PUG CHALLENGE EXCHANGE AMERICAS

Better Than Thinking Like a Hacker

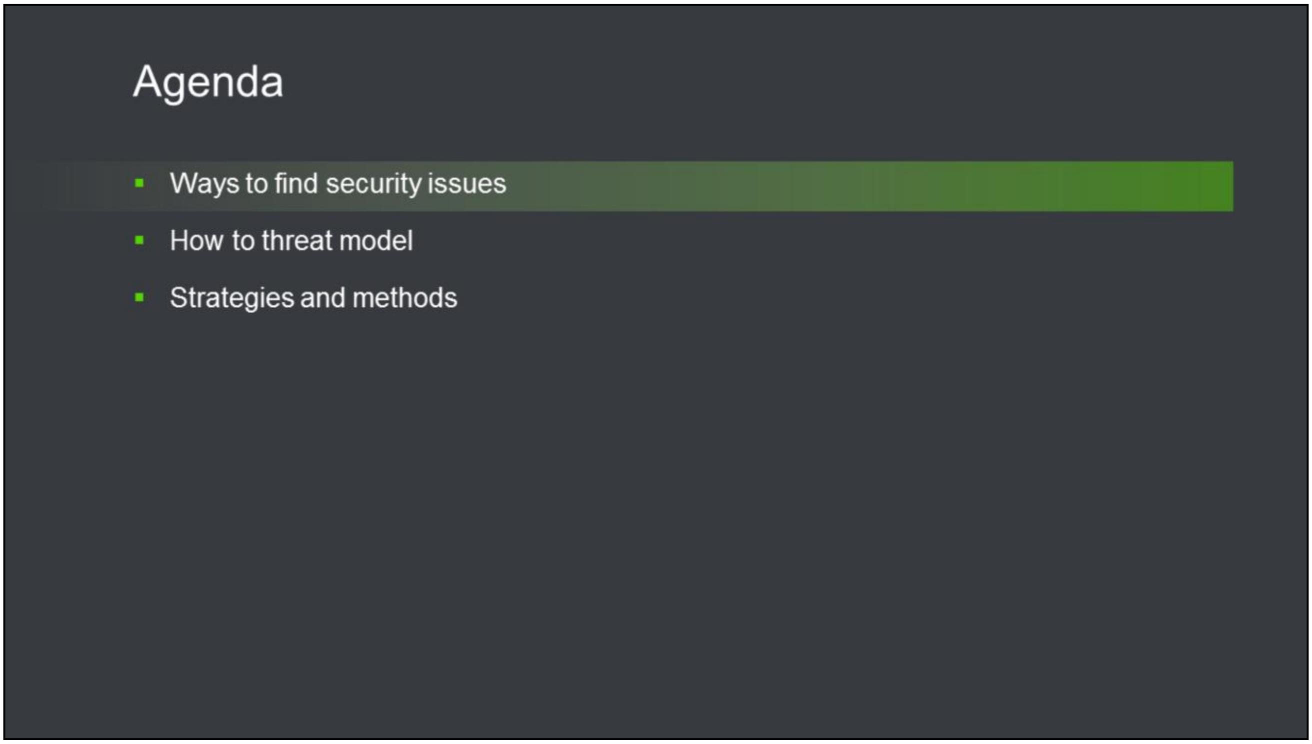
Threat Modeling - A Practical Way to Develop Secure Applications

Michael Solomon

June 2016

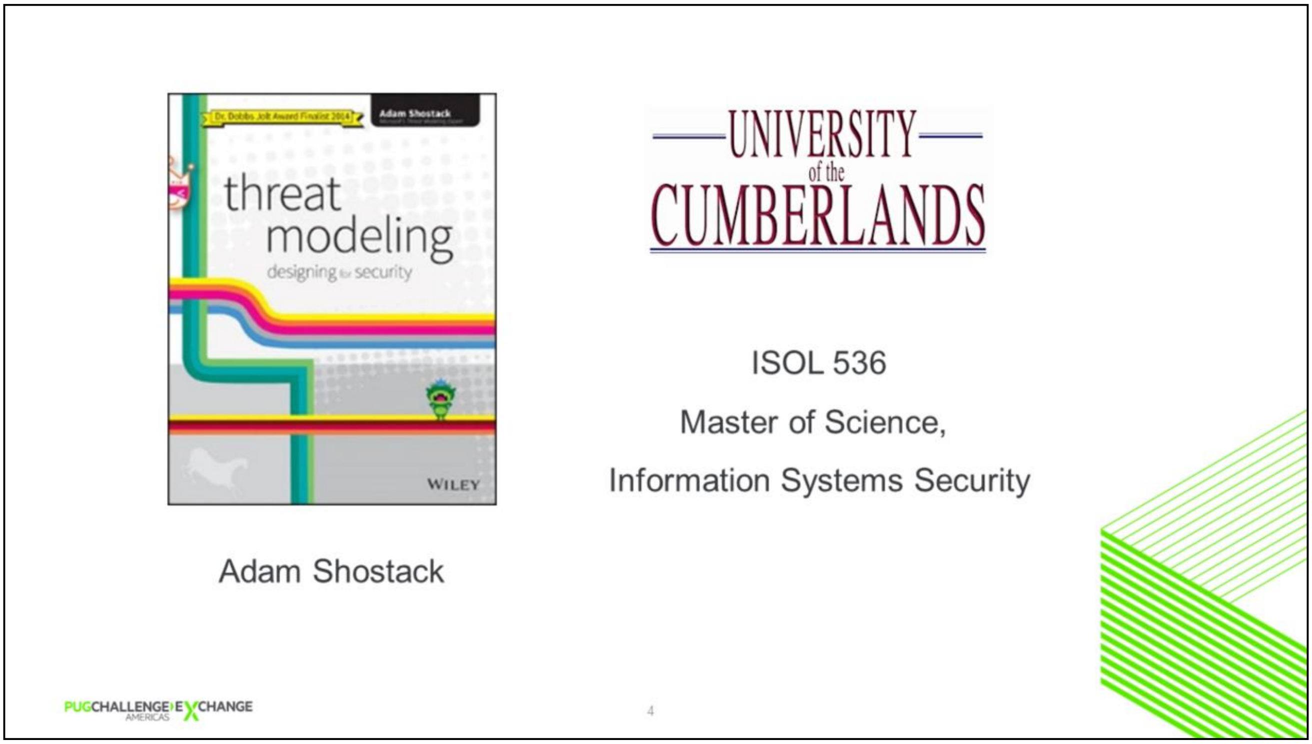
Wouldn’t it be great to find security issues before writing any code?

Security issues (problems/bugs) are hard to find. Intuitive approaches are limited (some in serious ways). Threat Modeling is effective, but difficult to do.



Agenda

- Ways to find security issues
- How to threat model
- Strategies and methods


Dr. Dobbs Jolt Award Finalist 2014
Adam Shostack
threat modeling
designing for security
WILEY

Adam Shostack

UNIVERSITY of the CUMBERLANDS

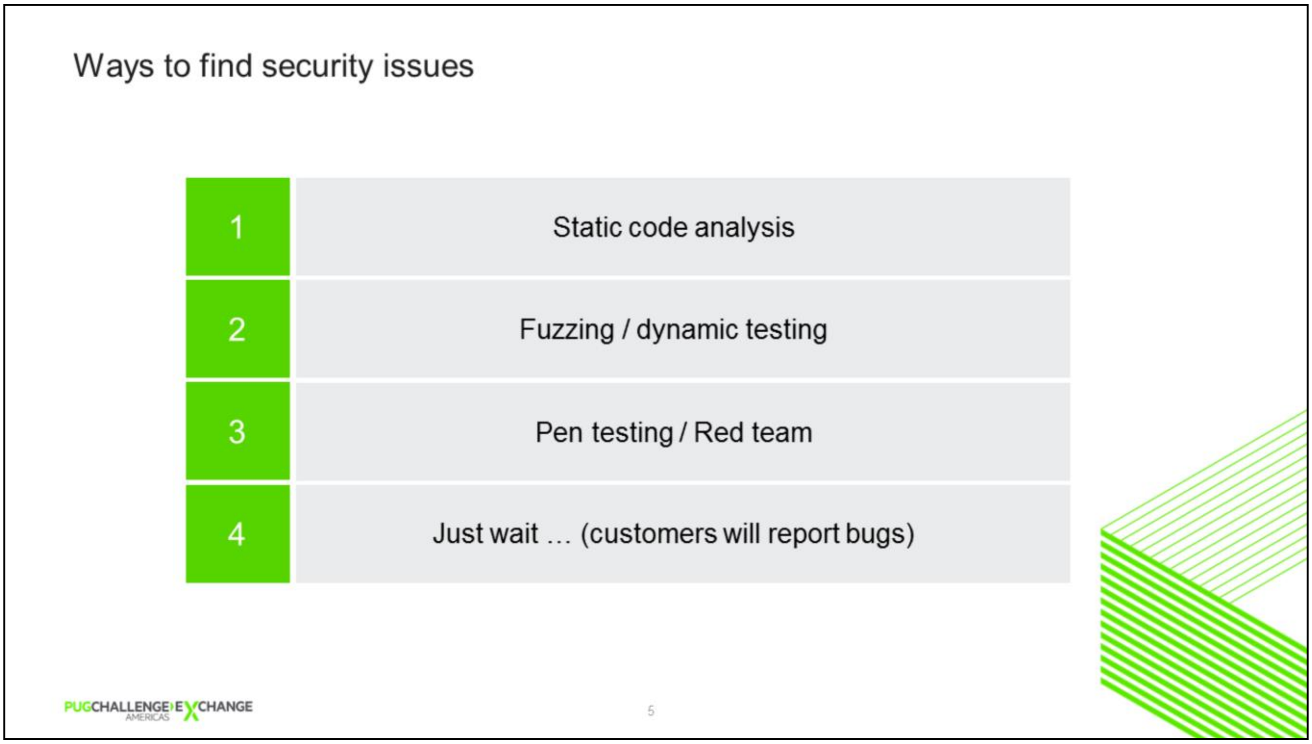
ISOL 536

Master of Science,

Information Systems Security

PUGCHALLENGE AMERICAS EXCHANGE
4

This talk is based on the book “Threat Modeling: Designing for Security” by Adam Shostack, and the course I taught (ISOL 536) for the University of the Cumberlands.



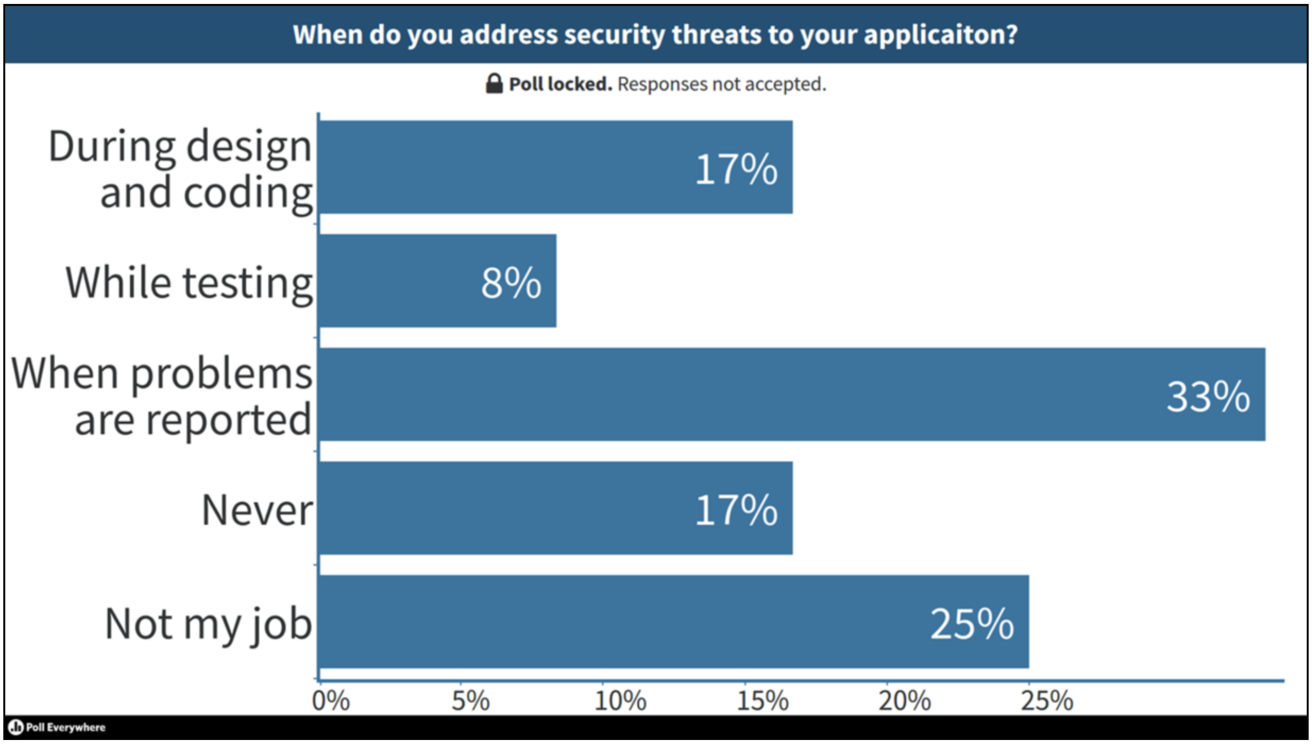
Ways to find security issues

1. Static code analysis
2. Fuzzing / dynamic testing
3. Pen testing / Red team
4. Just wait … (customers will report bugs)

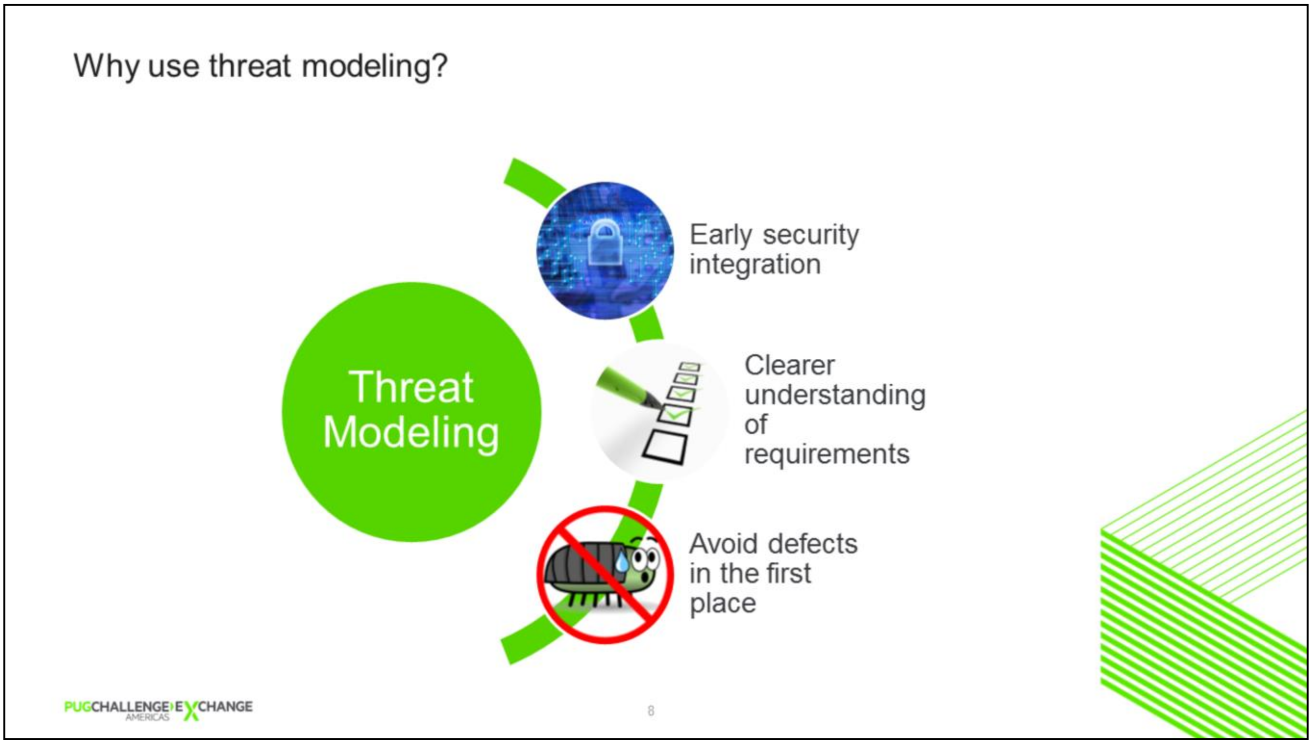
List progresses from early to late in the development process.

Late identification results in higher cost (impact) to fix.

Or … Threat modeling!



Results of a live poll of attendees of the talk at PUG Challenge Americas 2016.



Why use threat modeling?

Threat Modeling

Early security integration

Clearer understanding of requirements

Avoid defects in the first place

How to threat model

Think like an attacker? A popular approach.

But, how can you know what they know? Or, what they will do?

What if you get it wrong?

Similar to “cook like a professional chef”. You can’t really think like every chef – they’re all different!

Chefs meet the needs of their specific kitchens (and environments.) It is FAR BETTER to know how to be a chef, and then meet the needs of your situation.

To take this further, what would a state-sponsored attacker do to get at your data? How about a hacktivist?

Thinking like at attacker isn’t good enough – you’ll miss important things.

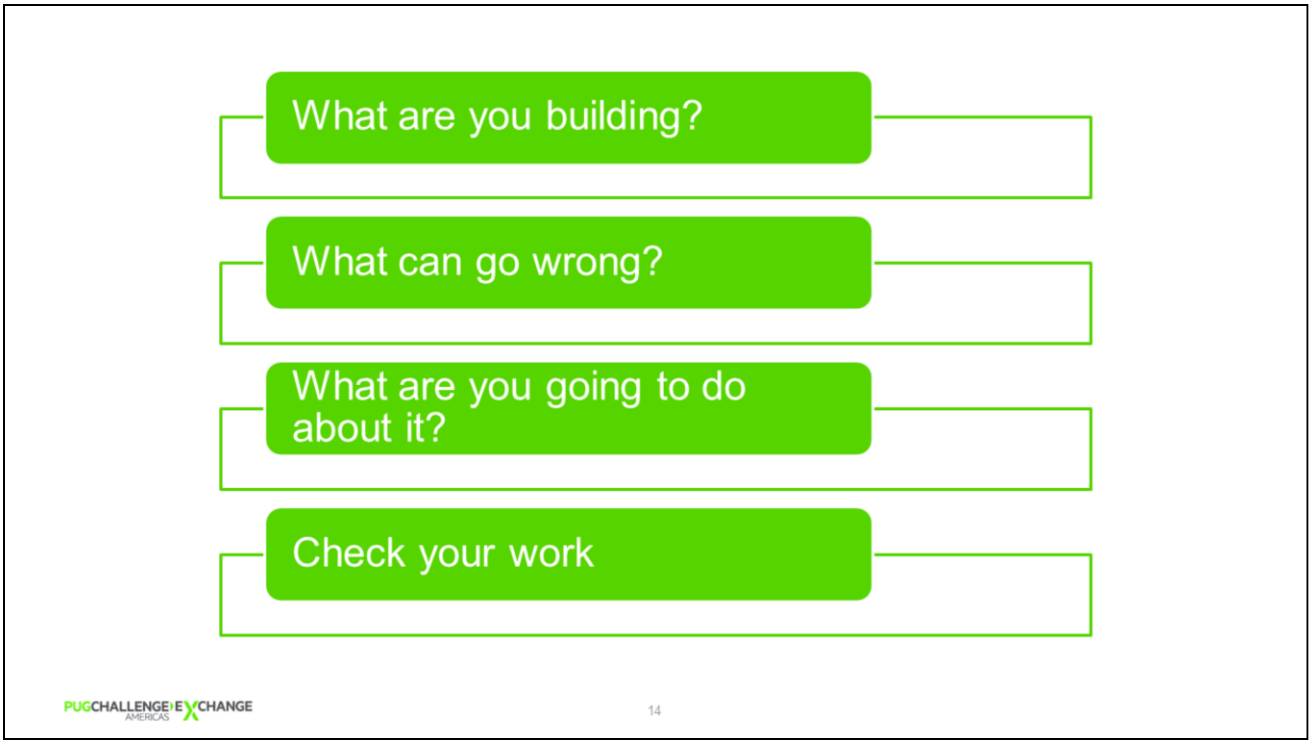
Assets

What does an attacker really want?

It’s hard to judge what an attacker will want: maybe they want a random computer to serve pirated music. What do you protect? Everything inside the perimeter, because every computer is a stepping stone to the inner-sanctum. So what do you learn by making an asset list?

Focus on what you’re building

You’re more likely to understand the thing you’re building than any other possible starting point.



What are you building?

What can go wrong?

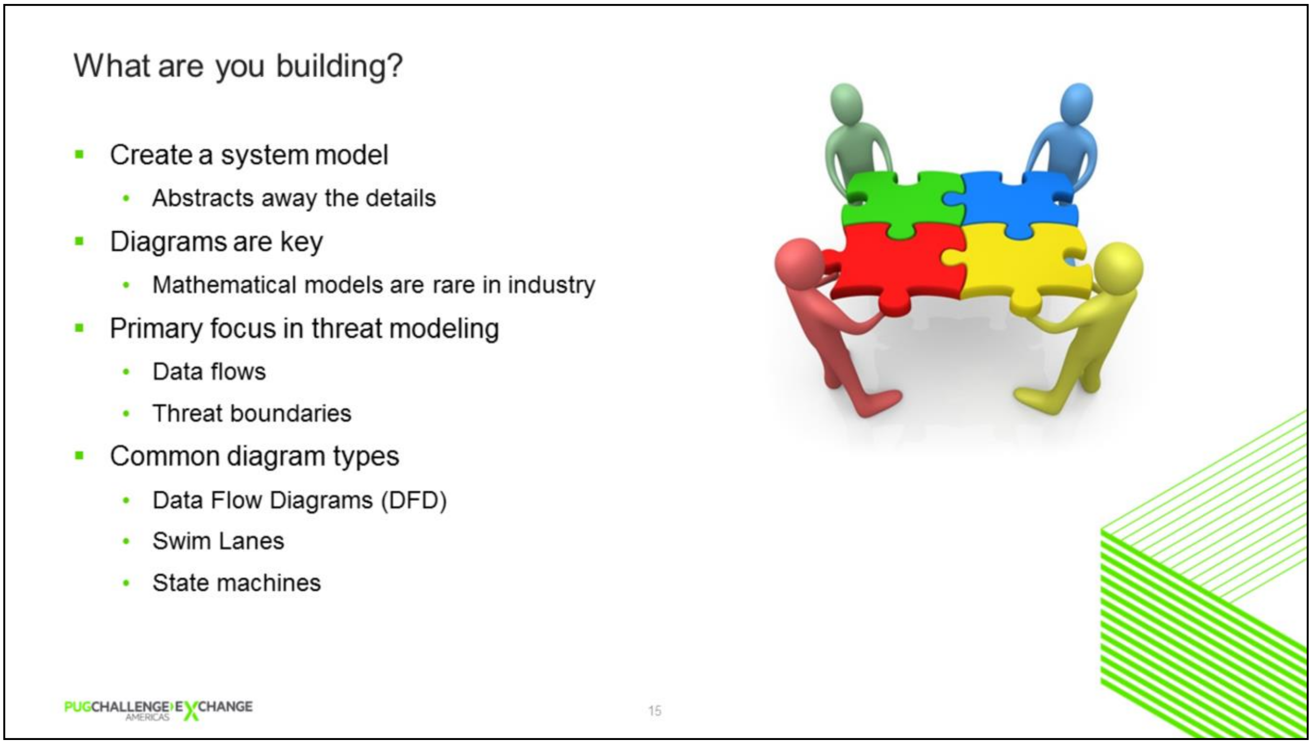
What are you going to do about it?

Check your work

PUGCHALLENGE AMERICAS EXCHANGE

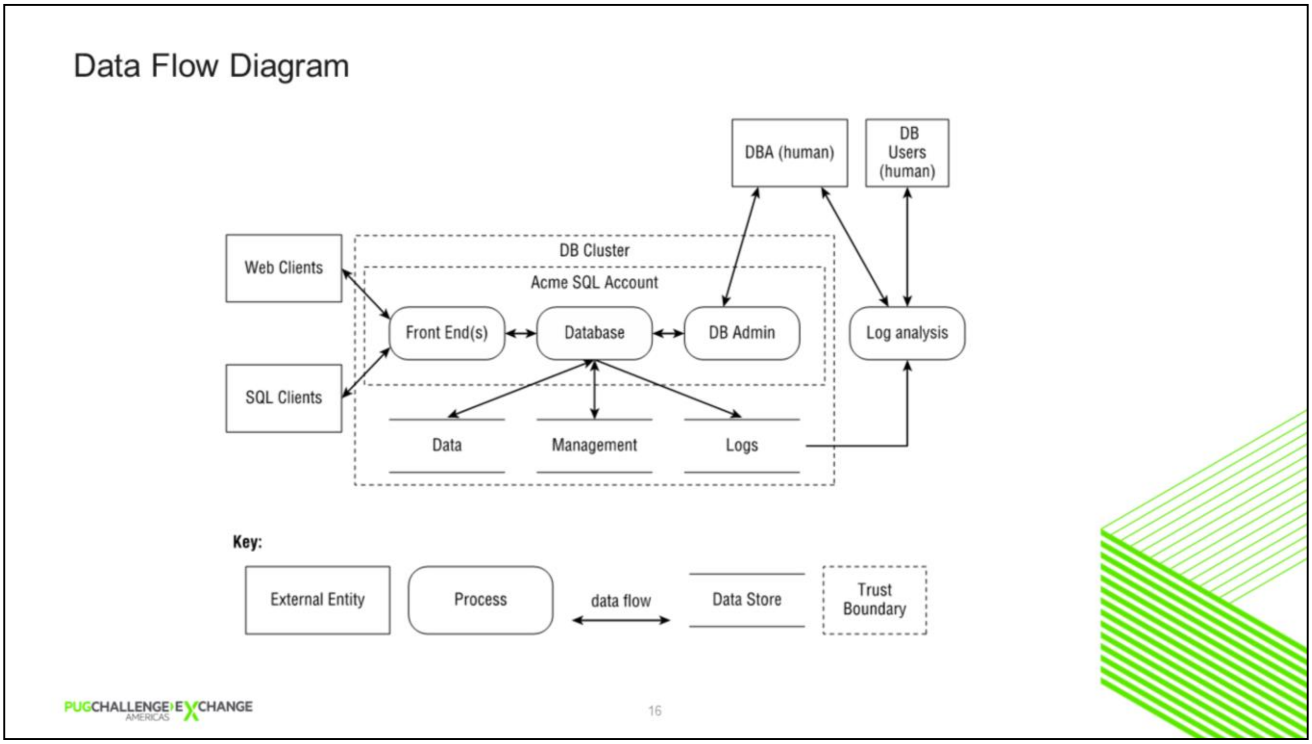
14

Threat Modeling outline



What are you building?

- Create a system model
  - Abstracts away the details
- Diagrams are key
  - Mathematical models are rare in industry
- Primary focus in threat modeling
  - Data flows
  - Threat boundaries
- Common diagram types
  - Data Flow Diagrams (DFD)
  - Swim Lanes
  - State machines



Data Flow Diagram

DBA (human)

DB Users (human)

Web Clients

SQL Clients

DB Cluster

Acme SQL Account

Front End(s)

Database

DB Admin

Log analysis

Data

Management

Logs

Key:

External Entity

Process

data flow

Data Store

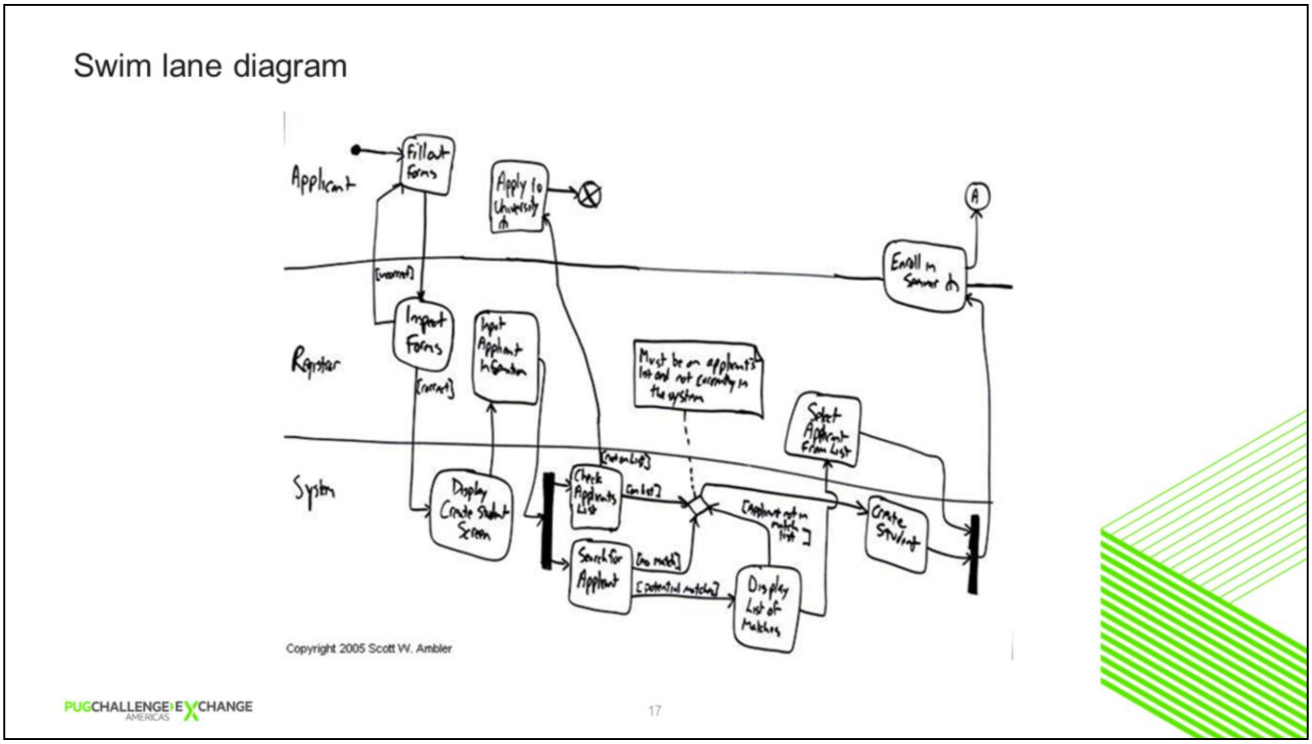
Trust Boundary

Developed in the early 70s, and still useful

- Simple: easy to learn, sketch
- Threats often follow data

Abstracts programs into:

- Processes: your code
- Data stores: files, databases, shared memory
- Data flows: connect processes to other elements
- External entities: everything but your code & data. Includes people & cloud software
- Trust boundaries now made explicit



Swim lane diagram

Copyright 2005 Scott W. Ambler

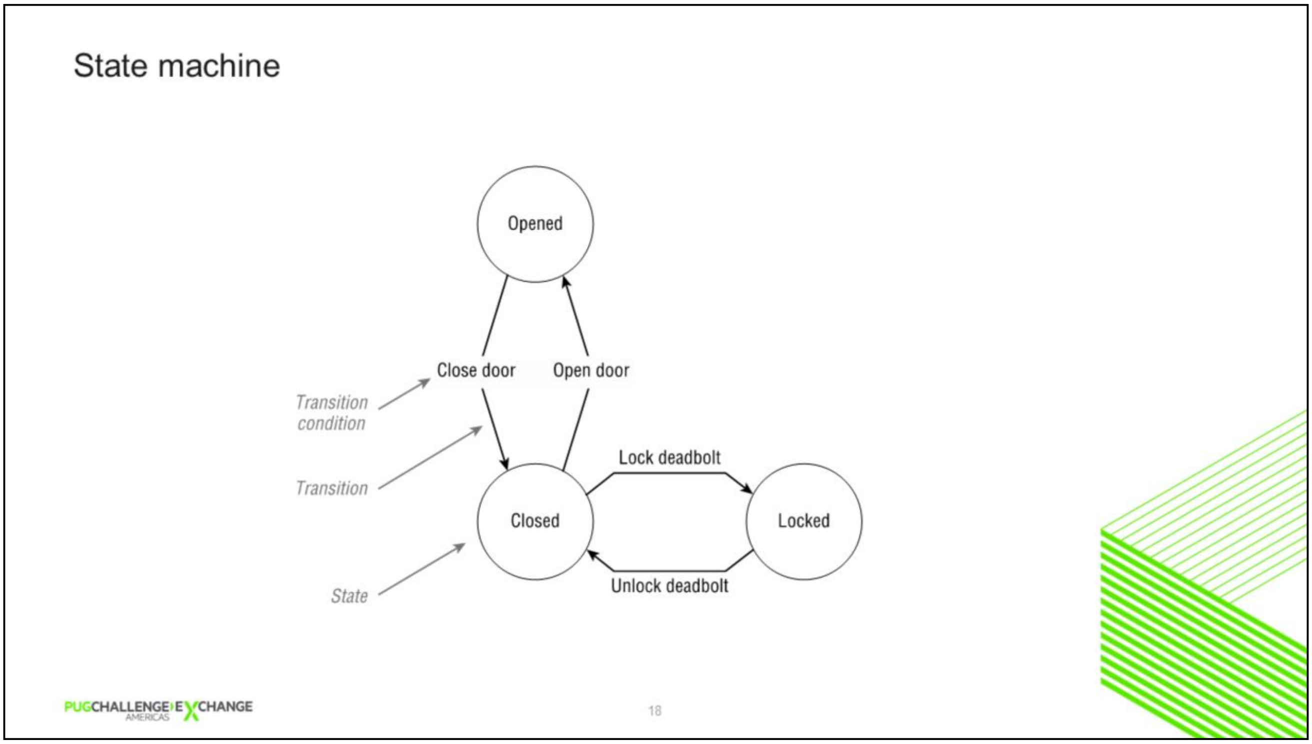
PUGCHALLENGE EXCHANGE AMERICAS

17

Show two or more entities communicating, each “in a lane”

Useful for network communication

Lanes have implicit boundaries between them

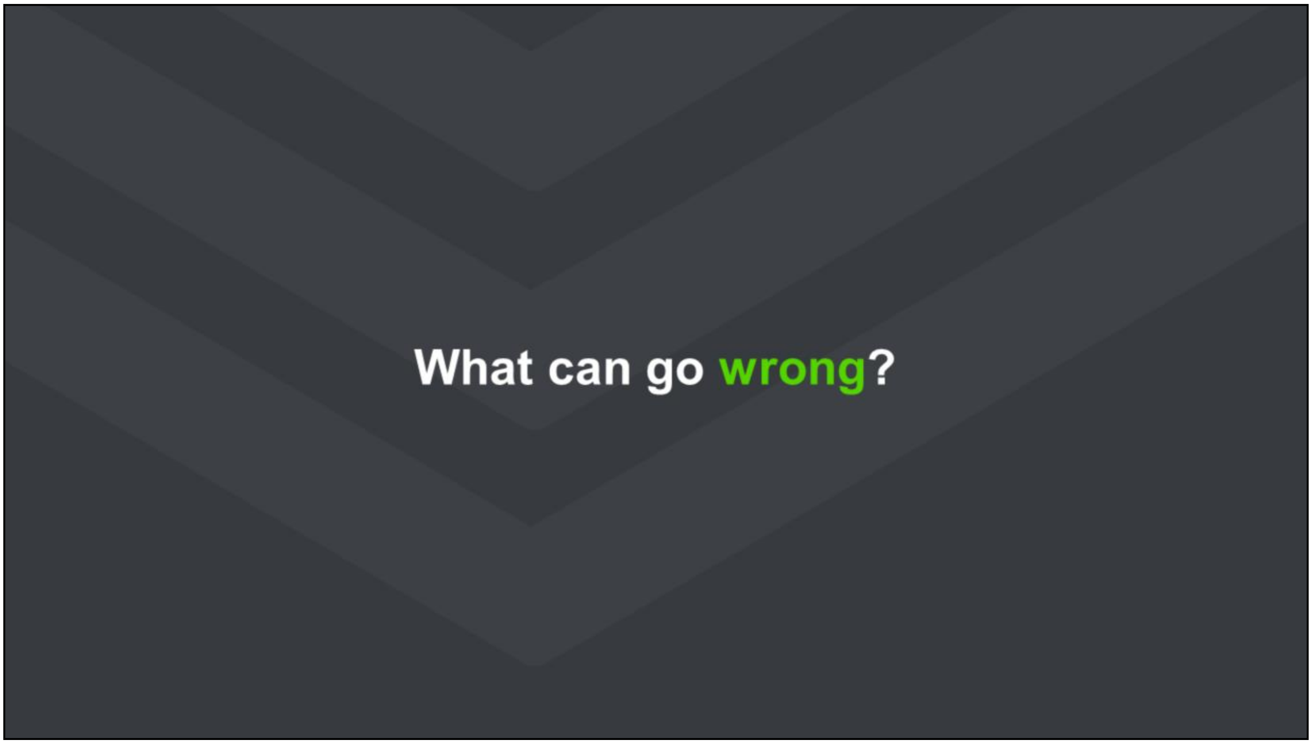


State machine

Helpful for considering what changes security state

- For example, unauthenticated to authenticated
- User to root/admin

Rarely shows boundaries



What can go wrong?

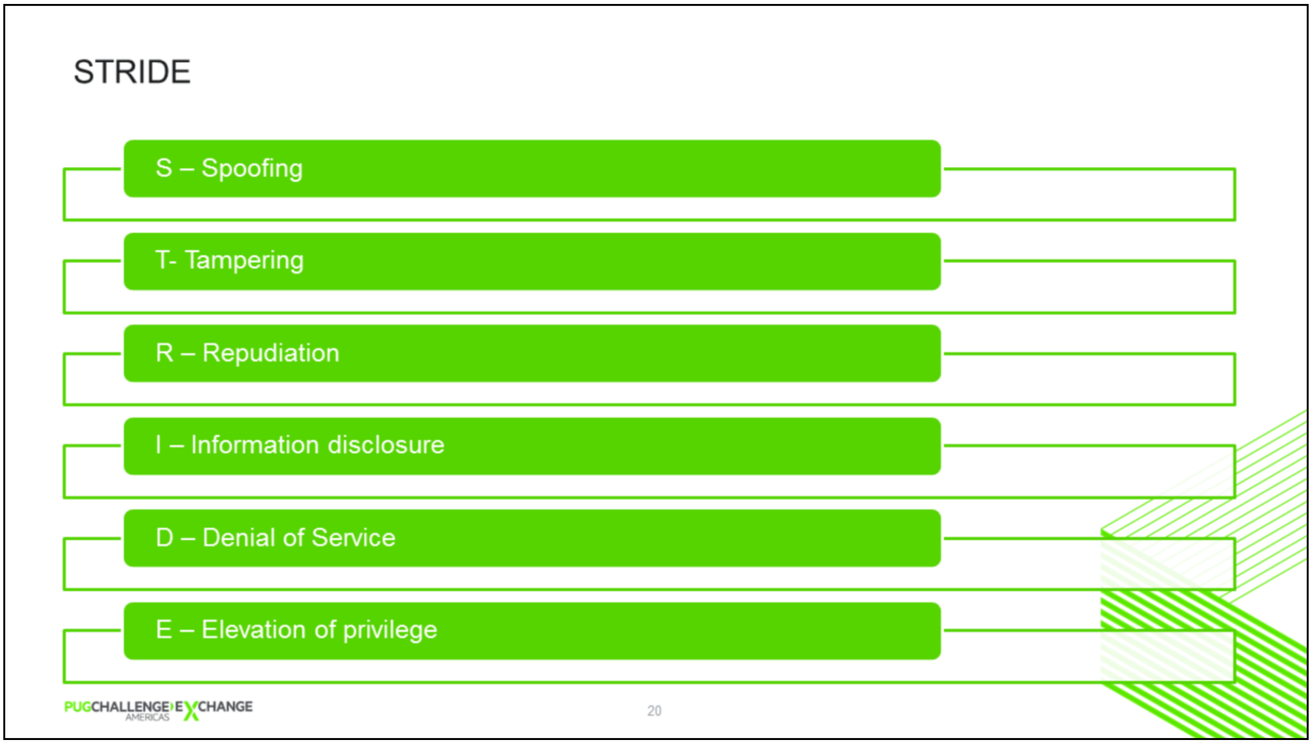
Fun to brainstorm

Mnemonics, trees or libraries of threats can all help structure thinking

Structure helps get you towards completeness and predictability

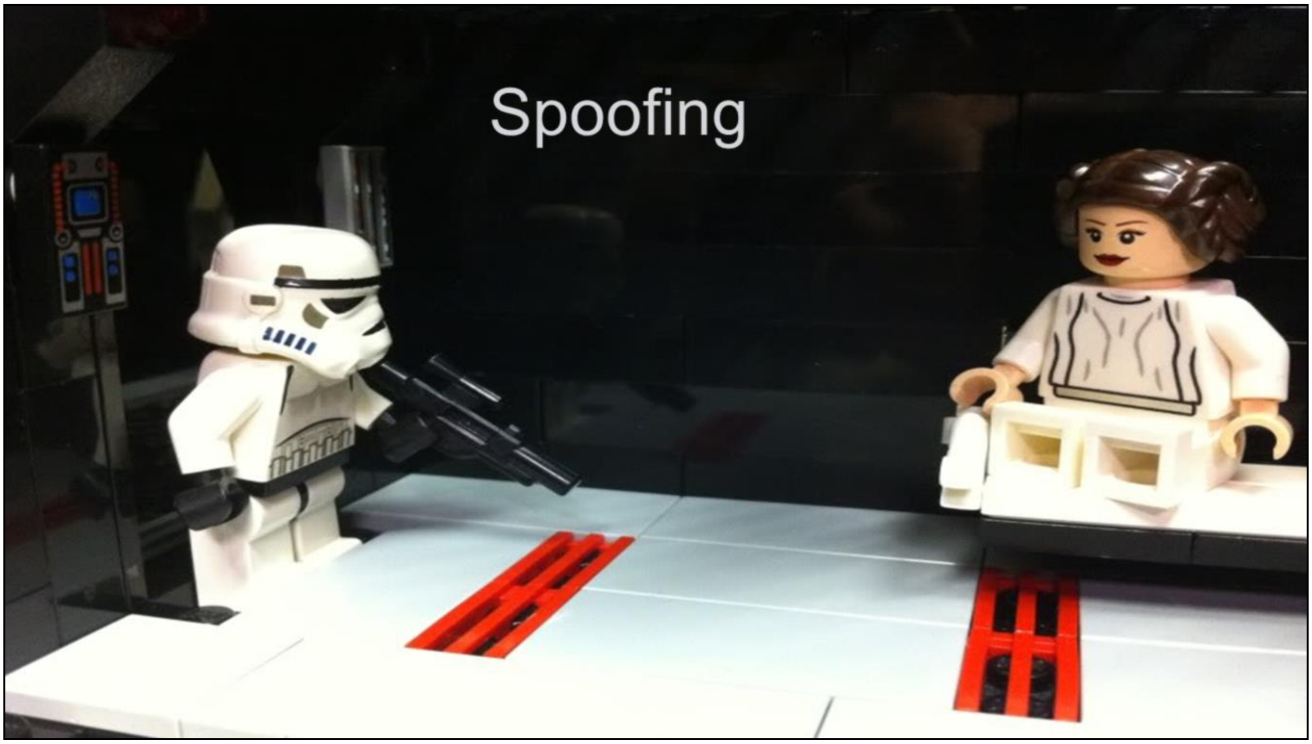
STRIDE is a mnemonic

- Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege



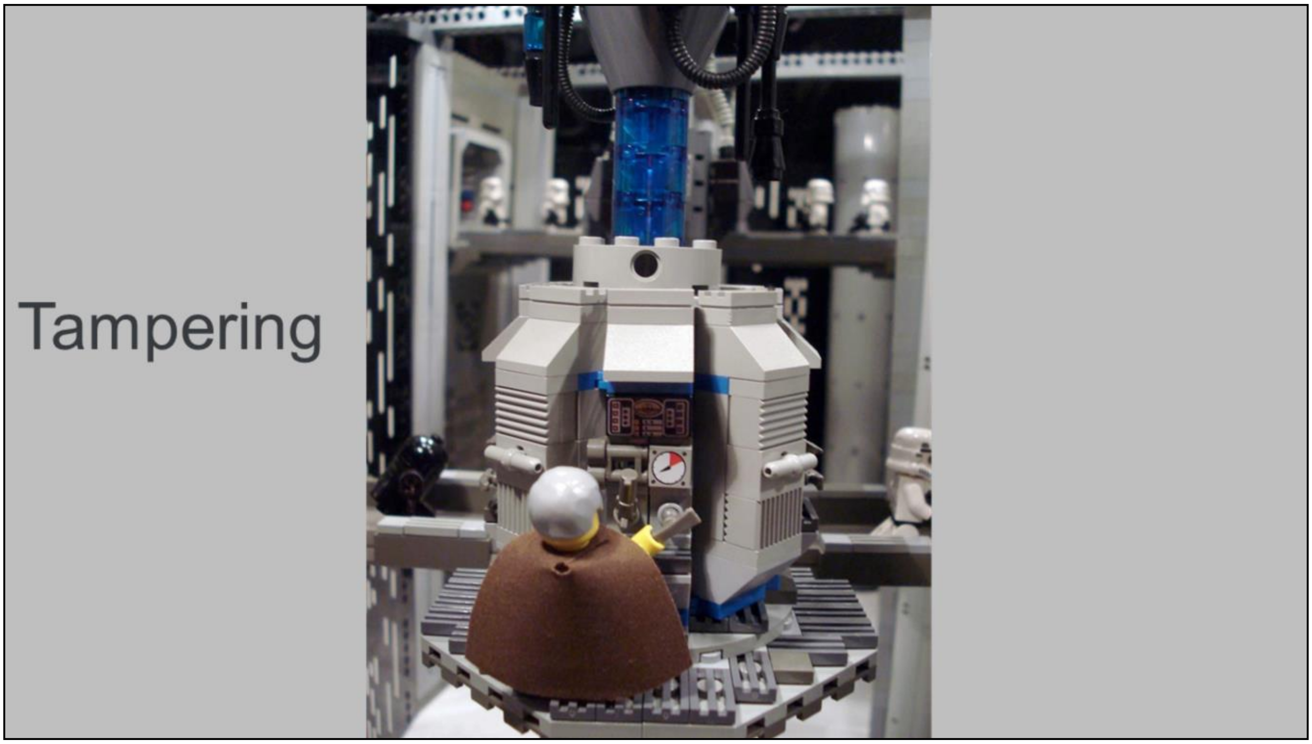
STRIDE

- S – Spoofing
- T- Tampering
- R – Repudiation
- I – Information disclosure
- D – Denial of Service
- E – Elevation of privilege



Spoofing

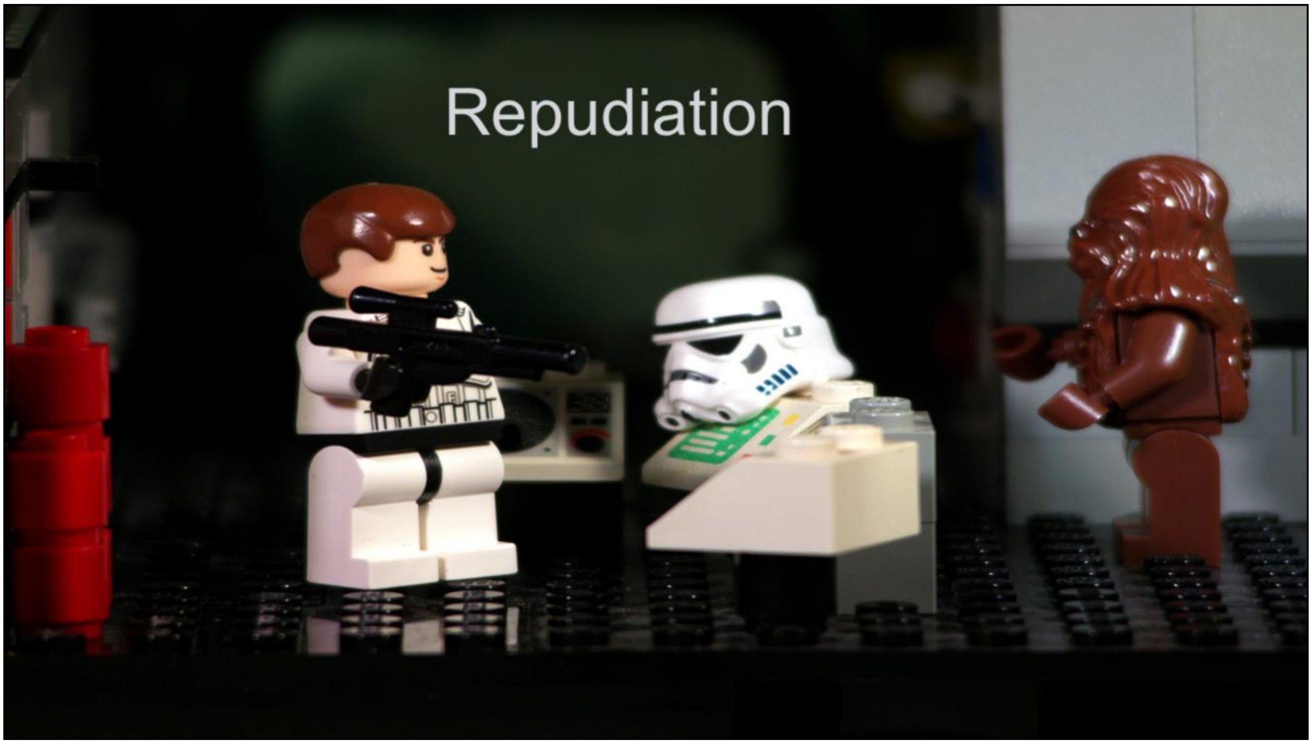
Spoofing is pretending to be something or someone you’re not, such as paypal.com, the C library, a Nigerian Prince with money to send you, or a Stormtrooper



Tampering

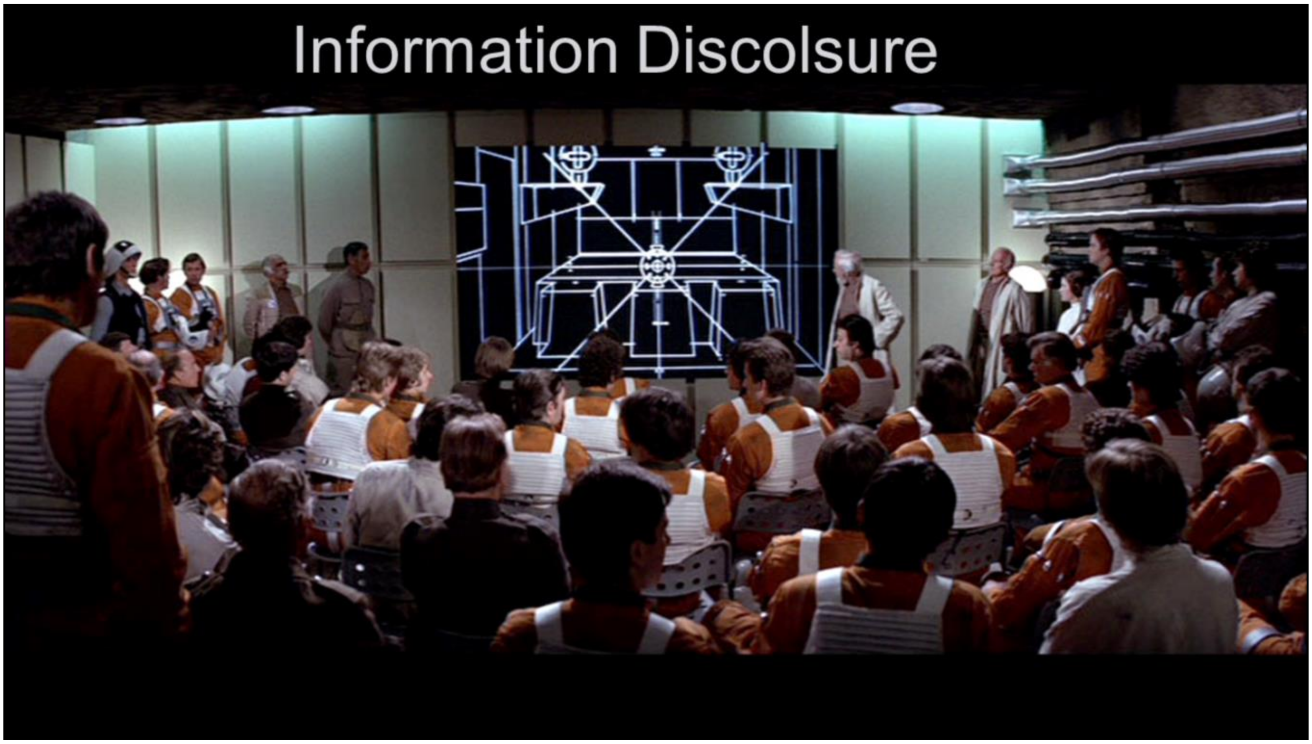
Tampering is modifying something you’re not authorized to modify, such as files on disk, packets on a network, data in memory, or the controls to a tractor beam.

http://pinlac.com/LegoDSTractorBeam.html



Repudiation

Repudiation is claiming that you’re not responsible for something, or lying to avoid responsibility. That might be saying that you didn’t eat the last cookie, you didn’t click buy it now, you never saw the email, or that you’ve had a reactor meltdown, and it’s very dangerous.

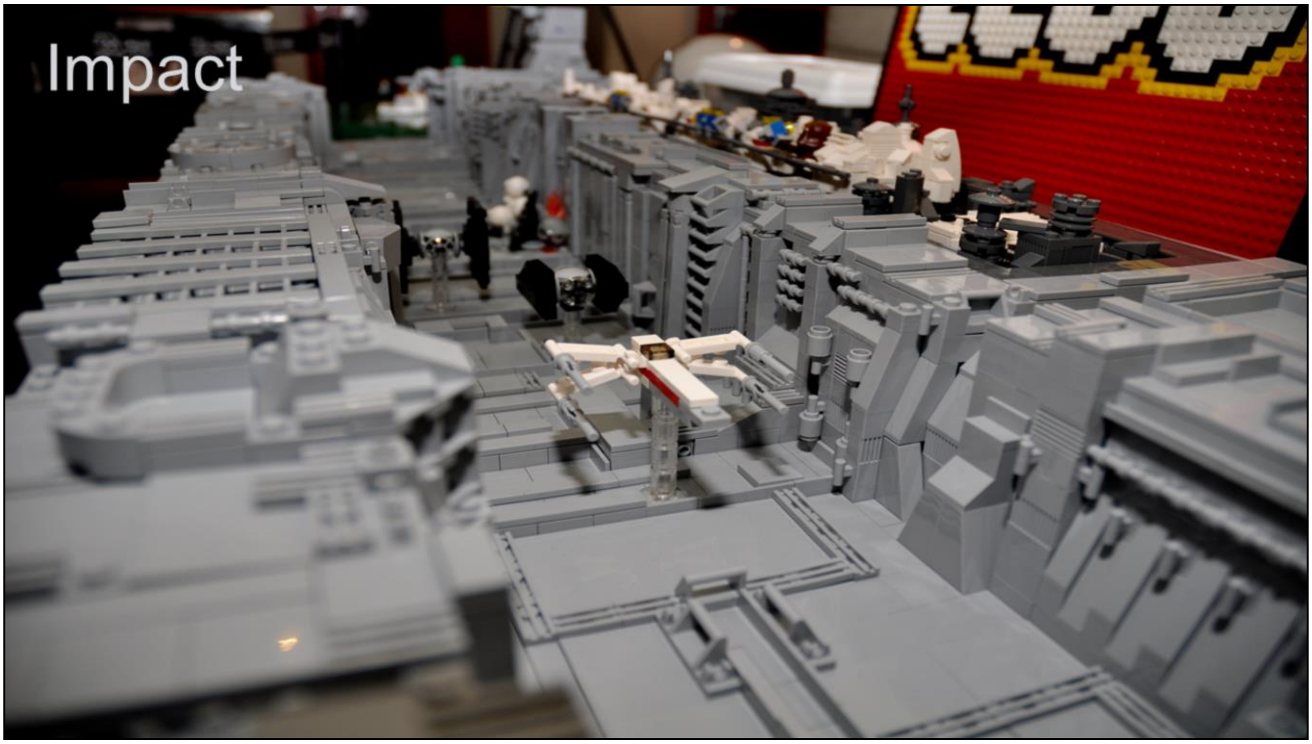


Information Discolsure

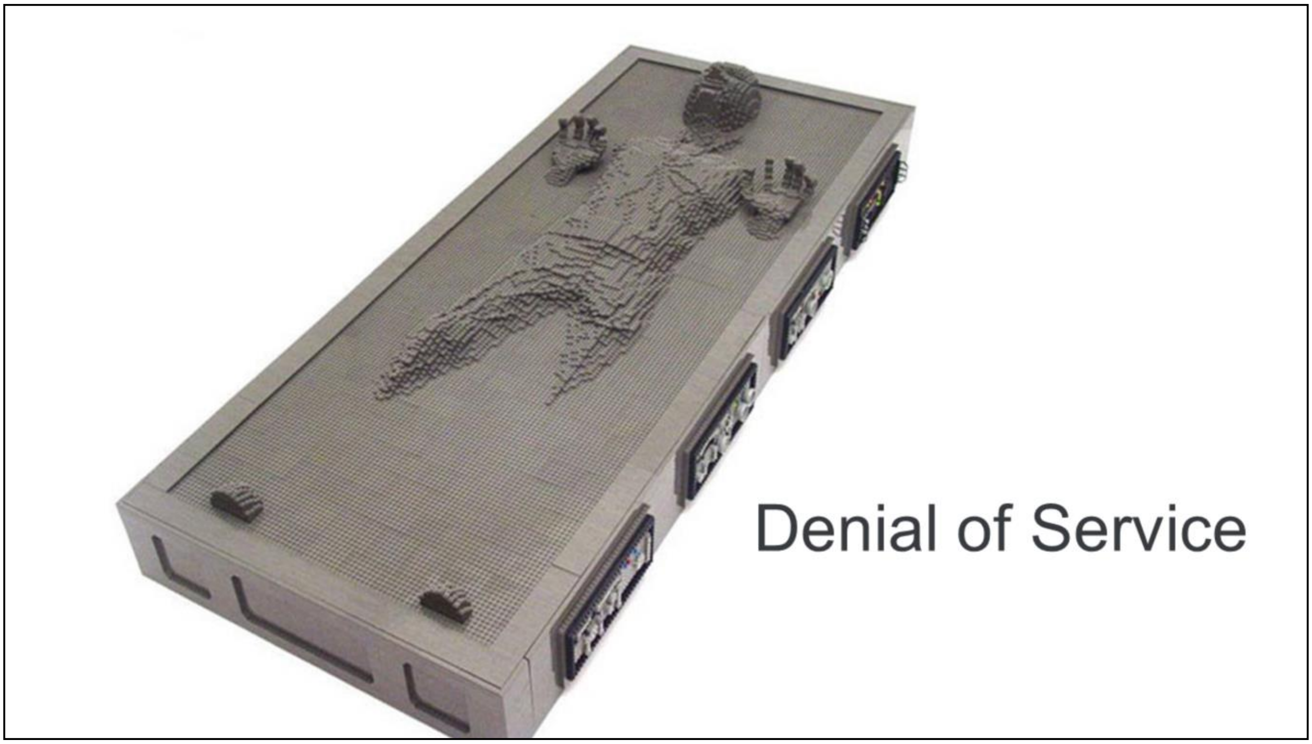
Information disclosure is about being able to read information that you’re not authorized to read. That might be files on disk, packets on a network, or the secret plans to a battle station.

Instructors might prefer the next (hidden) slide; this one gives you a chance to stress how central information disclosure is to the movie.

Information Disclosure



Impact



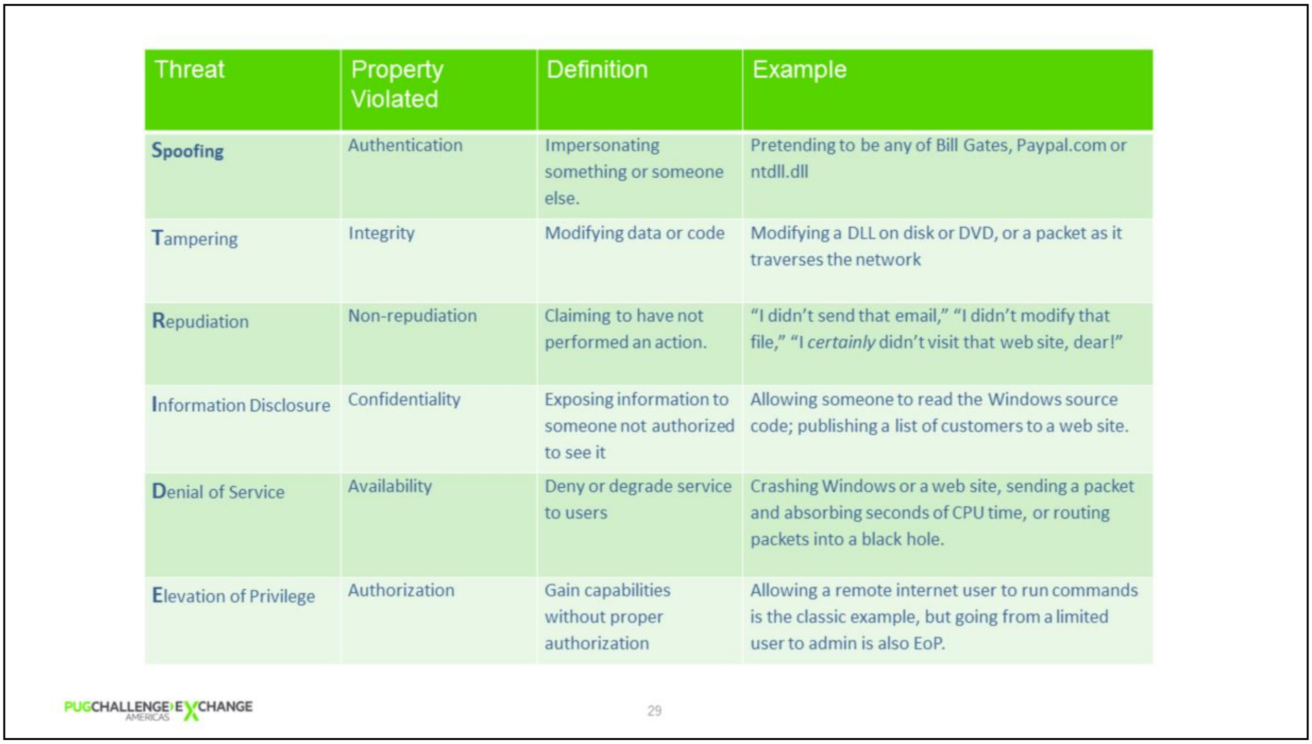
Denial of Service

Not having access to a resource, such as CPU, disk, bandwidth, or Han Solo

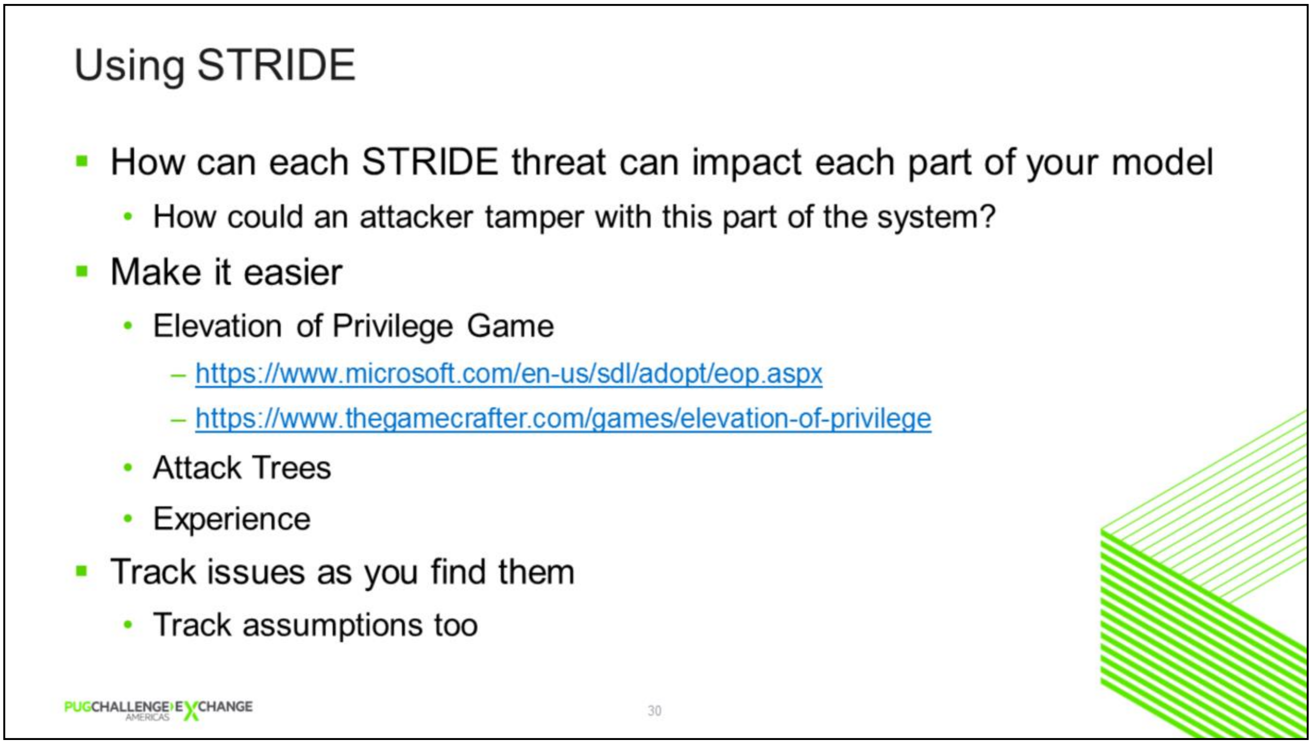
Alternately http://lego.wikia.com/wiki/Han_Solo

Elevation of Privilege

Doing things beyond your authorization, including running code as root, changing the files on a web server, getting a program to run your code, or telling stormtroopers that these aren’t the driods they’re looking for



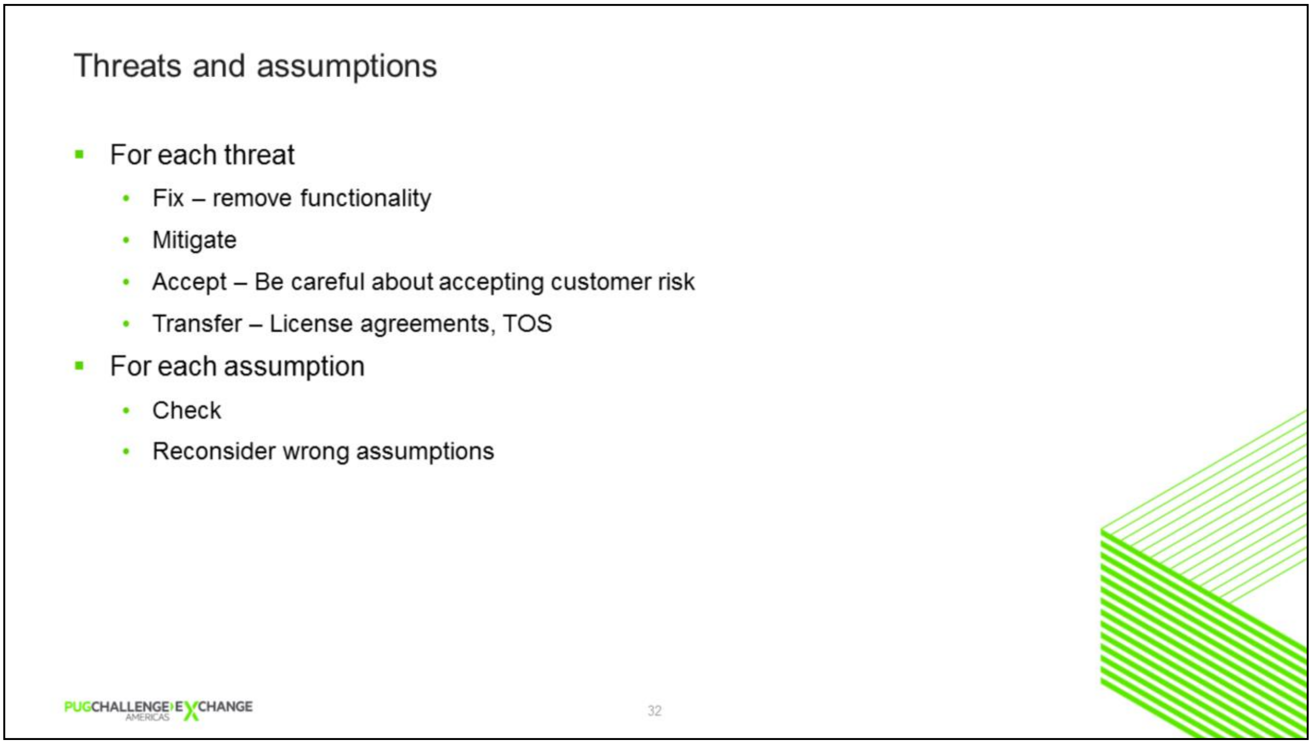
Threat	Property Violated	Definition	Example
Spoofing	Authentication	Impersonating something or someone else.	Pretending to be any of Bill Gates, Paypal.com or ntdll.dll
Tampering	Integrity	Modifying data or code	Modifying a DLL on disk or DVD, or a packet as it traverses the network
Repudiation	Non-repudiation	Claiming to have not performed an action.	"I didn't send that email," "I didn't modify that file," "I *certainly* didn't visit that web site, dear!"
Information Disclosure	Confidentiality	Exposing information to someone not authorized to see it	Allowing someone to read the Windows source code; publishing a list of customers to a web site.
Denial of Service	Availability	Deny or degrade service to users	Crashing Windows or a web site, sending a packet and absorbing seconds of CPU time, or routing packets into a black hole.
Elevation of Privilege	Authorization	Gain capabilities without proper authorization	Allowing a remote internet user to run commands is the classic example, but going from a limited user to admin is also EoP.



Using STRIDE

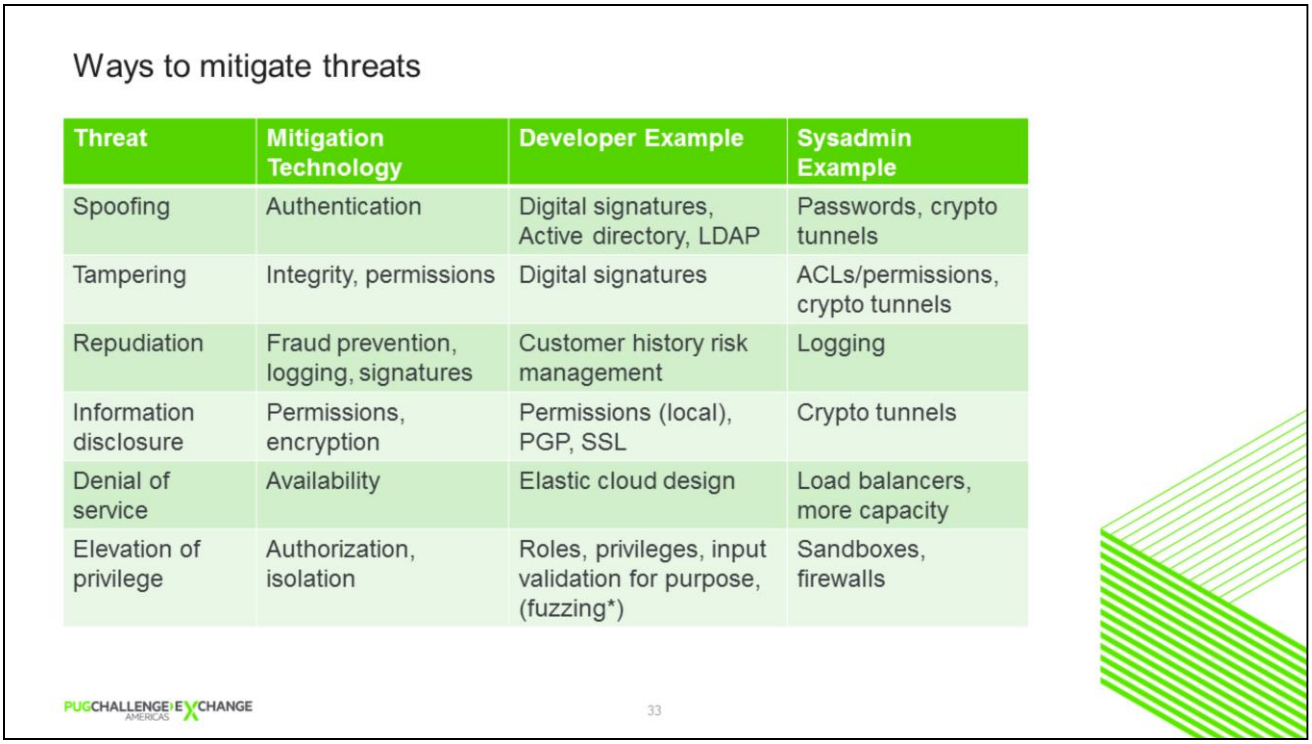
- How can each STRIDE threat can impact each part of your model
  - How could an attacker tamper with this part of the system?
- Make it easier
  - Elevation of Privilege Game
    - https://www.microsoft.com/en-us/sdl/adopt/eop.aspx
    - https://www.thegamecrafter.com/games/elevation-of-privilege
  - Attack Trees
  - Experience
- Track issues as you find them
  - Track assumptions too

What are you going to do about it?



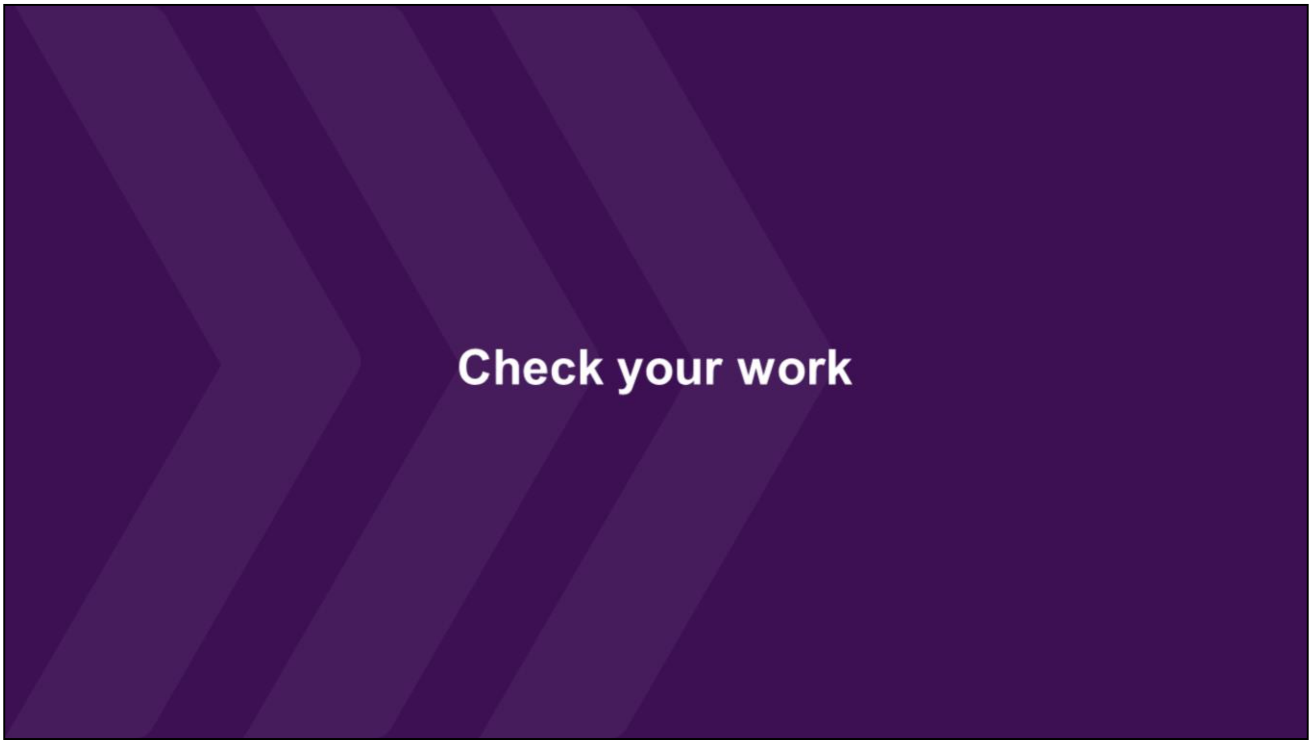
Threats and assumptions

- For each threat
  - Fix – remove functionality
  - Mitigate
  - Accept – Be careful about accepting customer risk
  - Transfer – License agreements, TOS
- For each assumption
  - Check
  - Reconsider wrong assumptions



Ways to mitigate threats

Threat	Mitigation Technology	Developer Example	Sysadmin Example
Spoofing	Authentication	Digital signatures, Active directory, LDAP	Passwords, crypto tunnels
Tampering	Integrity, permissions	Digital signatures	ACLs/permissions, crypto tunnels
Repudiation	Fraud prevention, logging, signatures	Customer history risk management	Logging
Information disclosure	Permissions, encryption	Permissions (local), PGP, SSL	Crypto tunnels
Denial of service	Availability	Elastic cloud design	Load balancers, more capacity
Elevation of privilege	Authorization, isolation	Roles, privileges, input validation for purpose, (fuzzing*)	Sandboxes, firewalls

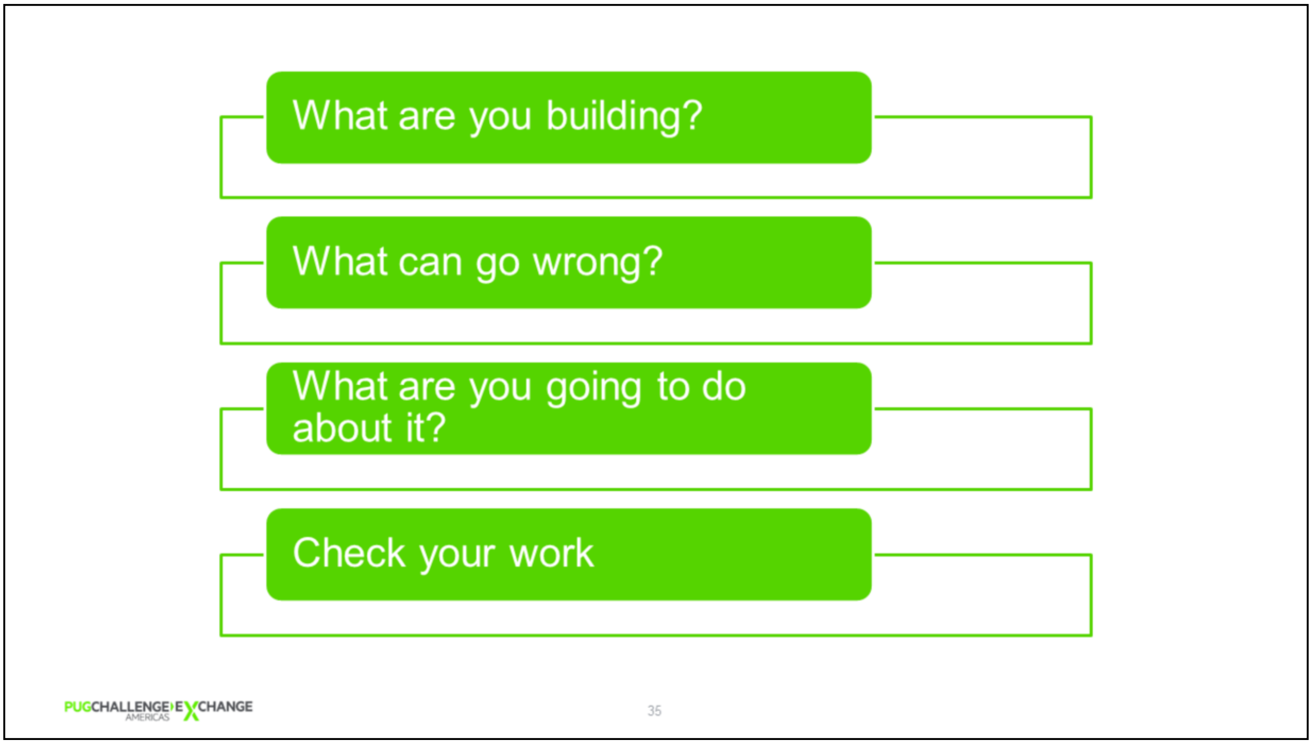


Check your work

Quality assurance

Check that you covered all the threats & assumptions

Check that each is covered well

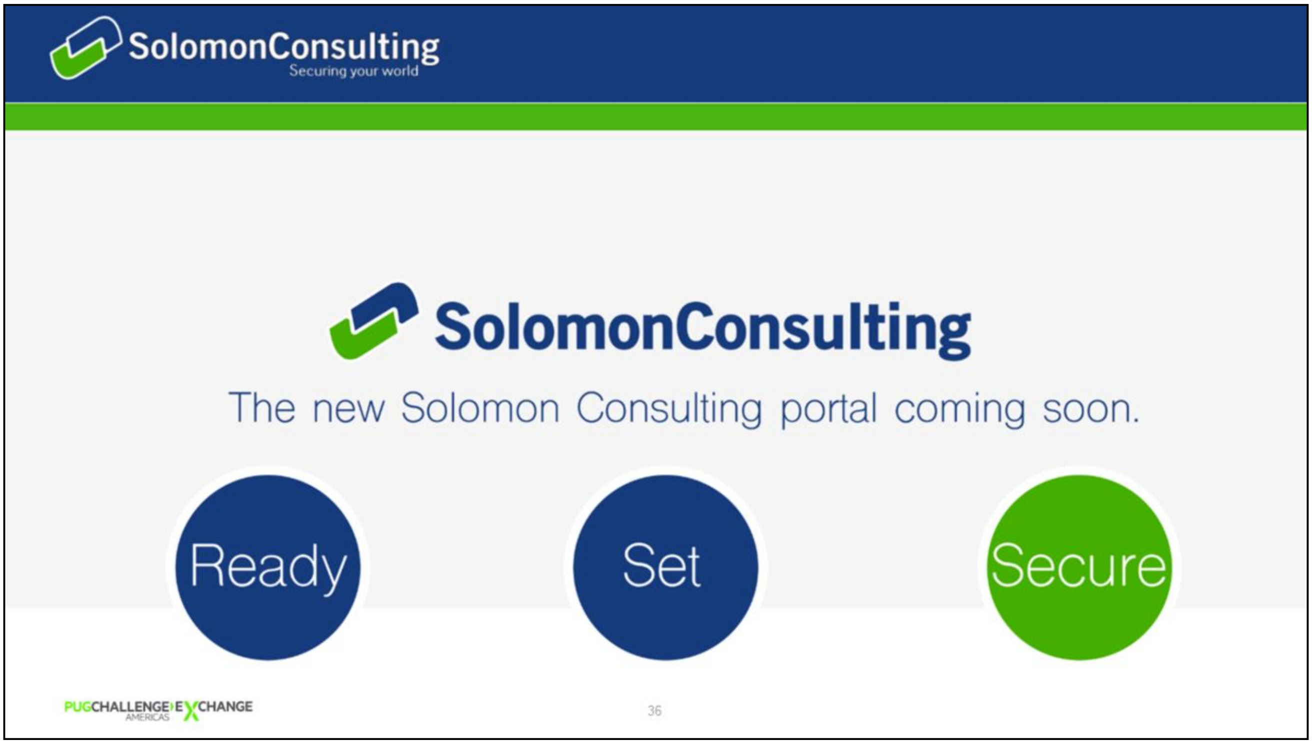


What are you building?

What can go wrong?

What are you going to do about it?

Check your work



SolomonConsulting
Securing your world

SolomonConsulting

The new Solomon Consulting portal coming soon.

Ready

Set

Secure

www.solomonconsulting.com