Point-and-Shoot Security Design: Can We Build Better Tools for Developers?

Sven Türpe
Fraunhofer Institute for Secure Information Technology (SIT)
Rheinstraße 75
64295 Darmstadt, Germany
sven.tuerpe@sit.fraunhofer.de

ABSTRACT

Security property degrees systematize the angles from which one can discuss the security of a system. Microscopic properties characterize how specific actions affect parts of a system. Mesoscopic properties describe how the pursuit of an attack objective may affect the system and the attacker. Macroscopic properties represent the interaction of a threat environment with a system. Properties of different degrees are interdependent, but not in a simple and universal manner.

Security design aims to control security properties, shaping them in a favorable way. Its objective is macroscopic control through design decisions on all three degrees. Design tools today occupy mostly the lower half of the property degree scale. A few macroscopic design aids exist but provide little guidance to engineers.

Security designers are thus in a similar situation as photographers, having to make fundamental design decisions without methodologies other than their private, homegrown approaches. This is essential for art but a deficiency in engineering. Standardized mechanization in point-and-shoot cameras helps inexpert photographers to a limited extent but can get in the way of the experienced and ambitious. Point-and-shoot security design, shorthand for current practice as well as a widely held expectation, may do the same to security engineers.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; D.2.0 [**Software Engineering**]: General—*Protection mechanisms*; D.2.2 [**Software Engineering**]: Design Tools and Techniques—*Computer-aided software engineering (CASE)*; D.2.1 [**Software Engineering**]: Requirements/Specifications—*Tools*

General Terms

Design, Security

NSPW'12, September 18–21, 2012, Bertinoro, Italy.

Keywords

Security engineering, security properties, property degree, macroscopic security, systematization, security tools, abstraction, security model, adversary, threat, philosophy, epistemology

1. INTRODUCTION

1.1 Science and Practice of Photography – a Parody

Year after year, computer scientists at the Metropolis Institute of Photography churn out reams of research papers, published in eminently respectable venues. A typical paper laments the state of affairs in photography, illustrates the point using widely known failures, and goes on developing a promising new approach. Using advanced image processing calculi, a novel way of analyzing, constructing, proving, or verifying photographic properties is proposed. The science revolves around the triad of hue, saturation, and value, sometimes amended with non-blurredness, properties concordantly believed to constitute the essence of photography. The scientists demonstrate their approach using a digital photo, 10 by 15 pixels large, of their lab coats. They conclude their results promise significant improvements of photography after further research.

Meanwhile, Macroshot Corporation makes a fortune selling shrink-wrapped stock photo collections. Macroshot's reputation had suffered a bit when the Internet came about. Their lomographies attracted lots of negative reviews online, whereas formerly their customers had silently discarded poor images and kept only the good ones. Under the pressure of criticism and competition, Macroshot introduced the Appealing Photo Life Cycle (MS APL), a collection of practices for better photography. Following APL, photographers apply the FACE (Focus, Abstraction, Composition, Exposure) checklist early in a project; the checklist generalizes how critics review a photo. All photos are thoroughly tested prior to release. Each must pass a discussion of drunk Internet trolls and the review by a critic wearing fuzzer goggles that simulate grandma's declining vision. The APL has become industry best practice. Nevertheless, released photo collections continue to require monthly patches replacing faulty images.

Three days after an investigative journalist accused the Cybersecurity Secretary of corruption, a blurry surveillance camera still appears on the title pages of newspapers throughout the country. A poor image in every conceivable respect, neither photo scientists nor industry practitioners would ever

dare calling it a photo. Barely recognizable, the image is nonetheless evidence of the Secretary's clandestine meeting with a lobbyist in the parking lot. The Secretary resigns at 2:15 p.m. On page 17 in one of the newspapers, an art gallery advertises a vernissage. The upcoming exhibition will feature photographs taken with pinhole cameras.

1.2 Wait a Moment, Photography?

Our understanding of security design as a discipline and practice remains inferior compared to other fields of creative endeavor. Security design is the methodical creation of practically secure information technology – systems, software, appliances, or infrastructures. Generalizing observations from the domain of photography and applying the resulting model to security engineering, this paper explores what is missing or less developed than it should be.

Photography and security engineering have little in common, except for the facts that in both, humans create – visual art in one field, secure systems in the other – and the results vary in quality. Developing a conception in one domain and transferring it into the other will result either in utter nonsense or in a functional analogy; this paper aims for the latter. Even a proper metaphor has nonetheless limitations, mapping only a subset of the source domain concepts and relationships meaningfully into the destination domain [35].

Despite all artistic freedom, photography is a skill that can be learned and taught to some degree. We expect the same of security design, which should in addition be methodical and repeatable. If we defined quality in photography exclusively as technical perfection – perfect sharpness, exposure, or perspective control – we could not explain how great photos can come out of a pinhole camera and poor ones out of expensive equipment, or how one perfect photographer creates expressive portraits while another shoots appealing landscapes.

When it comes to art and esthetics, we don't need an explanation. In engineering we do, or we would have to submit to luck and fate. If we define security exclusively as technical perfection in the design and arrangement of security mechanisms and in the prevention of defects, we cannot explain why some systems survive well despite known vulnerabilities, while others continue to be attacked despite all effort to secure them. As security engineers we cannot afford to rely on unexplained magic.

1.3 Security Design is Property Control

What does it mean to design a secure system? How can we effectively support developers in their security engineering work? Which security design tasks should we automate, which ones should we better leave to engineers, and which tools support security problem solving?

In the most general sense, security design means to control the security properties of an artifact in the process of creating it. This paper:

1. Establishes in Section 2 the notion of security property degrees. Microscopic properties characterize the impact of specific actions on parts of a system. Mesoscopic properties characterize how a system constrains attackers pursuing an objective. Macroscopic properties characterize the behavior of a threat environment in the presence of a system. Figure 1 outlines this idea.

2. Construes security design as property control in all three degrees in Section 3. To design a practically secure system, security engineers optimize lower-degree properties towards macroscopic objectives, exploring a design space for appropriate solutions.

3. Discusses security design tools in the framework of security property degrees, and identifies missing macroscopic and mesoscopic tools (Section 4). Our current design toolbox is too focused on microscopic work, making security design more of an art than an engineering discipline.

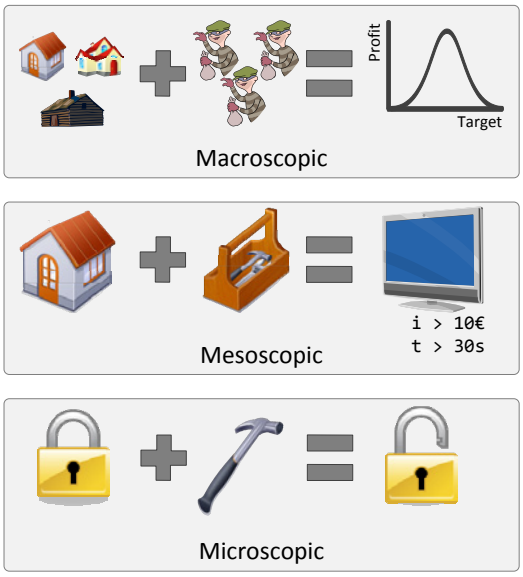

Figure 1: Security property degrees in a nutshell: microscopic properties (bottom) characterize parts under specific actions, mesoscopic properties (middle) characterize systems under attack, and macroscopic properties (top) characterize threat environments in the presence of systems.

Section 5 outlines an analogous model of photography from which the ideas in this paper were developed. Highly automated point-and-shoot cameras do not make photographers obsolete. Likewise, point-and-shoot security design tools solve only some of our problems. We need to address the macroscopic part of security design, too.

2. PROPERTY DEGREES

Properties characterize objects or entities; properties can in turn have second-order properties [65]. We introduce the *property degree* as a second-order property. The degree of a property characterizes the practical expressiveness of the language required to specify how to determine the value of the property. Property degrees form a continuum from microscopic to macroscopic:

- Microscopic properties can be expressed and evaluated locally, considering only parts or elements of an entity in isolation from any context.

- Macroscopic properties require expressions beyond the entity itself to evaluate them; their value may vary for the very same entity depending on the context in which we evaluate the property.

- Mesoscopic properties fill the range between these two extremes. Idealized, mesoscopic properties require expressions about the entity as a whole, but given a continuum of degrees this is just a tendency.

To simplify the discussion that follows, we treat property degrees as three distinct classes rather than as a continuum.

2.1 Security Properties of a System

Security properties of a system characterize how adversaries can affect the system's operations and vice versa. Adversaries have a range of intents or objectives, conditions they would like to attain; they have capabilities, subject to constraints in the environment such as availability of resources or the laws of nature; and their behavior is adaptive. Security properties describe what will happen if a system meets its adversaries.

Figure 2: Two different coin designs: euro coins (left) and embossed plastic chips (right).

We can think of security properties in different ways. Before discussing the differences, let's look as an example at two different designs of coins (Figure 2) and the payment systems they are part of. Coins serve as standardized tokens representing value; they are part of a system managing their lifecycle and supporting their use in transactions. Implementations range from simple plastic chips, used e.g. as consumption tokens at festivals, to supranational common currencies like the euro. A well-known security threat for coin-based payment is counterfeiting. Depending on the property degree considered, we get different descriptions and comparisons of the two systems with respect to security. Table 1 summarizes the discussion below.

2.1.1 Microscopic View

The strength of coin authentication characterizes the security of a payment system. During a transaction the recipient uses security features built into the coin to verify its authenticity and reject counterfeits. The more effort it takes to counterfeit coins that pass authentication, the more secure the payment system becomes.

Euro coins are made of special alloys and have design features difficult to replicate without original equipment. A counterfeit has to meet the appearance, weight, and metallic properties to pass a coin detector as genuine. Plastic consumption tokens have no such security features. Their authentication remains limited to appearance, which is easy to reproduce using common materials.

To keep the payment system secure it is important that coins are properly authenticated before accepting them. Plastic coins should not be used for payments since they do not provide strong authentication. Euro coins should be run through an electronic coin detector before accepting them.

Table 1: Selected security properties of two coin-based payment systems by property degree.

View	Euro Coins	Plastic Tokens
Macroscopic	Small fraction of circulating coins are counterfeit	Apparently few fraud incidents, low losses
Mesoscopic	Counterfeiting requires some effort; Soft limit on transaction size; Limited accumulation of counterfeits	Tight limits on transaction type, size, time, and location
Microscopic	Bimetal design; Special alloys; Visible features	No security features; Buy online or 3D-print

2.1.2 Mesoscopic View

The set of viable attacks by financially motivated adversary characterize the security of a payment system. An attack is a course of action with consequences for the system and the attacker. It is viable if executing the attack is technically feasible and, with sufficient likelihood, profitable. The system's design determines the attacks an adversary can execute and their consequences.

Counterfeiting is possible in both payment systems discussed here, yet some of their security properties differ. Euro coins require a higher investment, per attack as well as per coin, than plastic tokens to produce the same amount of usable counterfeits. In both systems, authentication of coins remains superficial in most transactions, such that counterfeits are detected only by chance.

Counterfeit euro coins can be used for arbitrary transactions, anywhere in several countries. Counterfeit consumption tokens can be used to buy food and drinks for the duration and within the premises of a festival using them. A limited amount can be changed into common currency without raising suspicion. Adversaries' profits are thus capped to a much lower amount in the plastic token than in the euro system. The plastic token system also has limited exposure.

The euro system removes counterfeits from circulation systematically. Regardless of weak authentication in everyday transactions, circulating coins eventually come by a point where they undergo a thorough check. The plastic token system may do the same but less systematically; counterfeiting attacks will be detected only by chance or when the system breaks down.

2.1.3 Macroscopic View

The expected or, after the fact, observed incident profile in an environment characterizes the security of a payment system. A payment system attracts financially motivated

threats with criminal energy, from individuals to organized crime. Some amount of counterfeiting activity may result.

For the euro system we have data on counterfeiting. 157,000 counterfeit coins were removed from circulation in 2011, one counterfeit per 100,000 genuine coins circulating [21]. The number of counterfeit banknotes has the same order of magnitude [20] but corresponds with a larger sum of money. No data seem to be available on counterfeiting in plastic token payment systems. The lack of incident reports and their continual use suggest that counterfeiting, if at all, occurs only at a small scale.

Both systems attract the same threats, and they compete with each other in doing so. For adversaries seriously considering payment systems as a target and counterfeiting as an attack technique, the euro system seems much more attractive. It is more accessible, giving the adversary flexibility, and promises higher profits. Given this factor, contrary to what the microscopic view suggests, the plastic token system seems equally secure in the threat environment.

2.2 Characteristics of Property Degrees

As we have seen, we can discuss security in microscopic, mesoscopic, or macroscopic terms, leading to different statements and conclusions about the same systems. What distinguishes the three property degrees from each other? While a precise definition requires further research, some tendencies become obvious from the example above. Table 2 outlines them.

2.2.1 Scope of Evaluation

To determine microscopic security properties one needs to consider only individual parts of a system, such as components, subsystems, interfaces, or processes. Mesoscopic properties require that we evaluate the entire system; a particular, identifiable part may create a mesoscopic property in a system, but we cannot determine whether the system has the property by looking at that part in isolation. Macroscopic properties extend the scope even further, requiring that we evaluate the supersystem – e.g. an ecosystem – in which a system interacts with threats and adversaries.

2.2.2 Threat Models

The type of threat model considered corresponds with the scope of property evaluation. A threat model serves as a frame of reference within which one defines and evaluates security. Microscopic threat models assume specific attacker actions as relevant and important. The better a security function resists these actions, the more secure it becomes in this view. Mesoscopic threat models represent adversary objectives, such as the objective of gaining control over certain assets. An attacker may pursue an objective in different ways; the threat model thus represents a range of specific attacks. Macroscopic threat models represent a threat environment, in which several threat communities and competing targets exist. This type of threat model allows a system to be secure simply by being unattractive for adversaries.

2.2.3 Quality of Expressions

Microscopic properties express what can or cannot happen under certain assumptions, or which assumptions one needs to make in order to deem something possible or impossible. Mesoscopic properties express the ramifications of adversarial activity, the impact on a system and on the adversary as a function of attack properties. Macroscopic properties express the results of exposing a system to threats. Microscopic expressions of security properties tend towards logic, mesoscopic expressions towards functional relationships, and macroscopic expressions towards probabilistic or statistical statements.

2.2.4 System Security Failure Modes

We might say the microscopic view deals with security functions and security properties of system functions, the mesoscopic view with security architecture, and the macroscopic view with effective security. A system fails microscopically if it has security defects. It fails mesoscopically if it has an inappropriate security design. The system fails macroscopically if it gets attacked with unacceptable consequences.

2.2.5 Invalidation of Security Properties

A formally correct security analysis – a description of a system by its security properties – is nonetheless wrong if it is based on false assumptions. A microscopic analysis is rendered useless by an adversary choosing an attack outside the assumptions made. The pursuit of unexpected objectives invalidates a mesoscopic security analysis. Macroscopic security properties become obsolete if the threat environment changes in unforseen ways.

2.3 Property Degrees Are Not Abstractions

Higher-degree properties are not abstractions of lower-degree properties. To abstract means to omit unnecessary detail and retain the essence of something [30]. To move from a microscopic to a mesoscopic and further up to a macroscopic view, we must add information and reevaluate what we already know. It seems therefore more appropriate to think of abstraction as orthogonal to property degrees.

Abstractions of properties are properties, too. Abstraction maintains the property degree; an abstraction of properties has the same degree as the properties it is derived from. As a consequence, we cannot draw conclusions about security properties of one degree from properties of another degree without understanding the mapping between the two. There is no universal mapping that would apply to the same system in all environments, or to all systems in the same environment.

Our inability to find meaningful definitions of *almost* or *sufficiently* secure is a symptom indicating that we often do not understand the dependencies between property degrees. Intuitively, approximation should work for macroscopic properties, considering their probabilistic nature. But quantifying security remains difficult [67] despite earlier promising attempts [59], and we have to find a way yet to confidently shrug off a microscopic security defect as macroscopically irrelevant.

3. SECURITY DESIGN

Designing an object means to shape the properties that characterize it. Security design thus requires deliberate control over the security properties of a system. The design objective is to shape macroscopic security properties, which can, however, be controlled only indirectly through lower-degree design. Comprehensive security design must therefore encompass considerations and decisions on all three property degrees.

Table 2: Characteristics of microscopic, mesoscopic, and macroscopic security properties.

Degree	Scope	Threat Model	Expressiveness	Definitions
Macroscopic	Supersystem	Populations of adversaries	Incident statistics	Behavioral
Mesoscopic	System	Adaptive attacker pursuing an objective	Bounds of attack impact on system and attacker	Dependent on system design
Microscopic	Parts; Subsystems	Specific actions	Possibility and prerequisites of actions	Context-free

Contemporary security engineering in contrast does not seem to follow this approach. We often try to get away with microscopic considerations, hoping the properties thus created would translate into the desired macroscopic properties.

3.1 Three Degrees of Property Control

3.1.1 Macroscopic Design

Macroscopic design means to constrain the emergent behavior of a threat environment with respect to a system. The result is an acceptable incident profile expressing desired constraints on the type, frequency, and impact of incidents over the life cycle of the system. Macroscopic design objectives are probabilistic: once a system is deployed and being used, it will induce some behavior of the adversary populations that constitute the threat environment. With uncontrolled macroscopic properties this behavior comes as a surprise, we don't know what amount of attacks to expect, what they aim at, and how together they affect the system. Controlled properties give us confidence that, despite all remaining uncertainty about the adversaries, incident and impact statistics likely remain within set constraints.

The threat environment consists of potential adversaries. Their intents and capabilities vary; in security, by definition, we have no direct control over the adversaries. Nonetheless we shape their behavior, as a population, in various ways. Perhaps the most obvious influence we exert simply by creating artifacts that become targets attracting adversaries. Classical security engineering starts there, making a list of assets and devising security functions to protect them. But our most precious assets are not necessarily the biggest attraction for adversaries following their own agenda.

The macroscopic design space extends beyond the technical system; defining the system boundary is part of the design task. A macroscopic security design may for example include the legal system and law enforcement, meaning that the designers either take advantage of what is there, or even advocate changes through campaigning and lobbying. A host of mesoscopic and microscopic design considerations will follow if one goes down that path. More importantly, macroscopic design involves fundamental questions and tradeoffs, such as whether and to what extent a system should be allowed kill or damage human beings.

Ideally we would base macroscopic considerations on a solid scientific foundation of data and theory about threat environments. For now, in default of such a foundation, we have to work with assumptions. If we develop products – software, appliances, etc. – rather than a particular system, macroscopic design needs to take into account the range of systems that can be built using the software, and in some cases the ecosystem that a mass-deployed product will create.

3.1.2 Mesoscopic Design

Mesoscopically we design a complicated function. This function has a system-specific space of attacker objectives as its domain and a space of consequences – for the attacker, the system, users, and other affected parties – as its range. An attacker is one instantiation of a macroscopic adversary, the generic intents translated into specific objectives – but not attack actions – within or outside the system. Pertinent consequences are for instance required effort, expected profit, and risk for the attacker, or losses and other impacts for the system and its stakeholders. The design objective is to constrain this function for all – or at least the most likely – instances of macroscopic adversaries, such that consequences for the system and stakeholders remain within defined boundaries and those for the attacker, outside.

An attacker-consequence function is abstract; it needs a concrete system design and architecture to create and support it. The result of mesoscopic design is therefore a set of candidate architectures capable of approximating the desired function. In a purely mesoscopic view, the function remains parameterized. Its parameters represent microscopic properties an architecture relies on, such as an encryption scheme being as strong as required, users not sharing their passwords inappropriately, or certain types of defects not occurring in the implementation.

The mesoscopic design space encompasses design dimensions like consistency, weakest protection, maximum impact, or maximum net profit. Consistency means for instance consistent protection of an abstract asset across all representations, instances, or access paths. To err safely, one should base the attacker-consequence function on the weakest protection and the maximum impact or attacker net profit.

The design space extends beyond the technology domain, taking into account for instance user behavior and capabilities or security management and incident response. After having made the macroscopic decision to consider the legal system a design dimension, for example, one could go on and engage in liability design [2].

Non-security requirements and unchangeable features, like the capabilities of humanoid subsystems, constrain the designer, making parts of the design space inaccessible. Abstract building blocks represent the underlying microscopic design and are here connected into a whole. Only in this context can we evaluate the contribution of a security function or the possible impact security defect.

3.1.3 Microscopic Design

Microscopic design shapes the security building blocks a system relies on. They can be pre-existing like common encryption schemes, or custom-designed for a particular system. A typical security building block creates a domain within which a security function governs the likelihood of particular events or states regardless of an attacker's at-

tempted actions within the domain. This domain may have a technical representation, like a sandbox or an access control function, or it may be virtual like the domain of entities seeing only the ciphertext of some encrypted data.

Security building blocks often have dangling dependencies and transform, rather than prevent attacks. An encryption scheme, for example, transforms the attack of reading cleartext data into the attack of reading ciphertext data and obtaining a decryption key for the same result (but possibly different side effects). The Java sandbox has multiple dependencies, such as on code signing and user decisions. To resolve these dependencies is a mesoscopic design task. Designers of building blocks only have to document assumptions and dependencies fully and precisely.

One example of microscopic design is the prevention of exploitable defects or functions wherever the mesoscopic design relies on properties of code, hardware, or other constituent elements of a system.

3.2 The State of the Art

The problem of security design is far from solved. The industry continues to crave tools, techniques and processes to build enough security into technology with limited budgets. It seems we have made little progress since, almost 20 years ago, Baskerville [3] identified three generations of security design methods: (1) checklists, (2) mechanistic engineering, and (3) logical transformational methods based on abstract models of the problem and solution space. We are still struggling to make the third generation happen [61, 47, 5] – we do not understand "how ... low-level mechanisms relate to the high-level security goals" [12]. Research in economics of information security and related disciplines may enable us in the long run to do systematic macroscopic security design. To date, however, our systematic security design practices cover only microscopic and a limited extent of mesoscopic design. The following selection of practices we use is representative, not comprehensive.

3.2.1 Building Blocks

Research seems predominantly concerned with technological matters [5] and supplies us with security building blocks: cryptography, protocols, security functions, and so on and so forth. Researchers love formal models and methods, making problems accessible to rigorous mathematical reasoning [68]. Theoretical applications are manifold and not limited to specific problems [69]. Successful practical applications, however, remain scarce [50] and the required abstractions limit their capabilities [54]. Nevertheless, formal methods have a place in the development of building blocks security designers can rely on.

3.2.2 Design Patterns

A small number of solid design patterns [8] have attracted continuous attention in research as well as in engineering practice. Patterns outline common solutions to common problems. Some examples are secure communication, reference monitors and security kernels [1], and security policies. Research, e.g. on access control policies, focuses on idealized versions of such patterns and prototypical problems, whereas practical applications integrate patterns into larger designs. Security patterns as we know them today contribute to mesoscopic design by providing structured and comprehensive documentation of common security building blocks.

There are ongoing efforts to create a structured pattern catalog containing variants of patterns [24]. However, even the most comprehensive catalog of security design patterns is rendered useless by new design constraints. For instance the strategies commonly applied to secure general-purpose personal computers – frequent patching, antivirus software, etc. – are less applicable to industrial control systems.

3.2.3 Development Processes

Security design in the industry focuses on hands-on approaches and on security assurance to build confidence. Security engineering has to blend in with existing development processes and contribute to a vendor's business. Developers need to pay some attention to security without losing sight of other objectives and requirements [44, 45]. A representative approach is Microsoft's Security Development Lifecycle (MS SDL) [31, 46], a collection of security engineering practices across the development process of a software product. MS SDL aims at a security baseline, encouraging developers to apply common security building blocks and helping them to avoid common types of security defects. The predominant practices in development processes are security testing, source code analysis, code review, and design reviews with or without security experts [19]. Process models like the MS SDL and related tools touch mesoscopic ground with some of their tools and practices but leave a lot to be desired. They may have their greatest strength in the prevention of common defects.

3.2.4 Security Assurance

Vendors and their customers alike have an interest in security assurance [62, 27, 7] to build confidence. Evaluation and certification frameworks like the Common Criteria [13] and less standardized practices like third-party security testing serve this purpose. As a downside, assurance methodologies tend to focus on ease of demonstration rather than on suitable security properties; the Common Criteria's obsession with security functions is an example. Assurance thus may create incentives for developers to aim for ease of auditing [63] rather than practical security, defending only against the weakest possible adversary [15]. Microscopic properties are easier to audit than higher-degree ones. Not only does this lower the bar too far, assurance can also get in the way of good design by patronizing security engineers with microscopic presecriptions in standards and certification schemes.

3.3 Design for Practical Security

To control macroscopic security properties means to design for reality. Our current practice and community culture, in contrast, seems stuck in microscopic views, limiting our ability to understand security. For example we pay a great deal of attention to vulnerabilities – implementation and design defects that might be exploited to undesirable ends – but studying them teaches us little more than what went wrong in a particular case. Machine learning beats vulnerability scoring when it comes to predicting which defects will be exploited [11]; we don't know how to generalize vulnerability studies beyond specific technologies and architectures [16]; and we cannot say whether a monthly patch day is a good or a bad sign. How much security would we gain if we could reliably prevent all vulnerabilities that we

have collected and analyzed over the years? We don't know, but probably we would gain less than we are hoping for, and it won't get better if we call a class of prevented vulnerabilities a security guarantee.

Macroscopic security design means to accept reality and to adjust our conception of security. Instead of aiming for unattainable perfection in the microscopic details, security engineers should firmly establish the weakest possible notion of security they can get away with in practice, and then optimize their security design for it. In this sense, some software vendors may already have gotten it right for some uses of some of their products. Software continues to have defects and vulnerabilities, but a combination of development practices, vulnerability management, and a highly automated maintenance infrastructure can keep security at an acceptable level for the majority of users and applications. It's not perfect, but good enough. Macroscopic security design invites us to jump a psychological barrier and make *good enough, statistically,* the objective of security design.

One meaning of *point-and-shoot security design* in the title of this paper is a description of our current state of affairs: we are collectively acting like someone trying to become a better photographer by buying better cameras. The new paradigm is the equivalent of recommending to this person to pay more attention to finding interesting subjects. We will discuss a second meaning when it comes to tools later on.

4. TOOLS FOR SECURITY DESIGNERS[1]

Security engineers need tools to support their work. Apart from their role – provide feedback and analysis, help developers to keep track of information, automate tasks, prevent errors, and so on – we can classify security design tools by the property degree they address. There are tools and aids for all degrees, but those addressing higher property degrees provide too little guidance, leaving room for improvement. On the other hand, massive automation, as we see it for instance in software testing, seems unrealistic for macroscopic and challenging for mesoscopic design. Other than the previous sections, which tried to be as generic as possible in their scope, the discussion here remains limited to the software technology domain.

4.1 Available Design Aids

4.1.1 Macroscopic Tools

Macroscopic design aids support the designer's thinking about the threat environment a system has to cope with. They do not force the designer into particular design decisions or considerations. Macroscopic design aids rather help the designer to understand the threat environment of a system and the effects of design decisions in this environment. This is not necessarily the same as security requirements in the traditional sense. Security requirements follow from what is valuable to some stakeholder, whereas threats are a consequence of something having value for an adversary. Macroscopic design aids therefore focus on the objectives, modi operandi, and capabilities of attackers; they help the designer to figure out *what* to achieve and leave the *how* to creativity – or to other tools.

Abuse cases [43] are an example of a macroscopic design aid. An abuse case describes the resources, skills, and objectives of a class of attackers, possible abusive interactions with a system, and their harmful consequences. As a design aid, an abuse case requires a refutation to some desired level of assurance: the developers need to argue how the system copes with each abuse case [42]. Abuse cases provide merely a notation; the result of their application thus depends on the skill, effort, and experience put into their development.

4.1.2 Microscopic Tools

Microscopic design aids support local security considerations, generally neglecting the system and threat context. They work with generic, static, possibilistic attack models instead: if the attacker does X, the consequence will be Y and we do Z to prevent it. Microscopic aids leave it to assumption whether any actual adversary has an interest in Y, which alternative ways exist to attain Y through or around Z, and thus which net effect Z has on the security of a system. Microscopic design aids tend to be prescriptive, and they seem easier to automate and formalize, as they abstract away the complexity of threats and attacker behavior.

Examples for microscopic design aids are (semi-)automated security testing tools, vulnerability and attack pattern databases, and formal methods. Automated security testing tools and database like CWE[2] and CAPEC[3] deal with common implementation and low-level design defects. Testing tools find defects in a program, thus giving the developer feedback, whereas databases provide background information. What makes them useful as a design aid are classes of defects that are a) common, as they are provoked by the design of technologies and platforms, and b) are almost always a problem regardless of specific requirements, as they are easy to exploit to a wide range of ends. In most system designs one does not need to understand the precise security requirements to consider a remote-exploitable buffer overflow defect a critical vulnerability.

Formal methods become microscopic design aids through the abstractions they make as a prerequisite for formalizing. They tend to prescribe security properties based on assumptions about what security means, and they seem applicable to security building blocks such as security kernels, access control policies, or cryptographic protocols, rather than to entire systems. System and software designers merely use these building blocks rather than designing them.

4.1.3 Mesoscopic Tools

Mesoscopic design aids support the designer in relating macroscopic and microscopic properties. They help developers to express design decisions and to analyze their impact on the security of a system, adding context to microscopic and design detail to macroscopic considerations. Mesoscopic aids may combine analytical and prescriptive elements or focus on one of these aspects.

Examples of mesoscopic aids are security design patterns, attack trees, Microsoft's threat modeling technique, and exploratory testing tools. Design patterns [8, 60] capture common design problems and proven solutions, generally at a degree of abstraction that takes some building blocks and their

[1]Apart from minor corrections and additions, this section is left in its pre-workshop condition; some inconsistencies may ensue.

[2]Common Weakness Enumeration, `http://cwe.mitre.org/`

[3]Common Attack Pattern Enumeration and Classification, `http://capec.mitre.org/`

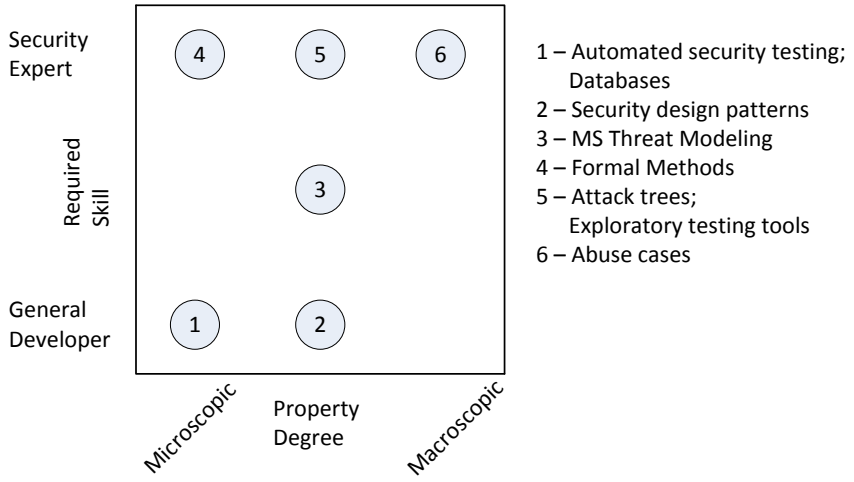

Figure 3: Rough classification of security design aids by property degree addressed and security-specific skill required.

properties for granted. To apply a pattern, the designer has to make macroscopic considerations to identify applicable patterns and to assess whether they really solve the problem at hand; the implementation of a pattern implies microscopic requirements. Little creativity and expertise is required to implement a pattern – the pattern description tells the designer what to do and what to care about.

Attack trees [57, 39] provide a lightweight notation for alternative attacks towards a common goal. Identifying attack goals is a matter of macroscopic analysis and beyond the scope of attack trees. Attack descriptions reflect the system design and can be refined to any level of detail. Once created, an attack tree can be used for model-based analyses of macroscopic properties like attack effort and feasibility. Since attack trees are just a notation, their power as a design aid depends on the designer's skill, expertise and creativity.

Microsoft's approach to security design analysis [64], often called threat modeling, relates rough design models to abstract attacker behavior. Developers model a system as a data flow diagram and assign STRIDE threats (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege) to its elements as abstract representations of common attacker behaviors and objectives. This process creates a checklist of attacker interactions that one might want to prevent. The technique does not prescribe what to do about any of the threats, and it does not specifically support the analysis of assets, requirements, or attacker objectives.

Exploratory testing tools support penetration testing, the interactive search for vulnerabilities in a system. Examples are interactive web proxies like WebScarab or Burp, Network packet manipulation tools like Scapy, generic protocol implementations like Netcat or Socat, fault injection tools, or debuggers. These tools allow a tester to interact with and observe a system in every possible way while testing flaw hypotheses. Their application requires expertise, skill, and creativity; their user needs to understand threats, attack success conditions, and attack techniques to get useful information out of them.

4.2 Room for Improvement

The challenge we are facing is to establish a discipline and support tools for macroscopic and mesoscopic security property control. We understand microscopic security design and how to support it with design aids. Increasingly we become able to support microscopic design tasks interactively as developers write code (e.g. [71]). However, we remain far from having a complete toolbox for security designers that would cover the entire scale of property degrees.

4.2.1 Macroscopic Design Aids

At the macroscopic end of the scale we have only notations, such as the abuse case notation, and vague notions like threat or asset. Security designers would benefit from techniques and aids for genuine threat analysis: Which assets attract attackers, what are the attacker modi operandi to consider, and which factors in the system design affect their effort, risk, and behavior? Macroscopic security design involves economic and ecosystem considerations that should remain valid regardless of some amount of noise and uncertainty about microscopic properties.

Some macroscopic design aids that might be useful if we could create them:

Design-invariant threat models and techniques to apply them to specific system designs. Can we describe the essence of attacker behaviors that remains after abstracting out microscopic detail? We might create models of, for example, opportunistic vs. targeted attackers or of single-system vs. ecosystem threats, and use them to understand the threat profile for a system under development.

An asset valuator or target attractiveness metrics to help engineers understand how valuable an asset, a collection of assets or an entire system is to a class of ad-

versaries. Stakeholders' and adversaries' priorities are not necessarily the same.

Abusability assessment techniques to show designers the most likely abuses of a system, considering its purpose and environment. For instance it would be nice to have an attack objective generator asking unpleasant questions about a system, such as: "What if an attacker wants something that isn't on your list of precious assets?"

Constraint analysis techniques that help the designer of a system to find a small set of enforceable constraints that make a system sufficiently secure against a type of threat. A financial service for instance might want to constrain the amount of money per unit of time that can be lost through a wide range of attacks.

Such analytic techniques and aids would allow engineers to explore the design space, evaluate prospective system designs, and compare approaches.

4.2.2 Mesoscopic Design Aids

Mesoscopic design aids serve three purposes, not necessarily all three in the same tool: to guide microscopic security considerations, to refine macroscopic models, and to connect, or translate between, macroscopic and microscopic considerations. A complete toolbox should support translation in both directions: deriving design directives and priorities from macroscopic considerations, and the evaluation of designs and design decisions for their macroscopic consequences. Evaluation may be the more important part, as design decisions are being made for a variety of reasons other than security, yet their consequences need assessment.

Tools and aids like the following may support security designers in their work:

Security-aware architecture modeling, highlighting aspects of system architectures and deployments that most likely affect their security properties. From runtime architecture visualization [22] to automated detection and flagging of fractures in the interpretation of information [16] there is a broad range of tools that might be useful but are not available today.

An asset tracker helping developers to keep track of all the manifestations of an asset in a system and protect them consistently. Often we do not really need to protect files or databases, but rather that which they represent and which may have further manifestations in a system.

Tools to analyze security dependencies between security building blocks and other components of a system. Many security mechanisms do not solve problems but rather transform them, creating secondary assets that impose new protection requirements and points from which security can be subverted. Trust distribution diagrams [37] look like a step in this direction. Such techniques may profit from better, standardized documentation of security functions: assumptions they rely on; which problems they do *not* solve; and which new attack paths they create.

Mechanism effect analysis would translate a security design into the macroscopic world and show how much of a difference a security mechanism makes with respect to a threat. Similarly, vulnerability landscape analysis might show security designers which possible security vulnerabilities will most likely be exploited by real attackers.

Such tools help security engineers to properly combine security building blocks and design decisions to attain those security properties that a system really needs.

4.3 Why Don't We Have Them Yet?

Several socio-economic factors may hamper the development and adoption of advanced security design tools. In the business of software and systems development, vendors may start from a low maturity level and go for the low-hanging fruits first – common bugs that threaten reputation. Taken to the extreme, this leads to a checklist mentality where a vendor focuses on avoiding the "Top Ten Security Bugs". As a community, we support this mentality by hunting just these bugs and celebrating loudly every time we have found another instance, sometimes neglecting honest assessment of their macroscopic impact. Microscopic security tools that are easy to use for general developers may be a larger market than advanced tools for the fewer architects and security experts involved in development. Typical development teams comprise many coders but few people responsible for higher-level design considerations.

The software supply chain is another factor. Off-the-shelf (COTS) software often serves as a template for systems, adapted and extended for specific applications with minimal effort. COTS developers thus make most of the design decisions, but they cannot tailor their software to the security needs of a particular application and system context. Design information gets lost at the interface between software developers and systems engineers; software often comes as a black box with only functional security documentation.

Finally, we have a tradition of focusing on microscopic security considerations. Even where we should understand the macroscopic requirements pretty well, such as for government systems having to protect confidential information, we rarely describe threats e.g. in terms of the characteristics of a foreign agency trying to obtain secrets. Seminal work in our field acknowledged the need for security considerations beyond the scope of policy enforcement within a confined computer system [53] but did not address the problem. Subsequent research largely followed suit. Whether this is due to the inherent difficulty of securing systems against adaptive adversaries or to biases in our community culture remains an open question.

5. THE POINT-AND-SHOOT ANALOGY

The property degree framework presented in this paper originates in a less-than-obvious analogy, which however reflects an unfulfilled desire. Wouldn't it be a great achievement if we could use technology to make security design as casual and easy as photography has become? Everyone can take good photos today knowing little more than how to hold the camera, letting the camera figure out exposure and all the other tricky stuff. Surely there is more to photography than that, but we have automated the basics and should try the same for security engineering. But when we listen to photographers we will find that, no, we haven't automated the basics, as we can see with just a bit of common

sense. The property degree framework abstracts this reasoning, which is not really specific to photography, and allows us to apply it to other fields.

Ignoring the artistic component, we can view photography as visual engineering. Photos have visual properties; the photographer designs them using a camera and other tools. How photographers work and what they consider in the process is documented in a large body of literature [23, 26, 58] and other sources [33].

5.1 Visual Properties of a Photo

What characterizes a photo? The answer to this seemingly simple question depends on how we interpret the photo – the property degree of the properties we consider. Figure 4 shows a sample photo from a microscopic, a mesoscopic, and a macroscopic perspective.

5.1.1 Microscopic View: Pixels and Details

A digital photo is completely and precisely specified by an array of pixels, each pixel carrying a color value. The word *pixel* means *picture element.* The color value of each pixel thus constitutes a property characterizing a portion of the picture; all pixels together characterize the entire photo.

What are the pixels telling us about a photo? Individually, not much: they allow us to compare photos, to qualify and quantify differences, but not in a very interesting or informative way. Maybe abstraction comes to our rescue, removing non-essential detail? We can indeed create abstractions of microscopic properties, such as the color histogram in Figure 4 (b) showing the distributions of primary colors over the pixel array. We may also apply image processing algorithms to determine abstract properties, such as edges or regions of highlight or shadow. But no matter how hard we try, it will remain very difficult to determine properties like: *"This photo has a horizontally centered composition,"* or: *"The horizon line is slightly skewed,"* or: *"This photo portrays an evening mood,"* for arbitrary photos, starting from pixel analysis.

5.1.2 Mesoscopic View: Composition

If we listen to photographers discussing photos, we will hear them use a different vocabulary as they discuss a different set of properties. A major part of their conversation will likely revolve around composition: the positioning and arrangement of objects, shapes, and lines; sharpness and blur in relation to the objects in the image; positive and negative space; visual emphasis; texture; or depth [23, 26, 58]. They discuss mesoscopic properties.

Figure 4 (c) shows one abstraction representative of the composition view, a dissection of a photo into positive and negative space. Negative space surrounds and separates subjects. This definition makes the notion dependent on interpretation of the content of an image. The concept of negative space does not translate uniquely and universally into microscopic properties; contrast in color, brightness, or texture, blur around sharp objects, and other visual effects can create and define negative space.

To discuss the mesoscopic properties of composition, we need to interpret the content of the entire photo. We can explain the underlying microscopic properties for a particular image, but we cannot universally derive composition from a microscopic analysis of pixels. Microscopic approximations may exist for restricted sets of pictures.

5.1.3 Macroscopic View: Message and Reaction

An analysis of composition tells us how a photo works, how the photographer decided to shoot. But only the photo itself tells us its message and evokes our reaction – thoughts, emotions, insight. A photo's macroscopic properties characterize its message. To discuss these properties we need to take into account the beholder. An image alone does not fully specify its message, the beholder's interpretation is an essential part of the macroscopic view: different people may react differently to the same image. A photo may work or fail for reasons independent of the photographer's skill and effort, such as the cultural background of the person viewing it or her personal relationship with that which the photo shows. Figure 4 (d) shows a photo; your reaction to it is part of its macroscopic properties.

Mesoscopic properties of a photo, such as the various aspects of composition, support its message by guiding and focusing the beholder's interpretation. However, composition alone does not characterize the message. We need to consider properties like facial expressions, situations, moods, symbols, or speculation about temporal context. Different compositions may support similar messages, and similar compositions of different subjects convey different messages.

A photo constitutes an abstraction of its message. While shaping and conveying a message is the photographer's ultimate objective, not every detail of the message has a visual representation in the photo. To the contrary, good photographers actively pursue photographic abstraction, reducing that which a photo shows to the essentials.

5.2 How Does a Photographer Create?

The task of a photographer is to convey a message visually. The message may be predetermined, as in applied photography, or the photographer may freely define it in the more artistic styles of photography. For a given message, some photos work better than others in conveying it. This makes photography a design task: the photographer optimizes a visual representation such that it communicates the intended message effectively to the beholder of the photo.

If we observe photographers at work [29] or listen to them explaining how they work [33], we will find an approach roughly like this:

1. Choose a subject and style
2. Find a suitable angle and composition
3. Adjust camera settings
4. Take the picture
5. Improve the result in postprocessing (optional)

Within the process we will likely see iterations, e.g. trying different angles and compositions, and the photographer may abort the process either without taking a picture or later by discarding it.

Property degrees allow us to describe this process in abstract terms. The process begins with macroscopic considerations, defining the message to be conveyed in a photo and some constraints on how to do it. The photographer then goes on and explores the design space, looking for favorable angles and compositions under the constraints imposed by macroscopic requirements and the environment. Microscopic decisions, such as camera settings or the position of

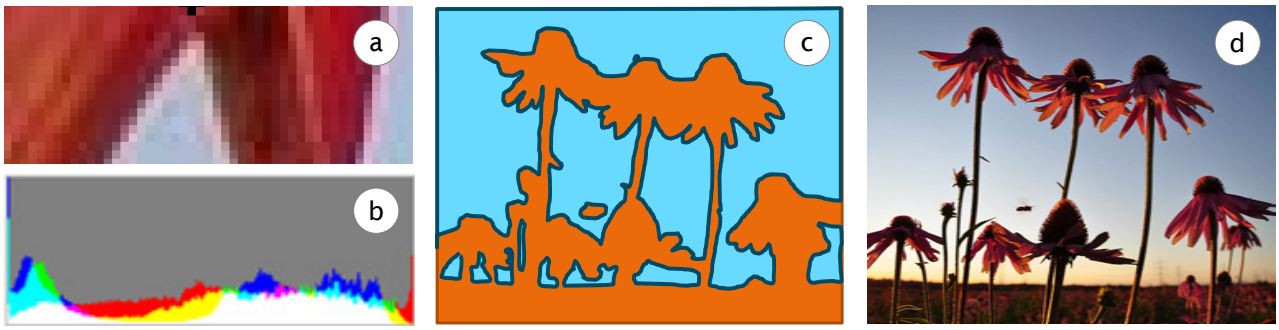

Figure 4: Property degrees illustrated: microscopic properties (a), a microscopic abstraction (b), a mesoscopic abstraction (c), and the whole photo (d).

the camera in space, follow as a consequence. Which microscopic properties matter for which part of the photo depends on higher-degree decisions.

Table 3: Some visual properties of the photo in Figure 4 and related means of property control.

Degree	Properties	Means of Control
Macroscopic	Flowers; Calm sunset mood	Selection of subject; Choosing time, location; "Working the scene"; Sense of esthetics
Mesoscopic	Directed light; Centered composition; Flat, little depth; Repetition; Low horizon	Camera angle; Use sky as negative space; Choice of lens; Choice of aperture
Microscopic	Vertical color gradient; Blurred horizontal edge; Irregular reddish shapes; Vertical edges	Focal length 16mm; Aperture f/3.5; Auto shutter speed; Exposure bias 0.67

We cannot simply reverse this process and start from a microscopic definition of "good" photography, e.g. in terms of sharpness or exposure. Doing so would constrain higher-degree design decisions and cut the photographer off from design options that would improve the result. However, a photographer can voluntarily set such constraints as part of his or her personal style, with the consequence of this style working better for some subjects than for others.

5.3 Better Photographers Have More Control

If the subject is variable, lower-degree design decisions depend on it, and everyone can push the shutter release button, what makes a good photographer? It is the degree of property control attained. Anybody can point a camera at something and take a shot. The result will have properties of all degrees, but most are left to chance and do not support the intended message. Starting from this baseline, one has to learn three skills to become an amazing photographer: controlling microscopic properties, such as focus and exposure; controlling mesoscopic properties, such as composition and lighting; and controlling the message of a photo by finding interesting subjects and ways of viewing them.

Figure 5 illustrates this idea with three different photos of snow in winter. Photo (a) shows little conscious control over any of its properties. Note, however, that we cannot ground this statement on an analysis of the photo alone. We have to take into account whether the photographer wanted it to look that way and how it resonates with an audience. Photo (b) shows a composition concept applied: diagonal lines. Without a message control of composition leads to formalist photos like this. Photo (c) in Figure 5 includes elements representative of a situation, thus attempting to tell a story.

This model is compatible with the common observation that neither a better camera nor a better understanding of camera controls makes better photographers.

5.4 The Point-and-Shoot Toolbox

Modern cameras greatly simplify the act of taking a picture; imaging software does the same for post-processing, which once required darkroom skills. Nevertheless, amateurs remain amateurs, and professionals often disable or override the automated helpers in their cameras. Which role play tools in photography and where are their limits?

We find rather little direct tool support for mesoscopic and macroscopic property control. Choosing subjects, deciding how to shoot them, and finding a suitable composition are tasks left to the photographer. The lack of tools reflects the lack of firm rules, the photographer's creative freedom. Indirectly, however, digital imaging technology supports the photographer by providing quick feedback on the back of the camera and by making it easier to modify photos experimentally in post-processing.

The majority of automated tools, such as auto exposure, autofocus, or preset programs, make it easier to control microscopic properties – if the requirements meet the assumptions the tool was designed for. Auto exposure, for instance, assumes average lighting and fails systematically for very bright or very dark scenes, and autofocus has its own model of what to focus on. Automation can thus get in the way of experienced photographers – their higher-degree design decisions often imply specific, non-standard requirements for microscopic property control.

Automation seems most useful where it helps inexperienced photographers to control at least some microscopic

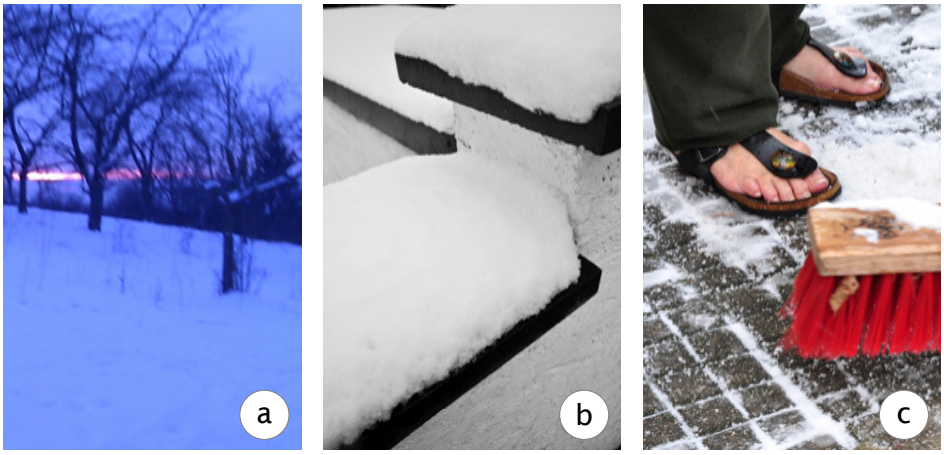

Figure 5: Photographs of snow with different degrees of property control: almost none in a smartphone snapshot (a), formalist control of composition (b), controlled message (c). Photo (a) courtesy Tim Kern.

properties in a better-than-nothing fashion. In a comprehensive design process, with a skilled photographer trying to control properties at all degrees, the role of tools becomes multifarious. Tools serve for example to:

- Support and simplify specific tasks while keeping the photographer in full control. Auto exposure for instance can be used to control only some of the parameters affecting exposure, leaving others for the photographer to control.

- Enlarge the design space. Technologies like image stabilization, high-ISO sensors or high dynamic range processing allow photos to be taken under adverse conditions or in a look that would have been difficult to create using film.

- Reduce the photographer's workload. Professionals don't reject automation where it helps them. In sports photography for instance, speed is important to capture the right moment; automation of mundane tasks helps the photographer to achieve that.

There is no magic. Point-and-shoot functions as tools do not replace the photographer. They improve the quality of the result if one starts from a very low baseline of property control, and they can sometimes relieve the photographer of complications and minor tasks.

5.5 From Art to Engineering

Using photography as an analogy to discuss security engineering embeds this paper in the age-old debate whether we should conceive software development as engineering [17, 70, 9, 48, 4], art [34, 28], science [10], craft [66], or something else, and whether we would be better off emphasizing one view or the other [18, 49, 14, 41]. The author does not take sides, but rather looks for lessons to be learnt from one field for another.

As Graham points out [28],

> "Hacking[4] and painting have a lot in common. In fact, of all the different types of people I've known, hackers and painters are among the most alike.
>
> What hackers and painters have in common is that they're both makers. Along with composers, architects, and writers, what hackers and painters are trying to do is make good things."

Art and engineering have in common the laborious process of design optimization through a series of design decisions. They differ in their objectives; engineering solves objective problems while art expresses subjective views. But both start with an idea and subsequently refine it into an optimized artifact, exploring a design space in the process. While in everyday snapshots we trade quality for speed, a careful photo design process can require just as much time and effort as an engineering project. As an example, according to a documentary [29] it took visual artist Andreas Gursky about two years and multiple collaborators to create just one of his photographs, *Hamm, Bergwerk Ost*. As another case in point, photographers developed a formal model of exposure control, the zone system [32, 36]. The zone system provides precise control of the mapping of brightness levels in a scene onto the limited dynamic range of film – microscopic property control.

During the workshop, as well as through the writing and revision of this paper, the photography analogy turned out a bit distracting. This is in part owed to its purpose: to illustrate notions and ideas in a manner that fits the medium. Conceptually it seems interchangeable with any other non-trivial creative practice. Photography just came in handy as something familiar to most of us and easy to demonstrate. Beyond this role, and a few general thoughts about the use and purpose of tools and automation, photography should not be taken for a proper model of security engineering. There are important differences between the two:

[4]Note that Graham uses the term *hacking* in the traditional sense [51], which has nothing to do with malicious activity.

- Photography is a single-objective task, whereas security design takes place as part of a multi-objective engineering process.
- Consequently, there is less teamwork in photography. While professional photographers often work with aides, the team remains small and has a clearly defined hierarchy with the photographer at the top.
- Compared to security engineering, the stakes in photography seem negligible.
- Artists can choose their audiences, or do so implicitly by their style and subject. Security engineers don't get to choose the threats their work result has to cope with.
- There is less pressure, at least in free artistic photography, to get results at all. A photographer, if not working on an assignment, can at any time stop and just not take a photo.
- Shots are cheap, facilitating experimentation. Shooting a large number of different attempts and discarding most of them is perfectly feasible. Exploring a design space may be much more costly in engineering.
- Engineering must be efficient. Engineers therefore prefer to reuse available building blocks over developing from scratch, limiting the range of feasible designs.
- More generally, a great deal of engineering takes place outside the immediate development process of a system or software product. Technologies evolve under the influence of a gazillion of factors and players.

Our idealized model of a design process may therefore be more appropriate for photography than it is for security engineering. This should however not invalidate the proposition that a design process needs to cover all property degrees.

6. CONCLUSION

The property degree framework systematizes the different ways in which we can think about the security of a system. Each degree of security properties corresponds with a class of adversary models: microscopic properties with specific attack actions; mesoscopic properties with attackers pursuing particular objectives; and macroscopic properties with populations of adversaries. The different viewpoints imply barriers between security properties of different degrees, barriers we cannot pass by simply abstracting, generalizing, or aggregating lower-degree properties according to universal rules. Rather, the mappings of properties from one degree to another depend on macroscopic threats and mesoscopic design decisions.

Effective security design controls macroscopic properties through design decisions spanning the entire range of degrees. To this end we need design practices and tools to support the developers of systems and software in their design process. Currently our collective toolbox is populated predominantly by microscopic and to some extent, mesoscopic tools. Other than safety engineers with their clearly defined notions of undesired events, causes, consequences, and contributing factors [52], we lack even the vocabulary to express and discuss macroscopic design objectives.

Point-and-shoot security design has a double meaning. First, it describes the state of the art in security design by analogy. Point-and-shoot automation in a camera covers only the microscopic part of a photographer's job, and does it well only in some situations. It is better than no property control for inexperienced photographers, but often gets in the way of the adepts and their macroscopic pursuits. So do contemporary security design practices and tools, with the difference that apparently we do not systematically teach and develop adepts of security design yet.

Second, point-and-shoot security design refers to the industry demand for tools to support developers unobtrusively in their security engineering tasks. This demand is legitimate and rational; it poses the question to what extent it can be fulfilled, in theory and in practice. The difference between art and engineering is that the artist is free to choose objectives and approaches, whereas the engineer needs to understand and solve particular problems. On the other hand, *"Humans have a funny knack for figuring out how to do something when there is no perfect answer."* [10], so there is a place for creativity in engineering. Point-and-shoot-style tools alone are not the right answer to the demand for security engineering tools.

The tools proposed in Section 4 are merely suggestions how to apply the property degree framework in practice. Beyond practical applications, the framework has implications for the science of security [56, 38, 40, 6]. If macroscopic security properties characterize how a system interacts with its threat environment, how can we describe them, to what extent can we control them, which environmental factors change them so we need to redesign a system, and how do desired macroscopic properties translate into design decisions? These are only some of the questions a science of security needs to ask. In the process of trying to answer them we may hit epistemological limits [55]: how far can we not only anticipate, but control the future [25]? This is after all what macroscopic security means, a decreased likelihood of undesirable outcomes in spite of enemy action.

7. ACKNOWLEDGMENTS

I thank Michael Locasto for his shepherding and guidance, all NSPW 2012 attendees and reviewers for their comments and discussion, my colleagues Andreas Poller and Jörn Eichler for their honest feedback on various drafts of this paper, and Tim Kern for contributing a photo.

The work presented in this paper was performed in the Software-Cluster project InDiNet (www.software-cluster.org), funded by the German Federal Ministry of Education and Research (BMBF) under grant no. 01IC10S04. The author assumes responsibility for the content.

8. REFERENCES

[1] J. Ames, S.R., M. Gasser, and R. R. Schell. Security kernel design and implementation: An introduction. *Computer*, 16(7):14–22, July 1983.

[2] R. Anderson. Liability and computer security: Nine principles. In D. Gollmann, editor, *Computer Security — ESORICS 94*, volume 875 of *LNCS*, pages 231–245. Springer Berlin / Heidelberg, 1994.

[3] R. Baskerville. Information systems security design methods: implications for information systems development. *ACM Comput. Surv.*, 25:375–414, December 1993.

[4] S. Berkun. Programmers, designers, and the Brooklyn bridge, Mar. 2004. http://www.scottberkun.com/essays/30-programmers-designers-and-the-brooklyn-bridge/.

[5] K. Beznosov and O. Beznosova. On the imbalance of the security problem space and its expected consequences. *Information Management & Computer Security*, 15(5):420–431, 2007.

[6] K. Bicakci and P. C. van Oorschot. A multi-word password proposal (gridword) and exploring questions about science in security research and usable security evaluation. In *Proceedings of the 2011 workshop on New security paradigms workshop*, NSPW '11, pages 25–36, New York, NY, USA, 2011. ACM.

[7] BITS software assurance framework. http://www.bits.org/publications/security/BITSSoftwareAssurance0112.pdf, Jan. 2012.

[8] B. Blakley and C. Heath. Security design patterns. Technical Guide G031, The Open Group, Apr. 2004.

[9] B. Boehm. A view of 20th and 21st century software engineering. In *Proc. 28th Intl. Conference on Software Engineering*, ICSE '06, pages 12–29, New York, NY, USA, 2006. ACM.

[10] T. Bollinger. The interplay of art and science in software. *Computer*, 30(10):128, 12–127, Oct. 1997.

[11] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker. Beyond heuristics: learning to classify vulnerabilities and predict exploits. In *Proc. 16th ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 105–114, New York, NY, USA, 2010. ACM.

[12] A. A. Cardenas, T. Roosta, and S. Sastry. Rethinking security properties, threat models, and the design space in sensor networks: A case study in SCADA systems. *Ad Hoc Netw.*, 7:1434–1447, Nov. 2009.

[13] *Common Criteria for Information Technology Security Evaluation*, July 2009. Version 3.1, revision 3.

[14] J. O. Coplien. Software development as science, art, and engineering. In L. Rising, editor, *The Patterns Handbook: Techniques, Strategies, and Applications*, SIGS reference library series, pages 321–332. Cambridge University Press, 1998.

[15] J. Corman. Intro to HDMoore's Law. http://blog.cognitivedissidents.com/2011/11/01/intro-to-hdmoores-law/.

[16] J. Crandall and D. Oliveira. Holographic vulnerability studies: Vulnerabilities as fractures in interpretation as information flows across abstraction boundaries. In *Proc. New Security Paradigms Workshop 2012*, NSPW '12, Sept. 2012.

[17] M. Davis. Will software engineering ever be engineering? *Commun. ACM*, 54:32–34, November 2011.

[18] T. DeMarco. Software engineering: An idea whose time has come and gone? *Software, IEEE*, 26(4):96, july-aug. 2009.

[19] J. Epstein. A survey of vendor software assurance practices. In *Proc. Annual Computer Security Applications Conference 2009 (ACSAC '09)*, pages 528–537, dec. 2009.

[20] European Central Bank. Biannual information on euro banknote counterfeiting. Press release, 16 July 2012, http://www.ecb.int/press/pr/date/2012/html/pr120716.en.html, July 2012.

[21] European Commission. Euro coin counterfeiting in 2011. Press release, 27 January 2012, http://ec.europa.eu/commission_2010-2014/semeta/headlines/news/2012/01/20120127_en.htm, Jan. 2012.

[22] W. Fang, B. P. Miller, and J. A. Kupsch. Automated tracing and visualization of software security structure and properties. In *Proc. 9th International Symposium on Visualization for Cyber Security*, VizSec '12, pages 9–16, New York, NY, USA, 2012. ACM.

[23] A. Feininger. *A manual of advanced photography.* Thames and Hudson, 1962.

[24] E. B. Fernandez, N. Yoshioka, H. Washizaki, J. Jürjens, M. VanHilst, and G. Pernul. Using security patterns to develop secure systems. In H. Mouratidis, editor, *Software Engineering for Secure Systems: Industrial and Research Perspectives*, pages 16–31. IGI Global, 2010.

[25] R. Ford and L. M. Mayron. All your base are belong to us. In *Proc. New Security Paradigms Workshop 2012*, NSPW '12, Sept. 2012.

[26] M. Freeman. *The Photographer's Mind.* Elsevier Science & Technology Books, 2010.

[27] K. M. Goertzel, T. Winograd, H. L. McKinley, L. J. Oh, M. Colon, T. McGibbon, E. Fedchak, and R. Vienneau. Software security assurance: A state-of-the-art report (SOAR), 2007.

[28] P. Graham. *Hackers and Painters*, pages 18–33. O'Reilly, Sebastopol, CA, May 2003.

[29] Long shot close up – Andreas Gursky. Art Documentary Film, Aug. 2010. Jan Schmidt-Garre (Director); Pars Media Film- und Fernsehproduktion; Co-produced by BR, ARTE and Tilk Filmproduktion.

[30] K. Henney. Down on the upside. http://www.artima.com/weblogs/viewpost.jsp?thread=341297, Mar. 2012.

[31] M. Howard and S. Lipner. *The Security Development Lifecycle.* Microsoft Press, 2006.

[32] C. Johnson. *The Practical Zone System for Film and Digital Photography: Classic Tool, Universal Applications.* Elsevier Science, 2012.

[33] S. Kelby. Crush the composition. Google+ Photographer's Conference presentation, http://www.youtube.com/watch?v=FpHMuK7Htic, May 2012.

[34] D. E. Knuth. Computer programming as an art. *Commun. ACM*, 17(12):667–673, Dec. 1974.

[35] G. Lakoff and M. Johnson. *Metaphors we live by.* Univ. of Chicago Press, 1996.

[36] B. Lav. *Zone System: Step by Step Guide for Photographers.* Amherst Media, 2002.

[37] M. E. Locasto, S. J. Greenwald, and S. Bratus. Trust distribution diagrams: Theory and applications. In *Fourth Annual Layered Assurance Workshop (LAW 2010)*, Dec. 2010.

[38] T. Longstaff, D. Balenson, and M. Matties. Barriers to science in security. In *Proceedings of the 26th Annual*

Computer Security Applications Conference, ACSAC '10, pages 127–129, New York, NY, USA, 2010. ACM.

[39] S. Mauw and M. Oostdijk. Foundations of attack trees. In D. Won and S. Kim, editors, *Information Security and Cryptology - ICISC 2005*, volume 3935 of *LNCS*, pages 186–198. Springer Berlin / Heidelberg, 2006.

[40] R. A. Maxion, T. A. Longstaff, and J. McHugh. Why is there no science in cyber science?: a panel discussion at nspw 2010. In *Proceedings of the 2010 workshop on New security paradigms*, NSPW '10, pages 1–6, New York, NY, USA, 2010. ACM.

[41] S. McConnell. The art, science, and engineering of software development. *Software, IEEE*, 15(1):120, 118–119, jan/feb 1998.

[42] J. McDermott. Abuse-case-based assurance arguments. In *Proc. 17th Annual Computer Security Applications Conference (ACSAC 2001)*, pages 366 – 374, Dec. 2001.

[43] J. McDermott and C. Fox. Using abuse case models for security requirements analysis. In *Proc. 15th Annual Computer Security Applications Conference (ACSAC '99)*, pages 55 –64, 1999.

[44] J. D. Meier. Web application security engineering. *IEEE Security & Privacy*, 4(4):16–24, July-Aug. 2006.

[45] P. H. Meland and J. Jensen. Secure software design in practice. In *Proc. 3rd Intl. Conf. on Availability, Reliability and Security, 2008 (ARES'08)*, pages 1164–1171, Mar. 2008.

[46] Microsoft. Microsoft security development lifecycle (SDL). Version 5.1.

[47] K. G. Olthoff. The high assurance brake job—a cautionary tale in five scenes. In *Proc. New security paradigms workshop 1999*, NSPW '99, pages 118–140, New York, NY, USA, 2000. ACM.

[48] D. Parnas. Teaching programming as engineering. In J. Bowen and M. Hinchey, editors, *ZUM '95: The Z Formal Specification Notation*, volume 967 of *LNCS*, pages 470–481. Springer Berlin / Heidelberg, 1995.

[49] D. Parnas. Software engineering: An unconsummated marriage (extended abstract). In M. Jazayeri and H. Schauer, editors, *Software Engineering – ESEC/FSE'97*, volume 1301 of *LNCS*, pages 1–3. Springer Berlin / Heidelberg, 1997.

[50] D. L. Parnas. Really rethinking 'formal methods'. *Computer*, 43(1):28–34, 2010.

[51] The jargon file. `http://catb.org/jargon/`, Oct. 2004. Version 4.4.8.

[52] J. Rushby. Critical system properties: survey and taxonomy. *Reliability Engineering & System Safety*, 43(2):189–219, 1994. Special Issue on Software Safety.

[53] J. H. Saltzer and M. D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[54] M. Schaefer. Symbol security condition considered harmful. In *Security and Privacy, 1989. Proceedings., 1989 IEEE Symposium on*, pages 20–46, May 1989.

[55] R. Schaefer. The epistemology of computer security. *SIGSOFT Softw. Eng. Notes*, 34:8–10, Dec. 2009.

[56] R. R. Schell. Information security: science, pseudoscience, and flying pigs. In *Computer Security Applications Conference, 2001. ACSAC 2001. Proceedings 17th Annual*, pages 205 – 216, dec. 2001.

[57] B. Schneier. Attack trees. *Dr. Dobb's journal*, 24(12):21–29, 1999.

[58] M. Schnelle-Schneyder. *Sehen und Photographie: Ästhetik und Bild*. X.media.press / publishing. Springer, 2nd edition, 2011.

[59] G. Schudel and B. Wood. Adversary work factor as a metric for information assurance. In *Proceedings of the 2000 workshop on New security paradigms*, NSPW '00, pages 23–30, New York, NY, USA, 2000. ACM.

[60] M. Schumacher. *Security Patterns: Integrating Security and Systems Engineering*. Wiley series in software design patterns. John Wiley & Sons, 2006.

[61] A. Shostack. Engineers are people too. Slide deck,I3P SAUSAGE workshop, `http://www.homeport.org/~adam/Engineers-are-people-too-SAUSAGE.pptx`, Apr. 2011.

[62] B. Snow. We need assurance! [assurance of computing quality, reliability, and safety]. In *Computer Security Applications Conference, 21st Annual (ACSAC'05)*, pages 7–10, Dec. 2005.

[63] C. Swan. Building security in – the audit paradox. `http://blog.thestateofme.com/2012/01/28/building-security-in-the-audit-paradox/`, Jan. 2012.

[64] F. Swiderski and W. Snyder. *Threat Modeling*. Microsoft Press, 2004.

[65] C. Swoyer and F. Orilia. Properties. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, CSLI, winter 2011 edition, 2011.

[66] P. Taylor. Vernacularism in software design practice: does craftsmanship have a place in software engineering? *Australasian Journal of Information Systems*, 11(1), 2003.

[67] V. Verendel. Quantified security is a weak hypothesis: a critical survey of results and assumptions. In *Proceedings of the 2009 workshop on New security paradigms workshop*, NSPW '09, pages 37–50, New York, NY, USA, 2009. ACM.

[68] J. M. Wing. A specifier's introduction to formal methods. *Computer*, 23(9):8–22, Sept. 1990.

[69] J. M. Wing. A symbiotic relationship between formal methods and security. In *Computer Security, Dependability and Assurance: From Needs to Solutions, 1998. Proceedings*, pages 26 –38, 1998.

[70] N. Wirth. A brief history of software engineering. *Annals of the History of Computing, IEEE*, 30(3):32 –39, july-sept. 2008.

[71] J. Xie, B. Chu, H. R. Lipford, and J. T. Melton. ASIDE: IDE support for web application security. In *Proc. 27th Annual Computer Security Applications Conference*, ACSAC '11, pages 267–276, New York, NY, USA, 2011. ACM.