On bugs and elephants: Mining for science of security

Dusko Pavlovic
Royal Holloway, Oxford and Twente
dusko.pavlovic@rhul.ac.uk

1 On security engineering

> *A number of blind men came to an elephant. Somebody told them that it was an elephant. The blind men asked, "What is the elephant like?" and they began to touch its body. One of them said: "It is like a pillar." This blind man had only touched its leg. Another man said, "The elephant is like a husking basket." This person had only touched its ears. Similarly, he who touched its trunk or its belly talked of it differently.*
>
> Ramakrishna Paramahamsa

Security means many things to many people. For a software engineer, it often means that there are no buffer overflows or dangling pointers in the code. For a cryptographer, it means that any successful attack on the cypher can be reduced to an algorithm for computing discrete logarithms, or to integer factorization. For a diplomat, security means that the enemy cannot read the confidential messages. For a credit card operator, it means that the total costs of the fraudulent transactions and of the measures to prevent them are low, relative to the revenue. For a bee, security means that no intruder into the beehive will escape her sting...

Is it an accident that all these different ideas go under the same name? What do they really have in common? They are studied in different sciences, ranging from computer science to biology, by a wide variety of different methods. Would it be useful to study them together?

1.1 What is security engineering?

If all avatars of security have one thing in common, it is surely the idea that *there are enemies and potential attackers out there*. All security concerns, from computation to politics and biology, come down to averting the adversarial processes in the Environment, that are poised to subvert the goals of the System. There are, for instance, many kinds of bugs in software, but only those that the hackers use are a security concern.

In all engineering disciplines, the System guarantees a functionality, provided that the Environment satisfies some assumptions. This is the standard *assume-guarantee* format of the engineering correctness statements. Such statements are useful when the Environment is passive, so that the assumptions about it remain valid for a while. The essence of security engineering is that the Environment actively seeks to invalidate System's assumptions.

Security is thus an *adversarial process*. In all engineering disciplines, failures usually arise from some engineering errors. In security, failures arise *in spite* of the compliance with the best engineering practices of the moment. Failures are the first class citizens of security. For all major software systems, we normally expect security updates, which usually arise from attacks, and often inspire them.

1.2 Where did security engineering come from?

The earliest examples of security technologies are found among the earliest documents of civilization. Fig. 1 shows security tokens with a tamper protection technology from almost 6000 years ago. Fig.2 depicts the situation where this technology was probably used. Alice has a lamb and Bob has built a secure vault, perhaps with multiple security levels, spacious enough to store both Bob's and Alice's assets. For each of Alice's assets deposited in the vault, Bob issues a clay token, with an inscription identifying the asset. Alice's tokens are then encased into a *bulla*, a round, hollow "envelope" of clay, which is then baked to prevent tampering. When she wants to withdraw her deposits, Alice submits her bulla to Bob, he breaks it, extracts the tokens, and returns the goods. Alice can also give her bulla to Carol, who can also submit it to Bob, to withdraw the goods, or pass on to Dave. Bullæ can thus be traded, and they facilitate exchange economy. The tokens used in the bullæ evolved into the earliest forms of money, and the inscriptions on them led to the earliest numeral systems, as well as to Sumerian cuneiform script, which was one of the earliest alphabets. Security thus predates literature, science, mathematics, and even money.

Figure 1: Tamper protection from 3700 BC: Bulla with tokens from Uruq (Irak)

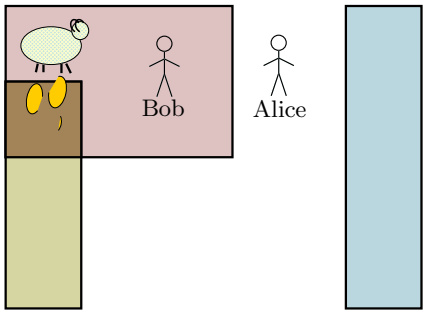

Figure 2: To withdraw her sheep from Bob's secure vault, Alice submits a tamper-proof token, like on Figure 1.

1.3 Where is security engineering going?

Through history, security technologies evolved gradually, serving the purposes of war and peace, protecting public resources and private property. As computers pervaded all aspects of social life, security became interlaced with computation, and security engineering came to be closely related with computer science. The developments in the realm of security are nowadays inseparable from the developments in the realm of computation. The most notable such development is, of course, *cyber space*.

A brief history of cyber space

In the beginning, engineers built computers, and wrote programs to control computations. The platform of computation was the computer, and it was used to execute algorithms and calculations, allowing people to discover, e.g., fractals, and to invent compilers, that allowed them to write and execute more algorithms and more calculations more efficiently. Then the operating system became the platform of computation, and software was developed on top of it. The era of personal computing and enterprise software broke out. And then the Internet happened, followed by cellular networks, and wireless networks, and ad hoc networks, and mixed networks. Cyber space emerged as the distance-free space of instant, costless communication. Nowadays software is developed to run in cyberspace. The Web is, strictly speaking, just a software system,

albeit a formidable one. A botnet is also a software system. As social space blends with cyber space, many social (business, collaborative) processes can be usefully construed as software systems, that ran on social networks as hardware. Many social and computational processes become inextricable. Table 1 summarizes the crude picture of the paradigm shifts which led to this remarkable situation.

age	*ancient times*	*middle ages*	*modern times*
platform	computer	operating system	network
applications	Quicksort, compilers	MS Word, Oracle	WWW, botnets
requirements	correctness, termination	liveness, safety	trust, privacy
tools	programming languages	specification languages	scripting languages

Table 1: Paradigms of computation

But as every person got connected to a computer, and every computer to a network, and every network to a network of networks, computation became interlaced with communication, and ceased to be programmable. The functioning of the Web and of web applications is not determined by the code in the same sense as in a traditional software system: after all, web applications do include the human users as a part of their runtime. The fusion of social and computational processes in cyber-social space leads to a new type of information processing, where the purposeful program executions at the network nodes are supplemented by spontaneous data-driven evolution of network links. While the network emerges as the new computer, data and metadata become inseparable, and a new type of security problems arises.

A brief history of cyber security

In early computer systems, security tasks mainly concerned sharing of the computing resources. In computer networks, security goals expanded to include information protection. Both computer security and information security essentially depend on a clear distinction between the secure areas, and the insecure areas, separated by a security perimeter. Security engineering caters for computer security and for information security by providing the tools to build the security perimeter. In cyber space, the secure areas are separated from the insecure areas by the "walls" of cryptography; and they are connected by the "gates" of cryptographic protocols.

But as networks of computers and devices spread through physical and social spaces, the distinctions between the secure and the insecure areas become blurred. And in such areas of cyber-social space, where

age	*middle ages*	*modern times*	*postmodern times*
space	computer center	cyber space	cyber-social space
assets	computing resources	information	public and private resources
requirements	availability, authorization	integrity, confidentiality	trust, privacy
tools	locks, tokens, passwords	cryptography, protocols	mining and classification

Table 2: Paradigms of security

information processing does not yield to programming, and cannot be secured by cryptography and protocols, security cannot be assured by the engineering methodologies alone. The methodologies of data mining and classification, needed to secure such areas, form a bridge from information science to a putative security science, which is discussed in the next section.

2 On security science

> *It is the aim of the natural scientist to discover mathematical theories, formally expressed as predicates describing the relevant observations that can be made of some [natural] system. [...] The aim of an engineer is complementary to that of the scientist. He starts with a specification, formally expressible as a predicate describing the desired observable behaviour. Then [...] he must design and construct a product that meets that specification.*
>
> Tony Hoare

This was the first paragraph of one in the first papers on formal methods for software engineering, published under the title *"Programs are predicates"*. Following this slogan, software has been formalized by logical methods, and viewed as an engineering task ever since. But computation evolved, permeated all aspects of social life, and came to include not just the purposeful program executions, but also the spontaneously evolving network processes. Data and metadata processing became inseparable. In cyber space, computations are not localized at network nodes, but also propagate with non-local data flows, and with the evolution of network links. While the local computations remain the subject of software engineering, network processes are also studied in the emerging software and information sciences, where the experimental validation of mathematical models has become the order of the day. Modern software engineering is therefore coupled with an empiric software science, as depicted on Fig. 3. In a similar way, modern security engineering needs to be coupled with an empiric security science.

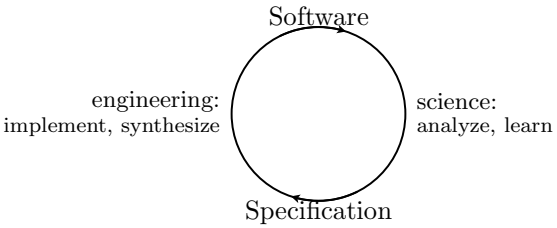

Figure 3: Conceptualization loop: The life cycle of computation

2.1 Why security science?

Security science is needed to secure those processes that do not yield to the static security distinctions between the insiders and the outsiders. Security engineering cannot get going without such distinctions: it requires a clear boundary between the System and the Environment. This boundary is blurred when cyber, physical and social spaces are conjoined by networks. The task of security science is to reconstruct the boundary, and to re-enable security engineering.

Let us have a closer look at the paradigm shift to postmodern cyber security in Table 2. It can be illustrated as the shift from Fig. 4 to Fig. 5. The fortress in Fig. 4 represents the static, architectural view of security. A fortress consists of walls and gates, separating the secure area within from the insecure area outside. The boundary between these two areas is the security perimeter. The secure area may be further subdivided into areas of higher security and areas of lower security. In cyber space, as we mentioned, the walls are realized using crypto systems, whereas the gates are authentication protocols. But as every fortress owner knows, the walls and the gates are not enough for security: you also need weapons, soldiers, and maybe even some detectives and judges. They take care of the dynamic aspects of security. Dynamic security evolves through social processes, such as trust, privacy, reputation, or influence. The static and the dynamic aspects depend on each other. E.g., the authentication on the gates is based on some credentials, intended to prove that

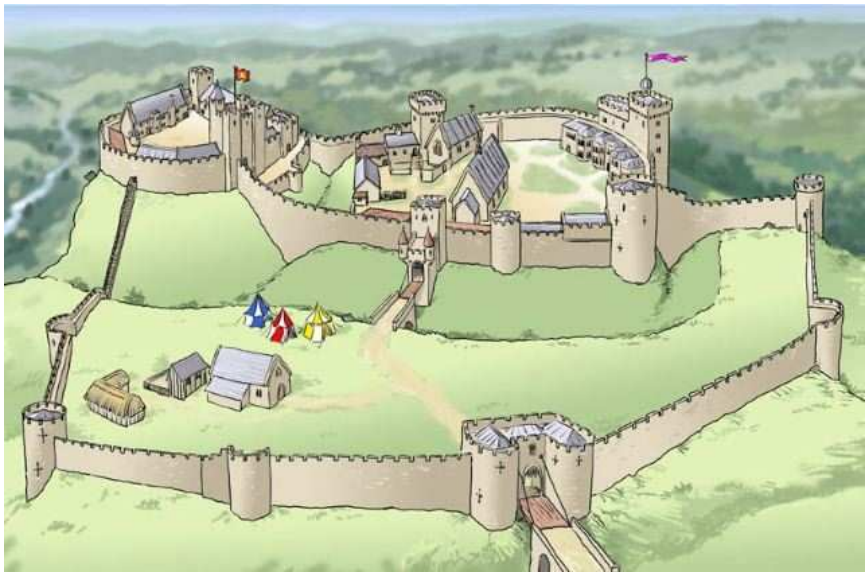

Figure 4: Static security: Multi-level architecture

the owner is honest. The credentials may be based on some older credentials; but down the line, a first credential must have resulted from a process of trust building, or from a trust decision, whereby principal's honesty was accepted with no credentials. The word "credential" has its root in Latin *"credo"*, which means "I believe".

The attacks mostly studied in security research can be roughly divided into cryptanalytic attacks and protocol attacks. They are the cyber versions of the simple frontal attacks on the walls and the gates of a fortress. Such attacks are static, in the sense that the attackers are outside, the defenders inside, and they are easily distinguished. The dynamic attacks come about when some attackers penetrate the security perimeter, and attack from within, like on Fig. 5. They may even blend with the defenders, and become spies. Some of them may build up trust and infiltrate the fortress earlier, and wait as moles. Some of the insiders may defect, and become attackers. The traitors and the spies are the *dynamic attackers*: they use the vulnerabilities in the process of trust. To deter them, all cultures reserve for the breaches of trust the harshest punishments imaginable, and Dante, in his description of Hell, places the traitors into the deepest, Ninth Circle. As a dynamic attack, treason was always much easier to punish than to prevent.

In cyber security, a brand new line of defense against dynamic attacks is opened by *predictive analytics*, based on mining the publicly available network data. But predictive modeling does not fit in the toolkit of the security engineering methodologies. It is a tool of security science.

2.2 What is security science?

Although the security Environment maliciously defies any System's assumptions that it can defy, security engineering still pursues its tasks strictly within the framework of the *assume-guarantee* methods. This is the only thing that an engineering discipline can do. E.g., the cryptographic techniques of security engineering are based on the fixed assumption that the Environment is computationally limited, and cannot solve certain hard problems.[1]

But sometimes, as we have seen, it is not realistic to assume even that there is a clear boundary between the

[1] Defy that, Environment!

Figure 5: Security dynamics: Threats within

System and the Environment. Such situations have become pervasive with the spread of networks, supporting not only social, commercial and collaborative applications, but also criminal and terrorist organizations. When there is a lot going on, you cannot be sure who is who. In large networks, with immense numbers of processes, the distinction between the System and the Environment becomes meaningless, and the engineering *assume-guarantee* approach must be supplemented by the *analyze-adapt* approach of science, whose task is to recover the distinction, whenever possible, albeit as a dynamic variable, and to adaptively follow its evolution. Similar situations, where engineering interventions are interleaved with scientific analyses arise not only in security, where they elicit security science to support security engineering, but also, e.g., in the context of health, where they elicit medical science to support health care. And just like health is not achieved by isolating the body from the external world, but by supporting its internal defense mechanisms, security is not achieved by erecting fortresses, but by supporting dynamic defenses, akin to the immune response. While security engineering provides blueprints and materials for the static defenses, it is the task of security science to provide guidance and adaptation methods for the dynamic defenses.

In general, science is the process of understanding the Environment, adapting the System to it, changing the Environment by the System, adapting to these changes, and so on. Science is thus an ongoing dialog of the System and the Environment, separated and conjoined along the ever changing boundaries. Dynamic security, on the other hand, is an ongoing battle between the ever changing teams of attackers and defenders. Only scientific probing and analyses of this battle can tell who is who at any particular moment.

In summary, if security engineering is a family of methods to keep the attackers out, security science is a family of methods to catch the attackers once they get in.

It may be interesting to note that these two families of methods, viewed as strategies in an abstract security game, turn out to have opposite winning odds. It is often observed that the attackers only need to find one attack vector to enter the fortress, whereas the defenders must defend all attack vectors to prevent them. But when the battle switches to the dynamic mode, and the defense moves inside, then the defenders only need to find one marker to recognize and catch the attackers, whereas the attackers must cover all their markers. This strategic advantage is also the critical aspect of the immune response, where the invading

organisms are purposely sampled and analyzed for chemical markers. In security science, this sampling and analyses take the form of data mining.

2.3 Where to look for security science?

The germs of a scientific approach to security, with data gathering, statistical analyses, and experimental validation, are already present in many intrusion detection and anti-virus systems, as well as in spam filters and some firewalls. Such systems use measurable inputs, have a quantifiable performance and model accuracy, and thus conform to the basic requirements of the scientific method. The collaborative processes for sharing data, comparing models, and re-testing and unifying results complete the social process of scientific research.

However, a broader range of deep security problems is still awaiting applications of a broader range of powerful scientific methods that are available in this realm. At least initially, the statistical methods of security science will need to be borrowed from information science. Security, however, imposes special data analysis requirements, some of which have been investigated in the existing work, and led to novel approaches. In the long run, security science will undoubtedly engender its own, domain specific data analysis methods.

In general, security engineering solutions are based on security infrastructure: the IPSec suites, RSA systems, ECC cryptography provide typical examples. In contrast, security science solutions emerge where the available infrastructure does not suffice for security. The examples abound. E.g., a Mobile Ad-hoc NETwork (MANET) is a network of nodes with no previous contacts, direct or indirect, and thus no previous infrastructure. Although advanced MANET technologies have been available for more than 15 years, *secure* MANETs are still a bit of a Holy Grail. Device pairing, social network and web commerce security also require secure ad-hoc interactions, akin to the social protocols that regulate new encounters in social space. Such protocols are invariably incremental, accumulating, analyzing and classifying the data from multiple channels, until a new link is established, or aborted. Powerful data mining methods have been developed and deployed in web commerce and financial security, but they are still awaiting systematic studies in non-commercial security research, and systematic applications in non-commercial security domains.