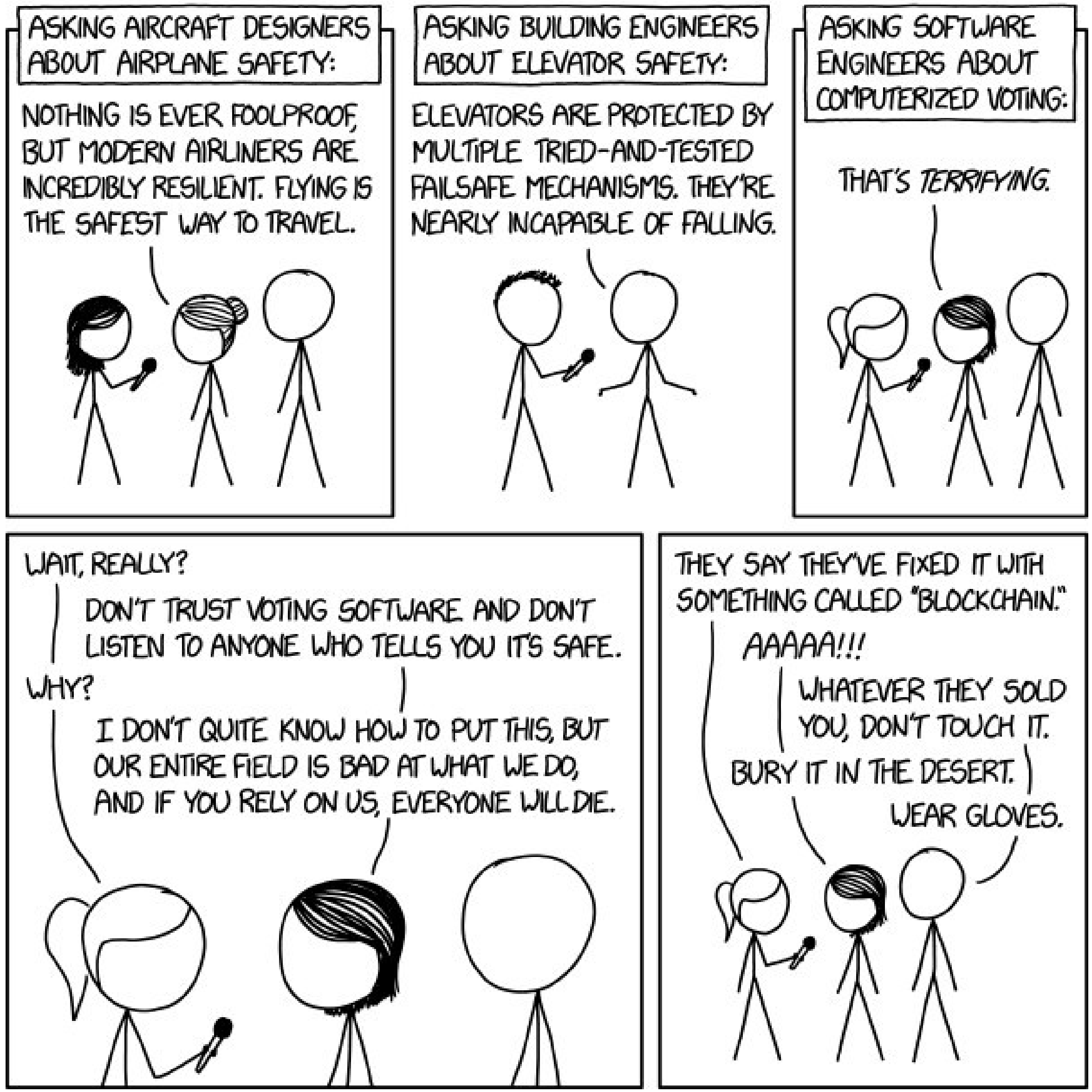
Threat modelling for developers

Arne Padmos

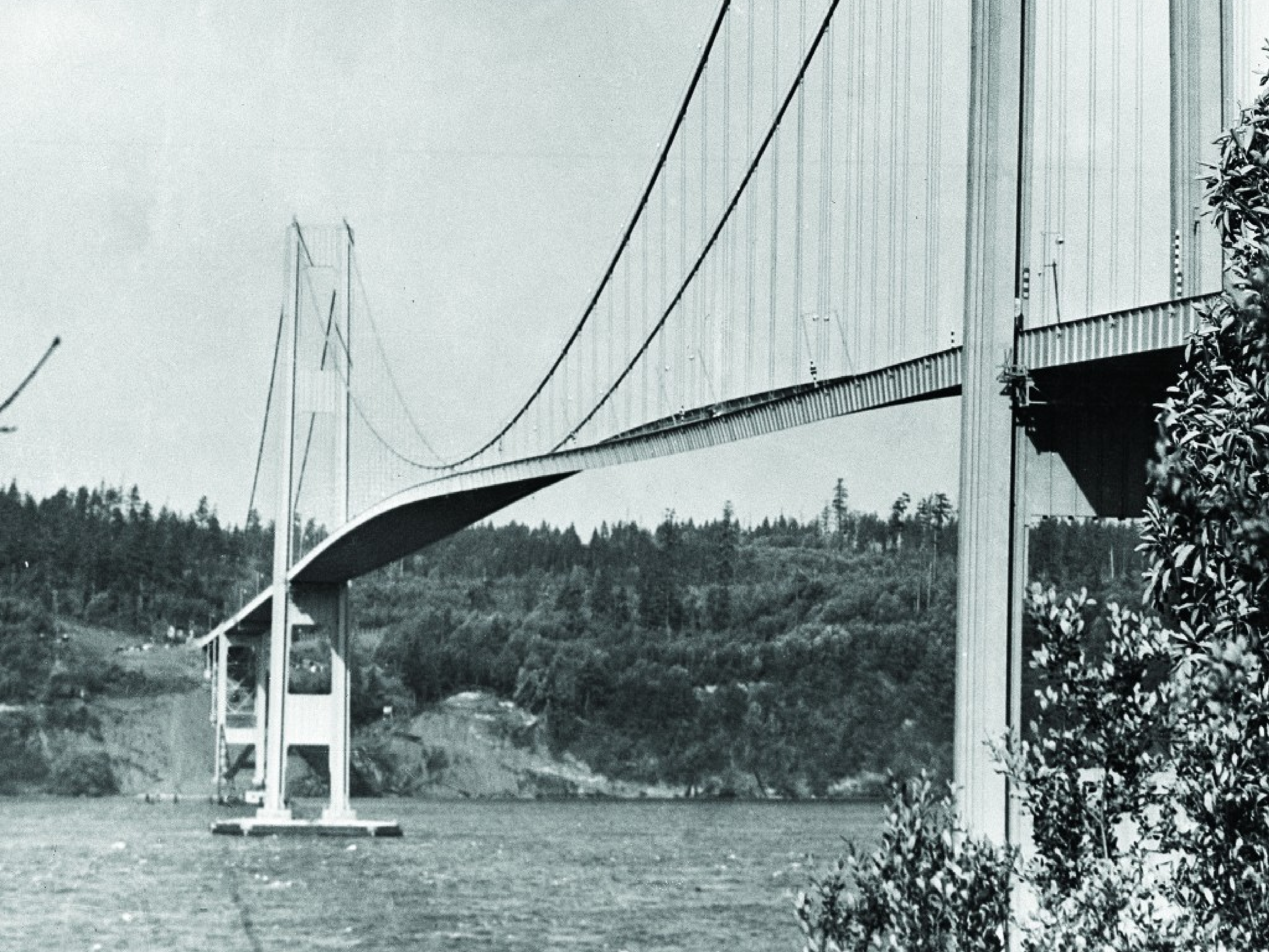

ASKING AIRCRAFT DESIGNERS ABOUT AIRPLANE SAFETY:
NOTHING IS EVER FOOLPROOF, BUT MODERN AIRLINERS ARE INCREDIBLY RESILIENT. FLYING IS THE SAFEST WAY TO TRAVEL.
ASKING BUILDING ENGINEERS ABOUT ELEVATOR SAFETY:
ELEVATORS ARE PROTECTED BY MULTIPLE TRIED-AND-TESTED FAILSAFE MECHANISMS. THEY'RE NEARLY INCAPABLE OF FALLING.
ASKING SOFTWARE ENGINEERS ABOUT COMPUTERIZED VOTING:
THAT'S *TERRIFYING.*
WAIT, REALLY?
DON'T TRUST VOTING SOFTWARE AND DON'T LISTEN TO ANYONE WHO TELLS YOU IT'S SAFE.
WHY?
I DON'T QUITE KNOW HOW TO PUT THIS, BUT OUR ENTIRE FIELD IS BAD AT WHAT WE DO, AND IF YOU RELY ON US, EVERYONE WILL DIE.
THEY SAY THEY'VE FIXED IT WITH SOMETHING CALLED "BLOCKCHAIN."
AAAAA!!!
WHATEVER THEY SOLD YOU, DON'T TOUCH IT.
BURY IT IN THE DESERT.
WEAR GLOVES.

xkcd

Safety

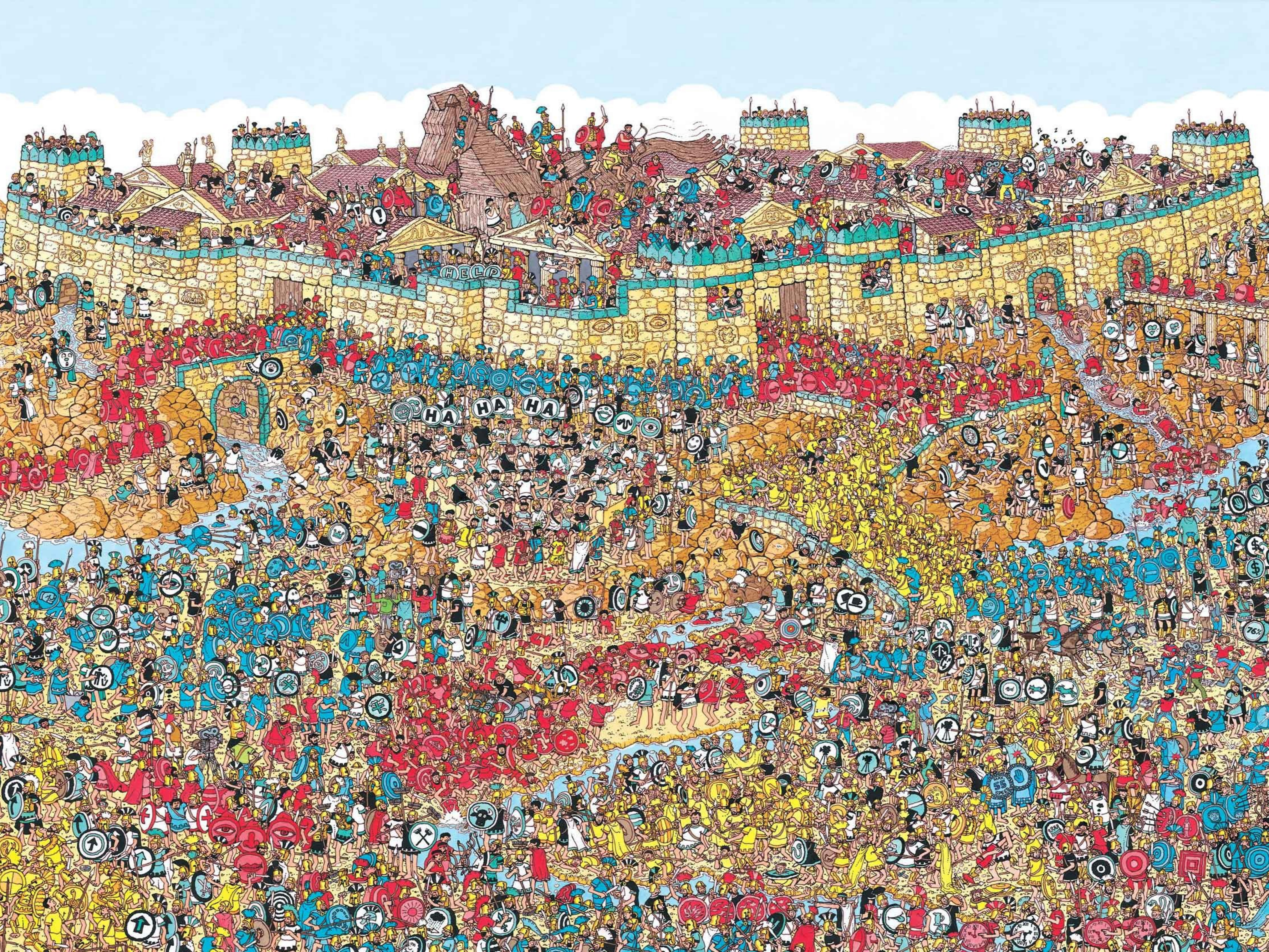
vs

Security



Warner Bros

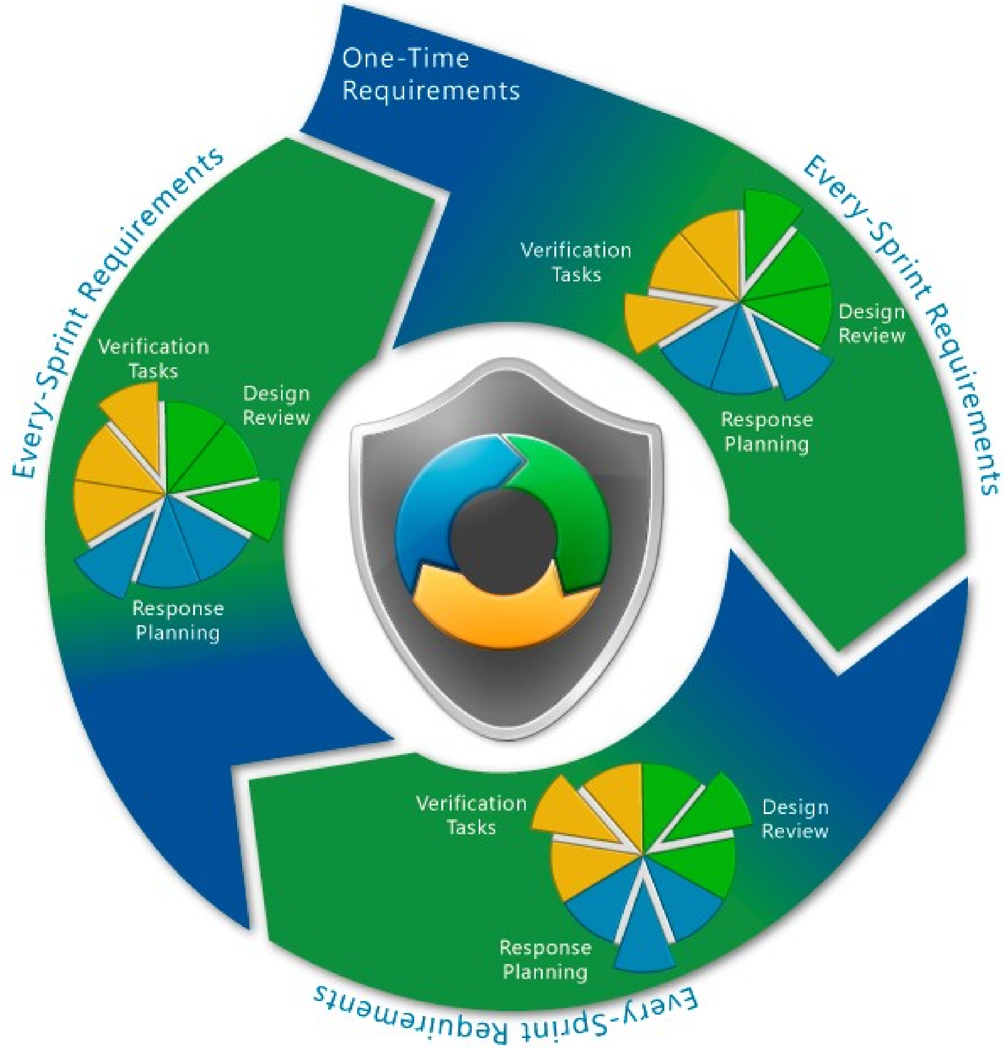
Are we doomed?

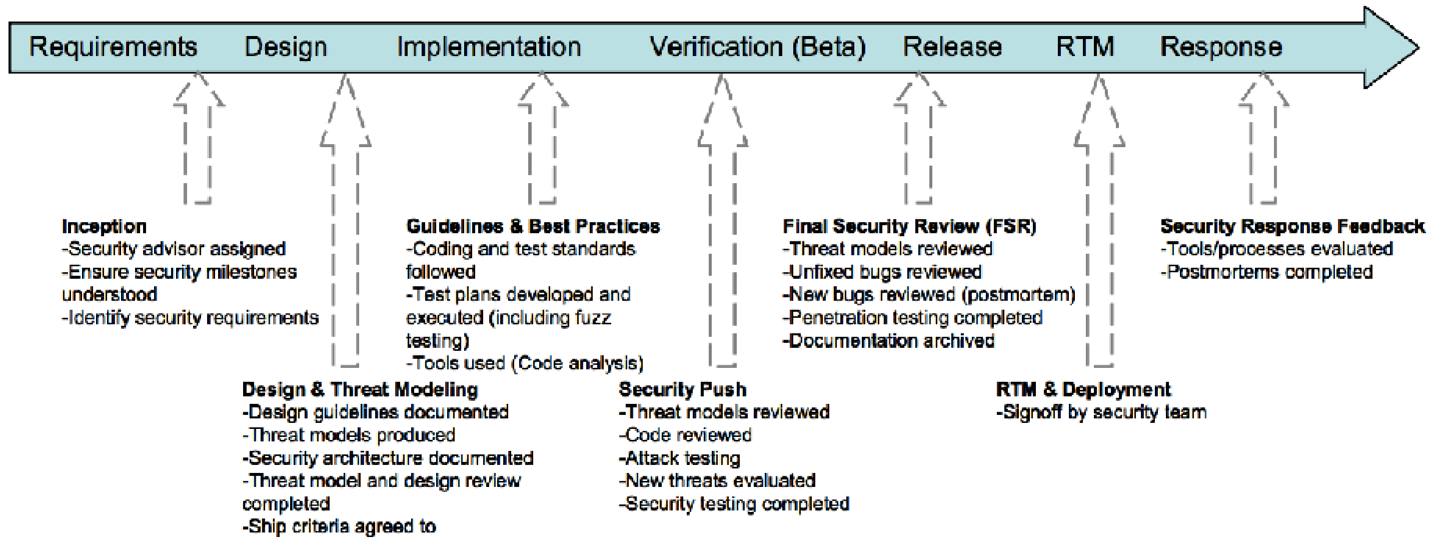

HELP
HA HA HA HA

“ Building security in ”

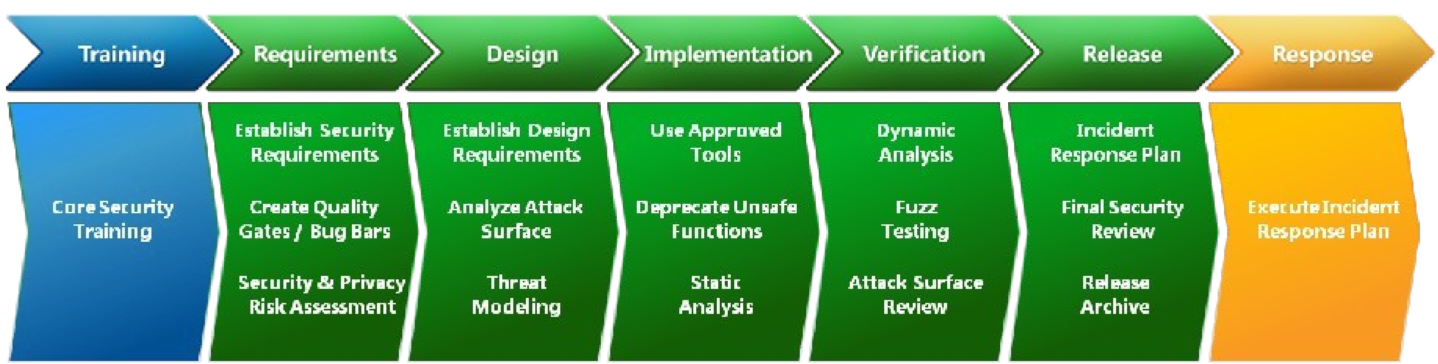
“ Security by design ”

“ Shifting security left ”


One-Time Requirements
Every-Sprint Requirements
Verification Tasks
Design Review
Response Planning
Every-Sprint Requirements
Verification Tasks
Design Review
Response Planning
Every-Sprint Requirements
Verification Tasks
Design Review
Response Planning



Requirements
Design
Implementation
Verification (Beta)
Release
RTM
Response
Inception
-Security advisor assigned
-Ensure security milestones understood
-Identify security requirements
Guidelines & Best Practices
-Coding and test standards followed
-Test plans developed and executed (including fuzz testing)
-Tools used (Code analysis)
Final Security Review (FSR)
-Threat models reviewed
-Unfixed bugs reviewed
-New bugs reviewed (postmortem)
-Penetration testing completed
-Documentation archived
Security Response Feedback
-Tools/processes evaluated
-Postmortems completed
Design & Threat Modeling
-Design guidelines documented
-Threat models produced
-Security architecture documented
-Threat model and design review completed
-Ship criteria agreed to
Security Push
-Threat models reviewed
-Code reviewed
-Attack testing
-New threats evaluated
-Security testing completed
RTM & Deployment
-Signoff by security team



Training
Requirements
Design
Implementation
Verification
Release
Response
Core Security Training
Establish Security Requirements
Create Quality Gates / Bug Bars
Security & Privacy Risk Assessment
Establish Design Requirements
Analyze Attack Surface
Threat Modeling
Use Approved Tools
Deprecate Unsafe Functions
Static Analysis
Dynamic Analysis
Fuzz Testing
Attack Surface Review
Incident Response Plan
Final Security Review
Release Archive
Execute Incident Response Plan

“*If we … could do only one thing to improve software security … we would do threat modelling every day of the week.*”

— Howard & Lipner

“ If we … could do only one thing to improve software security … we would do ***threat modelling*** *every day of the week.”*

— Howard & Lipner

Requirements engineering
&
Architectural analysis

What’s your threat model?

(security assumptions)

On the Security of Public Key Protocols

DANNY DOLEV AND ANDREW C. YAO, MEMBER, IEEE

***Abstract*—Recently the use of public key encryption to provide secure network communication has received considerable attention. Such public key systems are usually effective against passive eavesdroppers, who merely tap the lines and try to decipher the message. It has been pointed out, however, that an improperly designed protocol could be vulnerable to an active saboteur, one who may impersonate another user or alter the message being transmitted. Several models are formulated in which the security of protocols can be discussed precisely. Algorithms and characterizations that can be used to determine protocol security in these models are given.**

I. Introduction

THE USE of public key encryption [1], [11] to provide secure network communication has received considerable attention [2], [7], [8], [10]. Such public key systems are usually very effective against a "passive" eavesdropper, namely, one who merely taps the communication line and tries to decipher the intercepted message. However, as pointed out in Needham and Schroeder [8], an improperly designed protocol could be vulnerable to an "active" saboteur, one who may impersonate another user and may alter or replay the message. As a protocol might be compromised in a complex way, informal arguments that assert the security for a protocol are prone to errors. It is thus desirable to have a formal model in which the security issues can be discussed precisely. The models we introduce will enable us to study the security problem for families of protocols, with very few assumptions on the behavior of the saboteur.

We briefly recall the essence of public key encryption (see [1], [11] for more information). In a public key system, every user X has an *encryption function* E_x and a *decryption function* D_x, both are mappings from $\{0, 1\}^*$ (the set of all finite binary sequences) into $\{0, 1\}^*$. A secure public directory contains all the (X, E_x) pairs, while the decryption function D_x is known only to user X. The main requirements on E_x, D_x are:

1) $E_x D_x = D_x E_x = 1$, and
2) knowing $E_x(M)$ and the public directory does not reveal anything about the value M.

Thus everyone can send X a message $E_x(M)$, X will be able to decode it by forming $D_x(E_x(M)) = M$, but nobody other than X will be able to find M even if $E_x(M)$ is available to them.

We will be interested mainly in protocols for transmitting a secret plaintext M between two users. To give an idea of the way a saboteur may break a system, we consider a few examples. A message sent between parties in the network consists of three fields: the sender's name, the receiver's name, and the text. The text is the encrypted part of the message. We will write a message in the format: sender's name, text, receiver's name.

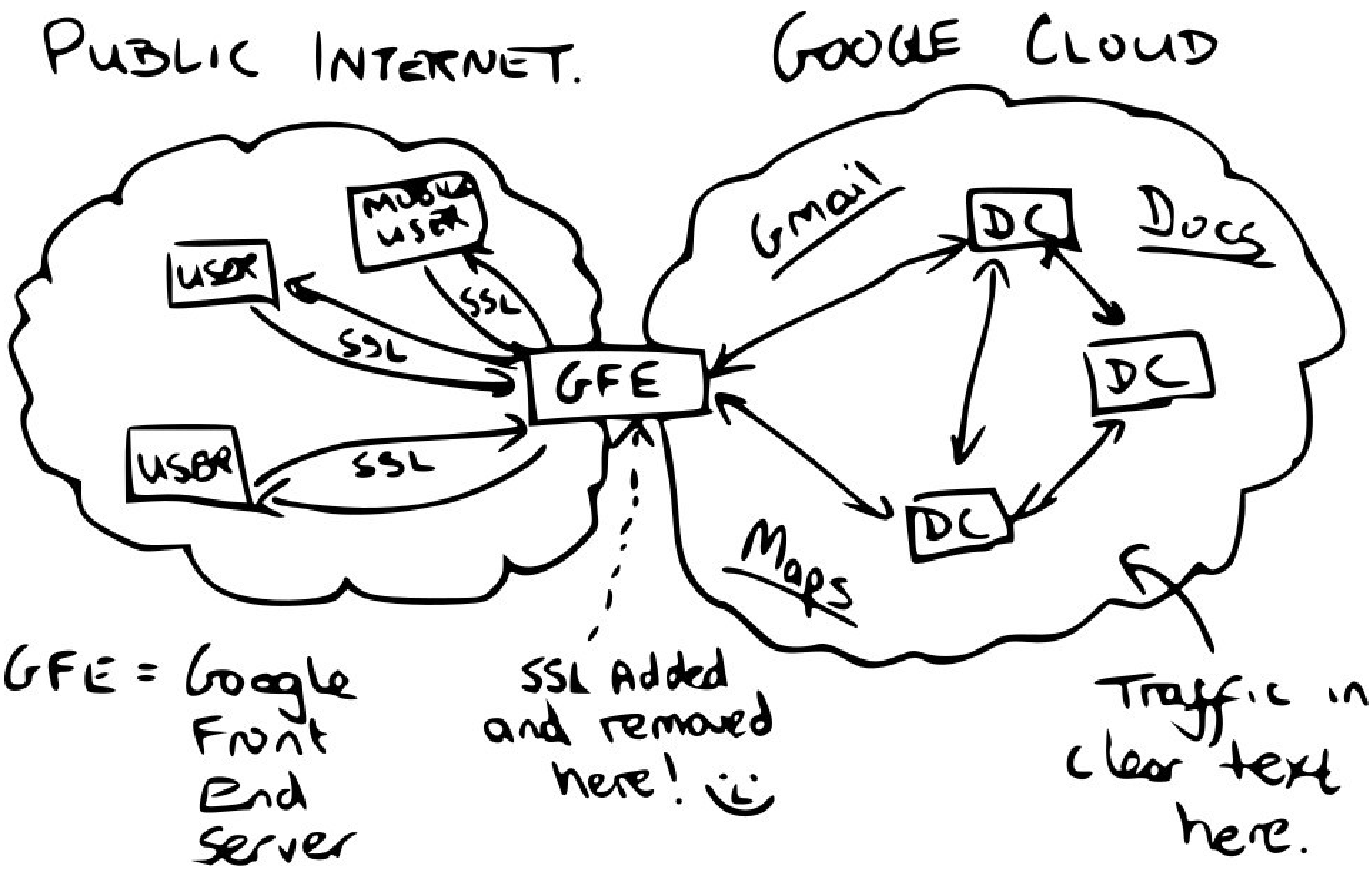
Example 1: Consider the following protocol for sending

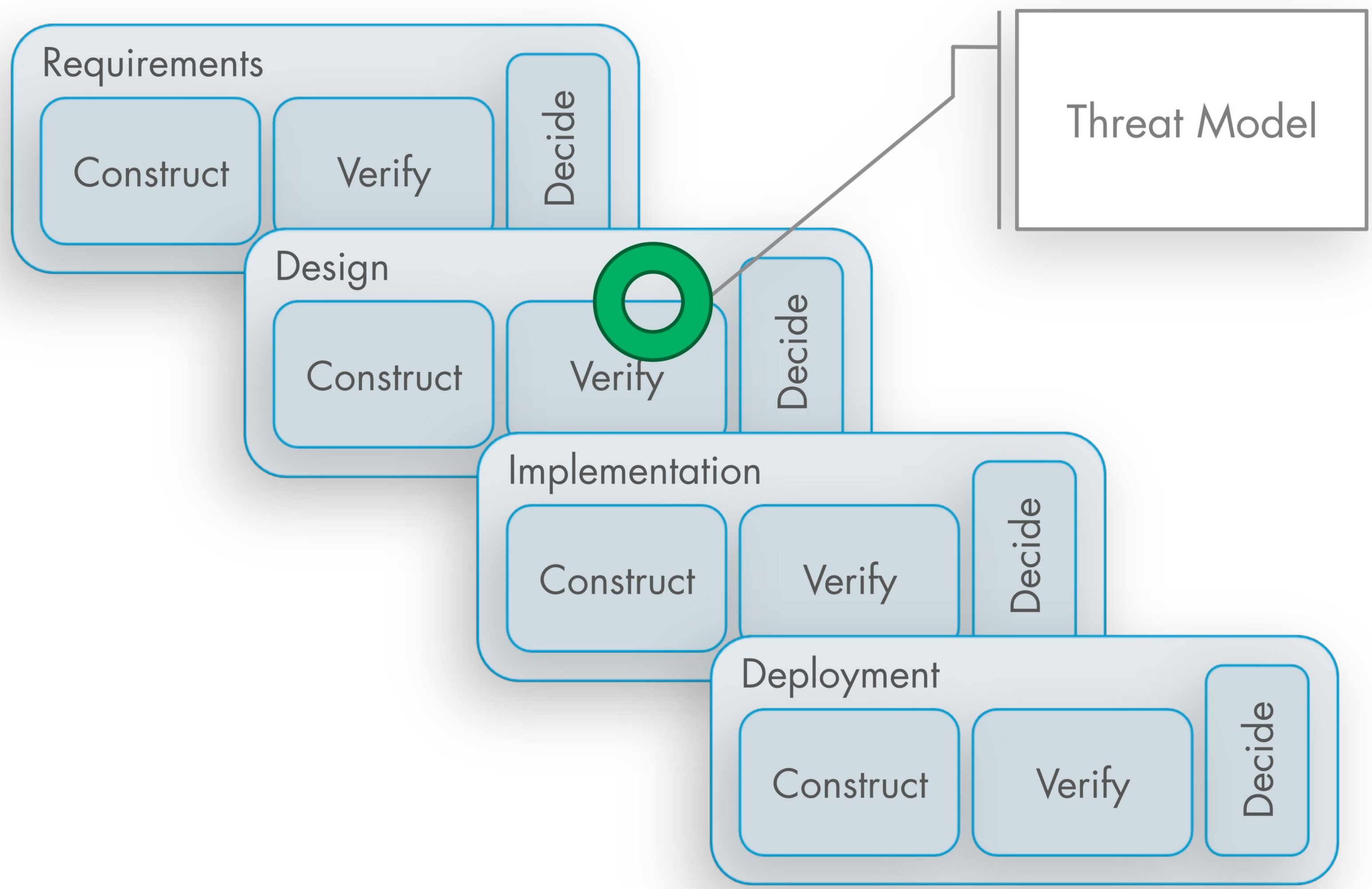
Manuscript received July 15, 1981; revised August 8, 1982. This work was supported in part by ARPA under Grant MDA-903-80-C-102 and by National Science Foundation under Grant MCS-77-05313-A01. This paper was partially presented at the 22nd Annual IEEE Symposium on Foundations of Computer Science, Nashville, TN, October 28–30, 1981.

“ *More precisely, we will assume the following about a saboteur:* ”

- obtain any message
- initiate any conversation
- be a receiver to any user

Utagawa Kuniyoshi


PUBLIC INTERNET.
GOOGLE CLOUD
USER
MOBILE USER
USER
SSL
SSL
SSL
GFE
Gmail
Docs
Maps
DC
DC
DC
DC
GFE = Google Front End server
SSL Added and removed here! :-)
Traffic in clear text here.



Eleanor Saitta

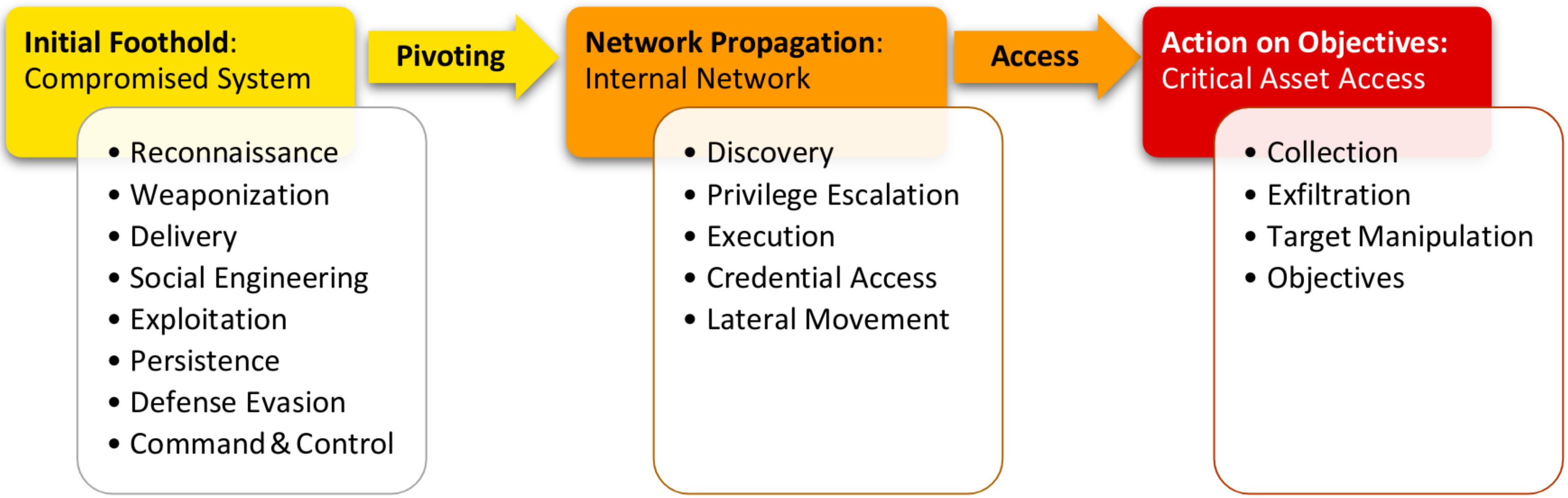
What could possibly go wrong?

What could possibly go wrong?

& how

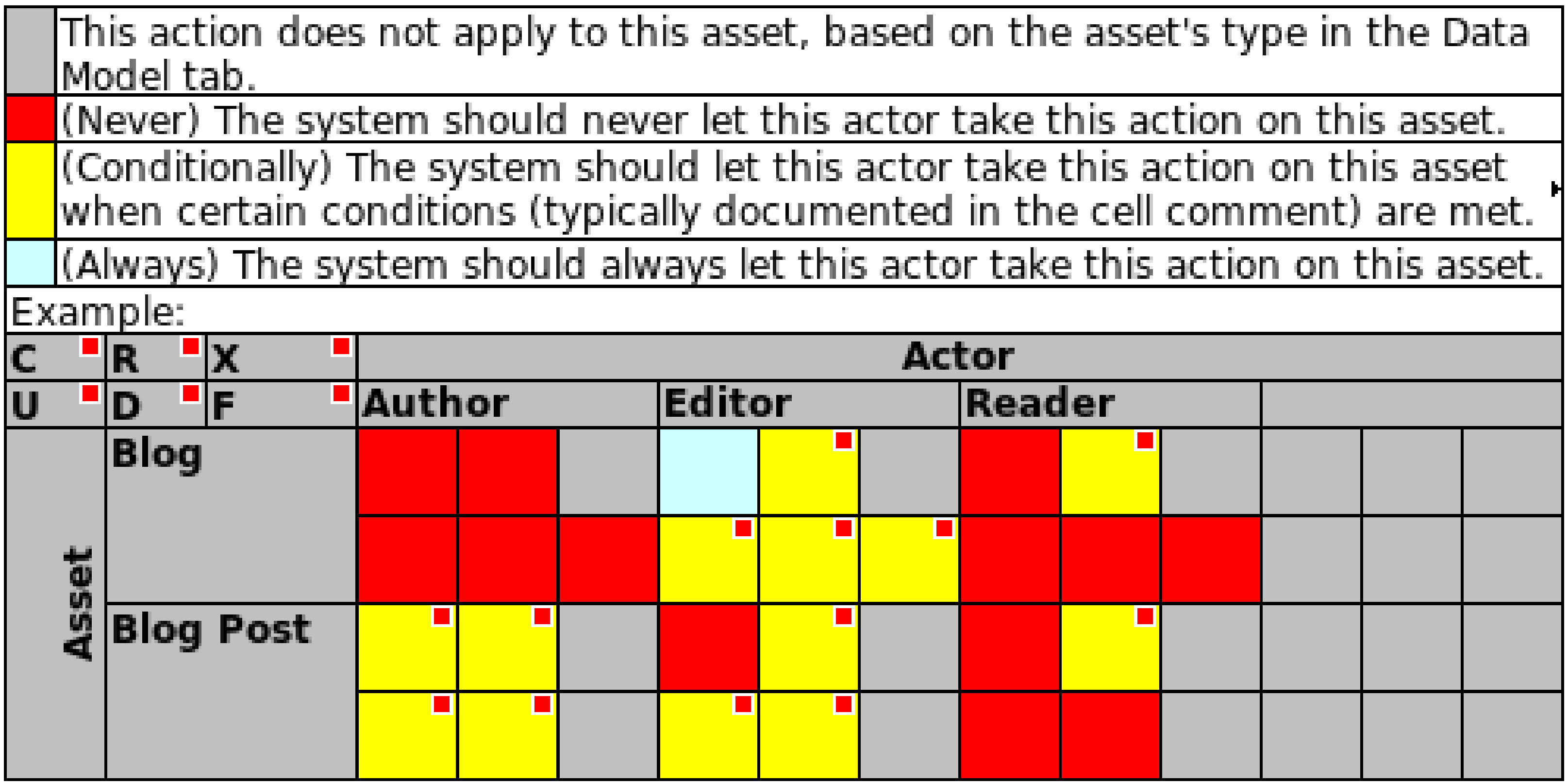
Types of threat modelling

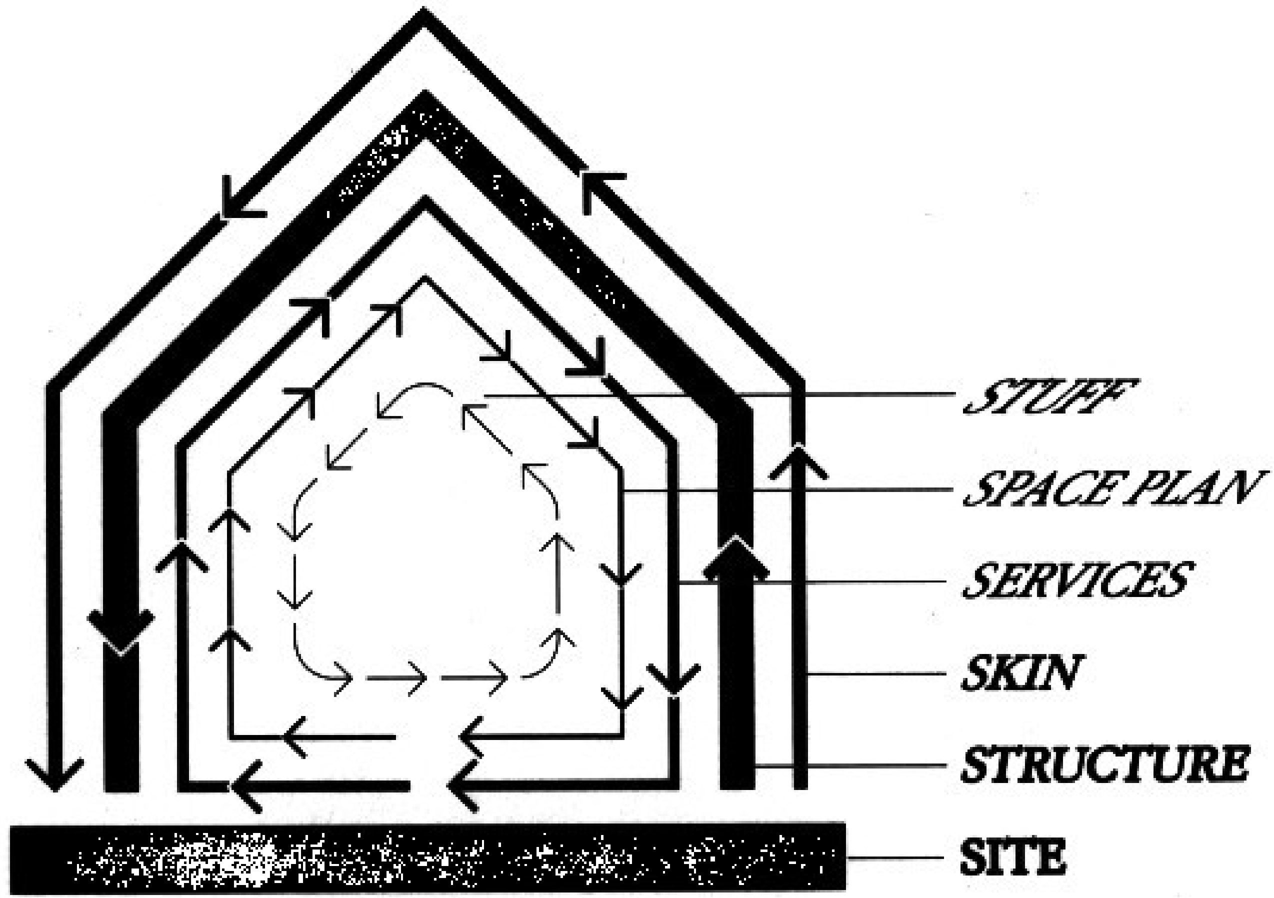
- Attacker-centric
- Asset-centric
- System-centric


Initial Foothold:
Compromised System
Pivoting
Network Propagation:
Internal Network
Access
Action on Objectives:
Critical Asset Access
• Reconnaissance
• Weaponization
• Delivery
• Social Engineering
• Exploitation
• Persistence
• Defense Evasion
• Command & Control
• Discovery
• Privilege Escalation
• Execution
• Credential Access
• Lateral Movement
• Collection
• Exfiltration
• Target Manipulation
• Objectives

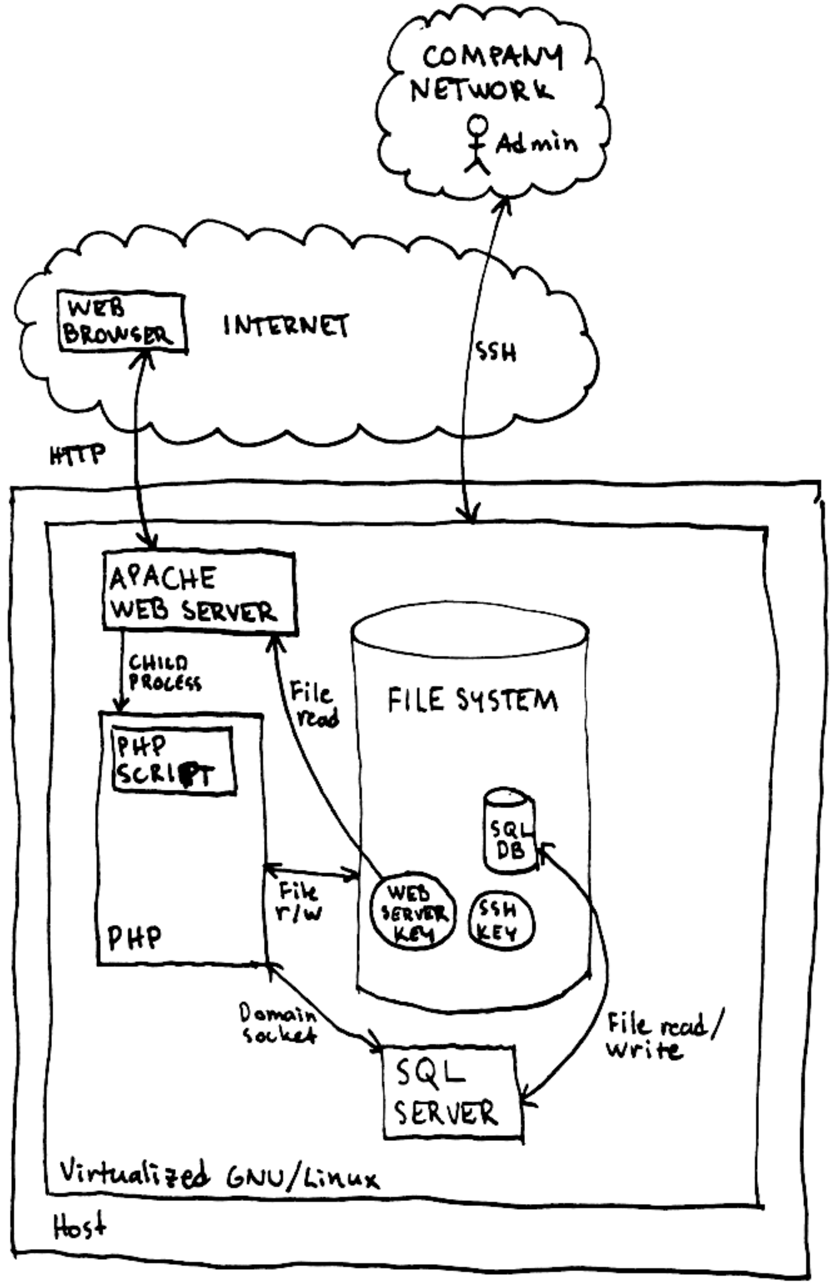
Paul Pols

Cyril Davenport


This action does not apply to this asset, based on the asset's type in the Data Model tab.
(Never) The system should never let this actor take this action on this asset.
(Conditionally) The system should let this actor take this action on this asset when certain conditions (typically documented in the cell comment) are met.
(Always) The system should always let this actor take this action on this asset.
Example:
C
R
X
Actor
U
D
F
Author
Editor
Reader
Asset
Blog
Blog Post



Stewart Brand


COMPANY NETWORK
Admin
WEB BROWSER
INTERNET
SSH
HTTP
APACHE WEB SERVER
CHILD PROCESS
File read
FILE SYSTEM
PHP SCRIPT
SQL DB
File r/w
WEB SERVER KEY
SSH KEY
PHP
Domain socket
File read/write
SQL SERVER
Virtualized GNU/Linux
Host

Popular approaches
(*system-centric*)

- STRIDE
- Trike
- PASTA

Relevant questions

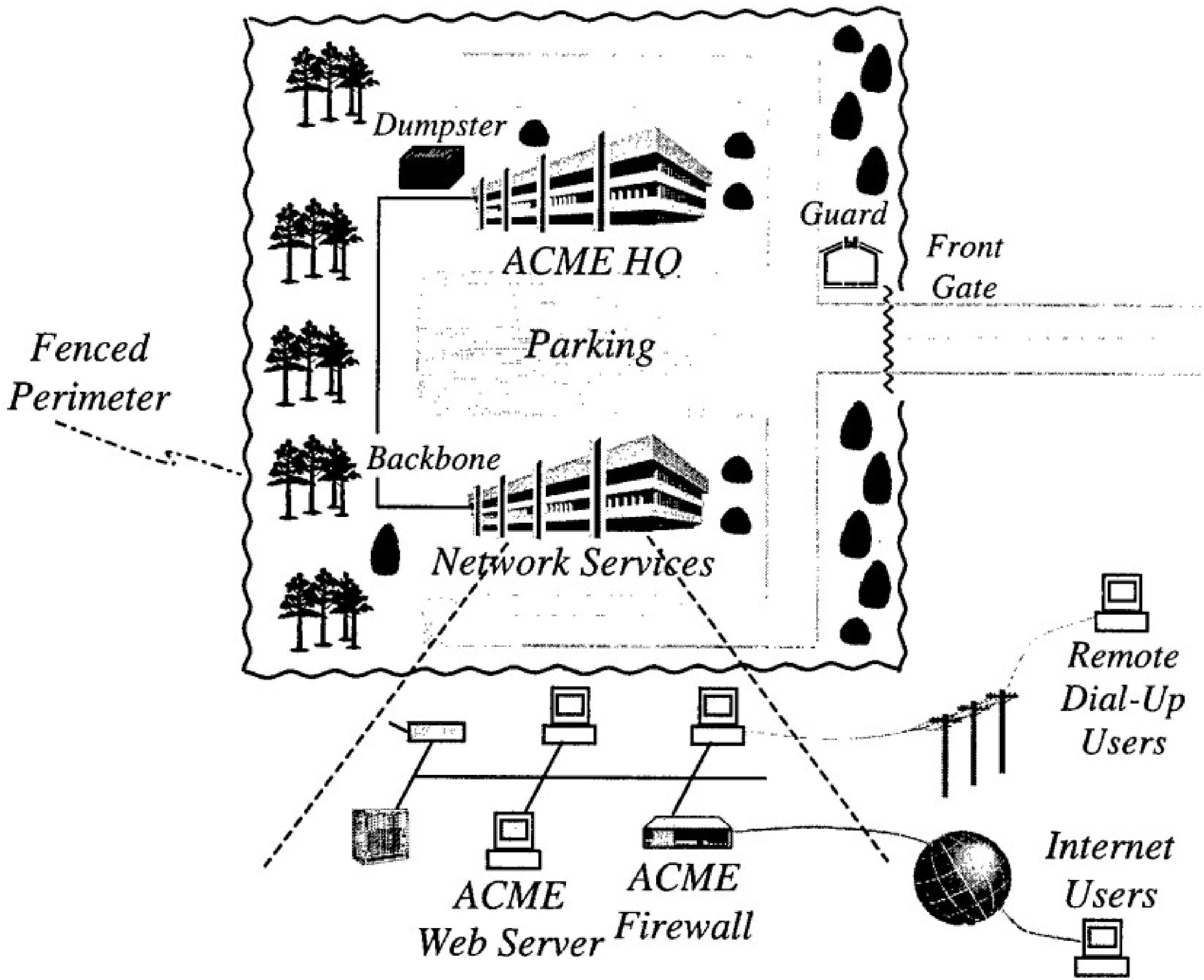
1. What are we working on?
2. What can go wrong?
3. What are we going to do?
4. Did we do a good job?

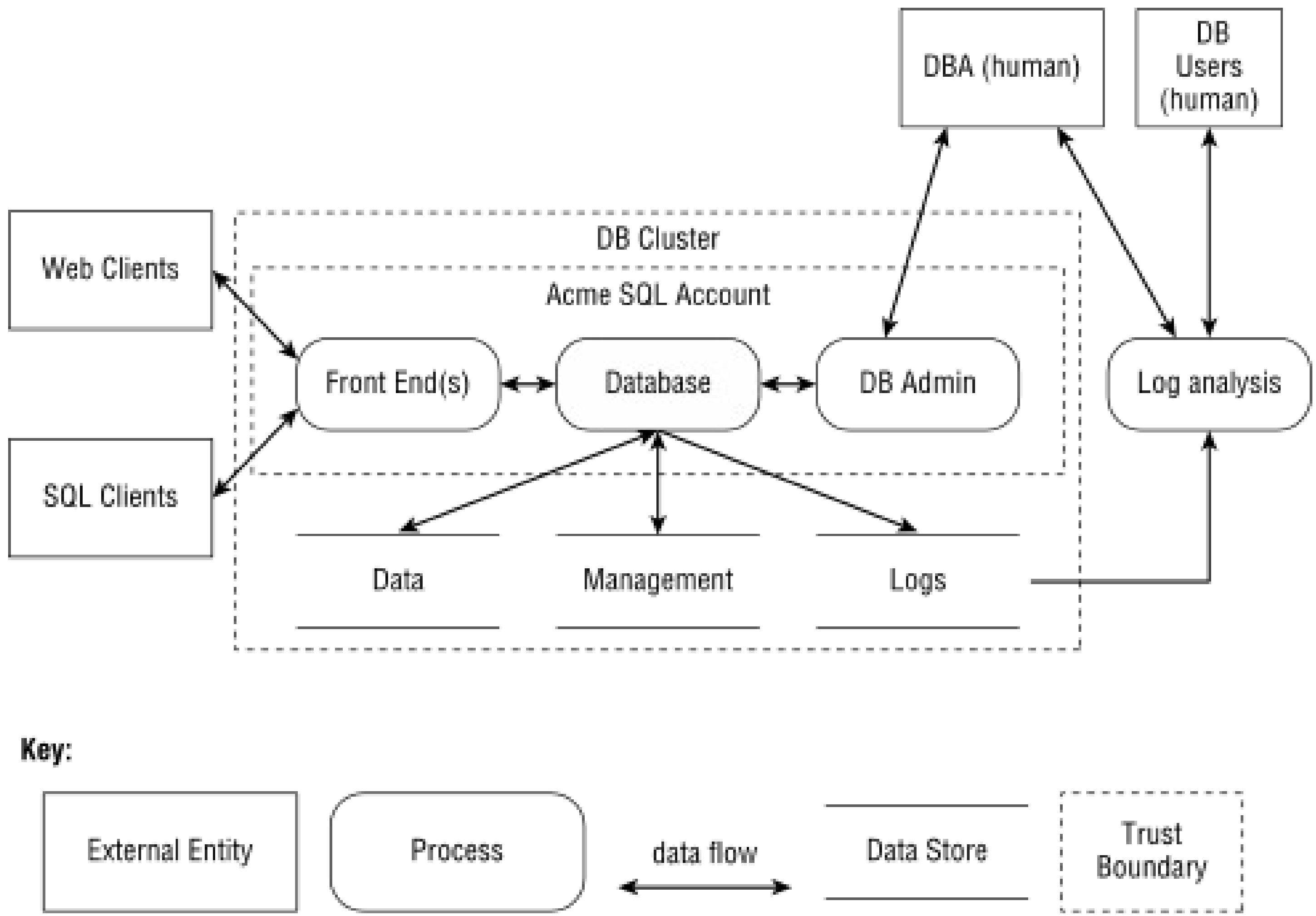
Lightweight methodology

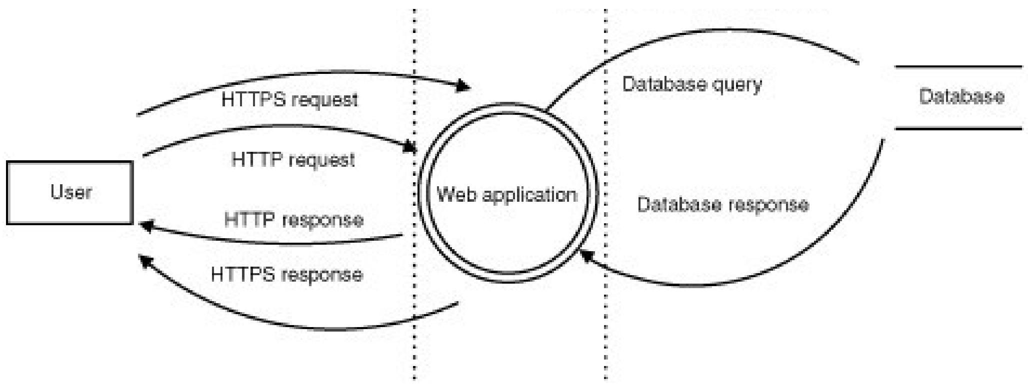
1. Draw data flows
2. Elicit threats
3. Ranking + controls
4. Check your work

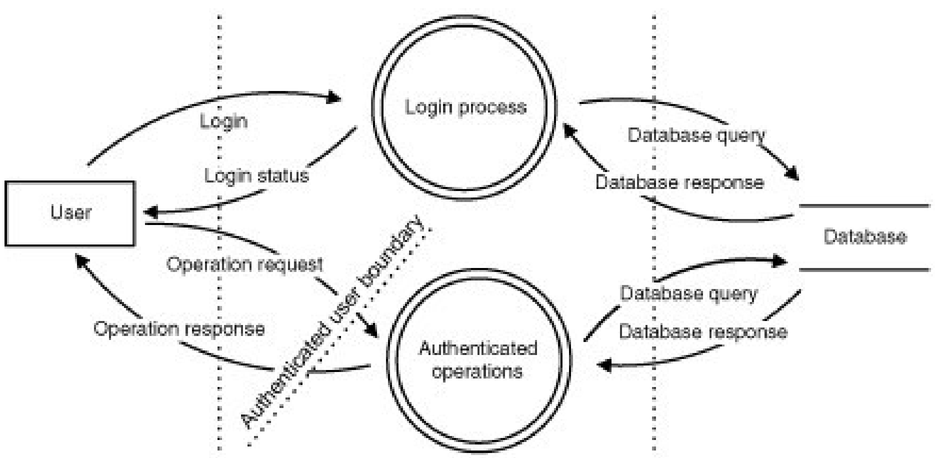
Lightweight methodology

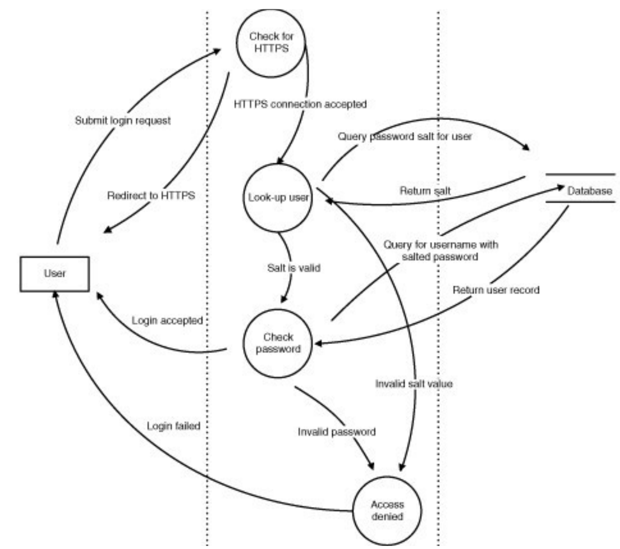
1. Draw data flows
2. Elicit threats
3. Ranking + controls
4. Check your work

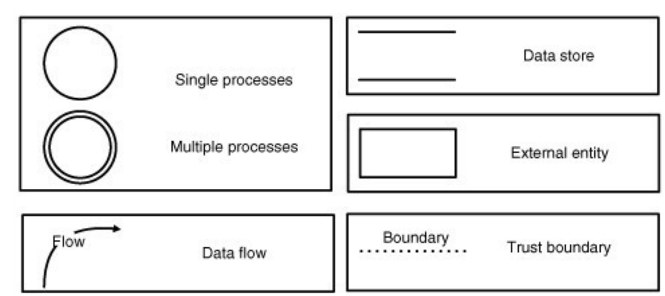

Dumpster
ACME HQ
Guard
Front
Gate
Parking
Fenced
Perimeter
Backbone
Network Services
Remote
Dial-Up
Users
Internet
Users
ACME
Web Server
ACME
Firewall


DBA (human)
DB
Users
(human)
Web Clients
DB Cluster
Acme SQL Account
Front End(s)
Database
DB Admin
Log analysis
SQL Clients
Data
Management
Logs
Key:
External Entity
Process
data flow
Data Store
Trust
Boundary

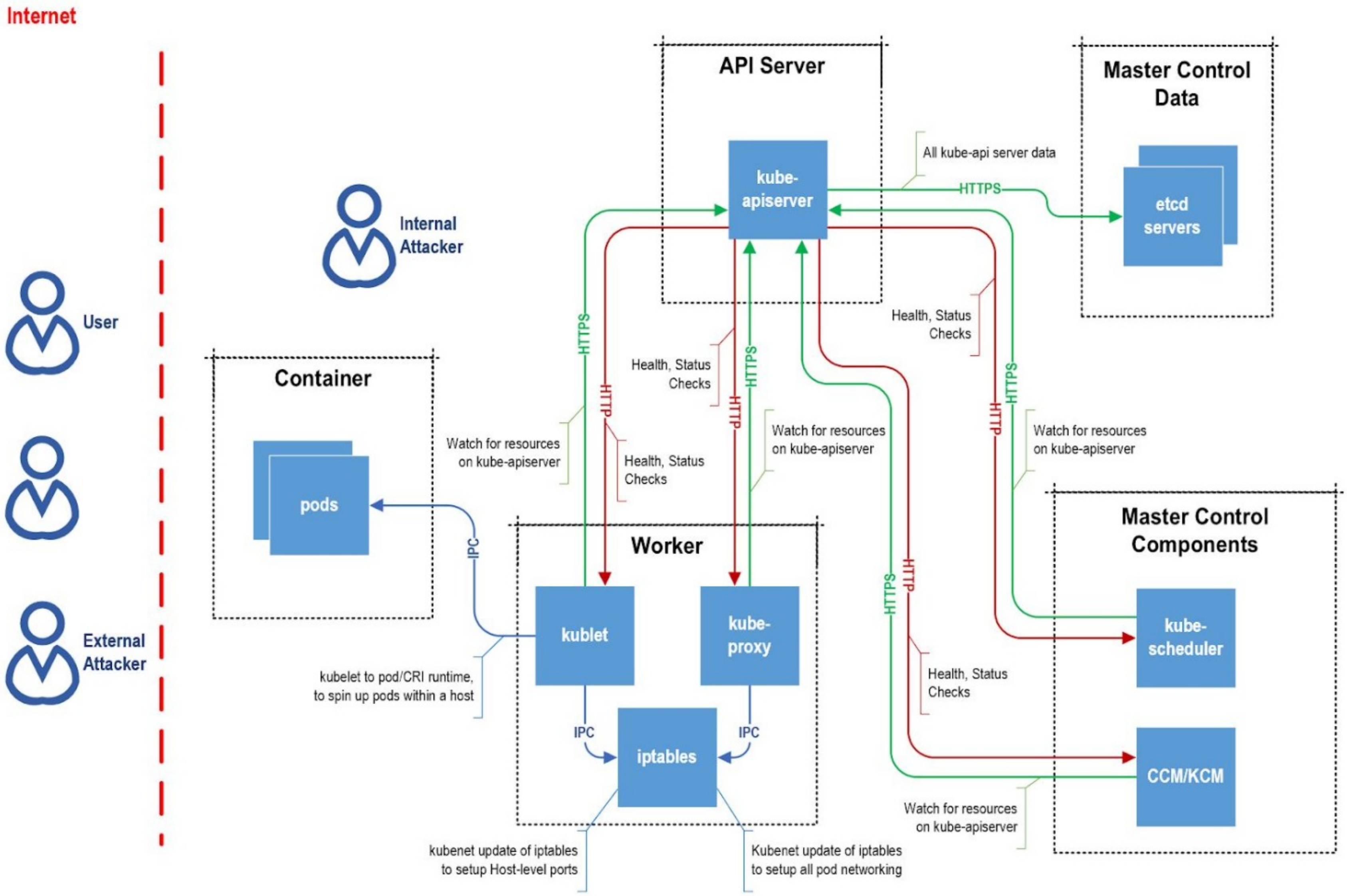








Mark Dowd et al.



Internet
API Server
Master Control
Data
kube-
apiserver
All kube-api server data
HTTPS
etcd
servers
Internal
Attacker
User
External
Attacker
Container
pods
IPC
HTTPS
HTTP
Health, Status
Checks
HTTPS
HTTP
Health, Status
Checks
HTTP
HTTPS
Watch for resources
on kube-apiserver
Watch for resources
on kube-apiserver
Watch for resources
on kube-apiserver
Health, Status
Checks
Worker
kublet
kube-
proxy
Master Control
Components
kube-
scheduler
HTTPS
HTTP
Health, Status
Checks
CCM/KCM
kubelet to pod/CRI runtime,
to spin up pods within a host
IPC
IPC
iptables
Watch for resources
on kube-apiserver
kubenet update of iptables
to setup Host-level ports
Kubenet update of iptables
to setup all pod networking

Lightweight methodology

1. Draw data flows
2. Elicit threats
3. Ranking + controls
4. Check your work

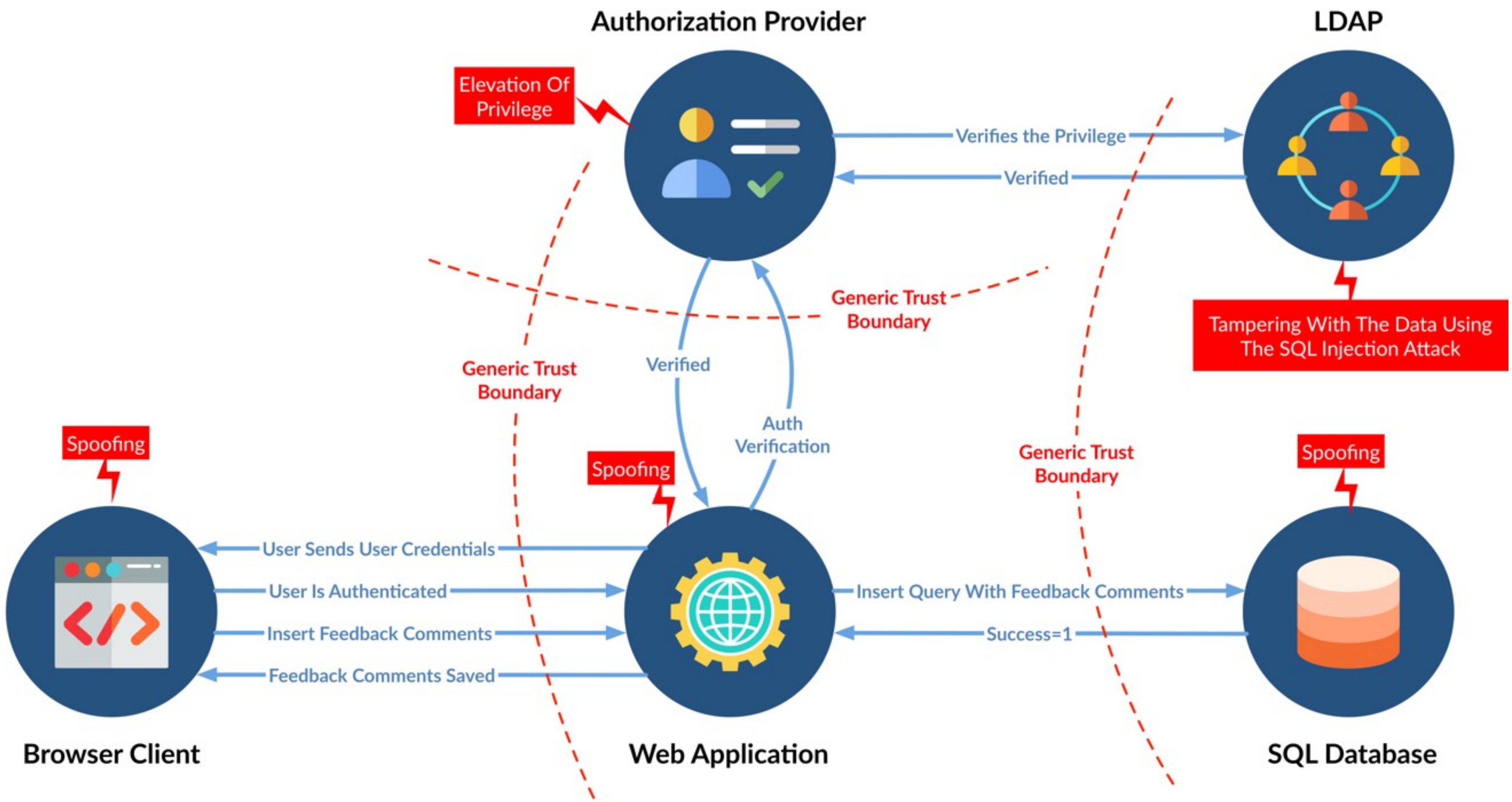
Confidentiality
Integrity
Availability

Authentication
Authorisation
Accountability

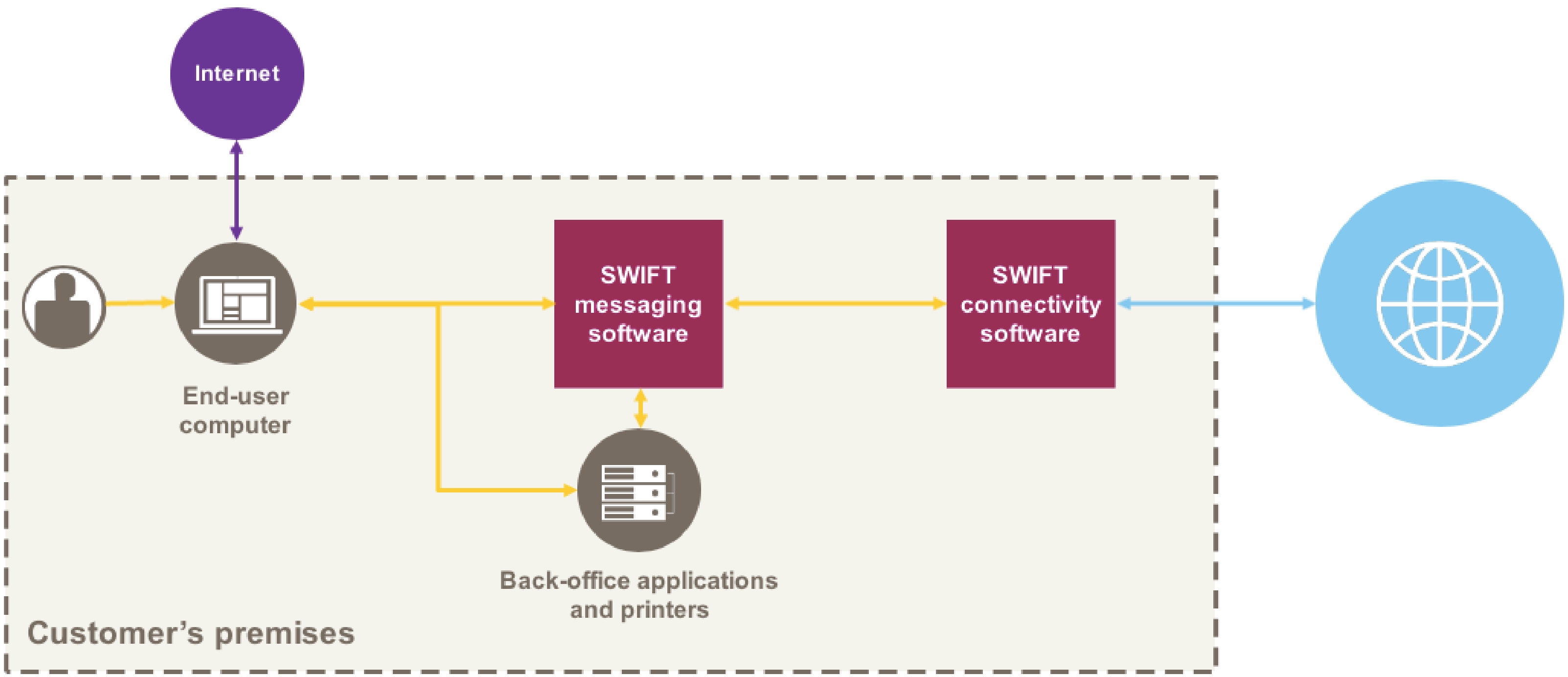
Information disclosure
Tampering
Denial of service

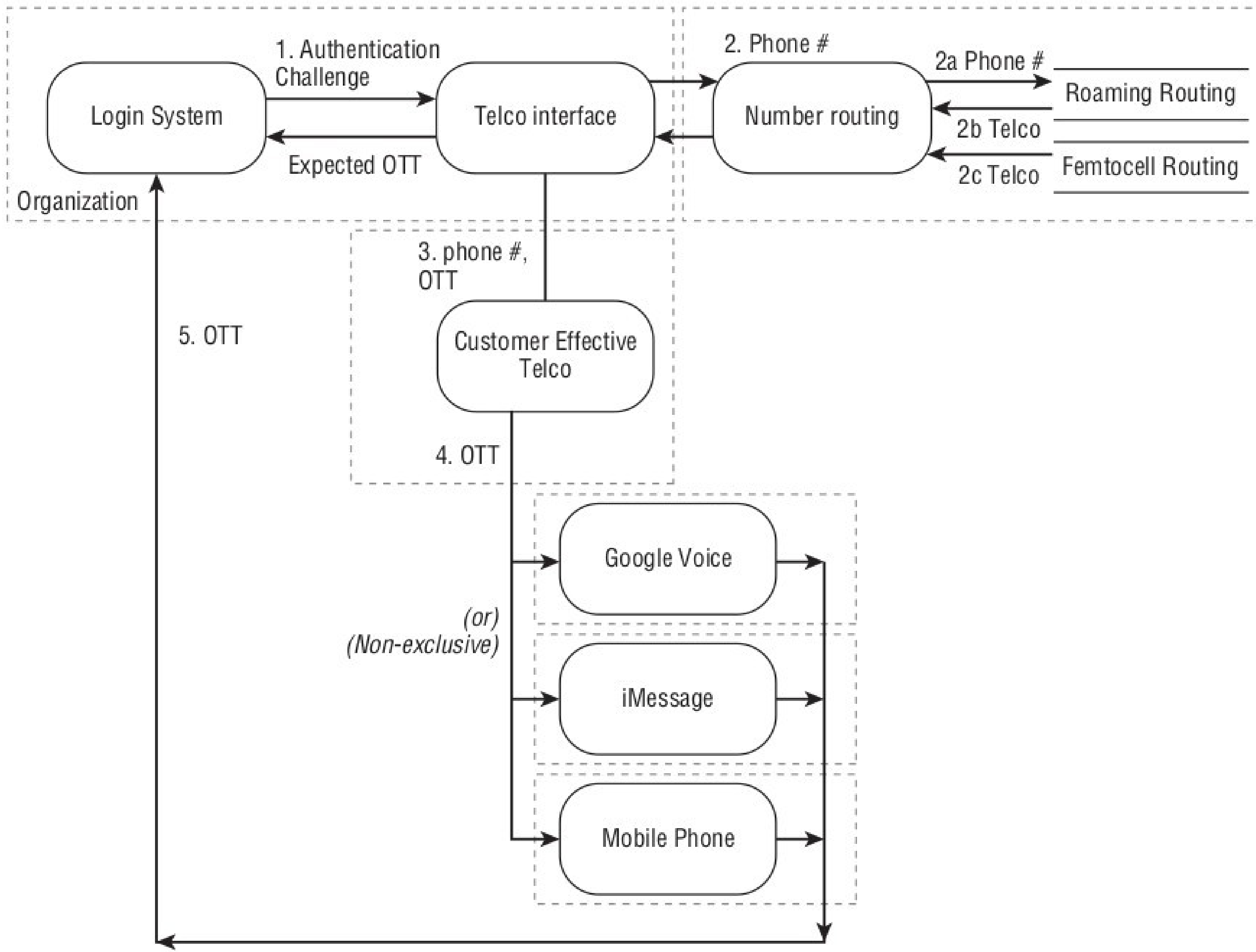
Spoofing
Elevation of privilege
Repudiation

"STRIDE"



Authorization Provider
LDAP
Elevation Of Privilege
Verifies the Privilege
Verified
Generic Trust Boundary
Tampering With The Data Using The SQL Injection Attack
Generic Trust Boundary
Verified
Auth Verification
Generic Trust Boundary
Spoofing
Spoofing
Spoofing
User Sends User Credentials
User Is Authenticated
Insert Feedback Comments
Feedback Comments Saved
Insert Query With Feedback Comments
Success=1
Browser Client
Web Application
SQL Database

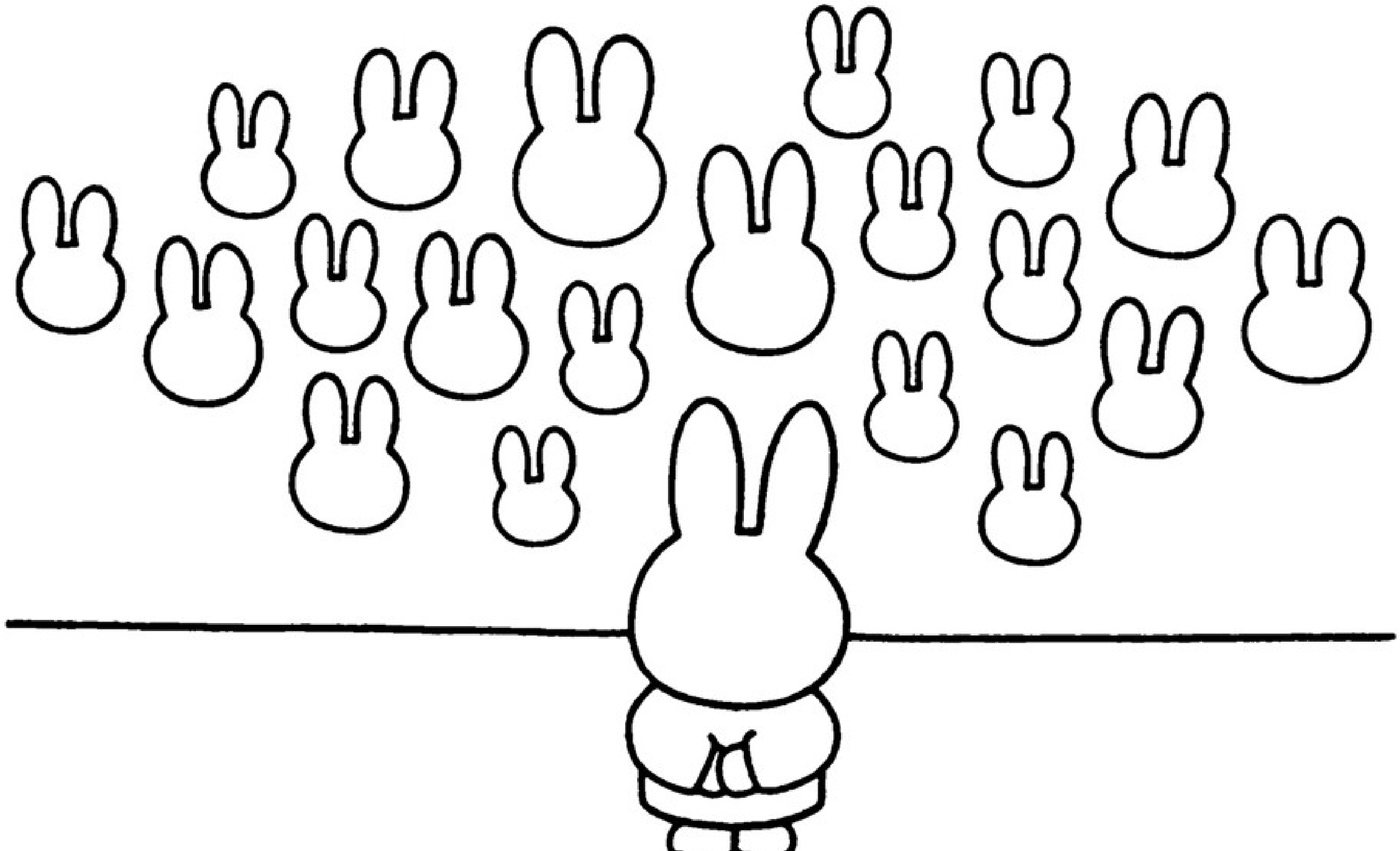

Internet
SWIFT messaging software
SWIFT connectivity software
End-user computer
Back-office applications and printers
Customer's premises



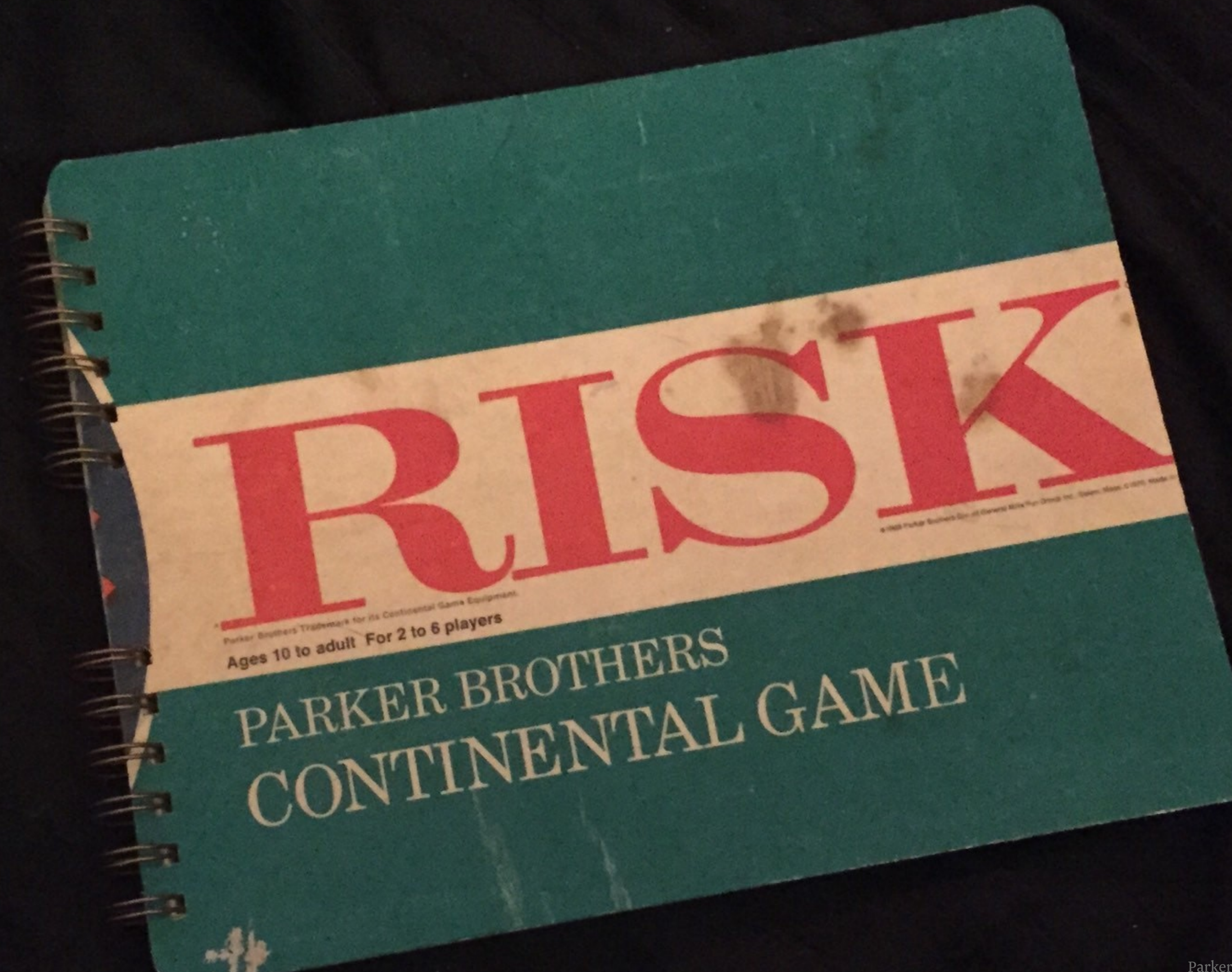
Adam Shostack

Lightweight methodology

1. Draw data flows
2. Elicit threats
3. Ranking + controls
4. Check your work

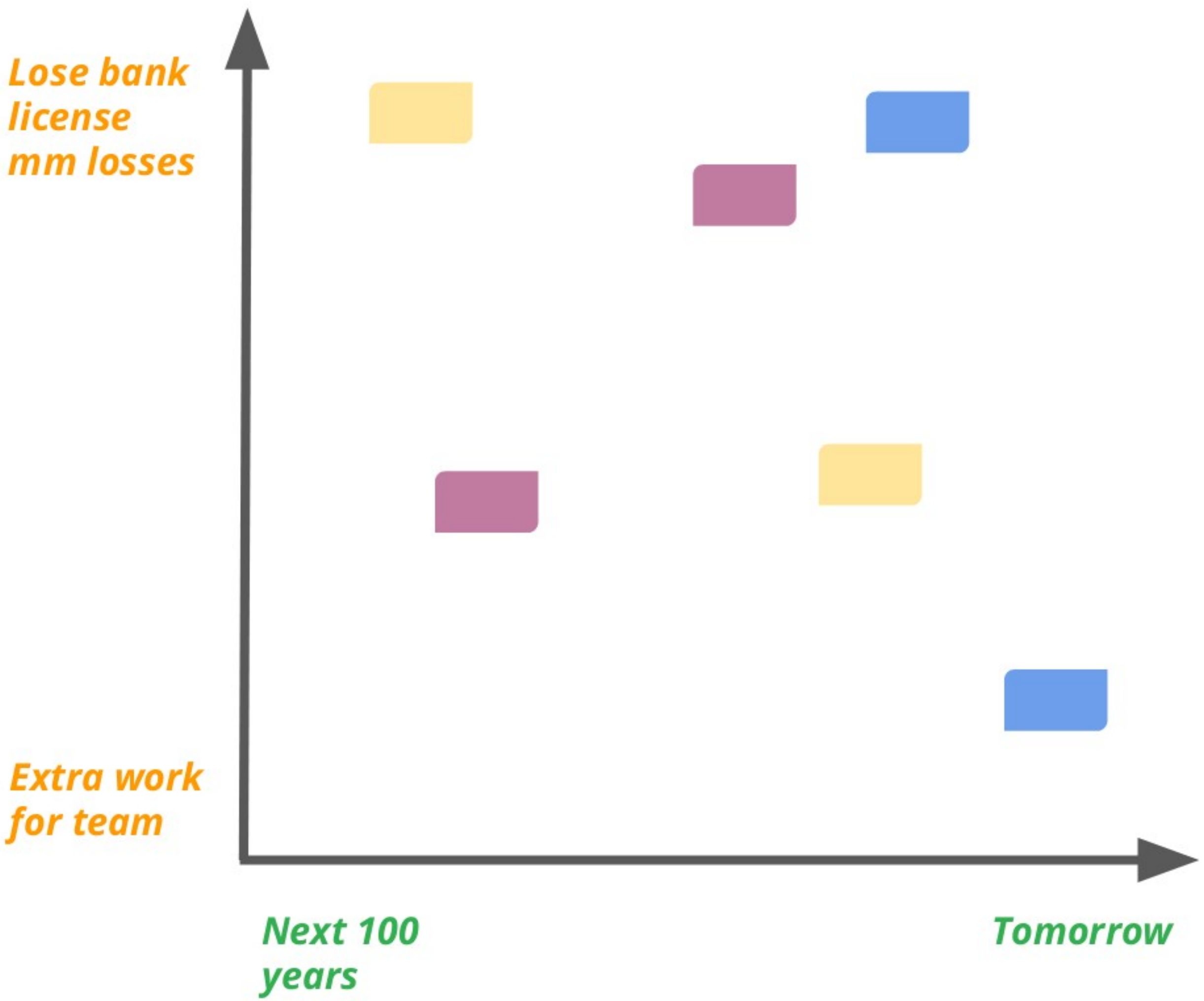


Dick Bruna

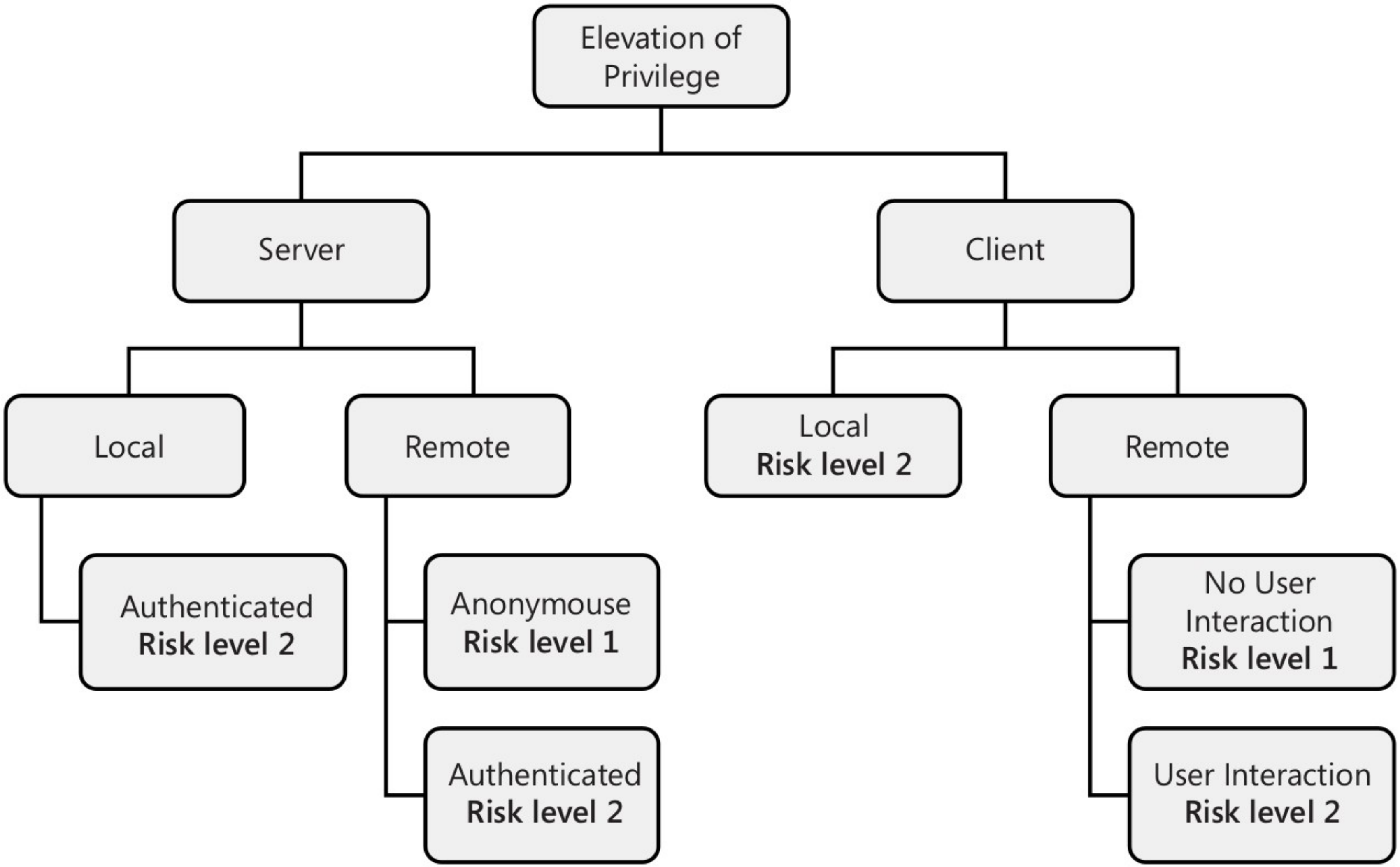


Parker Brothers

$$\text{Risk} \approx \text{likelihood} \times \text{impact}$$



Lose bank
license
mm losses
Extra work
for team
Next 100
years
Tomorrow

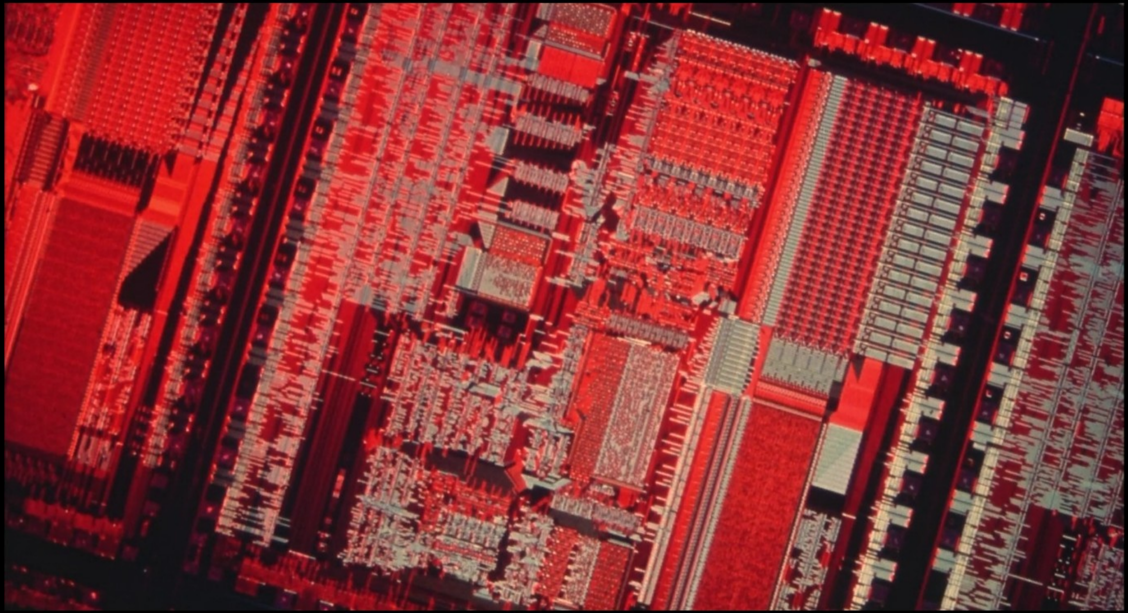

Elevation of Privilege
Server
Client
Local
Authenticated
Risk level 2
Remote
Anonymouse
Risk level 1
Authenticated
Risk level 2
Local
Risk level 2
Remote
No User Interaction
Risk level 1
User Interaction
Risk level 2

Lightweight methodology

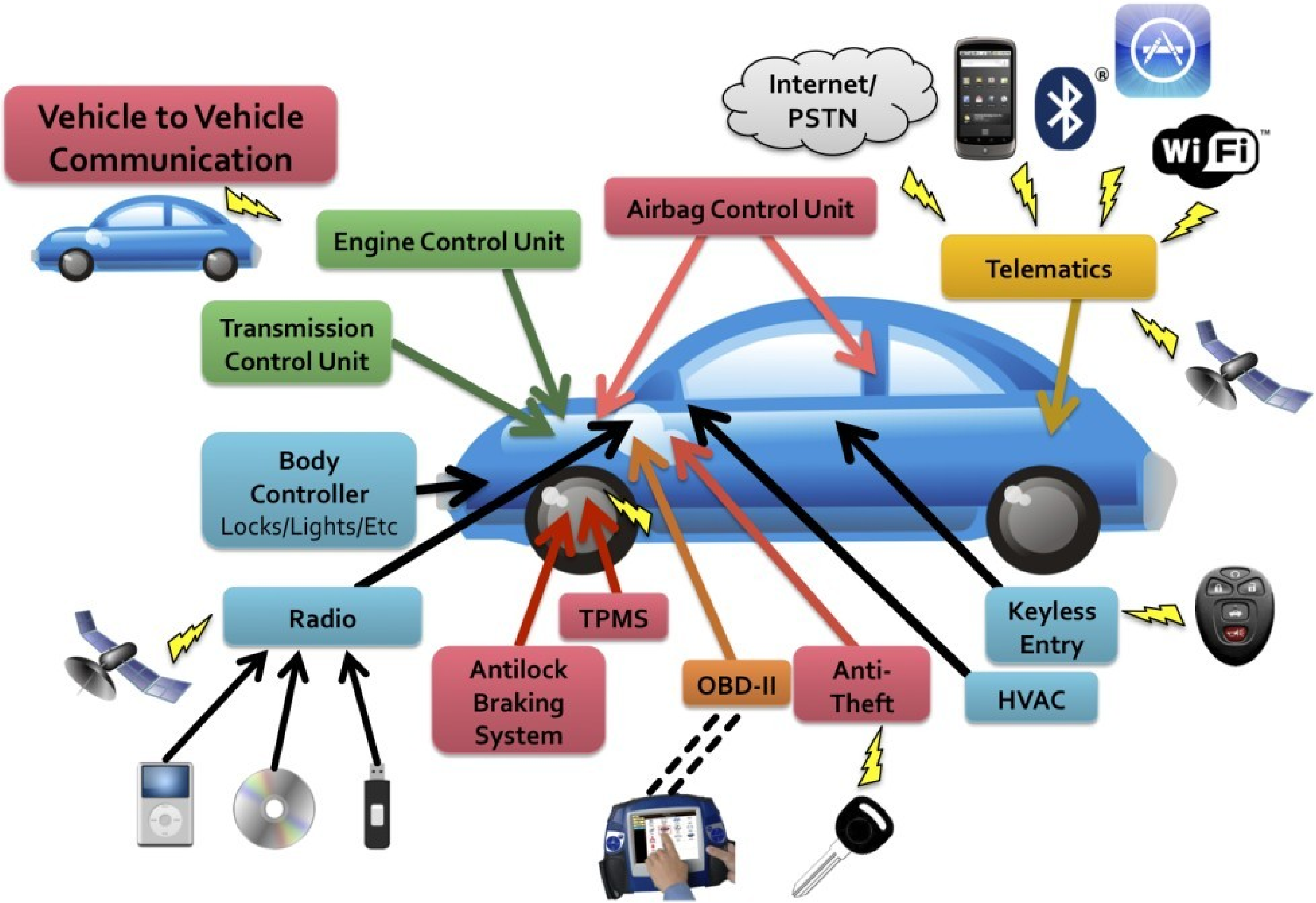
1. Draw data flows
2. Elicit threats
3. Ranking + controls
4. Check your work

“ All models are wrong,
some models are useful. ”

— George Box

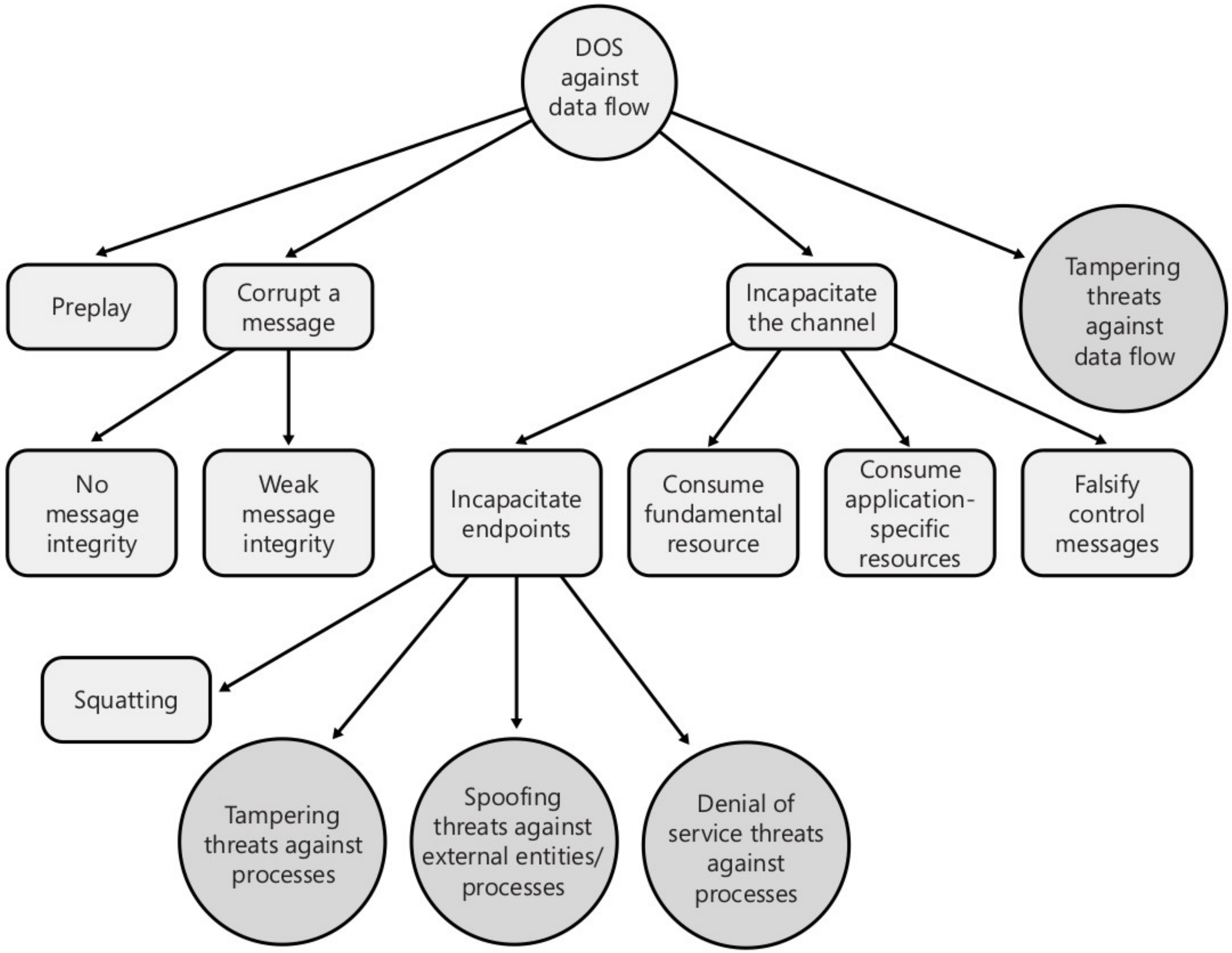


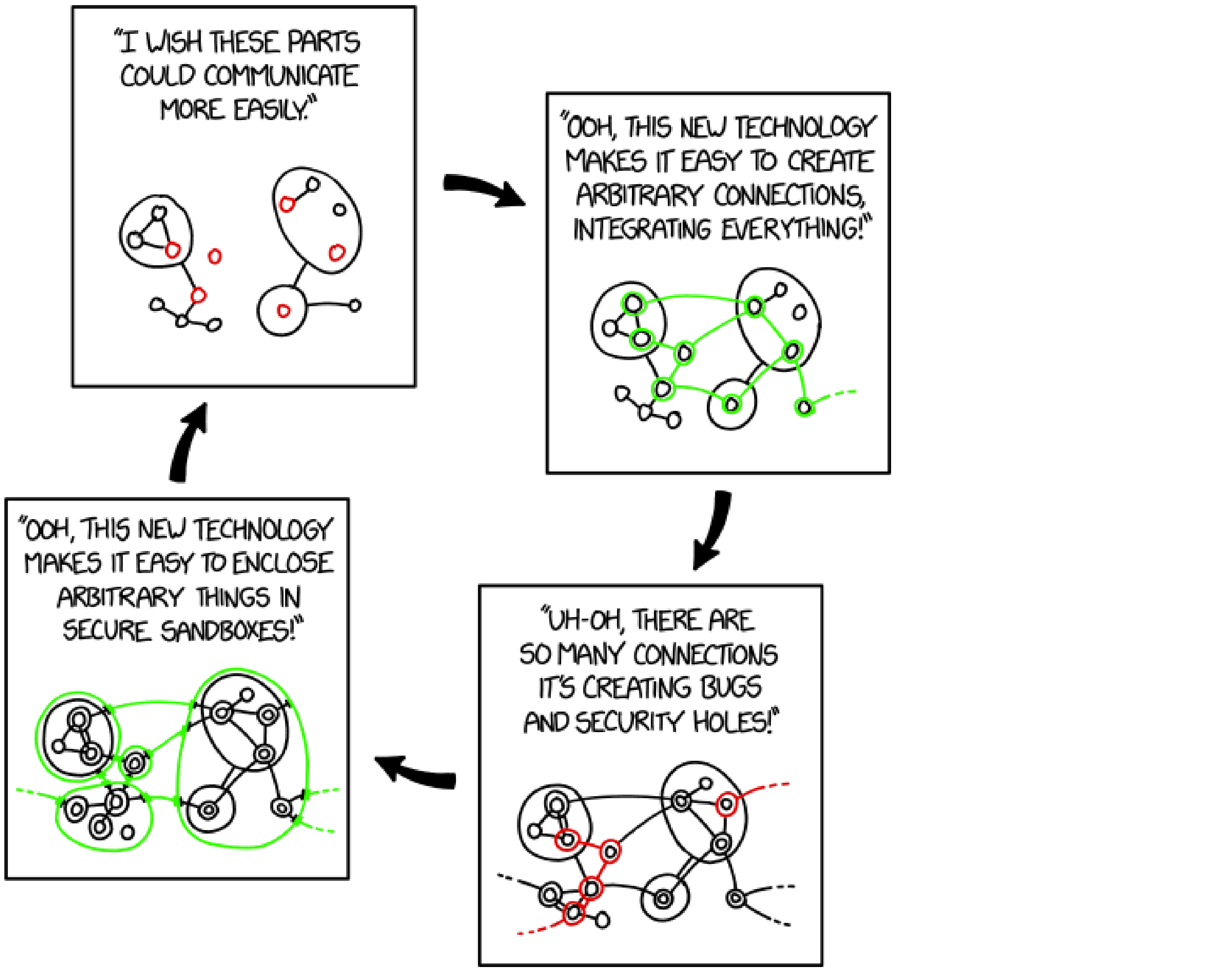
Koyaanisqatsi



Stephen Checkoway et al.

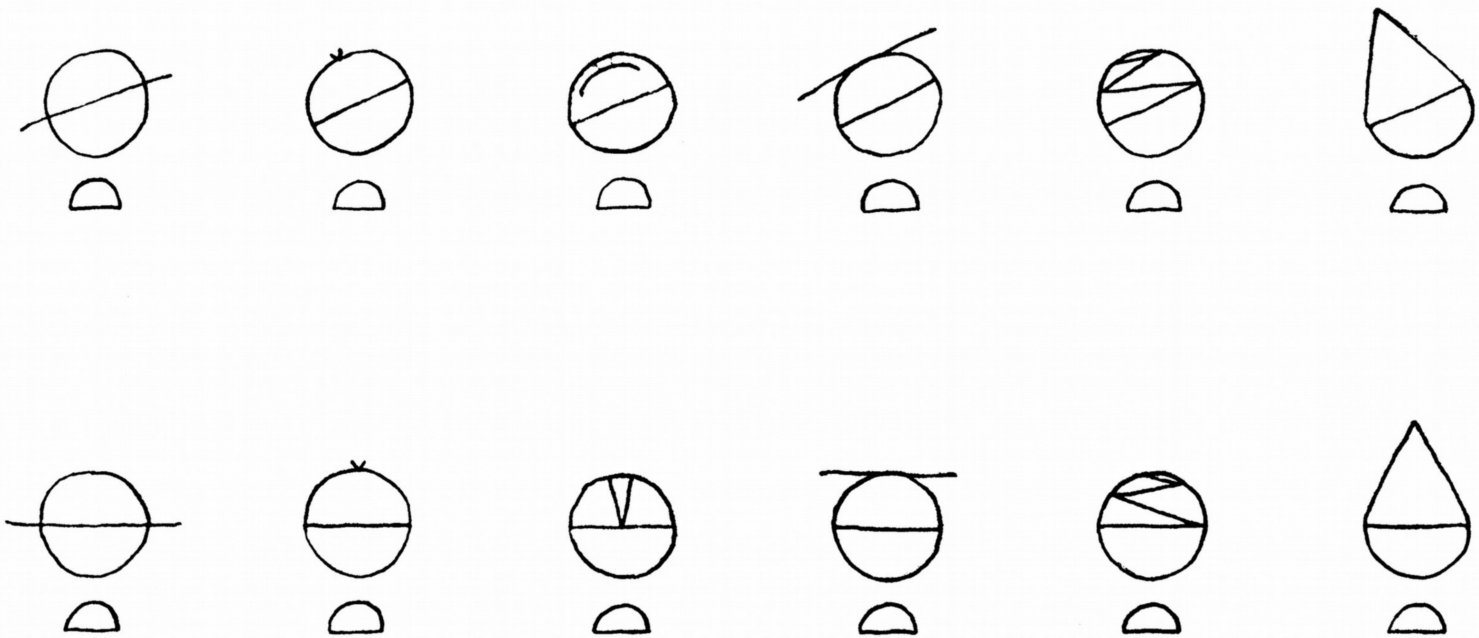
1. information disclosure
2. data interpreted as code
3. resource exhaustion/denial
4. race conditions
5. canonicalisation
6. insufficient access control
7. environment (mis) configuration
8. logic errors
9. predictability
10. poor usability

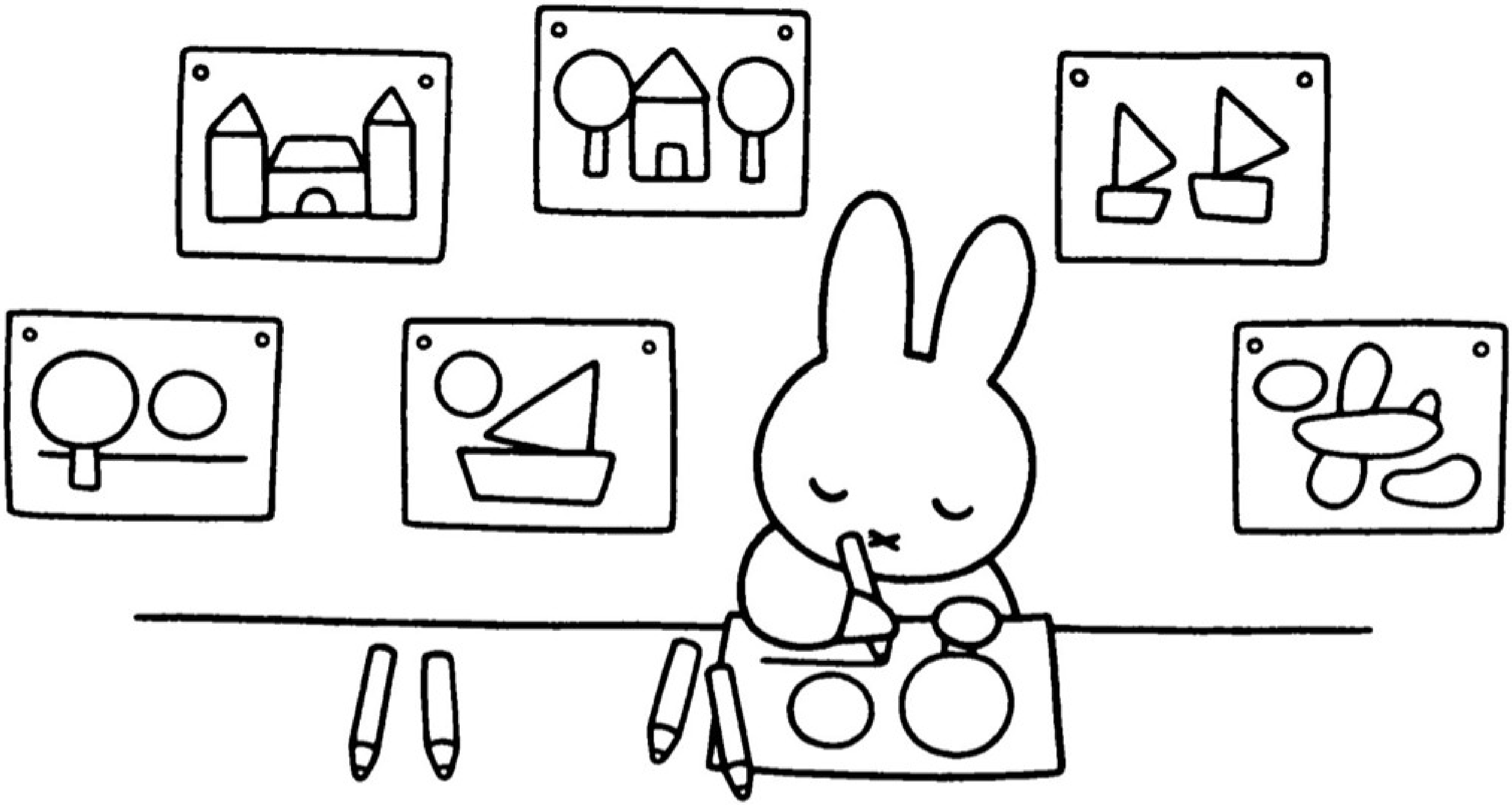

DOS against data flow
Preplay
Corrupt a message
Incapacitate the channel
Tampering threats against data flow
No message integrity
Weak message integrity
Incapacitate endpoints
Consume fundamental resource
Consume application-specific resources
Falsify control messages
Squatting
Tampering threats against processes
Spoofing threats against external entities/ processes
Denial of service threats against processes


"I WISH THESE PARTS COULD COMMUNICATE MORE EASILY."
"OOH, THIS NEW TECHNOLOGY MAKES IT EASY TO CREATE ARBITRARY CONNECTIONS, INTEGRATING EVERYTHING!"
"UH-OH, THERE ARE SO MANY CONNECTIONS IT'S CREATING BUGS AND SECURITY HOLES!"
"OOH, THIS NEW TECHNOLOGY MAKES IT EASY TO ENCLOSE ARBITRARY THINGS IN SECURE SANDBOXES!"
xkcd

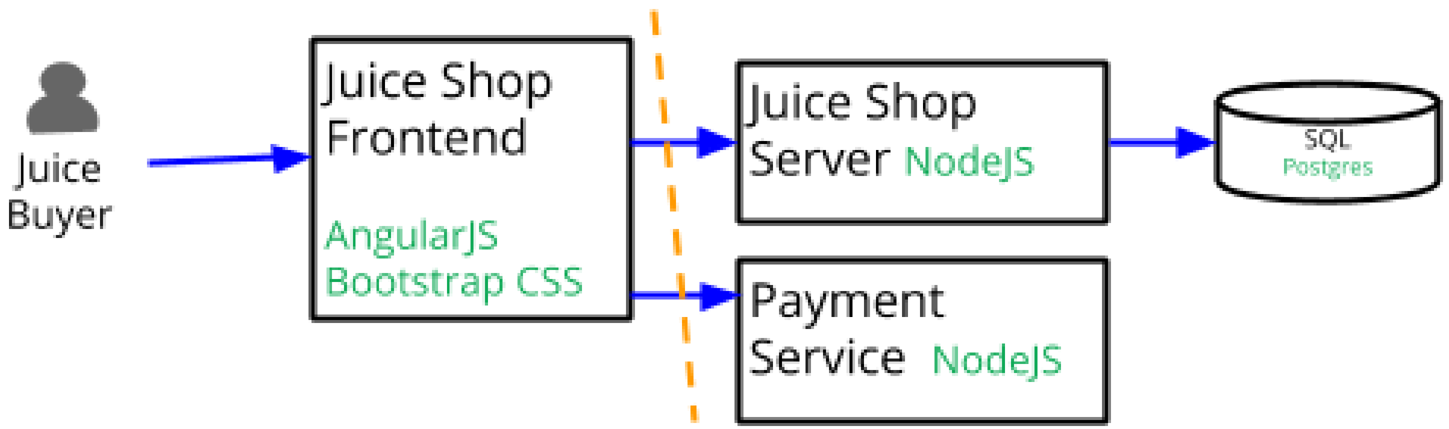
Lightweight methodology

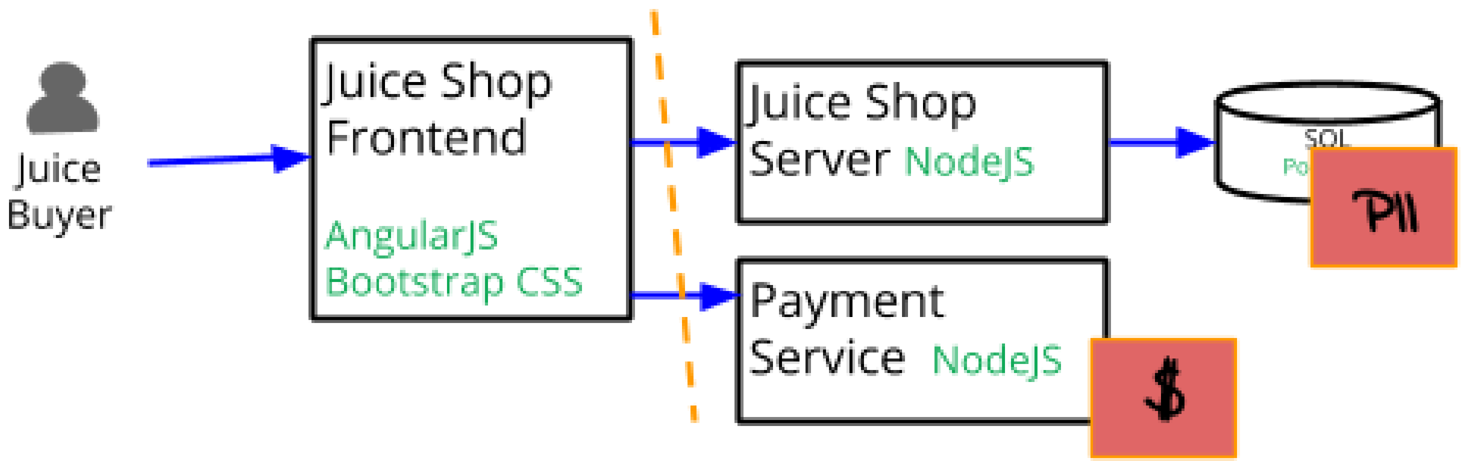
1. Draw data flows
2. Elicit threats
3. Ranking + controls
4. Check your work

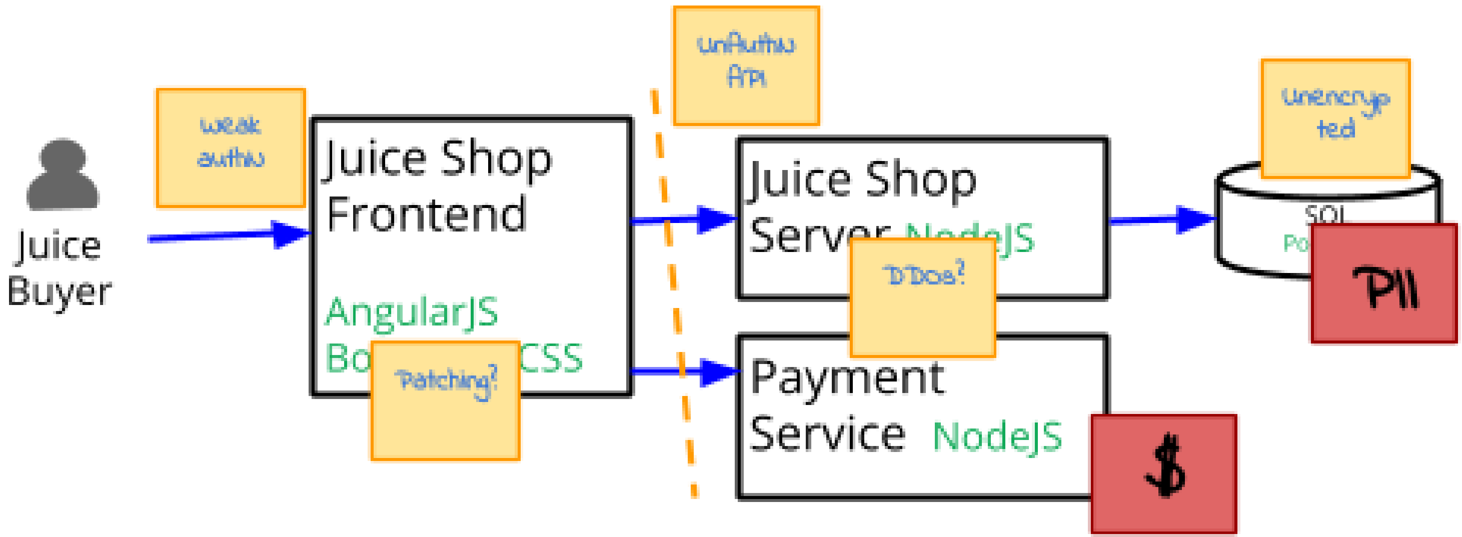




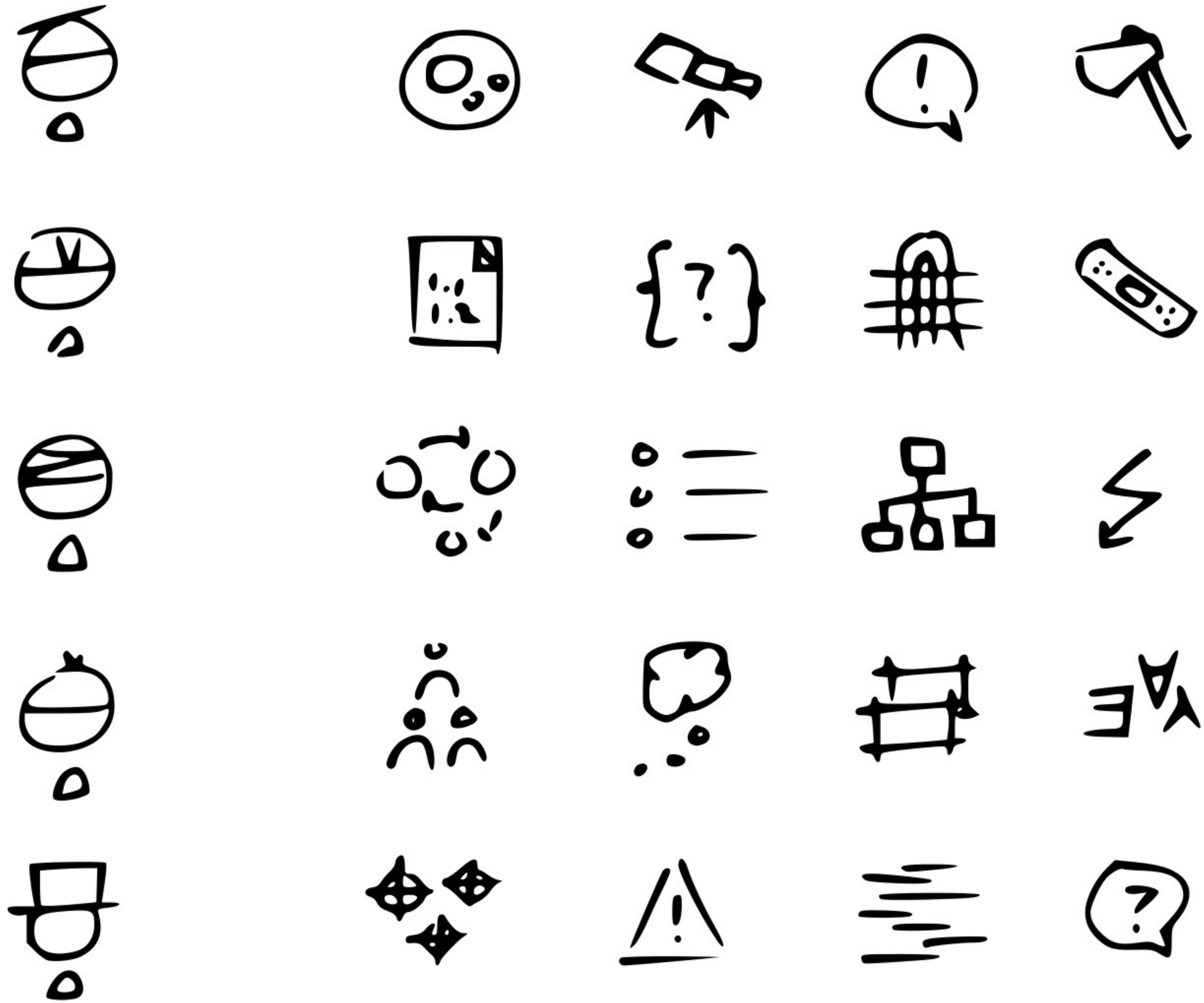
Dick Bruna


Juice Buyer
Juice Shop Frontend
AngularJS
Bootstrap CSS
Juice Shop Server NodeJS
Payment Service NodeJS
SQL
Postgres


Juice Buyer
Juice Shop Frontend
AngularJS
Bootstrap CSS
Juice Shop Server NodeJS
Payment Service NodeJS
$
SQL
PII


weak authn
Juice Buyer
Juice Shop Frontend
AngularJS
Bo
CSS
Patching?
unAuthn API
Juice Shop Server NodeJS
DDoS?
Payment Service NodeJS
$
Unencrypted
SQL
PII


AS disgruntled employees
WITH access to the organisation's systems and basic technical skills
SO THAT we can embarrass colleagues or impact the organisation's reputation
AS opportunist hacker forum kids
WITH access the public internet and basic technical skills
SO THAT we can show off to other kids on forums for 'the lulz'
AS cyber-criminals
WITH access to the public internet and strong technical skills
SO THAT we can profit or make monetary gain
AS inside attackers
WITH administrative access to the organisation's systems and strong technical skills
SO THAT we can profit by selling access to information
ELB
CLIENT
ADMIN
REDIS
S3
API GATEWAY
KAFKA
30 days
Github
GoCD
DEVELOPERS
- DEVELOP & ENHANCE
RELEASE AUTHORISER
- TO SIGN OFF ALL RELEASES
- VIA GO CD
AWS console
OVH consoles
AS a disgruntled or angry member of staff



Microsoft

secure software DEVELOPMENT SERIES

BEST PRACTICES

THE SECURITY DEVELOPMENT LIFECYCLE

SDL: A Process for Developing Demonstrably More Secure Software

Michael Howard and Steve Lipner

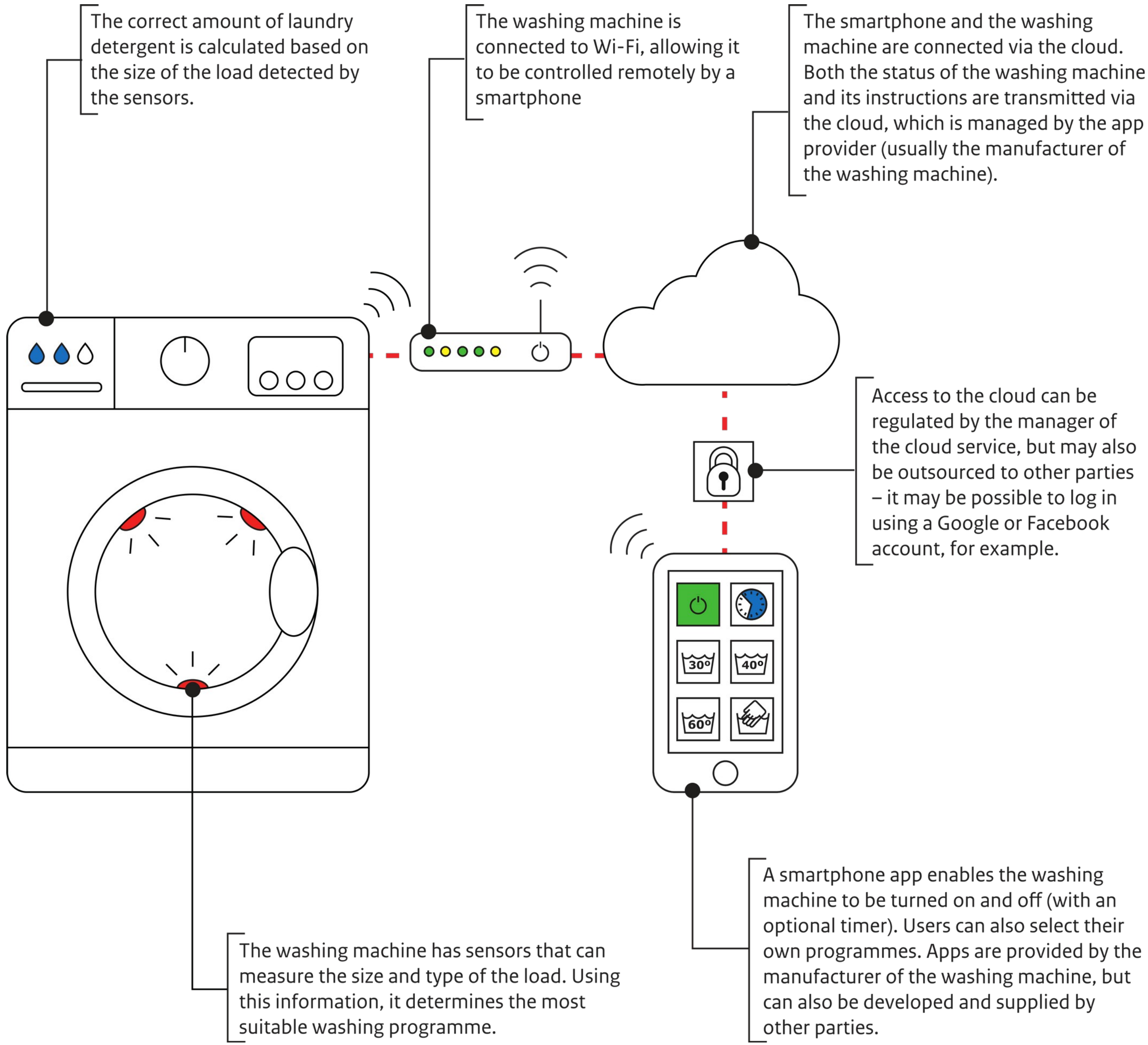
Foreword by Jim Allchin

Co-President, Platforms & Services Division, Microsoft Corporation

CD Includes:
- A security training class video
- Sample SDL documents

See back cover


4


The correct amount of laundry detergent is calculated based on the size of the load detected by the sensors.
The washing machine is connected to Wi-Fi, allowing it to be controlled remotely by a smartphone
The smartphone and the washing machine are connected via the cloud. Both the status of the washing machine and its instructions are transmitted via the cloud, which is managed by the app provider (usually the manufacturer of the washing machine).
Access to the cloud can be regulated by the manager of the cloud service, but may also be outsourced to other parties – it may be possible to log in using a Google or Facebook account, for example.
30°
40°
60°
The washing machine has sensors that can measure the size and type of the load. Using this information, it determines the most suitable washing programme.
A smartphone app enables the washing machine to be turned on and off (with an optional timer). Users can also select their own programmes. Apps are provided by the manufacturer of the washing machine, but can also be developed and supplied by other parties.

What could possibly go wrong?

Arne Padmos
hello@arnepadmos.com

Hanzehogeschool Groningen
University of Applied Sciences

IT Academy
Noord-Nederland

github.com
/arnepadmos/resources

my “toy collection”