The As, Bs, and Four Cs of Testing Cloud-Native Applications

Dan Cornell
March 5, 2020

Dan Cornell

- Founder and CTO of Denim Group
- Software developer by background
- OWASP San Antonio co-leader
- 20 years experience in software architecture, development, and security

DENIM GROUP

Building a world where technology is trusted

Denim Group is solely focused on helping build resilient software that will withstand attacks.

- Since 2001, helping secure software
- Development background
- Tools + services model

How we can help:

Advisory Services

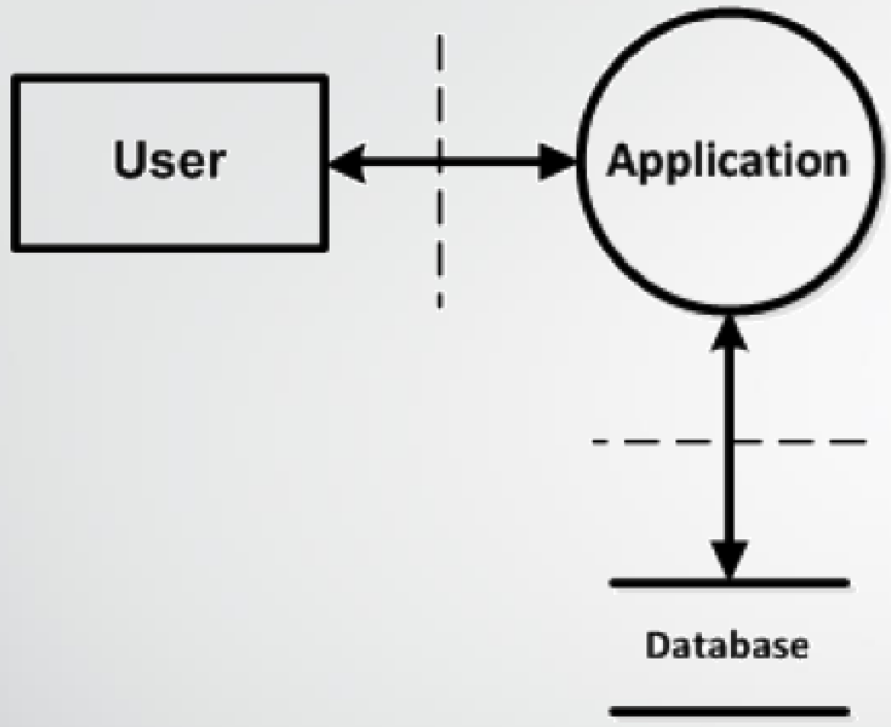
Assessment Services

Remediation Services

ThreadFix
Powered by Denim Group
Vulnerability Resolution Platform

Agenda

- The Good Old Days
- The More Interesting New Days
- **A**rchitectural **B**ill of Materials
- Four **C**s
- Reporting
- Tailoring
- Questions

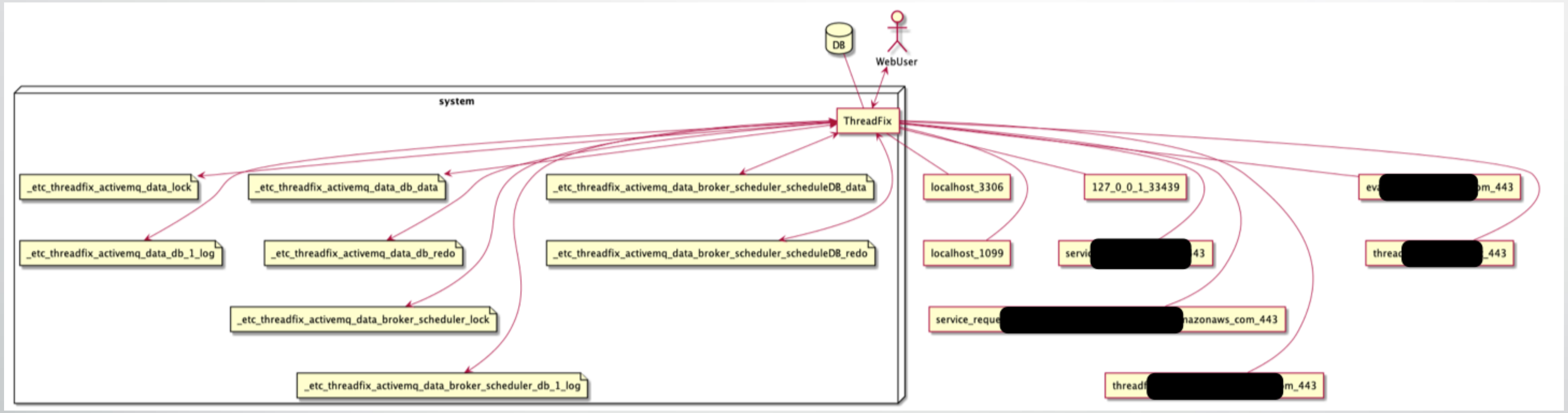


The Good Old Days

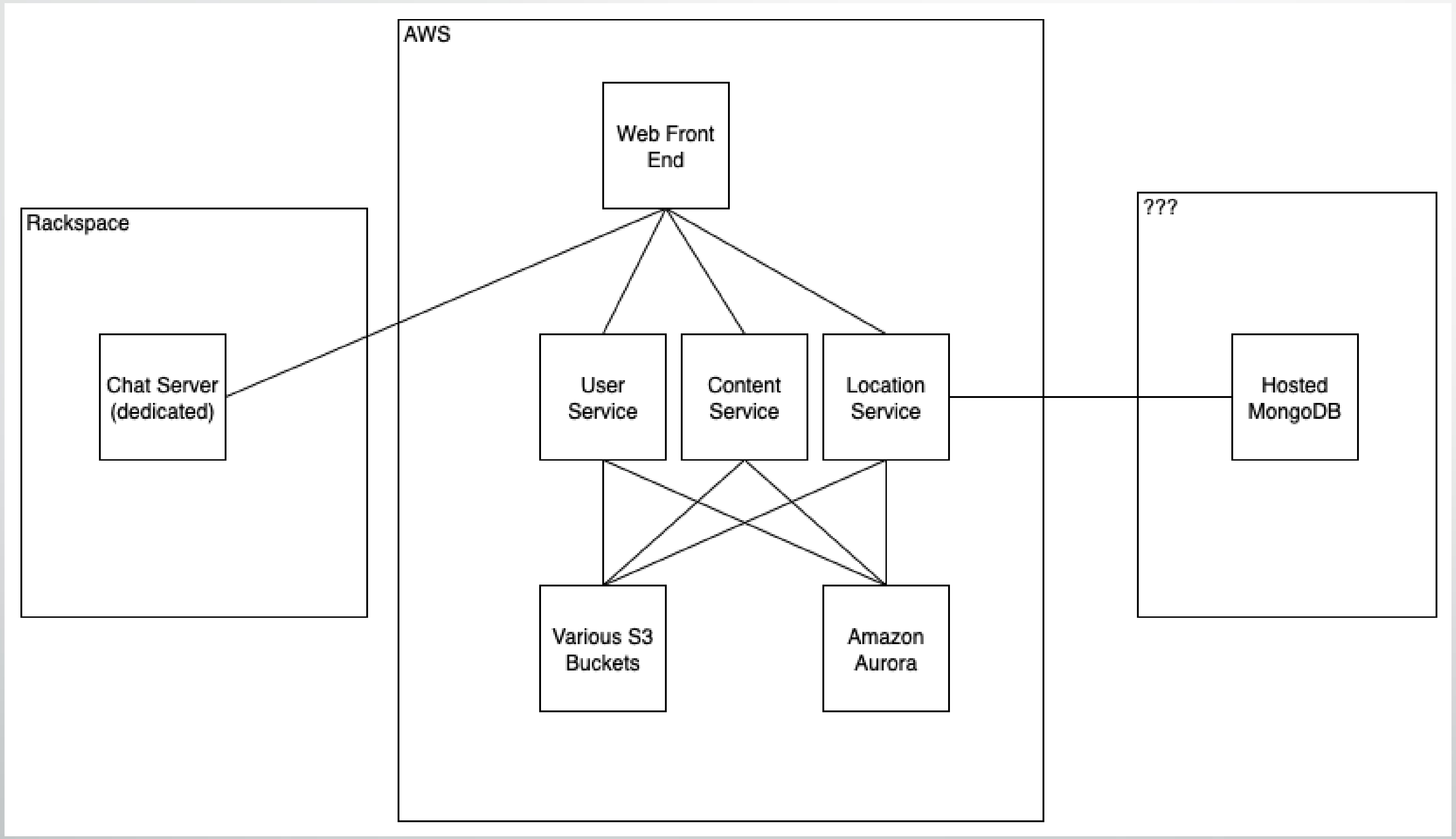
Blast it with SAST or DAST
Do some manual testing, and …

PDF

The More Interesting New Days

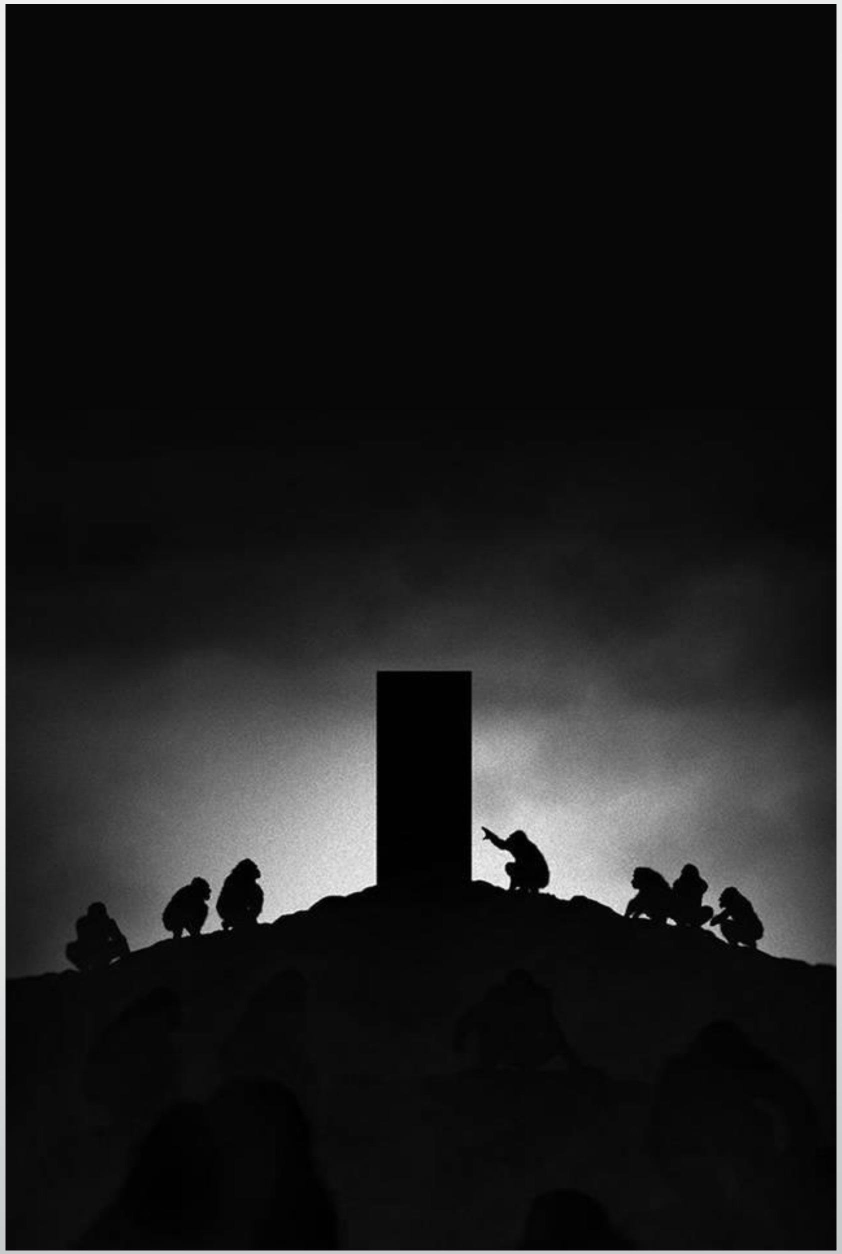

DB
WebUser
system
ThreadFix
_etc_threadfix_activemq_data_lock
_etc_threadfix_activemq_data_db_data
_etc_threadfix_activemq_data_broker_scheduler_scheduleDB_data
localhost_3306
127_0_0_1_33439
ev[redacted]m_443
_etc_threadfix_activemq_data_db_1_log
_etc_threadfix_activemq_data_db_redo
_etc_threadfix_activemq_data_broker_scheduler_scheduleDB_redo
localhost_1099
servi[redacted]43
threa[redacted]_443
_etc_threadfix_activemq_data_broker_scheduler_lock
service_reque[redacted]azonaws_com_443
_etc_threadfix_activemq_data_broker_scheduler_db_1_log
threadf[redacted]m_443

The Even *More* Interesting New Days



A Dedicated Server at Rackspace?!

What Changed?



An Aside: *Why* Did Things Change?

- Digital Transformation
  - The “risk” we talk about is crap
  - Falling behind creates existential risk for firms
- Must Go Faster?
  - Change culture to DevOps
- Culture has changed to DevOps?
  - Adopt new technologies to support mission

https://www.denimgroup.com/resources/whitepaper/security-the-other-side-of-digital-transformation/

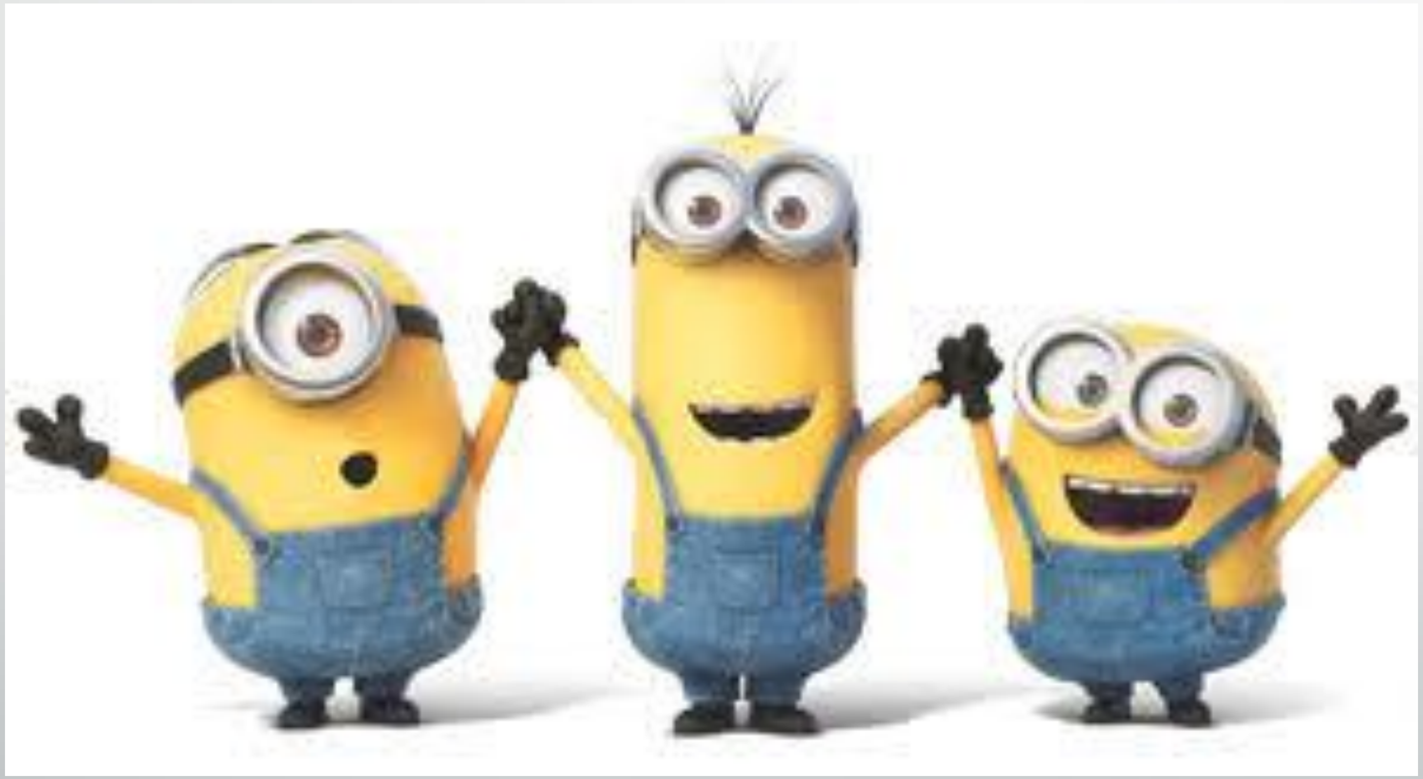
What Changed?

- Architecture
  - Monolithic -> Microservices
- Technology
  - Cloud servers
  - Cloud services
  - Containers
  - Serverless
  - CI/CD Pipelines

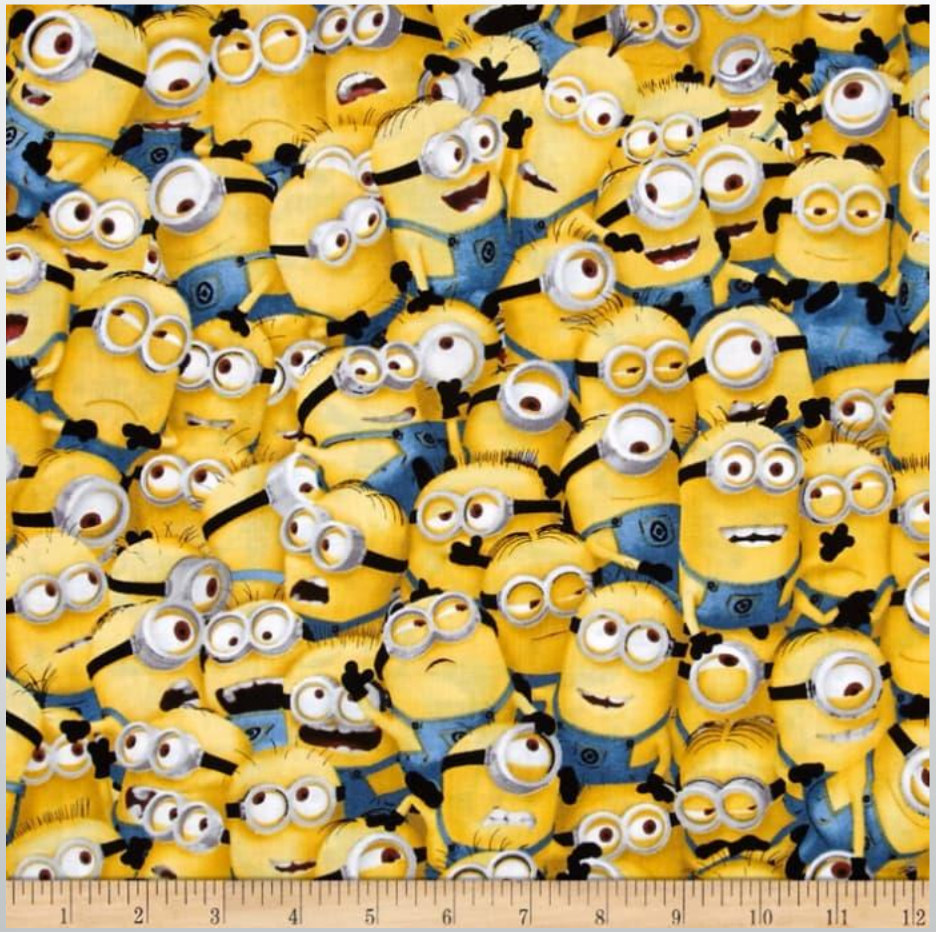
Microservices

If you couldn’t make one big thing work properly, what makes you think you can make thirty smaller things that need to talk to one another work properly?

How You Think Microservices Will Work



How Microservices Actually Work

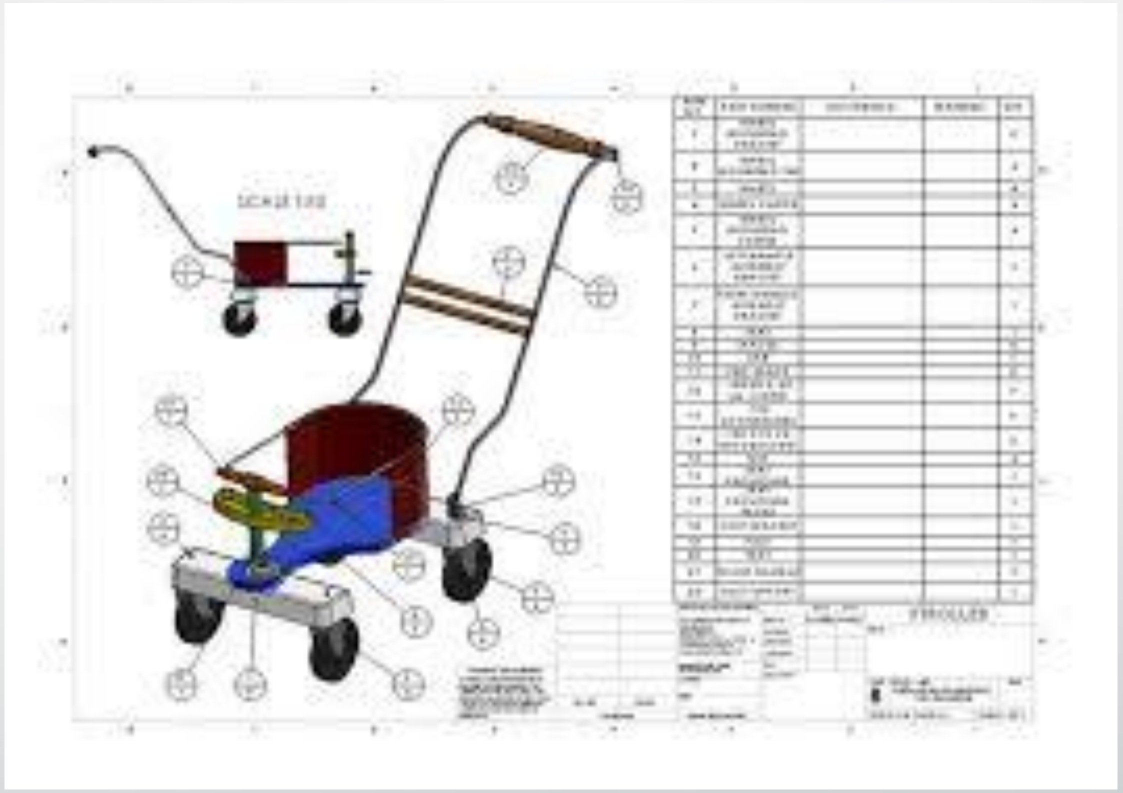


As, Bs, and Four Cs

- **A**rchitectural **B**ill of Materials
- Four Cs
  - **C**ode
  - **C**omponents
  - **C**ompute
  - **C**loud Configuration

Software Bill of Materials (SBOM)

- What is actually in the software I am shipping?
- Open source, etc



OWASP Dependency Track
https://www.owasp.org/index.php/OWASP_Dependency_Track_Project

Architectural Bill of Materials

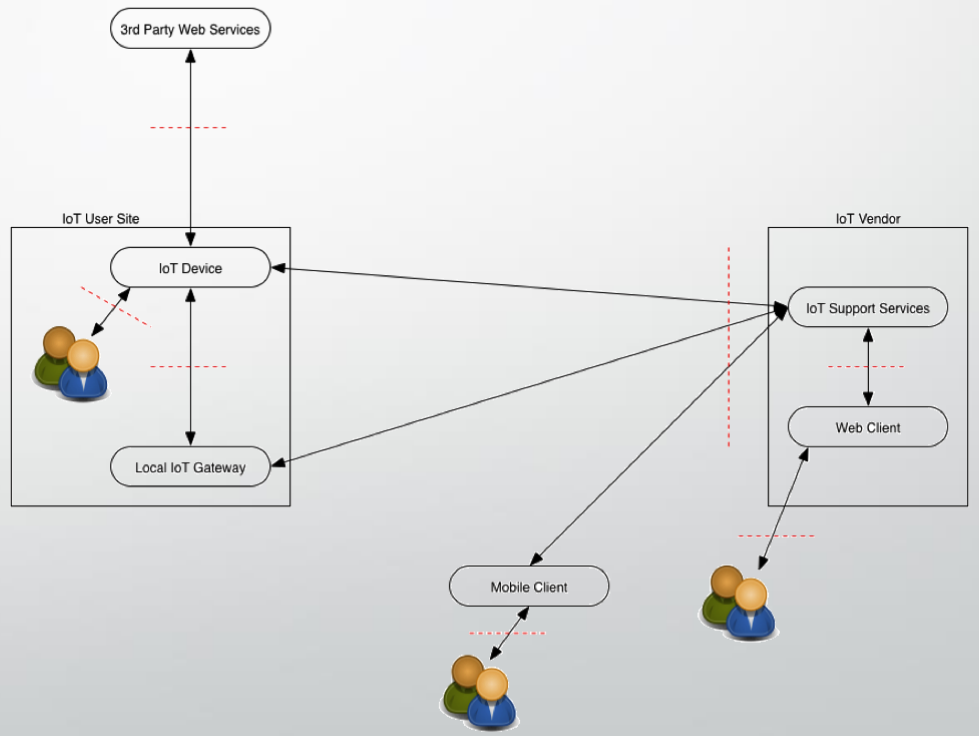
COPY/PASTE OPEN SOURCE CODE
IMPORT OPEN SOURCE COMPONENTS
SOFTWARE BILL OF MATERIALS
ARCHITECTURAL BILL OF MATERIALS
imgflip.com

Architectural Bill of Materials

- What are the pieces of the system we are looking at?
- Being able to answer:
  - What are the various parts of the system?
  - What do they consist of?
  - What do they do?
  - Where are they hosted?

Architectural Bill of Materials

- So a threat model?
- Yeah pretty much. A threat model.

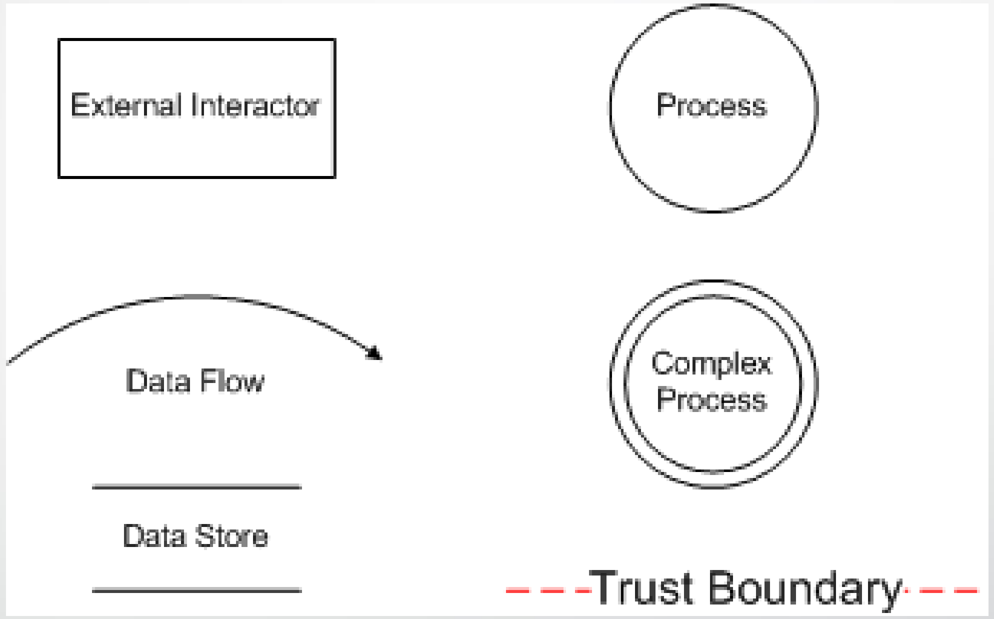


High Level Threat Modeling Concepts

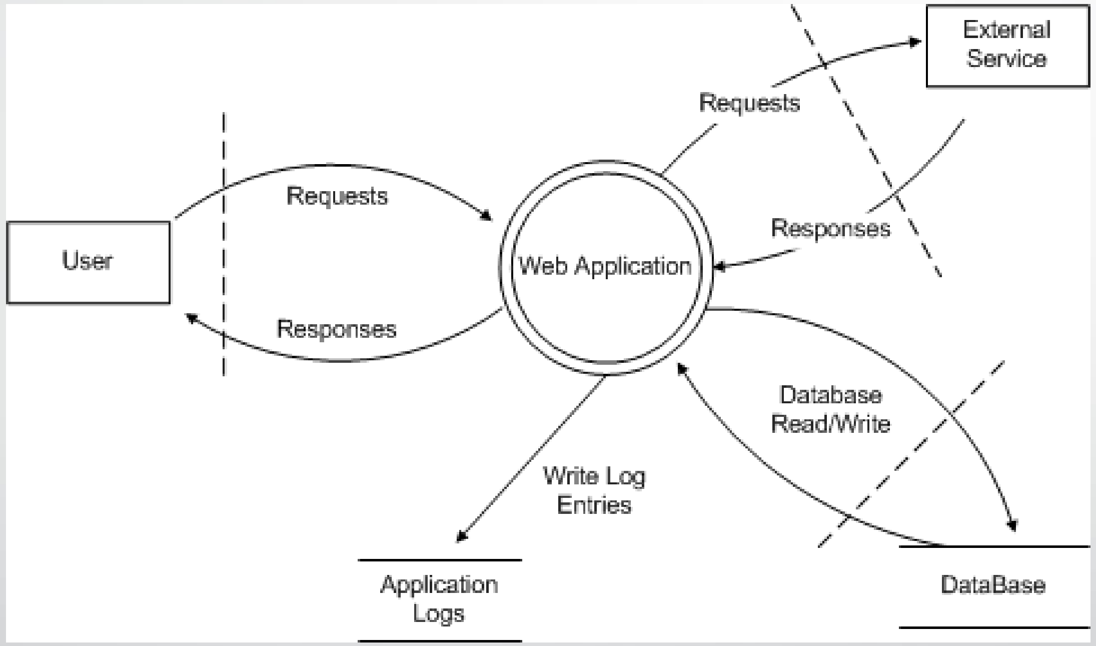
1. Decide on scope
2. Build your dataflow diagrams
3. Enumerate threats
4. Decide on mitigations

Creating Data Flow Diagrams (DFDs)

- Decompose the system into a series of processes and data flows
- Explicitly identify trust boundaries



Example Data Flow Diagram



Identifying Threats from the Data Flow

STRIDE is expansion of the common CIA threat types

- Confidentiality
- Integrity
- Availability

STRIDE

- Spoofing Identity
- Tampering with Data
- Repudiation
- Information Disclosure
- Denial of Service
- Elevation of Privilege

Mapping Threats to Asset Types

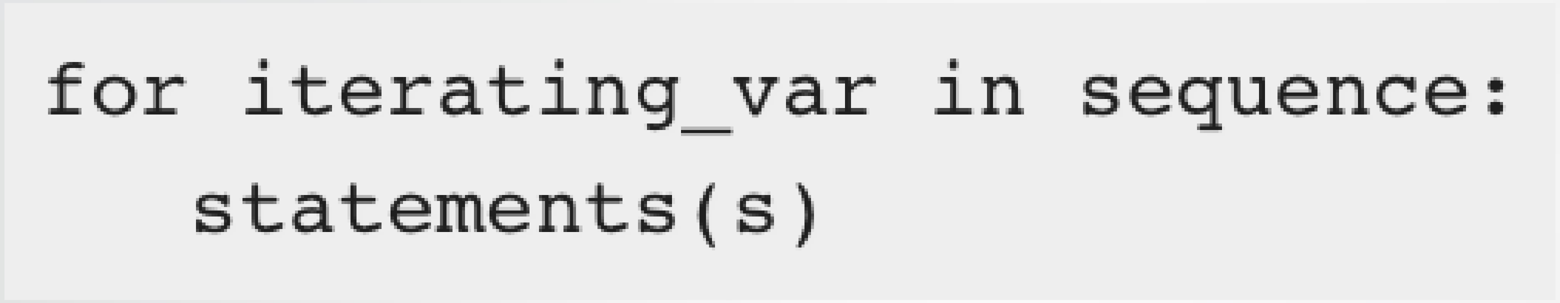
Threat Type	External Interactor	Process	Data Flow	Data Store
S – Spoofing	Yes	Yes		
T – Tampering		Yes	Yes	Yes
R – Repudiation	Yes	Yes		Yes
I – Information Disclosure		Yes	Yes	Yes
D – Denial of Service		Yes	Yes	Yes
E – Elevation of Privilege		Yes		

So What Does That Leave Us?

- Take all the assets
- Associate threat types with each asset
- Voila! List of things we need to worry about

ABOM

- We at least need the results of Steps 1 and 2 to get our asset list and the relationships
- May as well finish things off because we'll need the rest later on to provide context for reporting



```
for iterating_var in sequence:
    statements(s)
```

- We now need to look at the security of each of the pieces in the overall system
- Test them for security issues at various layers
- Aggregate the results

Four Cs

- Code
- Components
- Compute
- Cloud Configuration

Code

Code

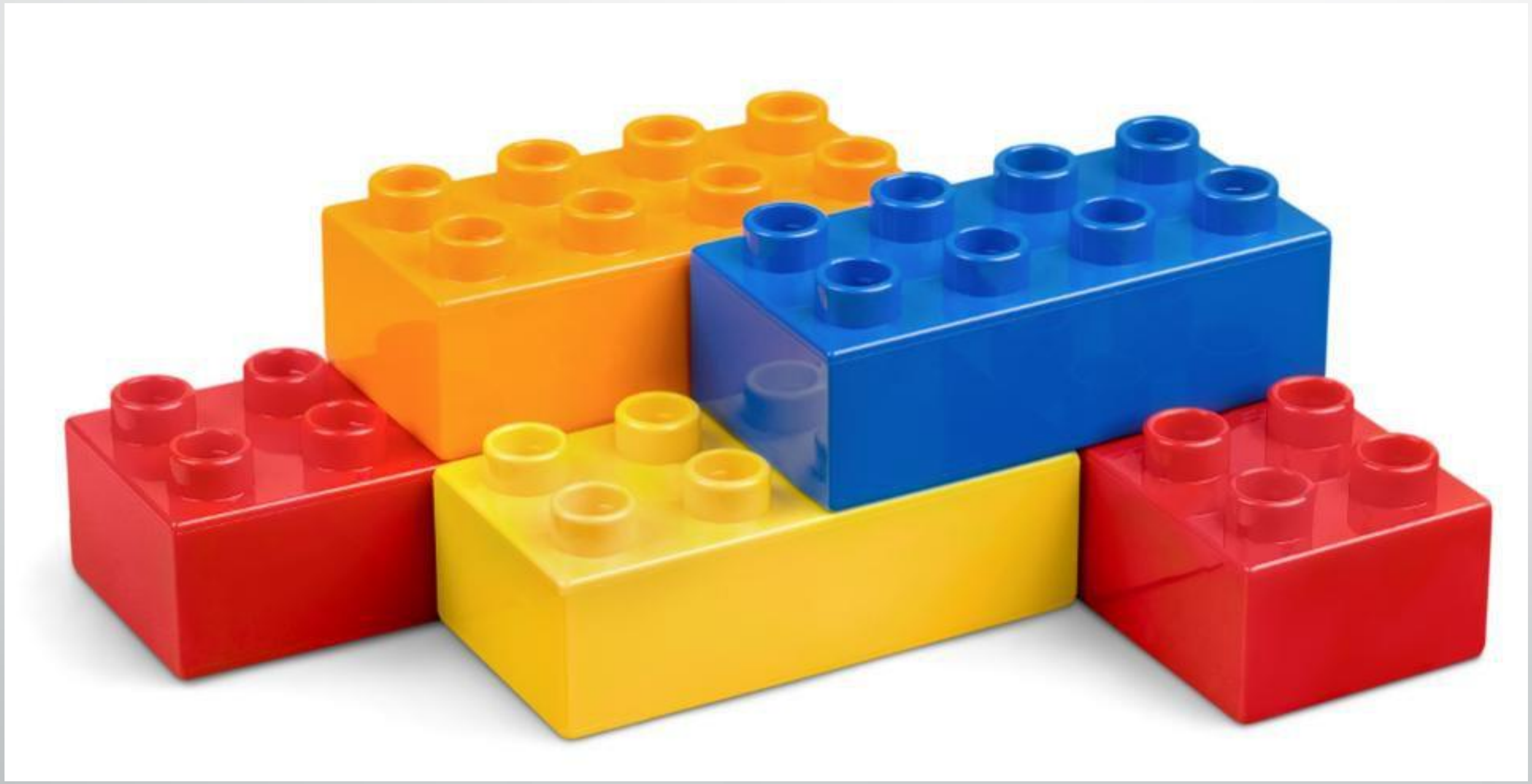
- This is the code you write
  - Business logic
  - Glue stuff together

- Traditional focus of OWASP/application security
- Automated testing with SAST, DAST, IAST
- Manual penetration testing and code review

Code – API Testing

- Great news – the DAST tools you depended on for web application testing might not work terribly well for APIs
- Some API-focused DAST tools
  - OWASP ZAP has some capabilities in this area
  - Always option to do manual testing

Components

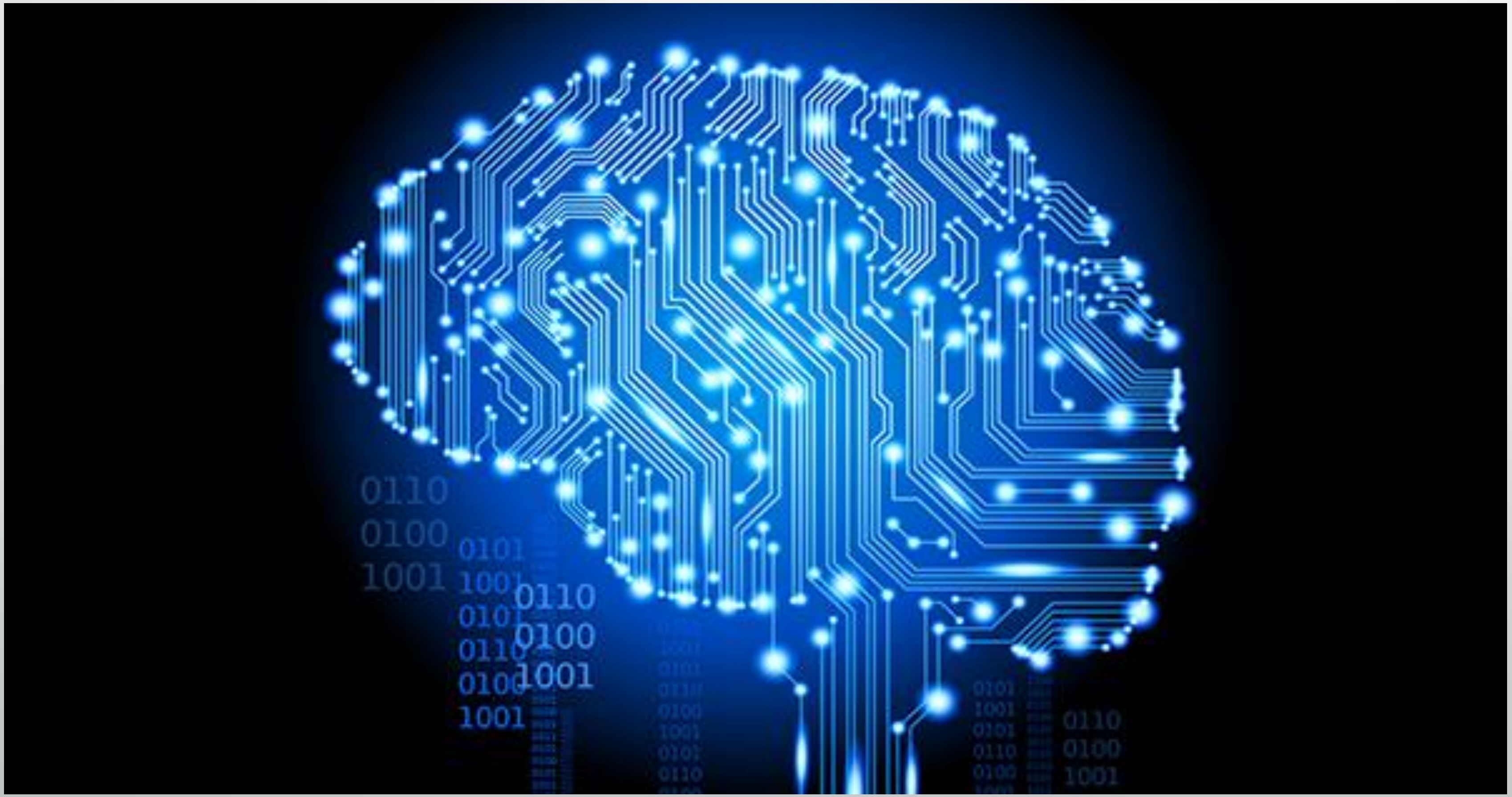


Components

- These are the open source components you include so that you don't have to write everything
  - Libraries
  - Frameworks
- Gained prominence with its introduction in the OWASP Top 10 2013
  - Gained notoriety with the Equifax breach
  - Thanks, Struts…
- Test with Software Composition Analysis (SCA)
  - Often need to manually validate impact
- Traditional SBOM scope

https://www.owasp.org/index.php/OWASP_Dependency_Check

Compute



Compute

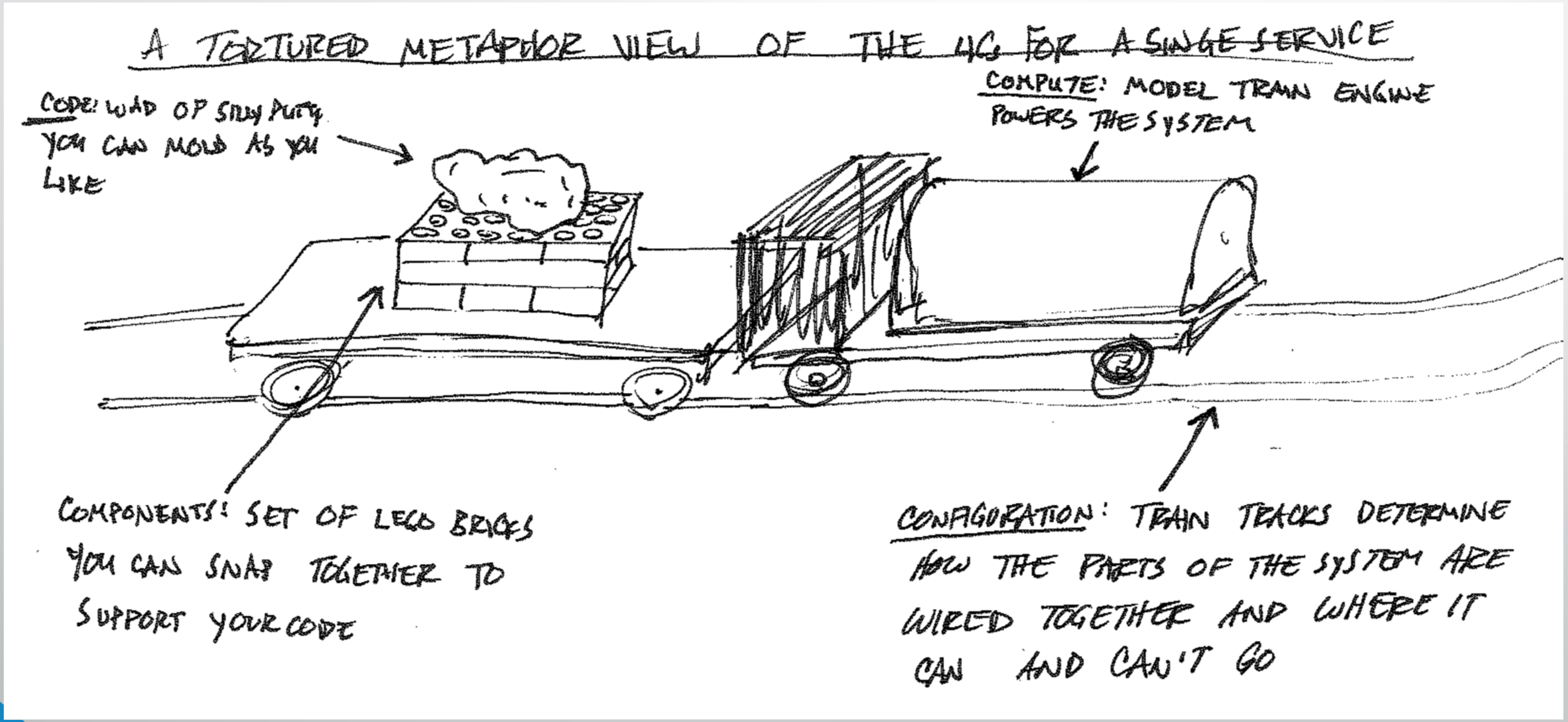
- Something has to run all this code…
- Virtual machines, cloud servers, containers
  - Serverless takes this to the extreme
  - Don’t forget dedicated servers
- Test with:
  - Traditional vulnerability scanning
  - Container scanning

Cloud Configuration

Cloud Configuration

- The squishiest of all the Cs
  - Maybe that's why it gets two Cs…
- Largely configuration checks
  - Open S3 buckets
  - Bad IAM set ups
- Will evolve over time
  - If this presentation were being given a couple of years ago, cloud servers might fall in this category
  - Move stable stuff – cloud servers – into their own **C**ategory

So What Does This All Look Like?


A TORTURED METAPHOR VIEW OF THE 4Cs FOR A SINGE SERVICE
CODE: WAD OF SILLY PUTTY YOU CAN MOLD AS YOU LIKE
COMPUTE: MODEL TRAIN ENGINE POWERS THE SYSTEM
COMPONENTS: SET OF LEGO BRICKS YOU CAN SNAP TOGETHER TO SUPPORT YOUR CODE
CONFIGURATION: TRAIN TRACKS DETERMINE HOW THE PARTS OF THE SYSTEM ARE WIRED TOGETHER AND WHERE IT CAN AND CAN'T GO

Reporting

- Know your audience(s)
- Who are you consumers?
  - Security/risk management
  - Individual service owners/developers
- Start with your ABOM to provide context

Security/Risk Management

- Risk = Impact x Likelihood
    - Likelihood is important in these complicated systems
- DREAD
- CVSS v*X* – Base + Environmental Metrics

- Will often require a narrative
    - "If A, then B, then C…"

- Base concerns for exposure
- Compliance
- Service Level Agreements (SLAs)

Service Owner/Developer

- Why should/must I care?
- How do I fix this?

Tailoring to Your Requirements

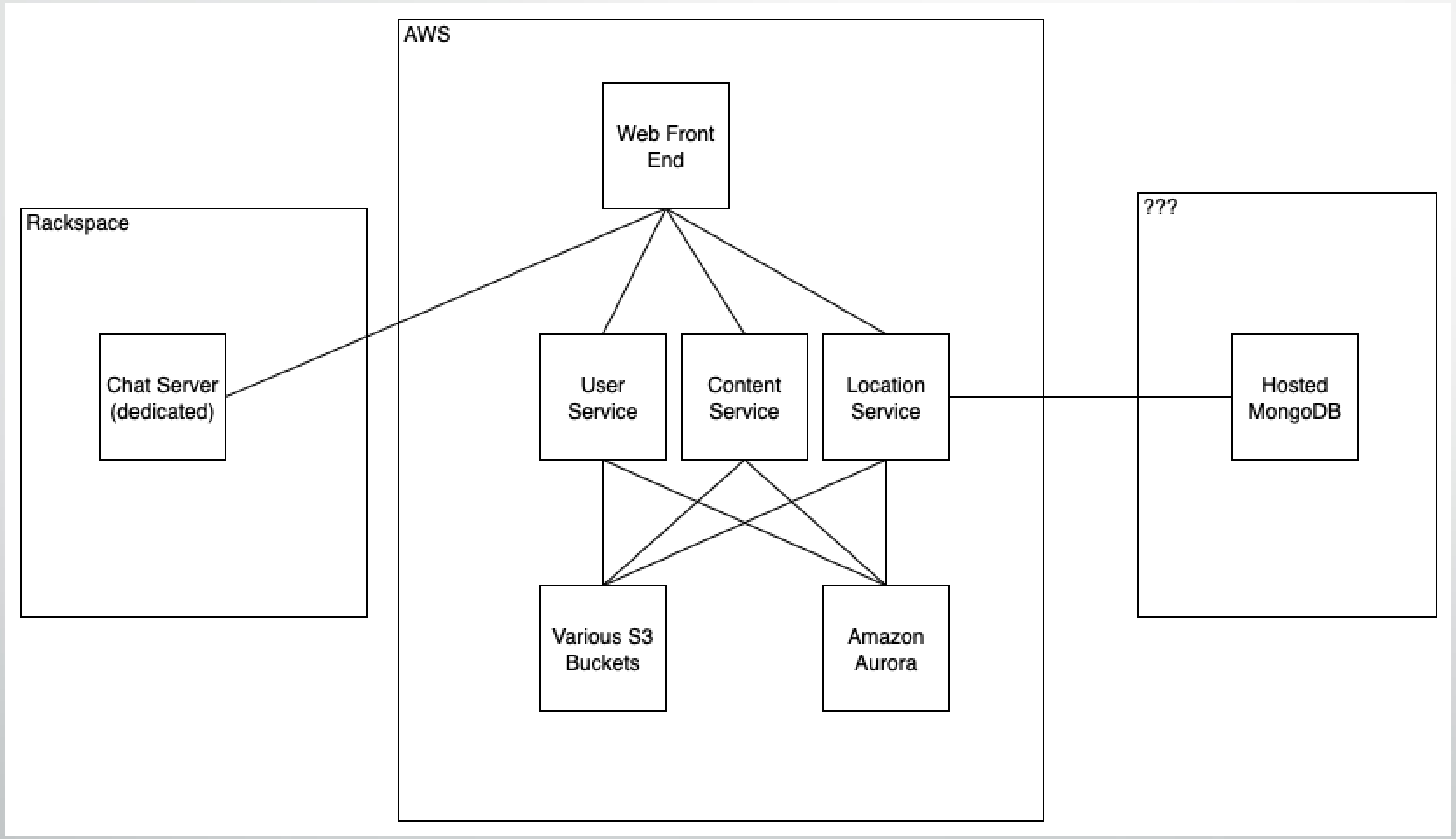
- Nobody has the resources to do everything they want
- If everything is important then nothing is important

- What services deal with the most critical data?
- What components of the system expose the most risk?
  - Are you more concerned that a container might have a blank root password or that your login routine might have Cross-Site Scripting (XSS) exposed?

Prioritized Testing

- Dynamic testing of public-facing sites and services
  - That’s what most bad guys will see
- Cloud configuration checks to identify potential unknown attack surface
  - Open S3 buckets, etc
- Prioritize additional activities based on resources

Tailoring to Your Requirements



Decisions You Might Make

- What's the attack surface?
  - Definitely known:
    - Web front end
    - Chat server
    - Hosted MongoDB
  - Need to determine additional exposure:
    - Scan exposed network assets
    - Check cloud configuration

Test Plan

- Enumerate assets to establish ABOM
- Cloud configuration check
    - Identify S3 buckets, gross IAM sins
- Network scan of exposed (and owned) IPs
- DAST scan of Web Front End
    - Maybe some manual penetration testing
- DAST/API scan of Chat Server
    - Again maybe some manual penetration testing

Questions?

Dan Cornell

@danielcornell

dan@denimgroup.com