Science of Security Lablets Hard Problems

1. **Leveraging the torrents of data.**
   Challenge: Torrents of data related to security are becoming available. Current analytical techniques emphasize results over causation of the findings and, as such, may not reveal fundamental data relationships or contribute to the science of security.

   Analytic techniques can be used to identify behavior patterns and anomalies (such as fraud or a security breach). Analytic techniques are often used for applications that emphasize results over causation of the findings. For example, analytic techniques may be used to determine patterns which population is most receptive to a particular postcard advertisement. The identified patterns may involve an almost infinite number of combinations of the many variables. Explaining the behavior (i.e. examining causation) by hypothesizing about the combinations of variables may not be economical and/or of interest to a user. A business, or an agency, may choose to act on the behavior without a focus on causation provided that the pattern has a high empirical probability of correctly identifying an issue. As such, current analytic techniques may not support the identification and advancement of fundamental scientific principles based upon an analysis of causation, or indeed science may not fully support use of the methods or conclusions based on those methods. The move from heuristics to science in analytics is fundamental to the advancement of the science of security.

2. **Measuring and modeling system behavior.**
   Challenge: Malicious use of a system must be automatically distinguishable from benevolent use of a system based upon security metrics and models.

   Automatically distinguishing between malicious and benevolent use of a system necessitates metrics and modeling of cyber-physical systems. Metrics are used to quantify dependent (some of which interrelate and deal with the system environment) and independent variables in the models with the aim of quantifying the resilience of software artifacts to cyber-attacks. Model development drives the exploration of hypotheses about system behavior. Models of cyber-physical systems need capture the bidirectional influence of cyber and physical on each other, the bidirectional influences on each other, and characterization of unique attacks where manipulation of one system impacts the other. With such models we address development of analysis of resiliency of cyber-physical systems to attacks.

   Research goals include improving the quality of metrics in the dual sense of (1) improved approximation to potential assured results, and (2) improved efficiency in evaluating artifacts and actions to develop metric values. Advancement on this hard problem contributes to the science in two ways: (1) improved ability to evaluate hypotheses through the use of better metrics and measurements, and (2) improved measurement capability, both in support of further science and in support of practice.

3. **Understanding attack behavior.**
   Challenge: The human component of cyber-security introduces variance and uncertainty.

   The interactions of human users with a system, including the framework of metaphors on which these are based, can be modeled as features of the operating environment and as features of the system itself. Researchers have developed methods to perform sophisticated task analysis for a

given problem domain, including security analytics (e.g., Designing for Situation Awareness: An Approach to User-Centered Design, M. R. Endsley). Behavioral psychology has methods to capture human mental models, which can then be used to describe the step-by-step strategies an analyst uses to solve specific tasks. Deviation from these patterns would suggest unexpected activity within a system, and one that may warrant further inspection. Once behavioral models are developed, they will need to be evaluated in the context of real analysts and security domains, with a particular focus on identifying commonalities across tasks and users. A final step would be to develop methods to integrate these models into real-world security systems. For example, given enough basic provenance information, and real-time information about user keystrokes, their order and their timings, or user mouse movements, it is currently possible to identify individual users, and their mood (e.g., anger, fear, happiness), through involuntary physiology and psychology driven changes in their typing and mouse movement behaviors. This type of work could be extend to "intent" that may be associated with security breaches and attacks. In short, we must infer behavior through low-level data analytics and the development of prediction models. These models of interactions include computer systems, the humans using them and the humans attacking them, with the goal of analyzing the resiliency of the computer system and its defenders against cyber attack.

4. **Scalability in the engineering of assured secure systems**

Challenge: Assuring security and quality in software-reliant systems is not only increasingly critical to operational success, but it is also increasingly challenging due to the continued growth in complexity, scale, and criticality. Success in developing and evaluating critical and infrastructural systems demands high levels of sophistication in the technical aspects of cybersecurity, software and hardware design, modeling and analysis for software and hardware, and human-systems interaction. Game-changing advances in development and assurance practices can address persistent challenges such as enabling the rapid evolution of assured systems, complex interlinking of multiple assured systems, assurance for self-adaptive and autonomous systems, integration of assured components with established socio-technical ecosystems, and the co-development of systems and the assurance cases that support them.

As systems grow in complexity and size, the challenge we face in reasoning about security attributes seems to grow even more rapidly. This is not surprising, given the quadratic reality of Metcalfe's Law, which observes that the number of potential interactions among components grows with the square of the number of components (i.e., the number of edges in a complete graph as a function of the number of nodes). Since security outcomes (at a technical level) derive from (a) the internal content of the components and (b) the overall configuration of components in relation to each other, a rapidly growing number of potential interactions must be considered.

This challenge is exacerbated by the reality of most modern systems, which is that these larger numbers of components are provided through more diverse and complex supply chains to create and sustain those components. The diversity of sourcing, nearly unavoidable for systems that involve web services, mobile devices, and other established socio-technical ecosystems, suggests that assurance cases for complex multi-component systems must increasingly rely on *direct evaluation of the product* rather than current practices which rely more heavily on *trust in the provider* and on *process compliance.*

There are further difficulties. Current regimes for certification and accreditation tend to focus on particular snapshots of system configurations. The reality of many modern systems is that they undergo continuous evolution and enhancement, are often self-adapting (e.g., in response to security threats), and include capability for dynamic reconfiguration (e.g., new device drivers, app installs, dynamic loading of compilation units, etc.). Additionally, systems increasingly

interact with other systems ("system of systems"), which creates a composition problem a much larger component scale. And, furthermore, larger systems are more likely to undergo a continual evolution in response to changes in the mission and operating environments, the technology infrastructure, doctrine, and other factors.

When evaluation regimes are not able to arrive at assurance judgments, systems may need to be simplified, at a cost to capability, agility, interlinking, performance, or other attributes significant to the mission. This suggests the need for significant improvements to our ability to construct secure systems and to assess, measure, and enhance security-related attributes of systems.

Composability is the principal key to scaling in overall size/complexity, in numbers of components, and in the diversity of sourcing of those components. Composition issues exist at nearly every level of software structure, ranging from large-scale frameworks and subsystems, to APIs and architecture, to individual code elements. Additionally, composition issues and challenges vary according to the particular quality attribute being assessed.

It is generally recognized that, in the general case of security evaluation, when two components are combined into an aggregate, the entire aggregate must be evaluated as a whole, and, additionally, independent assessments of the two components may not be useful to support this evaluation. In other words, all bets are off when systems are integrated or configured. But when models, components, and analyses are suitably arranged, it can be possible to achieve composability, wherein the independent assessments of components can contribute directly to an aggregate assessment without a need to reevaluate the component internals. In other words, it is not a necessary outcome that the assessments become invalid when components are combined. This kind of composability has been established for several significant security-related properties. For example, compositional type safety is now an established feature of nearly all mainstream languages (C++, Java, C#, Ada), but in the years prior to the emergence of these languages there were questions regarding whether this was technically feasible. Similarly, composition is now possible for an increasing range of critical quality attributes.

For many security-related attributes, such as related to confidentiality and locality of data, integrity of data, and availability of services, composability is, in general, difficult to achieve. But there is a growing set of approaches to modeling, analysis, and language, and these are offering great promise. This set is growing as a consequence of often deeply technical breakthroughs in the underlying theory and science, which inform particular design decisions in the development of modeling and analytic tools. These, in turn, inform the design of programming and evaluation practices as well as the languages and tools that support these.

This concept of composability encompasses dynamic and adaptive designs—it includes not only static models and structures defined during a development process, but also capability to scale and compose results from complementary dynamic models and tools that support self-adaptation of operational systems.

5. **Secure collaboration and analytics.**
   Challenge: Systems must to be developed to be legal- and policy-compliant and privacy preserving, complicated by collaborative use of the system by multiple parties. These same systems must not be so over constrained to support this collaboration that the utility of data analytics is unnecessarily reduced.

   Collaboration is complex because the collaborating parties are autonomous and heterogeneous: they can exercise different policies and laws, requirements, and risk aversion. Collaboration

exacerbates security risks because of such diversity, especially because the party's might inadvertently harm each other even when they are cooperative. The key problem is this: Security presumes that there is some correct or good outcome that we wish to protect and which an attacker may disrupt. Today, these standards of correctness are conceptualized at a low level and often miss what stakeholders need. Instead, we need a standard of correctness that captures the normative relationships between collaborating parties: that is, expresses their interactions in a technology-independent manner and yet maps to specific technical realizations. Such a representation would help infer whether one collaborating party may violate the norms of another, whether inadvertently or maliciously. This leads to two technical sub-problems. First, how may we model such normative relationships perspicuously? Second, how may we analyze combined data from multiple, possible interdependent sources to understand which norms are applicable for determining compliance and which are violated. Additionally, the analysis and combination of data from multiple, possible interdependent data sources may create knowledge for which the compliance of the use and sharing must be determined. Finally, big data privacy control requires us to rethink privacy protection in almost aspects, including privacy models, utility models, anonymization algorithms and paradigms, and evaluation metrics and methodologies.

6. **Usability in development and evaluation of systems**

Challenge: From a purely technical perspective, the interventions required to enhance scalability and flexibility in assured development may be extensive for complex systems with diverse security-related assurance attributes. How can the tools used by professional developers and evaluators be engineered in a way that acknowledges the diverse technical backgrounds of those who contribute to systems—and also recognizes the realities of the development process and, in particular, its measures and rewards?

Consideration of this leads to an identification of two features that are characteristic of success. The first of these is a need for practicing developers and evaluators to be able to *readily grasp the metaphors behind the technical models* used in the process of design and analysis. That is, the presentation to human developers and evaluators must be intellectually approachable and, ultimately, "intuitively comfortable"—regardless of the technical sophistication and depth of the underlying mathematics of the modeling and analysis embodied in the tool, language, and model designs.

The second feature characteristic of success is an *incremental approach*. The idea is that increments of effort by developers should be rewarded by increments of value back to them, in the form of enhancements to productivity or accretion of measurable evidence in support of an assurance claim. The purpose of this is to enhance the intrinsic motivation for developers to apply evidence-based techniques on a routine basis in development, enhancement, and evaluation.

The rationale for this kind of incrementality is the simple economics of cost, benefit, time, and risk. Suppose, from the developer or evaluator perspective, there is an apparent uncertainty associated with the benefits of a particular action related to assurance. In this case, the action is less likely to be taken if the benefits are far in the future or if the action itself is costly (e.g., in taking time away from other activities with a more immediate reward). In these circumstances, the present value of the benefits may be too low or have too high a variance. These, combined with the lack of immediate benefit, may deter or defer a developer or evaluator from taking the action. We focus, therefore, on actions that are more granular and for which the benefits can be made more immediate. This enables a more explicitly motivated engagement by developers and evaluators, with greater potential for adaptability and hence the extent and immediacy of the benefit. Motivation for these security and quality practices is further enhanced by capacity for better measurement of immediate benefits and prediction of potential benefits.

Success in usability for developers and evaluators could greatly facilitate evaluation practices, including both increasing the overall level of capability for highly critical systems and, additionally, greatly enhancing productivity and agility in evaluation and certification by providing technical and tool support for the concurrent development of system capabilities and the assurance cases to support their acceptance for deployment.

7. **Usability for human operators**
   Challenge: Most modern complex systems include participation by human operators within the engineered system. How can the models, metaphors, and interactions be constructed to lead to less human error in the operation of systems and enhanced operator awareness to better thwart social-engineering and insider attacks.

   One way to frame the goal is to design metaphors and interaction mechanisms such that there is an effective connectivity among the actuality of system operation, the policies and metaphors in the minds of the operators, and policy intent of the system designers. For example, how can a particular intended policy be realized in configuring and operating a system? And: For a given system configuration, what is the actual policy realized? The goal, in other words, is to define models that enhances the semantic connection between the implemented security and safety policies and the models in the minds of the human operators and users.

8. **Prediction of the rate of occurrence of security breaches.**
   Challenge: Software organizations need to assess whether software is ready to release given the desired security goals for the product.

   Analysis techniques for estimation of security metrics are based on huge state spaces where most of the metric values are derived from states rarely encountered. The rate of occurrence of security breaches will relate to the degree to which the system has been fortified to prevent both random and systematic attacks. The rate of occurrence due to systematic attacks is may increase significantly faster than the volume of use of the system. More popular software systems and those that process attractive data assets attract significantly more attackers.

   Empirical evaluation goes hand-in-hand with models and analysis. Severely lacking are scientifically grounded techniques in experimental design and statistical analysis of real systems (in the field or in a laboratory testbed) that provide measurements, which correlate well with the vulnerability of a system, presence of intrusion, or resistance to intrusion.

9. **Attack-tolerant systems.**
   Challenge: Systems for which security is especially critical should be designed to a) auto-detect possibly correlated attacks (e.g., based on redundancy, acceptance-test, or watchdog principles); b) isolate or interfere with the activities of a potential or actual attack; and (3) recover a secure state and continue.

   Fault tolerant and resilient systems rely on redundancy and/or recovery to continue normal operation despite the presence of hardware or software faults. Similarly, the science behind an attack-tolerant system would recognize anomalies and identify user “intent”, as well as identify appropriate defense and isolation mechanisms, such as isolating the using to a redundant system. For example, during this time the attacker could continue what he/she believes is regular use while the activities could be carefully analyzed and logged for later study. If a suspected attacker is later deemed benevolent, their use could be resumed on the main system.

10. Isolation of attack-susceptible components

Challenge: Systems should be designed to minimize the adversary surface of a system.

The presence of human users of the system will prevent a secure system from having a closed form. However, a system could be architected such that some components need to account for the unpredictability of human users (attack-susceptible) and others can be isolated and then able to have more closed forms of security (trusted). The goal would be to architect system to maximize trusted components and minimize attack-susceptible components.

11. Security economics

Challenge: System developers do not understand the implications of their development choices on the security of a system; system purchasers do not demand security of their vendors/suppliers and do not understand the cost of their security choices; system design has to try and optimize among competing concerns such as cost, reliability, availability, performance, safety, and security .

Lack of understanding of how security attributes affect and are affected by other system attributes exists at all levels of the system, from circuitry to applications. System developers should better understand the economic impact of their development design choices, processes, practices and testing on the security of software-intensive systems and be better able to make tradeoffs between alternatives to obtain desired levels of security. System developers should also understand how design decisions in systems architecture, application programming interface (API) structuring, language design, and other engineering elements influences the capacity to reason about a system with respect to each of these various attributes. It is evident, for example, that types in programming languages have a tremendous benefit in reasoning about data access and representational integrity. Another example is the role of finite state models to model protocols, object state, and other characteristics. System developers should also understand, for particular attributes, reasoning mechanisms for supporting assessment of a design, model, or system component with respect to a particular attribute. This includes understanding what sorts of specifications must be made at the perimeter of components and within components in order to express intent and support the reasoning process.

System purchasers should understand the importance of demanding and then paying for secure software solutions based upon the economic impact of buying insecure software.