Certain cybersecurity: the impossible dream

Kaled Aljebur, Mostfa Albdair, Ron Addie, *Member, IEEE,*

***Abstract*—The abstract goes here.**

I. Introduction

One of the most important themes in the history of the philosophy of science, initiated by David Hume, has been the difficulty of inductive reasoning, i.e., how are we able to infer facts from observations. Karl Popper's "solution" to this problem is that the scientific method is really about trying to find evidence *against* a hypothesis, and when such evidence cannot be found, despite our best efforts, this can be regarded as strong evidence *for* the hypothesis. Popper's resolution was, and remains, quite influential.

However, this debate has not ceased since Popper's contribution. Other key contributions were made by Kuhn, and xx, but there is still no universally accepted resolution.

However, more pragmatically, this question is also addressed by statistics. Furthermore, statistics has quantitative procedures.

Even so, statistics also has some really difficult foundational issues that are not just theoretical. Fundamentally, statistical reasoning relies on assumptions, just like deductive reasoning. In particular, we have to adopt a model. Naive practitioners (and that is the vast majority, esp given that most of them are not actually educated in statistics), like to believe that it is satisfactory to pick a standard statistical method and apply it. This is, after all, what they are taught to do. In fact, not behaving this way is regarded by most users of statistics as unorthodox, and unsound.

So, statistics is really the modern form of inductive reasoning, and it is certainly a lot better defined than the philosophical concept of inductive reasoning.

But, what about deductive reasoning?

The "default" viewpoint is that we only use this in mathematics, or perhaps for going from one set of assumptions to another. But cybersecurity, and especially public key cryptography, and also block-chain techniques, seem to suggest that we can effectively use deductive reasoning about the real world.

In cybersecurity, the essential problem is to prove that something is impossible. (You could call it "the impossible dream", i.e. not dreaming about achieving something impossible, but rather dreaming that we can make something not wanted impossible). This is, in general, difficult. However, here is a simple example of how deductive reasoning can easily achieve it.

Suppose I ask you to draw a right-angle triangle, on a flat surface, with sides of length 3, 4, and 6. (Yes: 6). The right answer to such a request is to say: "No, I can't do that. It is impossible.". You can show this by deductive reasoning. But the statement is about a real-world event: drawing a triangle. This has nothing to do with the precision of the measurements. The inaccuracy can be quantified. Even an approximately 3,4,6 right-triangle is impossible. There are events, even with approximate measurements, which can be proved to be impossible. Granted, the impossibility of non-pythagorean triangles doesn't seem to help a lot in cybersecurity, but what if we can entangle real world events in mathematically precise statements in such a way that the events themselves become impossible?

We can do this, and already do it, using similar reasoning to the above, and in this way we are able to conclude that certain cybersecurity events can't happen: for example, this happens when certificates are used, and digital signatures. Conventional wisdom, from the background of science, statistics, and experimental methods, strongly suggests that certainty of this sort is impossible. But the triangle shows that the problem is not with deducing impossibility. Deducing impossibility is possible! What is not clear is whether we can systematically employ such deductions for useful cybersecurity purposes, like preventing servers ferom being hacked.

In fact, we are already doing this, and there will be a lot more such deductions in the near future.

II. Problem Statement

Many problems in cybersecurity can be expressed as the requirement that certain events, or outcomes, should be *impossible*. For example, it is often an objective of an organisation with clients (or customers), that client data is not accessible except when used for the purpose of clients. In brief, invalid access to client data should be *impossible*.

The problem we tackle in this paper is:

How can we systematically, and rigorously, ensure, that certain outcomes are impossible?

III. Literature Review

A. The Importance of Security Policies

According to [4] the security reasoning can be needed according to the needed security environment. Not all organisations have developed their own version of security policies [7]. More research still needed to distinguish what could be a good security policies.The measure of what could be a good security policies should be state clearly.

B. Policies Development

Policies development it has to be efficient for the organisation security requirements. The security requirements should be reflected in the designed security policies.

C. Security breaches recent examples

Below some of the examples of the recent security breaches

1) Optus Security Breach: Security policy take pivotal position in providing the needed data security. Optus security breach in September 2022 was one of the very well known of. Around 10 millions data has been exposed to the attachers. Maintaining and developing the suitable policy may control the security level against such breach [1][2]. According to [9][3] an authenticated API was the breach used by the attacker. A testing API was used without authentication opened the way for millions of records belong to the company customers.

IDOR (Insecure Direct Object References) vulnerability can be considered as a big security breach if combined with an unauthenticated API. The an authenticated API from Optus was attacked using IDOR attack [8].

2) Medibank Security Breach: Optus security breach is not the fist and nor the last with this huge security impact national wide in Australia [6][5]. Medibank which is the largest health insurer in the country been under heavy attack of data security breach. In October 2022 Medibank security breach resulted exposing 9.7 millions of data to the attacker. The available security policy didn't help to prevent such attack [2].

IV. Examples

Turkey Dilemma

There is a concept that can be assumed here. People normally keep feeding a Turkey to be eaten later at a festival or party. The viewpoint of Turkey is that those people are very kind and generous, but in fact, Turkey still doesn't know the waiting destiny. We can borrow this example for what can be considered tested security solutions. Thus, logical and mathematical prove still required.

A. Certificate Validation

1) Objectives: The objectives of a system for validating certificates are:

CO-1 The system reports as valid, certificates that are valid;
CO-2 The system reports as invalid, certificates that are invalid;

2) Validation by Testing: If the software implementing the certificate validation is treated as a black box, and no constraints whatsoever are placed on how it is implemented, no amount of testing can effect a proof of any property of the system. It might even be the case that the system behaves differently at different times.

This is, in effect, the black swan problem, in the context of cybersecurity.

3) Validation by Testing of Logically Constrained Software: Now suppose the software is not a black box, but instead, there are certain valid rules that apply to its operation. In this case, it might be the case that a small amount of additional testing can ensure that the service meets one or more of the objectives. Consider, in particular, CO-1, and let us suppose that …

4) Impossibility of Certificate Fraud: Objectives may not be always able to be tested. The impossible of creating a fake certificated may cannot be completely tested. Such objective need to be proven logically and mathematically. Proving CO-2 is not testable and fully experimental. Thus logical proof will be required.

5) Formal Objectives: For CO-1 the predicated based formal objective may be:

$$\forall\ \mathrm{IsValid}(\mathit{certificate}) \supset \mathrm{system}\ \mathbf{Can}\ \mathrm{Report}(\mathrm{IsValid}(\mathit{certificate})) \tag{1}$$

For CO-2 the predicated based formal objective may be:

$$\forall\neg\ \mathrm{IsValid}(\mathit{certificate}) \supset \mathrm{system}\ \mathbf{Can}\ \mathrm{Report}(\neg\ \mathrm{IsValid}(\mathit{certificate})) \tag{2}$$

B. Access to Client Data

1) Objectives: The objectives of a system for controlling access to client data are:

DO-1 The system allows clients to update their own data;
DO-2 The system prevents anyone except the client themselves from updating their data.

2) Validation by Testing: As in the case of certificate validation, if the software implementing the certificate validation is treated as a black box, and no constraints whatsoever are placed on how it is implemented, no amount of testing can effect a proof of any property of the system.

3) Validation by Testing of Logically Constrained Software:

4) Impossibility of Alteration of Client Data by someone other than the client:

C. Prevention of SQL Injection

An attacker can modify SQL statements that are being executed by a web application by using a sort of attack known as SQL injection. Attacks of this nature may result in system compromise or even unauthorised access to sensitive data. The following actions can be taken to avoid SQL injection:

- Utilize parameterized queries: Sanitized values are substituted for placeholders used for user input in parameterized queries before the SQL query is executed. As a result, user input cannot be used to execute SQL code.
- Verify that user input adheres to the desired format and length by validating it. For instance, if a field is supposed to include a date, make sure the input follows the format for dates and isn't lengthier than necessary.
- Use input sanitization: Eliminate any special characters or appropriately escape them to stop them from being seen as SQL code. You can do this by creating your own code to sanitise input or by using a library.
- Restrict permissions: Make sure database users only have the rights they need to carry out their duties. This restricts the potential SQL injection attack's range.
- Use stored procedures: Stored procedures are precompiled SQL statements that an application can call. They are safer since they don't disclose critical information or permit dynamic SQL execution.
- Maintain software updates: Update your database and application software regularly to guarantee that all known security holes are fixed.

V. Conclusion

References

[1] Nicholas Biddle, Matthew Gray, and Steven McEachern. Data trust and data privacy: A brake on the data and digital dividend? *Centre for Social Research and Methods*, 2022.

[2] Nicholas Biddle, Matthew Gray, and Steven McEachern. Public exposure and responses to data breaches in australia: October 2022. *Centre for Social Research and Methods*, 2022.

[3] John Davidson. All optus customers can do is hope, 2022. https://www.afr.com/technology/all-optus-customers-can-do-is-hope-20220925-p5bku9.

[4] Janice Glasgow, Glenn MacEwen, and Prakash Panangaden. A Logic for Reasoning About Security. *ACM Transactions on Computer Systems (TOCS)*, 10(3):226–264, 1992.

[5] Maurice Blackburn Lawyers. Medibank data breach investigation and complaint, 2023. https://www.mauriceblackburn.com.au/class-actions/join-a-class-action/medibank-data-breach/.

[6] Medibank. Cyber event timeline, 2022. https://www.medibank.com.au/health-insurance/info/cyber-security/timeline/.

[7] Hanna Paananen, Michael Lapke, and Mikko Siponen. State of the art in information security policy development. *Computers & Security*, 88:101608, 2020.

[8] James Piskorz. Optus api hack analysis, 2022. https://terem.tech/optus-api-hack-analysis/.

[9] Paul Smith. Inside the optus hack that woke up australia, 2022. https://www.afr.com/technology/inside-the-optus-hack-that-woke-up-australia-20221123-p5c0lm.