Emergent Behavior in Cybersecurity

HotSoS'14

Shouhuai Xu
Department of Computer Science, University of Texas at San Antonio

Abstract

We argue that emergent behavior is inherent to cybersecurity.

Informal Definition and Implication

Emergent behavior is a core concept in Complexity Science, although there is no universally accepted definition.

Definition (informal): A security property of a cybersystem exhibits ***emergent behavior*** if the property is not possessed by the underlying lower-level components of the cybersystem. (Simplest example: "1 + 1 > 2" effect)

Implication: At least some security properties cannot be understood by considering the lower-level components individually; instead, we must explicitly consider the interactions between the lower-level components. In other words, the composition approach has some fundamental limitations (like reductionism in Physics).

Inspiration from Cryptography

Complexity Science Comes to Rescue Again?

The (envisioned) Science of Cybersecurity:

- **Soul: Security (concepts)**
- **Brain: (Cybersecurity) Dynamics (kind of Complexity Science)**
- **Muscle & Blood: Complex System/Network, Stochastic Process, Dynamical System, Statistical Physics, Control Theory, Game Theory, Statistics, Algebraic Graph Theory, Algorithms, Software, Programming Language, etc.**

The Science of Cryptography:

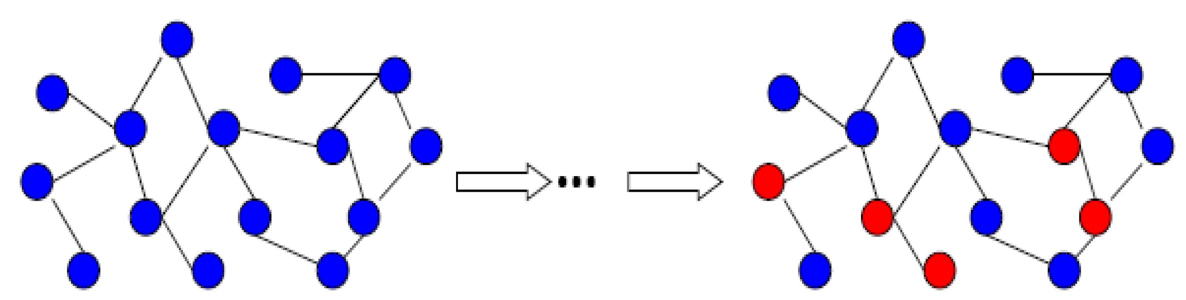
- **Soul: Security (concepts)**
- **Brain: Comp. Complexity Theory (kind of Complexity Science)**
- **Muscle & Blood: Probability Theory, Number Theory, Abstract Algebra, etc.**

For information about Cybersecurity Dynamics, see

http://www.cs.utsa.edu/~shxu/socs/

Example 1: Cyber Epidemics

Scenario: Illustration of cyber epidemics model (which is a specific kind of Cybersecurity Dynamics model).



(secure node compromised node)

Consider the simplest cyber epidemic model in two component networks: $G_i = (V_i, E_i)$, where V_i is the node set and E_i is the edge set for $i = 1, 2$.

$\lambda_1(G)$: the largest eigenvalue of the adjacency matrix of graph G
β: the defense capability in detecting and cleaning infected nodes
γ: the attack capability in infecting secure nodes

Suppose G_i is a complete graph with n_i nodes for $i = 1, 2$. Then, $\lambda_1(G_1) = n_1 - 1$ and $\lambda_1(G_2) = n_2 - 1$.

$$\lambda_1(G_i) < \beta/\gamma \quad \Rightarrow \quad \text{spreading dies out in } G_i$$
$$\lambda_1(G_i) > \beta/\gamma \quad \Rightarrow \quad \text{spreading does not die out in } G_i$$

Consider cybersystem $G_{1,2}$ obtained by interconnecting G_1 and G_2.

Suppose $G_{1,2}$ is also a complete graph with $n_1 + n_2$ nodes. Then, $\lambda_1(G_{1,2}) = n_1 + n_2 - 1$.

In many (if not all) cases, the defense capability β' and the attack capability γ' in $G_{1,2}$ are respectively the same as the defense capability β and the attack capability γ in G_1 and G_2. Since

$$\left.\begin{array}{lll} \lambda_1(G_i) & < & \beta/\gamma \\ \lambda_1(G_2) & < & \beta/\gamma \end{array}\right\} \not\Rightarrow \lambda_1(G_{1,2}) < \beta'/\gamma' = \beta/\gamma,$$

the spreading dies out in the two underlying component cybersystems, but does not die out in the interconnected cybersystem as long as $\lambda_1(G_{1,2}) > \beta/\gamma$.

This phenomenon applies to a class of cyber epidemic models.

Insight: Cyber epidemics properties exhibit emergent behavior because properties of cyber epidemics in a greater cybersystem cannot be determined by properties in the component cybersystems alone.

Example 2: Program Verification

Trace properties: In the field of program verification, Lamport proposed the safety-liveness framework of trace properties for analyzing concurrent programs. A trace is a finite or infinite sequence of states corresponding to an execution of a program; a trace property is a set of traces such that every trace, ***in isolation***, satisfies the same predicate. A safety property says that no ``bad thing" happens during the course of a program execution; a liveness property says that ``good thing" will eventually happen during the course of a program execution. **Both safety and liveness are trace properties**; every trace property is the intersection of a safety property and a liveness property (Alpern and Schneider).

Security properties are not (necessarily) trace properties (Goguen and Meseguer, Clarkson and Schneider, etc): (I) **Noninterference** is no trace property because it cannot be verified without examining the other traces in question. (II) **Information-flow** is no trace property because it cannot be verified by examining each trace alone. (III) **Average service response time** is no trace property because it depends on the response time in all traces.

Security properties can be trace hyperproperties: Clarkson and Schenider extended the concept of trace properties to trace hyperproperties, which are **sets of trace properties**. For example, information-flow, integrity and availability are trace hyperproperties (and intersections of some safety hyperproperties and some liveness hyperproperties).

Insight: Hyperproperties exhibit emergent behavior because the verification procedure must examine across multiple traces, which can accommodate interactions between component systems.

Example 3: Cryptography

Cryptographic secure multiparty computation allows multiple parties $P_1, \ldots, P_m$, each having a respective secret $x_1, \ldots, x_m$, to compute a function $f(x_1, \ldots, x_m)$ such that no information about the x_i's is leaked except for what is implied by the output of the function.

Feasibility result (Yao; Goldreich et al.): Under some standard cryptographic assumptions, any polynomial-time computable function f(.,…) can be securely computed in the ***standalone*** setting (i.e., the protocol executes in isolation).

Standalone vs. concurrent execution: When cryptographic protocols are used as building-blocks in larger applications/systems, they may execute concurrently (rather than in isolation). Are the cryptographic protocols, which are provably secure when executed in isolation, still secure when they are concurrently called by larger applications/systems?

Impossibility result: There exist classes of functions that can be securely computed by running some cryptographic protocols in isolation, **but cannot be securely computed when the protocols execute concurrently**. In order to make cryptographic multiparty computation protocols secure when they are used as building-blocks for constructing larger cybersystems, we need to make extra assumptions, such as that majority of the parties P_1, …, P_m are not compromised.

Insight: Cryptographic properties exhibit emergent behavior because there are functions that can be securely computed in the standalone setting but cannot be securely executed in the concurrent setting.

Acknowledgement

AFOSR Grant # FA9550-09-1-0165 and ARO Grant # W911NF-13-1-0370.