Give me $1M
Give me $1M
Quantifying Risk

QCon SF 2019

Markus De Shon
(mdeshon@netflix.com)
Measuring and Managing Information Risk

Jack Freund

& Jack Jones

fairinstitute.org
Frequency $\times$ Magnitude ($$) \quad (\text{of Loss})$
What is a loss?
First steps of a risk analysis

- Assets
- Architecture
- Control architecture
- Loss scenarios
Meet Sam the Sponge
His best friend Peter
His boss Mr. Prawn
The Prawn Patty
The secret recipe
Controls Architecture

- Only one copy
- Not memorized
- Kept in safe
- Trusted handlers
Recipe loss scenarios

- Confidentiality
  - Competitor
  - Public
- Integrity
  - crUD
- Availability
  - Unavailable
Threat
Hazard
Tardigrade
Estimate frequency

Security Engineers

Range

0 ——— ∞
Calibration

0.1

0.01

0.001
Tardigrade steals recipe
Patty Pirate steals recipe
Estimate magnitude

- Asset owner
- Decompose
- Low $\rightarrow$ High (90% CI)
- US$
Model magnitude with lognormal

Low loss

High loss

90% CI
Why Money?

- Composable (A+B)
- Comparable (A>B)
- Interpretable by business

What about:

- Priceless? → Implicit valuation
- Intangible? → Inverse of ROI on existing investments
Magnitude: Tardigrade

- Recipe unavailable → sales stop (primary)
  - 1 day @ $10K → $10K
  - 100 days → $1M
- Knockoffs at Tardigrade’s. Lose customers (primary)
  - 10 @ $100 → $1K
  - 1,000 → $100K
- Total:
  - Low: $11,000
  - High: $1,100,000

Expected Loss: $2,930
Magnitude: Patty Pirate

Recipe unavailable → lost sales (Primary loss)
- 10 days @ $10K → $100K
- 100 days → $1M

No Prawn Patties anywhere
→ immediate collapse, fires. dystopia. (Secondary, external)
- 10 days @ $1M → $10M
- 100 days → $100M

Totals:
- Low: $10,100,000
- High: $101,000,000

Expected Loss: $4,080,000
Engineering a Safer World
Systems Thinking Applied to Safety

Nancy G. Leveson
Controller and process
● Identify Assets
● Study Architecture
● Define Control architecture
● Identify loss scenarios
● Estimate frequency
● Estimate low/high magnitude
● Calculate expected loss
import math
import numpy as np
from scipy.stats import lognorm, norm

def get_magnitude(lo, hi):
    # Calculate the mean mu in log space
    mu = (math.log(lo) + math.log(hi)) / 2.
    factor = -0.5 / norm.ppf(0.05)
    sigma = factor * (math.log(hi) - math.log(lo))
    distribution = lognorm(sigma, scale=math.exp(mu))
    return distribution

0.01 * get_magnitude(11000, 1100000).mean()