goto;

GOTO
AARHUS 2023

#GOTOaar

Principles for Secure and Reliable Systems

GOTO Aarhus 2023

Eleanor Saitta / Systems Structure Ltd.

What is a System?

Systems exist to do things in the world

To be useful, they need to have certain emergent properties

Whole-system properties which occur in a specific context

Require unified effort to deliver

Properties you care about:

- Correctness
- Performance
- Efficiency
- Reliability
- Observability
- Security
- Resilience

What is Security?

A secure system is one that:

- Enables a chosen set of people to predictably accomplish specific goals
- Does so in the face of actions by a chosen set of adversaries
- Predictably prevents that chosen set of adversaries from

What is Security?

Alternately:

- Reliability and correctness of outcomes in the presence of an adversary
- Close-loop defense of outcomes

What is Resilience?

The ability of a system to deal with unforeseen modes of failure without complete failure

Resilience is a property of humans, not code

Designing for Resilient Security

Designing both processes and technical systems in accordance with specific principles leads to desired emergent properties

Properties of technical artifacts vs. properties of human processes

Component Principles

A selection of interesting system design principles:

- Statelessness/Logiclessness
- Immutability and Ephemerality
- Canonical Stores
- Unlinkability
- Least Mechanism

State and Logic

Services should either do computation or hold state, not both

Complex components are unpredictable

Immutability and Ephemerality

Data, configuration, and memory are all state

Immutable systems eliminate unnecessary state

Respinning a cluster resets state

Minimal, Canonical State

Every piece of state should exist canonically in exactly one place

As few systems as possible should be stores of state

Any duplicated state must be validated

Unlinkability

Privacy and anonymity are ill-defined

X piece of data is unlinkable to Y piece of data under these assumptions

Code is Not an Asset

- Complexity adds potential vulnerabilities geometrically
- Use the simplest working mechanism for features
- Have as few features as needed for what you do

- You spend lines of code to buy features
- Every line of code is an ongoing cost
- Every tool and library is also an ongoing cost
- Velocity averages out; technical debt is drag
- Most security debt is dark

Process Principles

And a few for the human side of the org:

- Declare and Generate
- Design for Failure
- Decide at the Edge
- Slack
- Incentivize What You Want

Declare, don't Program

Declarative configurations are easier for both humans and computers to create, compose, and validate

Use automatic memory management, parser generators, strongly typed languages, and state machine generators

Mitigations Always Fail

Kill Bug Classes

Security engineering changes that don't involve killing bug classes are emergency response work

...unless those changes kill traversal instead

Make a plan for each class and layer in advance and crosscheck

Design for Failure

Failure and compromise are inevitable

Design components and systems to handle both predictable and unpredictable types of failures

Think about security controls as a whole, assuming that some layers will always fail

Build the system you'd like to have during a compromise or outage

Decentralize Decisionmaking

Empower teams and engineers to work autonomously, so decisions can happen where people have full context

Focus on coordination and communication over control

Ensure teams have thick horizontal relationships outside of formal processes

Slack

Resilience requires teams to have downtime

Improving any emergent property takes more time than the bare minimum

Apply hard caps to feature velocity, ensure people take vacations, have large on-call rotations, and track out of hours work

Incentives

People do not do work they're told to do but incentivized not to do

Look at what your management, bonus, evaluation, and promotion structure encourages

Use Conway's law intentionally; your team structure is part of your technical architecture

Product Security

- You get to design your attacker's motivation level and the problems they have to solve
- Spend as much time designing unhappy paths as happy ones
- Know where each automated business or security decision in your flows
- Document this before each sprint and check it after

You are responsible for the impact of your work on people's lives.

Personas to Examine

- A domestic violence victim seeking an abortion
- A queer teen
- A union organizer

Quick tips for starting from zero

Ditch all your Windows boxes — use Macs/Chromebooks and ditch Office for Workspace

Yubikeys for everyone, everything tied to SSO

Get some Thinkst Canaries in prod/if you have an office

Backup everything and make sure it's tested

Track where your data is and be careful where it goes

Treat code as an expense, not an asset

Include maintenance when costing new SaaS tools

Startup looking to get serious about security?

Let's talk.

ella@structures.systems
@dymaxion@infosec.exchange

Eleanor Saitta
Systems Structure Ltd.

Don’t forget to
vote for this session
in the **GOTO Guide app**