Computation

Jon K. Burmeister and Ziyuan Meng

College of Mount Saint Vincent
Drew University

jon.burmeister@cmsv.edu, zmeng@drew.edu

Abstract[1]

This paper argues that Kant's philosophy of mind sheds light on Heinz Von Foerster's cybernetic thinking, and that both thinkers help us identify dubious theoretical assumptions within computer science and cybersecurity. Specifically, these two thinkers discuss the importance of integration within systems, a position which contrasts with a reductionist form of thinking currently common in computer science. We argue that such a reductionist and narrowly technocentric approach leads to the design of insecure software systems. To develop an improved theory of security and vulnerability, we look for inspiration to Kant and von Foerster.

Our approach focuses on two types of integration within Kant's philosophy of mind – the "unity of apperception," and the unity of the mental faculties – and then traces these same themes in the thought of von Foerster. Building on that, we argue two points: 1.) a secure software system never directly takes its structure or operations from the external environement, and 2.) the more integrated a software system is, the more secure it is. To illustrate these points, we analyze a case study of a code injection attack against a vulnerable web application, and show how such a system is vulnerable to cyberattack when it fails to maintain its integrated form in response to inputs from the environment.

Keywords: *Philosophy of Mind, Cybernetics, Cybersecurity, Information Systems, Kant, Heinz von Foerster, Integration, Reductionism.*

1. Introduction

The philosophical tradition of German Idealism and the field of cybernetics share a common impulse: both strive to think systematically and at a high level of generality, with the goal of discovering foundational principles that apply across a wide variety of domains. Immanuel Kant and Heinz von

[1] We are grateful to Dr. Barry Burd, Dr. Emily Hill, and Dr. David Storey for serving as peer-editors on this article.

Foerster are classic representatives of this shared impulse within these two traditions.

Like Von Foerster, Kant's curiosity was voracious, and his corpus covers a mind boggling array of topics, from meteorology to biology to logic to mind to physics to art. Had Kant lived in the 20th century, there is no doubt he would have also written about computing machines, particularly given his intense focus on questions about cognition and mathematics. This essay presents von Foerster as a kind of bridge between Kant's highly abstract philosophy of mind and the highly concrete world of computer science and cybersecurity. There are at least two reasons to treat von Foerster as such a bridge. One is that his own philosophy of mind strongly overlaps with that of Kant. The second reason is that, as a cyberneticist, he seeks to discover structural similarities between the mind and computers that Kant simply could not explore due to his time in history. Given how vast both the benefits and the dangers of computing technologies are, it is prudent to muster the greatest intellectual resources possible in order to understand its possibilities. This article is one such attempt.

Computing technologies, which in many respects are the offspring of the cybernetics movement, are being increasingly woven into our everyday lives. From online shopping to communication, health care, transportation, etc., software continues (as the saying goes) to eat the world. But this rapid growth comes with a cost. As human beings rely more and more on software, the damage that can be inflicted by malware and software vulnerabilities increases proportionately. As more and more aspects of human life are 'virtualized,' and as virtual and augmented reality take steps toward mass adoption, advanced cybersecurity must be a central pillar of any developed society.

Yet the current dominant approaches to cybersecurity are clearly inadequate. One does not need to be an expert to recognize this, given the frequency of high profile cyber breaches in the news. In 2020 for example, in the most damaging reported cyberattack in U.S. history, a group of cybercriminals gained access to the publishing system of IT management software firm SolarWind and injected malware into the update package of their network monitoring software product, used by the intelligence services

and military of the U.S government among many others (FireEye, 2020).[2] Speaking generally, we believe that one reason for the failure of traditional approaches to cybersecurity is that they take a reductionist and technocentric attitude, focusing solely on technical explanations of cyberattack incidents. The dominant thinking in cybersecurity is based on theoretical assumptions that are both unexamined and dubious.

For all of these reasons, new and bold approaches to cybersecurity are needed, and this reality inspires our approach of looking for inspiration in Kant's philosophy of mind and von Foerster's cybernetic thinking. We will not be making strong claims about any kind of structural identity between minds and possible computing machines, but rather will use analogical reasoning to see what ideas might arise that are useful for improving cybersecurity.

To further justify this unorthodox interdisciplinary approach, let us consider a ground-level question: what is the basic problem that cybersecurity attempts to solve? The problem is how to enable a computing system to be in one respect *open* to what is outside of itself, but in another respect *closed*: that is, to be open to 'friendly' information (emails from friends, ebooks we purchased) but closed to 'malicious' information (ransomware, spyware, etc.). To defend a system that is entirely closed would require no work at all, and to defend a system that is entirely open would be impossible. So the main challenge of cybersecurity is to create a system that is both open and closed in precisely the right manner, and with precisely the right balance. We will argue below that the ability of a computer system to defend itself stands in direct proportion to that system's level of *integration,* or unificiation. Thus, the more integrated a system is, the more secure that system will be.

To make that argument, we will first examine Immanuel Kant's 'unity of apperception' and unity of the cognitive faculties as a model of an integrated system. This will lead into a discussion of von Foersters' constructivism and his views on cognitive unity. What becomes apparent is

[2] Solar Wind's software product Orion became a trojan for the attackers to further compromise the security of these organizations and perform cyber espionage remotely. The malware embedded in Orion is so sophisticated that it evaded detection for more than eight months. The scale and depth of impact of SolarWind hack is still under investigation as this article is being written.

that Kant and von Foerster's shared constructivist epistemology – also seen through the cybernetic idea of 'operational closure' – springs out of a shared anti-reductionist view of mind, i.e., an emphasis on the integration of the mind. These two themes inform our thesis of two necessary (but not sufficient) conditions of secure computation:

a) A secure software system never directly takes its structure or operations from the external environment.
b) A secure software system's components are functionally integrated during the runtime execution.

Finally, we will evaluate this thesis in a case study of code injection vulnerability in a web application. We will not provide a technically specific solution for this kind of vulnerability, but rather will use this case study to illustrate the two necessary conditions for secure computation mentioned above.

2. Kant on the Mind: Constructivist and Anti-Reductionist

As with any other discipline, computer science did not emerge from a cultural and intellectual vacuum. In addition to advances in electrical engineering, prior advances within the philosophical tradition helped make computer science possible. Some of those advances were very old, such as Aristotle's founding of the discipline of formal logic (including the syllogism), while some were more recent, such as work in logic and mathematics in early 20th century analytic philosophy. Computer science's indebtedness to the western philosophical tradition for some of its most fundamental concepts is an historical fact which encourages us to consider what that much older tradition might *continue* to contribute to it, including to the field of cybersecurity.

Since the time of Plato, many Western philosophers have focused on both the unity of the mind itself and the mind's unifying powers, viewing these two qualities as central to what makes something a mind in the first place.[3] Immanuel Kant stands in this tradition. In the *Critique of Pure Reason*, he gives an account of what the human mind can and cannot know, and thus necessarily delves into the nature of how the mind itself functions. Kant

[3] E.g., see Plato's *Phaedrus* 249B-C.

views the mind as having a number of capacities, two of which are a receptive faculty which he calls 'sensibility,' which receives sensations from the outside world, and an active faculty which he calls the 'understanding' and that applies concepts to those sensations to produce knowledge (Kant, 1965, p. 93). In thinking about computers in this light, we must take great care not to anthropomorphize them; yet we will argue that Kant's view of the mind as an integrated and integrating set of processes may shed light on how computational processes might be constructed in a similar fashion.

2.1. Kant's Constructivism

As is well known, Kant's views on the nature of the mind and the nature of knowledge are (among other things) constructivist and anti-empiricist, in that they are a response to what he sees as the untenable philosophy of mind proposed by empiricist thinkers such as Locke and Hume. Against the empiricist view that knowledge consists of sense perceptions, where the mind functions like melted wax and passively receives the imprint of sensations, Kant famously argues that knowledge only arises when sense perceptions received by the mind are combined with concepts already present in the mind. That is, he argues that knowledge only arises by being *constructed* by the mind. In contemporary philosophy of mind, Wilfred Sellars coins a useful phrase to describe the anti-empiricist approach when he speaks of the "myth of the given" – i.e., the myth that knowledge is produced in the mind simply by the environment giving the mind sensory impressions (Sellars, 1997).

Against the empiricist position, Kant views the understanding (the 'active' capacity of the mind) as unifying the many representations and sensations that the mind receives. The understanding combines the numerous sensations that enter the mind (e.g., various kinds of visual information) into a single cognition, by means of the application of a concept (e.g., "That is a dog."). This mental act of combination, Kant says, is "an act of the self-activity of the subject," and thus "it cannot be executed except by the subject itself" (Kant, 1965, p. 152). In other words, the mind passively receives an unorganized stream of sensations but then actively engages in the integration of those sensations, thereby lending order and wholeness to that previously unorganized stream (Kant, 1965, p. 134).

Kant talks about this unifying activity of the mind as being the "unity of apperception." In his words, knowledge is made possible by "the formal unity of consciousness in the synthesis of the manifold of representations" (Kant, 1965, p. 135). Said otherwise, the mind receives a manifold or multiplicity of sensory representations and it then synthesizes or unifies that manifold by means of a concept.

2.2. Kant's Anti-Reductionism

But how is the mind able to do this? What enables it to act as such a unifying, integrating force? Crucially, Kant argues that the mind's own unity is what enables this (Kant, 1965, p. 135). The mind is the "original" unity (Kant, 1965, p. 152), which then grants unity to the stream of sensations that it receives, resulting in our various meaningful experiences of the world. As we will see, Kant's view of the mind as a unity is an 'anti-reductionist' view of the mind, insofar as it does not reduce the mind to being a mere combination of its particular facilities or capacities. It is this unity and irreducibility of the mind -- i.e., not the unity that the mind grants to sensory representations but rather the mind's own unity -- that we must explore in greater depth. And it is this unity and irreducibility which we will connect closely with the thought of Heinz von Foerster.

Before doing so, however, we should take a moment to reflect on the connections between the ideas sketched out above. For Kant, there is a causal relationship between his anti-empiricist view of knowledge, on the one hand, and his 'unity/unifier' view of the mind, on the other. The connection is this: sensory information from the environment does not (a la empiricism) passively and directly produce knowledge in the mind, and the *reason why* this is the case is that the mind -- which is itself an 'original unity' -- actively unifies sensory information with concepts to produce knowledge. In other words, Kant's view that the mind integrates sensory perceptions with concepts is the cause of his anti-empiricism. Kant is claiming that the human mind is simply not the sort of system which receives sensory information and passively holds it in an unstructured manner; instead, the mind alters or 'processes' that information by applying concepts to it, and by unifying it, while at the same time being itself a unified system.

Now we can turn to a more detailed discussion of Kant's anti-reductionist view of the mind, and to what he means in calling the mind an "original' unity. The aspect of the mind's unity that we will focus on is Kant's view of how the faculties of the mind (e.g., sensation, understanding, reason, etc.) stand in relation to the mind itself. The key point for Kant is that the faculties of the mind are not *parts* of the mind, but rather are *powers* of the mind. That is, the faculties of the mind can be separated from a conceptual point of view, but they are not 'sections' of the mind which could be separated from each other in reality. Rather, they are the powers by which the mind -- the mind as a unity -- accomplishes particular things. For example, the faculty of the understanding is not what understands, but instead the understanding is the power by which the *mind* understands. By way of analogy, we could speak of a lion as possessing the powers (or faculties) of 'raising young,' 'sleeping,' and 'hunting.' Clearly we would not say that the power of hunting is what hunts, but rather the lion hunts: hunting is the power *by which* the lion hunts. Hunting is not a 'part' of a lion but rather a power which enables an activity.

Kant's view that the mind's faculties are not 'parts' but rather 'powers' can be seen simply in the words that he chooses in the German. Instead of referring to *Teilen* (parts) of the mind, he refers to the word *Vermögen* or the *Fähigkeiten* of the mind, each of which can be translated as 'faculties' or 'powers' or 'capacities.' Furthermore, in the Introduction to the *Critique of Judgment* Kant not only uses this language of *Vermögen* but also stresses the unity of the facilities within the mind. For example, in the first edition of that text he entitles a section as "On the System of All of the Powers of the Human Mind" ("*Von dem System Aller des Meschlichen Gemüts*") (Kant, 1974, p. 18). And in the second edition, he speaks of the *Seelenvermögen*, or powers of the soul, as all operating within a larger whole (Kant, 1987, p. 16). It is, then, *one* mind that engages in various distinct activities, which we then describe in terms of various faculties being at work.

Another way that Kant talks about this topic is simply by speaking of the "subject" or the "I" which possesses the various mental faculties. As Kant puts it in the *Critique of Pure Reason*, "The manifold of representations, which are given in an intuition, would not be one and all *my* representations if they did not all belong to one self-consciousness" (Kant, 1965, p. 153).

In all of these ways, then, Kant argues for the unity and the irreducibility of the mind, and the inseparability of its various faculties. Its faculties are not 'parts' of the mind but powers, and they function as activities of one unified, integrated entity.

3. Von Foerster on Cognition: Constructivist and Anti-Reductionist

Kant's rejection of empiricist epistemology and his emphasis on the mind as an original unity continues to shape Western thought to this day. One place that we see this is in the thought of Heinz von Foerster, widely seen as the founder of second-order cybernetics. The ideas of Kant were alive and well – even if often just as a foil – in early 20th century Vienna, where much of von Foerster's formation took place. The shadow of Kant hung over both the Vienna Circle and Ludwig Wittgenstein, two of von Foerster's main philosophical influences. Yet this essay will not explore Kant's role in the historical genealogy of von Foersters' ideas; rather, it will focus on von Foerster's ideas themselves, and two ideas in particular that bear a striking resemblance to the ideas of Kant discussed above.

From a practical perspective, our goal is to find insights in the philosophy of mind that are relevant to computer science and to improving cybersecurity. Von Foerster himself desired to transfer insights from his studies of the human mind to the study of computing machines. In his essay on human cognition and human memory that we focus on below, he states, "We hope to provide with these studies the foundation for a new architecture of future computers…." (Foerster, 2003, p. 123). In this essay, however, we will not propose any new computing architecture but rather will apply ideas from Kant and von Forester to the current Turing-based computing systems, to develop a theory of what makes for secure computing.

3.1. Von Foerster's Constructivism

The first and most obvious way in which von Foerster's thinking about the mind resembles Kant's thinking is by following in Kant's anti-empiricist, constructivist footsteps. Von Foerster's essay "On Constructing a Reality"

contains a straightforward and unambiguous statement of this position in one of its section headings: “The Environment as We Perceive It Is Our Invention” (von Foerster, 2003, p. 212). That essay deals primarily with neurophysiology, but von Foerster articulates a similar position in his “Thoughts and Notes on Cognition,” where he challenges the traditional view of the ontological status of information. This traditional view is that information is a free-floating “token” transmitted to a receiving agent from the environment. By contrast, von Foester argues that instead of being passively received, information is the *outcome* of a cognitive process. That is, information is constructed:

> “Cognitive processes create descriptions of, that is information, about the environment.
> The environment contains no information. The environment is as it is.” (von Foerster, 2003, p. 189)

Von Foerster’s claim entails that inputs from the environment into a functioning cognitive system never directly determine the next state of that system. Rather, the system integrates the inputs from the environment with its own internal-state to generate meaningful responses. (von Foerster, 2003, p. 110).

Von Foerster’s anti-empiricist, constructivist view of the mind and knowledge are of a piece with another idea in his tool-kit, the idea of ‘operational closure.’ For Von Foerster, a system is operationally closed when “each output becomes the next input just as soon as it is produced” (von Foerster, 2003, p. 314). This way of describing operational closure requires that we rethink what we actually mean by the word ‘input,’ since an input that is produced by the system itself is not an input in the traditional sense, i.e., one which enters into the system from *outside* the system. In light of this new sense of ‘input,’ we can see that operational closure entails that the system in question is not directly structured by its environment, but rather by its own self.

It is crucial to note that operational closure does not require that the system be *entirely* closed off from its environment; rather, it simply means that the system does not take structures or its operations *directly* from its

environment. Put in this way, we can see how operational closure at work within the human mind is just another way of talking about constructivism.

Niklas Luhmann, the German sociologist and friend of von Foerster, addresses the implications of operational closure for epistemology, describing it as a "radical shift" away from a representational view of knowledge. Speaking of operational closure and the related concept of autopoesis, Luhmann states that these concepts are a rejection of the view that "something from the environment enters into the one who cognizes, and that the environment is represented, mirrored, imitated, or simulated within a cognizing system" (Luhmann, 2009, pp. 153-4). The anti-empiricist dimensions of operational closure are clear: knowledge is produced in the mind not by sensory impressions being received and 'mirrored' by the mind, but rather are produced through the activities of the mind itself. Applied to the human mental system, operational closure entails a constructivist epistemology.

3.2. Von Foerster's Anti-Reductionism

The second way in which von Foerster's philosophy of mind resembles Kant's is its strong anti-reductionism, i.e., its emphasis on the mind as an integrated functional whole. Just as with Kant, von Foerster's anti-reductionism regarding the faculties is the cause of his anti-empiricism. For example, empiricism becomes untenable when one rejects the idea that the faculty of perception can operate independently of concepts. A constructivist epistemology is the necessary result of viewing the faculties of conceptualizing and perceiving as part of a larger, integrated whole.

In his essay "What Is Memory that It May Have Hindsight and Foresight As Well," von Foerster proposes a thesis that the human cognitive architecture is a unified whole (von Foerster, 2003, pp. 101). He argues that the various faculties of our cognitive architecture are separable when conceptually analyzed, but *inseparable* in the actual cognitive experience.

> In the stream of cognitive processes, one can conceptually isolate certain components, for instance (i) the faculty to see (ii) the faculty to remember (iii) the faculty to infer. But if one wishes to isolate

> these faculties functionally or locally, one is doomed to fail. Consequently, if the mechanisms that are responsible for any of these faculties are to be discovered, then the totality of cognitive processes must be considered" (von Foerster, 2002, p. 105)

Von Foerster's claim is that viewing cognitive faculties as operating in an isolated and independent manner will lead to a confused view of mental functions. As an example, he points in the same essay to the mnemon (a 'unit' of memory) in higher mammals to illustrate that memory is not an isolated component to store and retrieve the past data, but rather is an integral part of the neural cognitive process as a whole (von Foerster, 2003, p. 110). As he puts it, such processes are "subservient to the maintenance of the integrity of the organism as a functioning unit" (von Foerster, 2003, p. 172).

Yet von Foerster does not develop this anti-reductionist line of thinking just to discuss human cognition. His ultimate goal in the "What is Memory" article is to lay the conceptual foundations for a new, more integrated architecture for the future of computing (von Foerster, 2003, pp. 123).[4] Our essay will not explore this ambitious topic since we are focusing on a theory of secure computing for traditional computer architectures. Nonetheless, von Foerster is clear that his critique of cognitive reductionism is a critique that applies to the common way of thinking about computers as well. Speaking about both the computer scientists and the biologists of his day, he says that they are hesitant to view mental faculties as activities of a single cognitive entity. They have, he says, a "reluctance to adopt a conceptual framework in which apparently separable higher mental faculties as, for example, 'to learn,' 'to remember,' 'to perceive,' 'to recall,' 'to predict,' etc., are seen as various manifestations of a single, more inclusive phenomenon, namely, 'cognition'...." (von Foerster, 2003 p. 172). Von Foerster provides a plausible explanation for why many thinkers tend to take this reductionist approach: it makes the work of understanding things (seemingly) *easier*, because it entails that the components of a system "can be reduced to rather trivial mechanisms" (von Foerster, 2003, p. 172).

[4] One architecture that he proposed is to integrate memory, logic inference and sensory capabilities in an adaptive computing unit called "cognitive tile" (von Foerster, 2003, pp. 119–123). He argues that a complex computing system can be built by combining multiple cognitive tiles.

An example of this mistake, he says, is reducing perception to mere "inputs"; in other words, as we have argued above, a non-integrated view of the faculties leads directly to an empiricist epistemology. When computer scientists attempt to recreate human cognitive processes within computers, it is simply easier to assume that those processes operate in isolation and then to model computer processes accordingly. But according to von Foerster, "by separating these functions from the totality of cognitive processes one has abandoned the original problem, and now searches for mechanisms that implement entirely different functions…" (von Foerster, 2003, p. 172). In the sections below, our goal is to take a small step toward the more difficult but hopefully more productive approach of viewing the various computing capacities as subservient to the integrity of the whole, and thus gain more insight into secure and insecure computation.

4. A Constructivist and Anti-Reductionist Theory of Cybersecurity

What can computer scientists and cybersecurity specialists learn from Kantian philosophy of mind and von Foerster's thesis on cognitive systems?

First, both Kant's and von Foerster's constructivism, and opposition to empiricism, entail the *indirectness* of perceptions' impact on the mind. A cognitive system never allows input from the external world to directly give rise to higher level mental functions. Similarly, a secure software system must always interact with the environment only indirectly. The inputs – which are analogous to perceptions – should not *directly* generate the outcome of the computation without reference to pre-established internal structure (which are analogous to concepts in the mind). However, in many cybersecurity attack case studies, we see that the systems are vulnerable precisely because they allow an attacker to directly inject code into the system and instruct the system to perform unintended computational tasks. A successful code injection attack can occur only when a software system *lacks* operational closure, since the system directly takes its structure and order from its environment without the participation of the system's own internal properties, e.g., its algorithms or its memory state. (By 'software

system' we mean software both at the application level and at the operating system level, such as the kernel.)

These observations lead to our first postulation: a secure software system never allows an input from the external environment to determine the meaning of the computation *without reference to its previously established internal state*. That is, a secure software system never allows an input *alone* to determine the meaning of a computation, since that would involve bypassing the system's previously established internal state. This is not to say that a secure system is nondeterministic, or cannot be predicted; it is simply to say that a secure system must compute an outcome based on both the input *and* its own internal state.

Second, cognitive faculties mentioned by von Foerster find their analogues in software design. For instance, one commonly adopted web application design pattern models a website as a system consisting of 1) a user interface component with the ability to take input data, 2) a data storage component to retain the past "experience" of the system, 3) a logic inference component, i.e., an algorithm (Buschmann et al., 1996, p. 3). These components correspond to the cognitive faculties of perception, memory, and inference in von Foerster's cognitive architecture, respectively. However, unlike Kantian epistemology and von Forester's thesis on cognition which emphasize unity, the computer science and Information Technology industry today often adopts a reductionist and atomistic approach in software engineering. They often develop these components as isolated functional units, and then only later on find a way to connect them in the process of software engineering. Independently developed functional units are also often reused in multiple software projects for the purpose of cost savings. Today, software developers can easily speed up their engineering process by using pre-built platforms shared by other projects. However, the fact that these components are interconnected does not mean that they are integrated, i.e., made into a unified whole. We show in the essay's next section that the weak links of mere interconnectedness (versus the strong links of integration) are what lead to vulnerabilities that can be exploited in high profile cyber attacks against web applications. This observation leads us to our second postulation that the unity of functional components is vital to the security of a software system, just as the unity of

cognitive faculties is vital to human consciousness. The more that the integration of the software system resembles the integration of cognition, the more secure it will be. It should be noted, however, that our thesis about the necessary conditions of secure computing does not depend on whether there actually exists any structural identity between cognition and computing.

Before looking at a case study, it is useful to restate our thesis about two necessary (but not sufficient) conditions of secure computing:

a) A secure software system never directly takes its structure or operations from the external environment.
b) A secure software system's different components are integrated during the runtime execution.

4.1. Case Study

In this section we will use a famous security vulnerability case study, "SQL injection," to examine the cyber attack's effect on the functional unity and operational closure of a software system. SQL injection is a form of code injection attack in which an attacker inserts carefully crafted malicious code into a database component of a software system (e.g., a running web application) and forces the system to execute certain tasks which it is not intended to perform in the original design (Clarke et al., 2012; N. Singh et al., 2016).

Strategies to thwart SQL injections are well studied, as seen for example in the approach of using "prepared statements."[5] Our goal in this paper is not to propose any specific mitigation method, but rather to use SQL injection as an empirical study to develop a general theory of some of the necessary conditions of secure computation. We will use a simple example of a login web application to illustrate and compare secure and insecure computations. We will examine our two postulations in both scenarios.

[5] Solutions to mitigate SQL injection have been developed since 2005. The most effective one is to use *prepared statements*. Details of using prepared statements to prevent SQL injection is outside of the scope of this paper (Clarke et al., 2012, p. 342-349; Castillo et al., 2019, p. 171-175). The solution essentially translates the syntax structure of the SQL query into an immutable function called a prepared statement, then has all the incoming user inputs to be bound as the data parameters of the function. It, therefore, effectively prevents the user provided data from modifying the original syntax structure of the query.

4.1.1. A Primer on Web Apps: In this section, we will use a simplified authentication web application as an example to give a basic review of how a modern website works. A typical web application consists of a client component and a server component. The client component consists of web pages running in a user's web browser. These pages are essentially the user interfaces for various functions. The server component is running at a remote server which can be further divided into two subcomponents: a database system which is responsible for data storage and management, and a web application's control logic (i.e., its algorithms). The relationship of the three components -- the client's user interface, the database, and the control logic -- is illustrated in Figure 1. A user can send a request to the server, and upon receiving the request, the server processes the request, generates a response, and then sends it back to the user. It is this request-response cycle that constitutes a web application.

An authentication website typically supports a login function. Let us consider a scenario where a user tries to login to the system. In this scenario, a user interacts with a login web page (depicted in Figure 1 left) where he/she can fill in user information. When the user clicks the "Sign In" button, a login request including the username and password is sent to the server. Upon receiving the user's request, the web application's control logic will first buffer the user-submitted username and password in two variables, **$U** and **$P**, and then will query the database to see if it contains a matching user record. In this case study, we assume that the database stores the records of the registered users in a table named "User_info" with two columns: "UserName" and "Pwd":

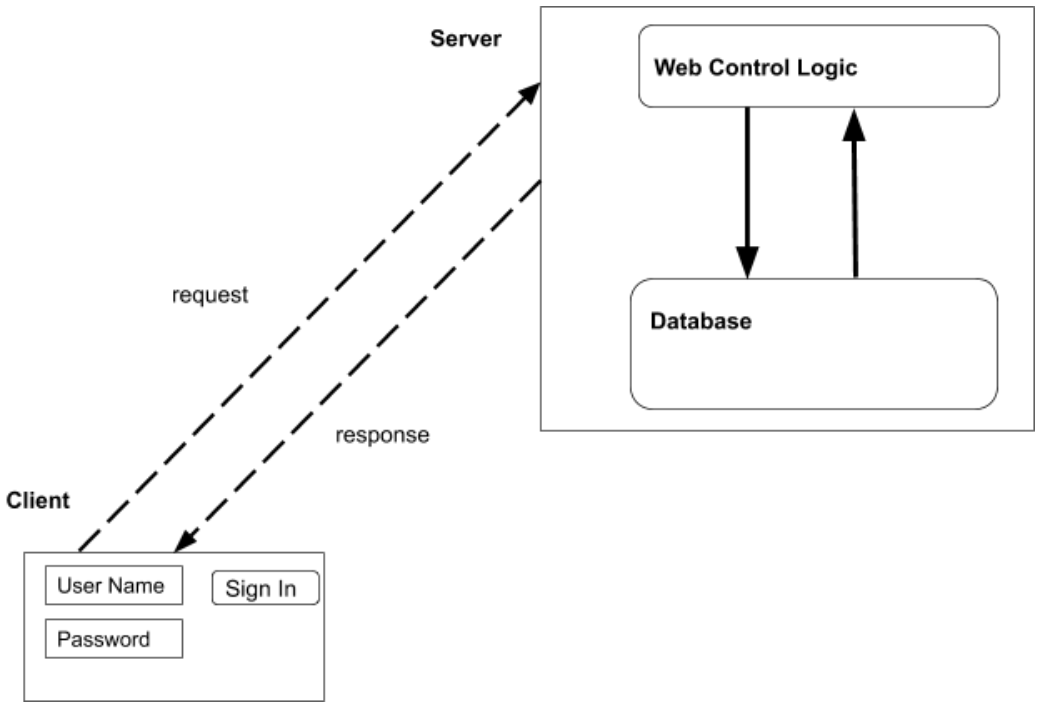

Figure 1. Client - Server Web Architecture for a login function of a website. The client side has a login web page displayed.

Table 1. An example of database table "User_info"

UserName	Pwd
Bob	76ddk8
Alice	6754gh
Joe	12345

The web application queries its database by using a standard database query language called "Structured Query Language" (SQL) (Melton & Simon, 1993). The following is the SQL query to verify the requesting user's credentials:

```
SELECT * FROM User_info WHERE UserName = '$U' AND Pwd
= '$P'
```

The query is pre-stored in the system's memory. The control logic first plugs the contents of **$U** and **$P** into the query. The query is then submitted to the database engine to execute.

For example, if a user submits a username **Joe** and a password **12345**, the control logic will form the following query and send it to the database engine to execute:

> SELECT * FROM User_info WHERE UserName = **'Joe'** AND Pwd = **'12345'**

The above query asks the database engine to fetch all records with full details (symbolized by *) from the "User_info" table where the UserName is equal to **Joe** and Password is equal to **12345**. The WHERE clause is used to specify the criteria to filter the records. It limits the query to fetch only the necessary records which meet the criteria.

If the query finds any matching rows from the "User_info" table, the login is successful, as seen in Figure 2. Otherwise, the login fails.

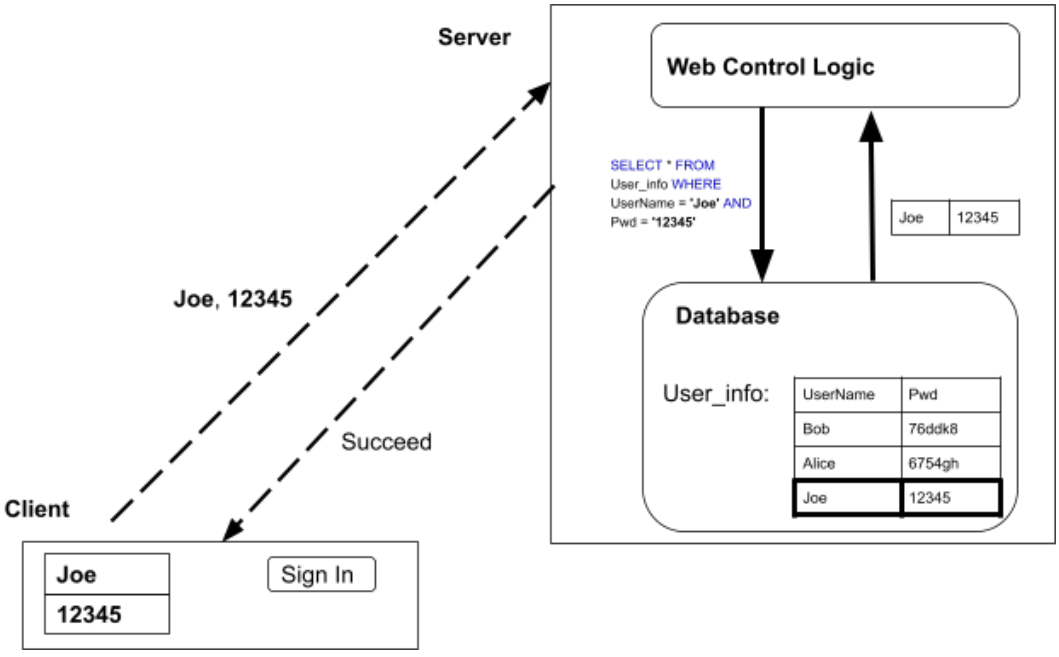

Figure 2. A successful login request with **Joe** and **12345** as the credential.

From the perspective of operational closure, the above computing process of login manifests the constructivist quality of *indirectness*. The user login request, as input, influences the outcome of the authentication process only after the pre-established control logic algorithm performs a comparison with the system's internal state: the state of the database table "User_info". The user login request *alone* does not dictate the outcome of the computation; rather, the user login request must be *processed within* the control logic algorithm before the outcome of the computation is produced.

Here, we can also apply von Forester's thesis on cognitive unity to examine the functional unity of the system during the login process. The ability of the system's user interface component to receive the user's input is analogous to the cognitive faculty of perception, while its database component is analogous to the faculty of memory, and the web application's control logic component is analogous to the faculty of inference. In a secure login computing process, the activities of these three components are inseparable from each other just like their cognitive counterparts are inseparable in any meaningful conscious experience involving sense data. In other words, the activities of these three components form a functional unity. Thus, the outcome of the computing process consists of the integration of the user input, the state of the database table, and the structure of the control logic. This integration is illustrated in the following flow figure:

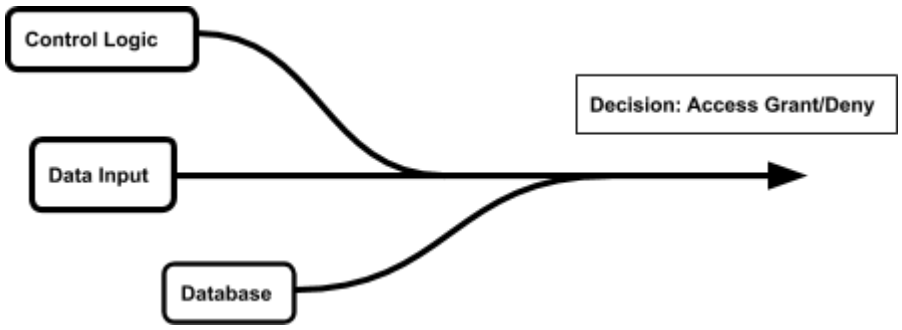

Figure 3. A login process must integrate all three components: user input, control logic and the database to decide whether to grant access to the user.

While the above login system seems simple and can securely handle user inputs most of the time, certain malicious inputs can trigger the system to malfunction. As we will see in the following section, the system contains a

code injection vulnerability which allows a malicious user to directly inject arbitrary code to execute and violate its unified form.

4.1.2. SQL injection to bypass authentication: SQL injection is one of the most common hacking techniques against a website (Clarke et al., 2012). It is a code injection technique used to attack web applications, in which malicious SQL statements are inserted into a target system for execution. SQL injection is not the only way an attacker can inject malicious code; similar code injection vulnerabilities have been discovered in the past two decades in many programming languages such as C, C++ (Seacord, 2005, pp. 33–47) and Java (Long et al., 2013, pp. 20–30). The Open Web Application Security Project (OWASP) has consistently listed code injection vulnerability as the most common and most dangerous type of vulnerability in web applications (OWASP, 2017). Although this paper focuses only on SQL injections, our treatment of this topic holds promise to shed light on other kinds of code injections as well. As mentioned earlier, we will not be providing a technically specific solution for SQL injections, but rather will use the SQL injection case study below to illustrate the two necessary conditions for secure computation developed in the pages above.

To see how a SQL injection attack can disrupt the functional unity of a target system, let us consider a hacker who does not have any credential and tries to bypass the authentication process without providing any correct username and password. To do this, the hacker submits the username as **Santa** and the password as **xyz' OR 'a' = 'a**.[6] After the control logic plugs them into the query, the query statement becomes:

> SELECT * FROM User_info WHERE UserName = **'Santa'** AND Pwd = **'xyz' OR 'a' = 'a'**

Here, adding **OR 'a'='a** in user data alters the structure and the meaning of the original SQL query -- that is, it alters the algorithm itself. Since **'a'='a'** is always ***True***, UserName = **'Santa'** AND Pwd = **'xyz' OR 'a' = 'a'** always

[6] Note that the hacker's specific use of *single* quotes is based on his desire to have the code injection fit correctly into the syntax of the *query's* use of single quotes: SELECT * FROM User_info WHERE UserName = **'$U'** AND Pwd = **'$P'**

evaluate to ***True***.[7] As a result, the query becomes equivalent to this completely nonspecific, nondiscriminating selection:

```
SELECT * FROM User_info
```

In other words, the query returns *all* the entries from the "User_info" table unconditionally, thus causing the web app control logic to mistakenly grant access to the attacker. This process is illustrated in Figure 4.

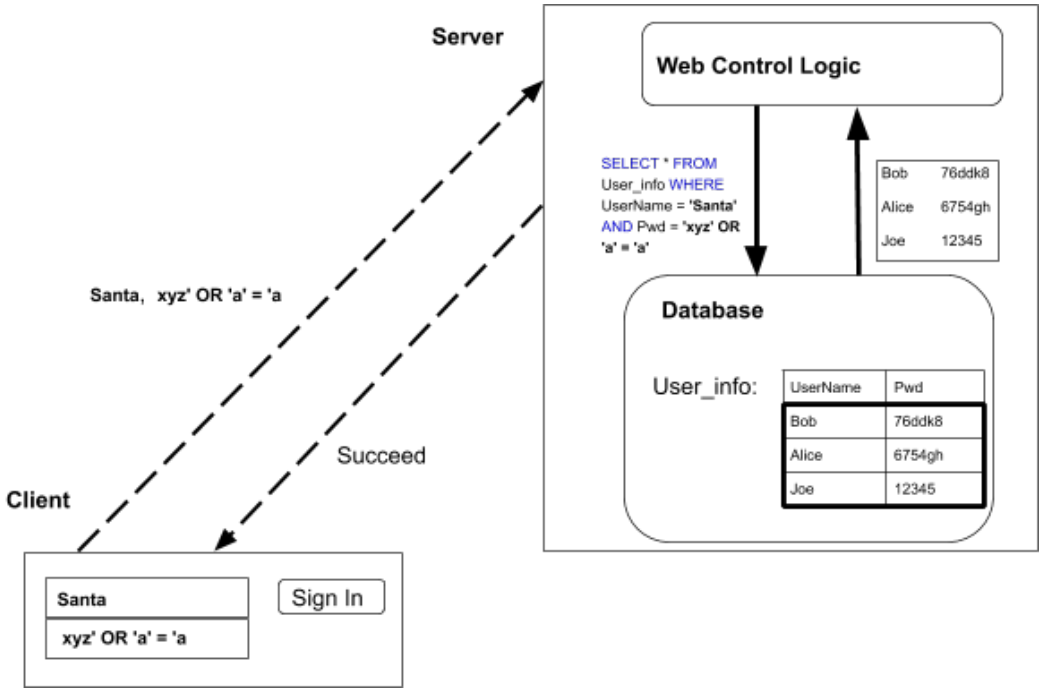

Figure 4. SQL injection to bypass authentication.

It is crucial to see that the code injection violates the *indirectness* of the system's interaction with the external environment, since originally the pre-established algorithm served as an intermediary between the environment and the system as a whole. But now the injected code directly *alters* that pre-established algorithm, by mixing itself with the algorithmic structure stored in the memory. In doing so, the user data becomes code!

Additionally, the code injection violates the system's functional unity. The state of the database ("User_info" table) has now become causally irrelevant to the rest of the system. As long as the "User_info" table is not empty, the

[7] This is the case because, in SQL, the AND operator has higher precedence over OR operator.

attacker will successfully login to the system. The SQL injection sets the control logic to *directly* grant the user access in a way that cuts the database content entirely out of the causal loop. That is, the code injection can easily pry apart the system as a whole and reduce it to a less integrated system. The following flow figure illustrates the effects of the injection attack on the system's integration:

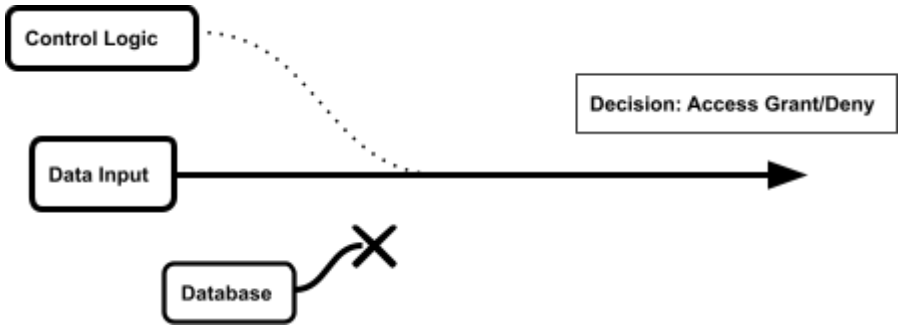

Figure 5. When an SQL injection attack occurs, the database becomes separated from the login process. The dotted line from the Control Logic (compared to the solid line in Figure 3) represents the SQL injection's violation and alteration of the original algorithm.

In fact, once the attacker discovers a SQL injection vulnerability, he/she can take a bold action that even further compromises the system's operational closure, by injecting code to directly modify the database (see Appendix A).

We should note that a SQL injection is just one kind of code injection, and that the principles of indirectness and integration can be used to analyze the vulnerabilities of computational systems to other kinds of code injection as well. For example, in the C programming language, the well known buffer-overflow injection vulnerability allows an attacker to inject binary code to a target machine and get executed (Erickson, 2008, pp. 115–193).[8]

[8] Our thesis can be extended to scenarios in which an exception occurs: an event which disrupts the normal execution of a program. In the context of cybersecurity, only the exceptions induced by user-provided inputs are relevant. If a user input causes an anticipated exception and there is code in a program which safely handles the exception and quits the program if necessary, the execution process is secure. It satisfies the necessary conditions of secure computation since the computational outcome still depends not only on the user input but also on the state of the exception and the part of an algorithm which handles it. If a user input causes an unanticipated exception and eventually leads the software system to shut down, the program is by definition vulnerable to denial of service attacks.

5. Conclusion

In summary, we propose an anti-empiricist, non-reductionist thesis of two necessary (but not sufficient) conditions of secure computing:

1. **Security requires operational closure**, which means that the software system cannot take its structure or operations *directly* from its external environment. For a system to be secure, an input can generate computational outcomes *only after* being processed within the system's pre-existing internal structure. If a system lacks operational closure, it will be insecure by definition, since it is excessively open to external manipulation. Put differently, it is not conceivable for a software system which is completely open to direct external manipulation to be a secure system.

2. **Security requires functional integration.** The only way that a software system can possess operational closure is by integrating its subcomponents during the runtime execution. If it fails to do this, it becomes insecure. Said otherwise, it not intelligible for a software system to be secure if its functional unity has disintegrated during the runtime execution, because this would compromise its operational closure.

Our findings having presented a constructivist understanding of what makes secure computation possible, and thus what constitutes vulnerabilities. Instead of focusing on how accurately computations simulate objective reality, our thesis takes a cue from Kantian transcendental idealism by examining the conditions for a computation to be secure. Our approach shifts the thinking of cybersecurity from *what* is computed to *how* it is computed.

In terms of future research, there are several technical and theoretical directions that might build on the findings of this essay. An immediate follow-up empirical study would be to expand our thesis to other forms of code injection. This could include close examinations of code injection vulnerabilities across different programming languages and the persistent code injection techniques employed by malware. In the long run, our

findings can also inspire a new generation of runtime vulnerability/malware detection systems to protect software applications from code injection. One can imagine a runtime self-monitoring system which checks the integration of various functional components in a given system and blocks an ongoing execution when it detects a violation of the system's functional unity.

Additional future research might explore the connections between Kant and von Foerster on the topic of self-reference. For example, in Kantian transcendental idealism, the 'subject' is constituted by being self-conscious, and we see a similar theme in von Foerster's thoughts on recursivity in both biological and computing systems (often in the forms of feedback loops), and particularly in his second-order cybernetics approach of including the observer into its own observations. We would like to know whether this sort of self-referential, self-observing structure can bring higher levels of integration to a system, i.e., not just integration of the system's subcomponents, but the integration of the system as a whole with itself.

A final point of potential future research is the novel architecture for computing *hardware* which von Foerster proposed, based on his concept of a "cognitive tile": a computing unit with integrated memory, perception and inference functions (von Foerster, 2003, pp. 119–123). This architecture deserves a resurrection from the University of Illinois' Biological Computer Lab archives, for this long forgotten creature might provide deeper insight into the nature of a truly integrated and truly secure computing system.

Acknowledgments

We thank Dr. Patrick Gamez and Dr. Joshua Shmikler for agreeing to be non-anonymous reviewers of this paper.

Appendix A. An example of using SQL injection to modify database content

To see an example of using SQL injection to modify the database, consider a scenario in the web login system introduced in the case study section: an attacker submits the username as **Santa'; INSERT INTO User_info VALUES ("John", "123abc") --** and password as **xyz**. The query then becomes:

> SELECT * FROM User_info WHERE UserName = **'Santa'; INSERT INTO User_info VALUES (''John'', ''123abc'') --** AND Pwd = **'xyz'**

In the syntax of SQL, the symbol **--** is reserved as the mark of the beginning of comments. When the SQL interpreter of the database sees the mark **--** in the query, it ignores everything that follows. The second condition in the WHERE clause, AND Pwd = **'123xyz'**, is consequently *blocked from execution by the database engine*.

The above query first selects all the user records with the name **Santa**, then inserts a new row **(''John'', ''123abc'')** into "User_info". This computational effect is directly caused by injected code without reference to the pre-stored algorithm and database. This process is illustrated in Figure 6 below.

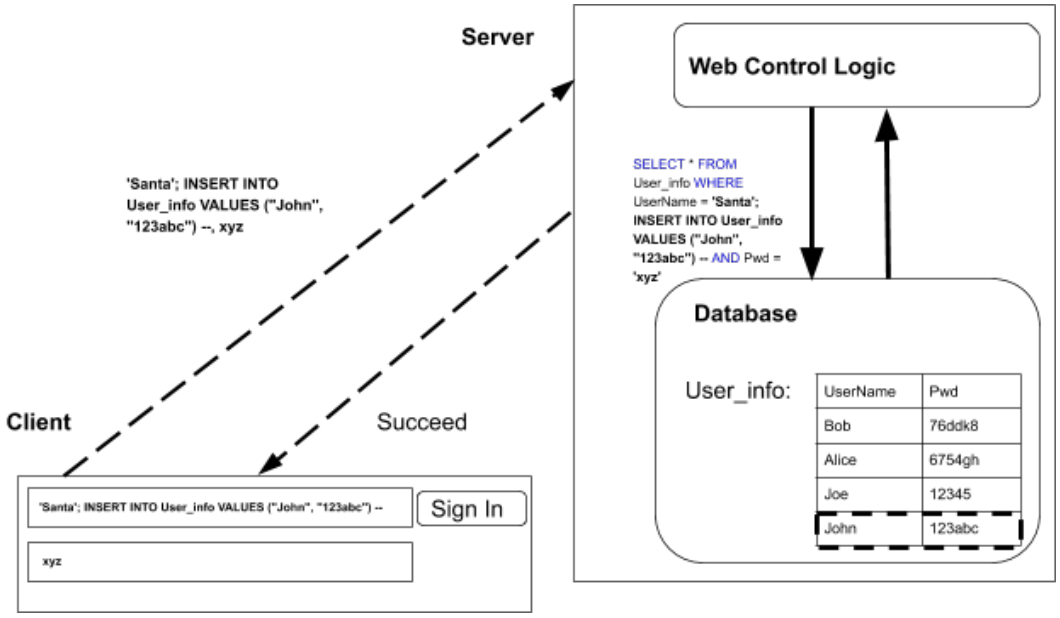

Figure 6. An SQL injection attack to insert a record **(John, 123abc)** into the database table: "User_info".

References

Buschmann, F., Meunier, Regine, Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-Oriented Software Architecture Volume 1: A System of Patterns* (Volume 1 ed.). Wiley.

Castillo, R.E., Caliwag, J.A., Pagaduan, R.A., and Nagua, A.C. (2019). Prevention of SQL Injection Attacks to Login Page of a Website Application using Prepared Statement Technique. In *Proceedings of the 2019 2nd International Conference on Information Science and Systems* (*ICISS 2019*). Association for Computing Machinery, New York, NY, USA, 171–175. https://doi.org/10.1145/3322645.3322704

Clarke, J., Fowler, K., Oftedal, E., Alvarez, R. M., Hartley, D., Kornbrust, A., O'Leary-Steele, G., Revelli, A., Siddharth, S., & Slaviero, M. (2012). *SQL Injection Attacks and Defense* (2nd ed.). Syngress.

Erickson, J. (2008). *Hacking: The Art of Exploitation, 2nd Edition* (2nd ed.). No Starch Press.

FireEye. (2020, December 13). *Highly Evasive Attacker Leverages SolarWinds Supply Chain to Compromise Multiple Global Victims With SUNBURST Backdoor*. https://www.fireeye.com/blog/threat-research/2020/12/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor.html

Kant, I. (1987). *Critique of Judgment* (1st ed.). Hackett Publishing.

Kant, I. (1965). *Critique of Pure Reason* (unabridged edition). St. Martin's Press.

Long, F., Mohindra, D., Seacord, R., Sutherland, D., & Svoboda, D. (2013). *Java Coding Guidelines: 75 Recommendations for Reliable and Secure Programs*. Addison-Wesley Professional.

Luhmann, N. (2009). Self-Organization and Autopoiesis. In M. B. N. Hansen & B. Clarke (Eds.), *Emergence and Embodiment: New Essays on Second-Order Systems Theory* (pp. 143–156). Duke University Press Books.

Melton, J., & Simon, A. R. (1993). *Understanding the New SQL: A Complete Guide (The Morgan Kaufmann Series in Data Management Systems)* (1st ed.). Morgan Kaufmann.

Open Web Application Security Project. (n.d.). *OWASP Top Ten 2017*. Owasp.Org. Retrieved January 15, 2021, from https://owasp.org/www-project-top-ten/2017/

Sajjadi, S. M. S., & Tajalli Pour, B. (2013). Study of SQL Injection Attacks and Countermeasures. *International Journal of Computer and Communication Engineering*, 539–542. https://doi.org/10.7763/ijcce.2013.v2.244

Seacord, R. C. (2005). *Secure Coding in C And C++* (1st ed.). Addison-Wesley Professional.

Sellars, W. (1997). *Empiricism and the Philosophy of Mind* (2nd Edition). Harvard University Press.

Singh, N., Dayal, M., Raw R.S., and Kumar, S. (2016). SQL injection: Types, Methodology, Attack Queries and Prevention. *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India, 2016, 2872-2876.

von Foerster, H. (2003). *Understanding Understanding: Essays on Cybernetics and Cognition*. Springer Publishing.