Velocity

@aaronrinehart @verico_io #chaosengineering
Security Precognition

@aaronrinehart  @verico_io #chaosengineering
“Resilience is the story of the outage that never happened.”
- John Allspaw
About A.A. Ron

- CTO of Stealthy Startup
- Former Chief Security Architect @UnitedHealth responsible for security engineering strategy
- Led the DevOps and Open Source Transformation at UnitedHealth Group
- Former (DOD, NASA, DHS, CollegeBoard)
- Frequent speaker and author on Chaos Engineering & Security
- Pioneer behind Security Chaos Engineering
- Led ChaoSlingr team at UnitedHealth
In this Session we will cover
Our systems have evolved beyond human ability to mentally model their behavior.
Our systems have evolved beyond human ability to mentally model their behavior.
Complex?

Continuous Delivery

Distributed Systems

Blue/Green Deployments

Containers

Immutable Infrastructure

Infracode

Service Mesh

Microservice Architectures

Automation Pipelines

Continuous Integration

Cloud Computing

CI/CD

DevOps

API

Auto Canaries

Circuit Breaker Patterns
Security?

Mostly Monolithic
- Prevention focused
- Defense in Depth

Expert Systems
- Poorly Aligned
- Requires Domain Knowledge

Stateful in nature
- Adversary Focused
- DevSecOps not widely adopted

DevSecOps not widely adopted
Simplify?
## Software Only Increases in Complexity

<table>
<thead>
<tr>
<th>More Abstract</th>
<th>Scripting / interpreted languages</th>
<th>High / middle level languages</th>
<th>Assembly language</th>
<th>Machine code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Perl, Python, Shell, Java</td>
<td>C, C++</td>
<td>Intel X86, etc (first layer of human-readable code)</td>
<td>Hexadecimal representations of binary code read by the operating system</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Binary code</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Binary code

Binary code read by hardware - not human-readable
Software Complexity

Accidental Complexity

Essential Complexity
Woods Theorem: “As the complexity of a system increases, the accuracy of any single agent’s own model of that system decreases”

- Dr. David Woods
How well do you really understand how your system works?
Difficult to Grok behavior
So what does all of this have to do with **Security**?
Failure Happens.

Amazon: Sorry something went wrong on our end. Please go back and try again or go to Amazon's home page.

Emergency Alert: BALLISTIC MISSILE THREAT INBOUND TO HAWAII. SEEK IMMEDIATE SHELTER. THIS IS NOT A DRILL.

Right Now: COMPUTER OUTAGE IMPACTS SOUTHWEST AIRLINES. LONG LINES, DELAYS SEEN AT AIRPORTS NATIONWIDE.
Incidents & System Outages are Expensive
Security Incidents are **Subjective** in Nature
We really don't know very much

Where?  Why?  Who?

How?  What?
Lets face it, **when outages happen**...
Teams spend too much time reacting to outages instead of building more resilient systems.
“Response” is the problem with Incident Response
“Chaos Engineering is the discipline of experimenting on a distributed system in order to build confidence in the system’s ability to withstand turbulent conditions.”
Who is doing Chaos?
PRINCIPLES OF CHAOS ENGINEERING

Last Update: 2017 April

Chaos Engineering is the discipline of experimenting on a distributed system in order to build confidence in the system’s capability to withstand turbulent conditions in production.
People Operate Differently when they expect things to fail
OMG!

What are your robot serial numbers?
Awesome!
The **Normal Condition** of a Human & Systems they Build is to **FAIL**
We need failure to **Learn** & **Grow**
Let's Flip the Model

Post Mortem = Preparation
Bring Order through Chaos
Use Chaos Engineering to initiate Objective Feedback Loops about Security Effectiveness
Proactively Manage & Measure

- Validate Runbooks
- Measure Team Skills
- Determine Control Effectiveness
- Learn new insights into system behavior
- Transfer knowledge
- Build a learning culture
Testing vs. Experimentation
Security Crayon Differences

Noisy distributed system behavior
Not geared for Cascading Events
Point-in-time even if Automated
Performed by Security Teams with Specialized skill sets
Security Chaos Differences

Distributed Systems Focus
Goal: Experimentation
Human Factors focused
Small Isolated Scope
Focus on Cascading Events
Performed by Mixed Engineering Teams in Gameday
During business hours
2018 Causes of Data Breaches

Ponemon Institute

27% Human error

48% Malicious or criminal attack

25% System glitch
2018 Causes of Data Breaches

Ponemon Institute

- 27% Human error
- 25% System glitch
- 48% Malicious or criminal attack
2018 Causes of Data Breaches

- Human error: 27%
- System glitch: 25%
- Malicious or criminal attack: 48%
2018 Causes of Data Breaches

27% Human error

48% Malicious or criminal attack

25% System glitch
Negligent insiders are individuals who cause a data breach because of their carelessness, as determined in a post data breach investigation. Malicious attacks can be caused by hackers or criminal insiders (employees, contractors or other third parties).
Proactively
Manage & Measure
Continuous SECURITY Validation
Build Confidence in What Actually Works
So how does it work?
Stop looking for better answers and start asking better questions.

- John Allspaw
What is the system actually doing?
What is the system actually doing?

Has it done this before?
What is the system actually doing?
Has it done this before?
Why is it behaving that way?
What is the system actually doing?
Has it done this before?
Why is it behaving that way?
What is it supposed to do next?
What is the system actually doing?

Has it done this before?

Why is it behaving that way?

What is it supposed to do next?

How did it get into this state?
How does My Security Really Work?
What evidence do I have to prove it?
An Open Source Tool
ChatOps Integration
Configuration-as-Code
Example Code & Open Framework

Serverless App in AWS
100% Native AWS
Configurable Operational Mode & Frequency
Opt-In | Opt-Out Model
Hypothesis: If someone accidentally or maliciously introduced a misconfigured port then we would immediately detect, block, and alert on the event.
Result: Hypothesis disproved. Firewall did not detect or block the change on all instances. Standard Port AAA security policy out of sync on the Portal Team instances. Port change did not trigger an alert and log data indicated successful change audit. However we unexpectedly learned the configuration mgmt tool caught change and alerted the SoC.
More Experiment Examples

- Software Secret Clear Text Disclosure
- Permission collision in Shared IAM Role Policy
- Disabled Service Event Logging
- Introduce Latency on Security Controls
- API Gateway Shutdown

- Internet exposed Kubernetes API
- Unauthorized Bad Container Repo
- Unencrypted S3 Bucket
- Disable MFA
- Bad AWS Automated Block Rule
Q&A

@aaronrinehart  aaron@verica.io
Thank you!

@aaronrinehart  aaron@verica.io