endava

SECURITY BEYOND THE LIBRARIES

software security fundamentals

Eoin Woods
Endava
@eoinwoodz

INTRODUCTION

endava

INTRODUCTION

- **Security** is a **difficult** thing to achieve
- **Development teams** often start with **technologies**
  - "SSL" "Spring Security" "SSO" "OAuth" "FindBugs" "Fortify" "Tripwire"
- This is completely the **wrong way around**
  - need to **understand** your **risks before** finding **solutions**
- In this talk we discuss how to base **security** on **risks**

endava

CAVEATS

- This talk is **introductory** in nature
  - some things aren't talked about & some things are just introduced
- Talk is for **system developers** not security engineers
  - subtleties are skipped, some things simplified to their essentials
  - you still probably need a security specialist
- Don't talk much about technologies or coding practice

INTRODUCING SECURITY

endava

THE NEED FOR SECURITY

- We need systems that are **dependable** in spite of
  - **Malice**, **Error** and **Mischance**
  - People are sometimes bad, stupid or just unlucky
- System security attempts to **mitigate** these situations
- Anything of **value** may attract unwelcome attention
  - Theft, Fraud, Destruction, Disruption

THE NEED FOR SECURITY

- **Why do we care** about these factors?

- Each of them implies a **loss** of some sort
  - Time
  - Money
  - Privacy
  - Reputation
  - Advantage

endava

THINKING POINT

What **risks of loss** are there in your system?

think beyond money and personal data …

endava

WHAT IS SECURITY?

- Security is the business of **managing risks**
  - Security is a type of insurance
  - Balances **cost** and **effort** against risk of **loss**

- Some basic terminology
  - **resources** - things of value that (may) need protection
  - **principals** (or actors) - people ("entities") interacting with the system
  - **policies** - the rules to control access to the resources
  - **threats** - the reason that the rules may be broken

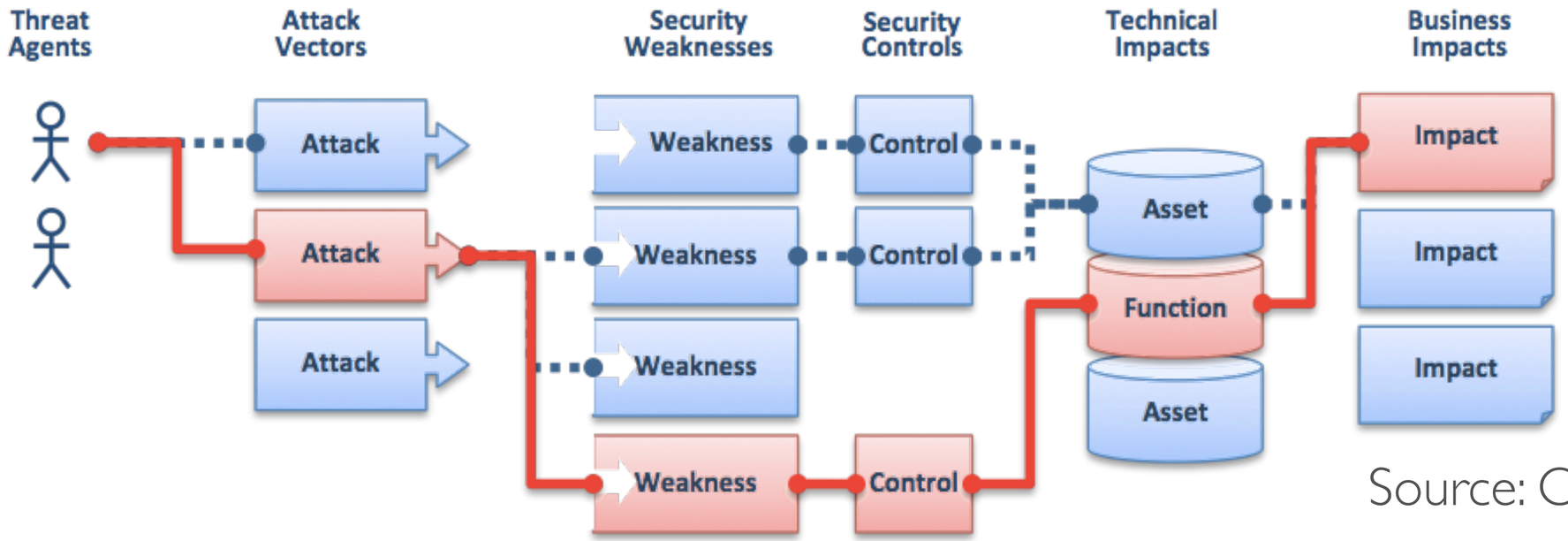
WHAT IS SECURITY?

- Security is **multi-dimensional**
  - **People**
    - Users, administrators, security experts (and … attackers)
  - **Process**
    - Design, operation, control, monitoring, …
  - **Technology**
    - What to apply, how to use it, how to integrate it

- Remember: you're as **secure** as your **weakest link**

"Security is not a product -- it's a process" — Bruce Schneier

RISKS, THREATS AND ATTACKS



Source: OWASP

- **Vulnerability** = a weakness in a security mechanism
- **Threat** = Vulnerability + Attacker + Motivation
- **Attack** = when the attacker puts a plan into action
- **Risk** = threat x likelihood x impact

KEY SECURITY REQUIREMENTS

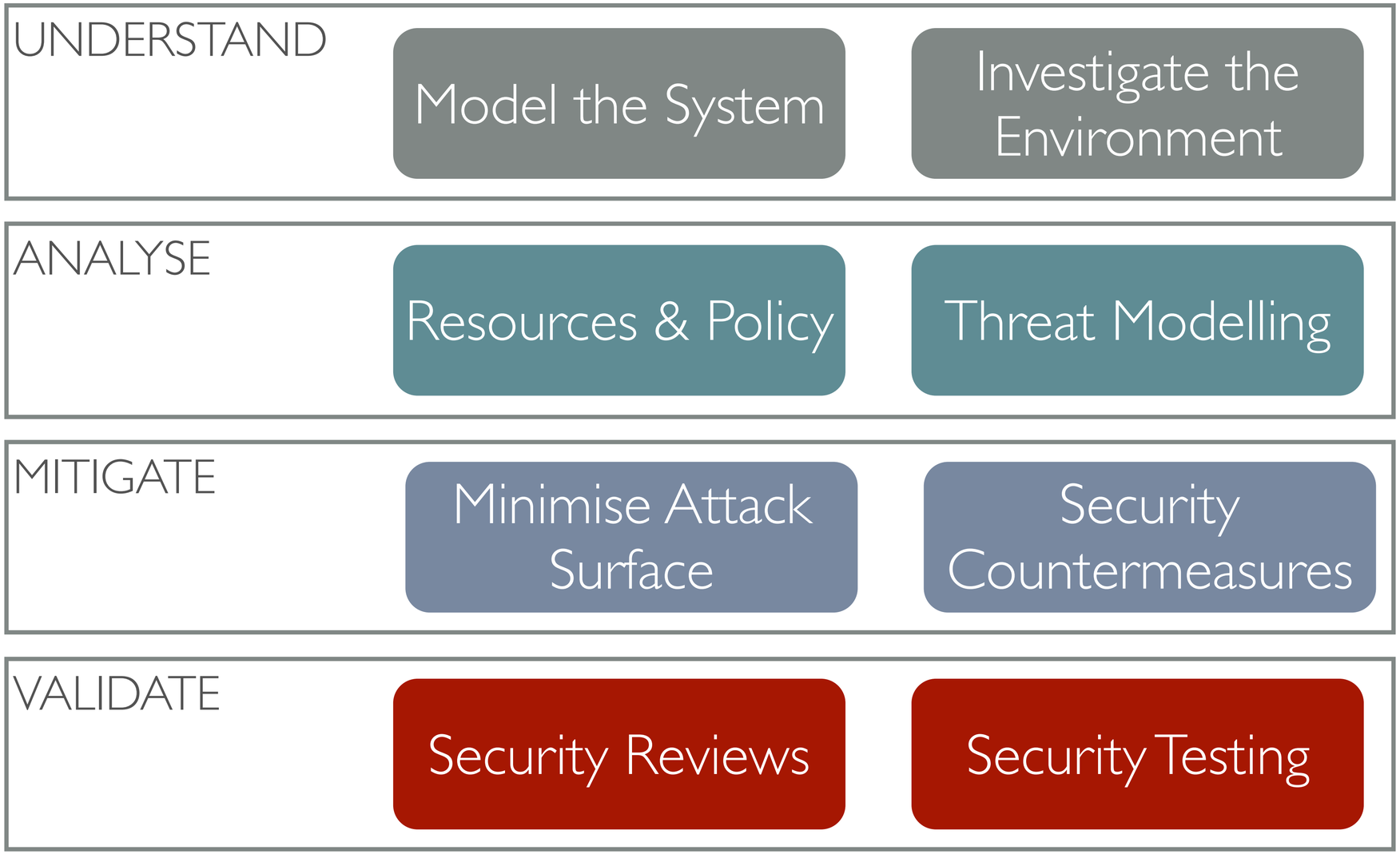
Confidentiality (or Privacy)	Prevent unauthorised access to information
Integrity	Prevent tampering or destruction
Availability	Prevent disruption to users of systems
Accountability (or "Non-Repudiation")	Know who does what, when

endava

SECURING SYSTEMS

endava

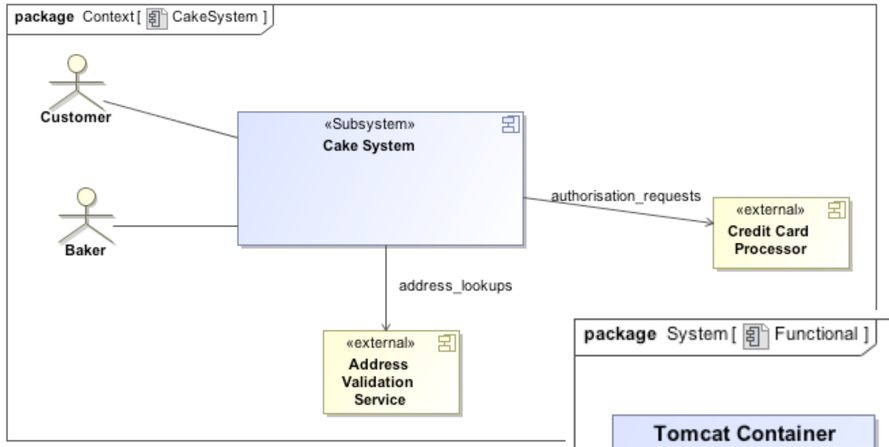
SECURING A SYSTEM



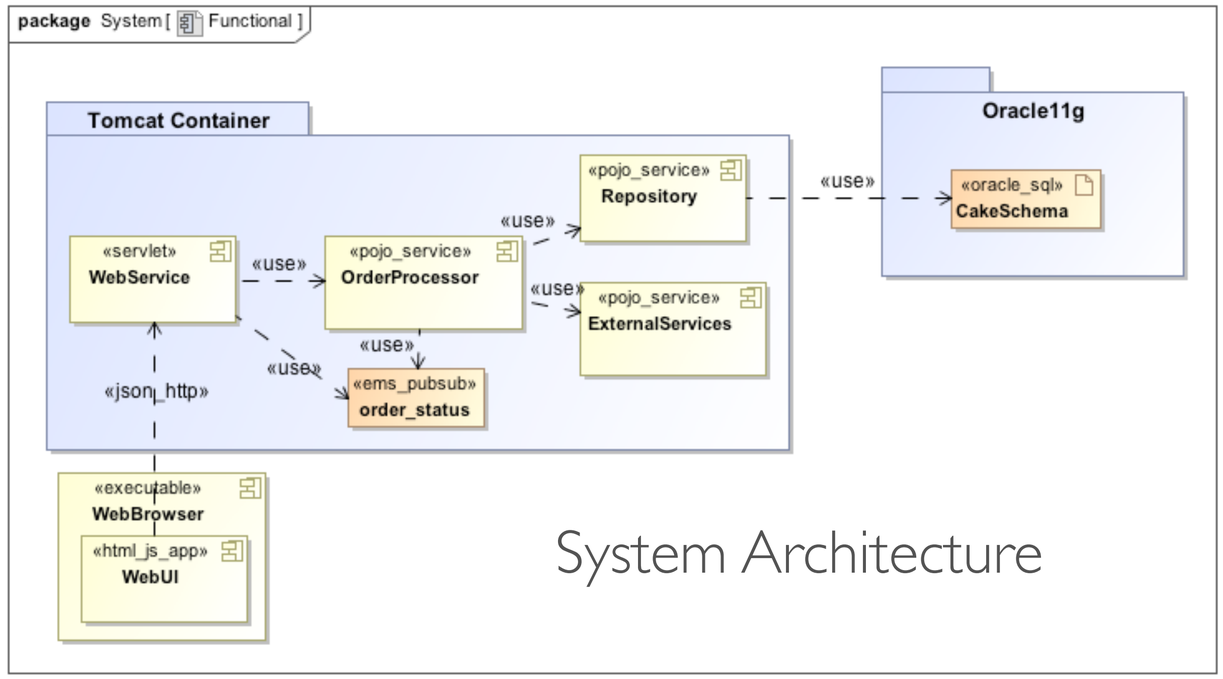
endava

MODELLING THE SYSTEM AND ENVIRONMENT

Understand



System Context



System Architecture

endava

Analyse

RESOURCES - IDENTIFY VALUE

- What is **valuable** is often self-evident
  - client information … damaging if lost
  - but what is of value for an **external attacker**? (e.g. configuration files?)
- **Operations** as well as **data**
  - viewing a payment might be fine … releasing one probably not!
- May require **fine-grained** consideration
  - HR data - work phone numbers vs home address

endava

Analyse

POLICY - DEFINE CONTROLS

- **Security policy** is a security **specification**
  - **controls** and **guarantees** needed in the system
  - **WHO** will use the system? (**principals**)
  - **WHAT** will they **work on**? (**resource types**)
  - and **WHAT** may they **do**? (**actions** on resources)

endava

Analyse

SECURITY POLICY

	Clients	Orders	Refunds <= £100	Refunds > £100
Onshore Service Agents	Create, View, Modify (Un)Suspend	All	Create, View, Authorise	View
Offshore Service Agents	View, (Un)Suspend	View, Cancel	View	View
Supervisors	All	All	All	Create, View, Cancel
Finance	View	View	View, Authorise	All

THREAT MODELLING

- **Threat** is a possible **breach** in security policy
  - System/process/people may (will) have **vulnerabilities**
  - **Attackers** have **motivation** and **goals**
  - **Threat** is an **attacker exploiting** a **vulnerability**
- **Identifying threats** is a key part of security design
  - threats are where you focus your security effort
  - threat modelling is the key activity

endava

Analyse

THREAT MODELLING

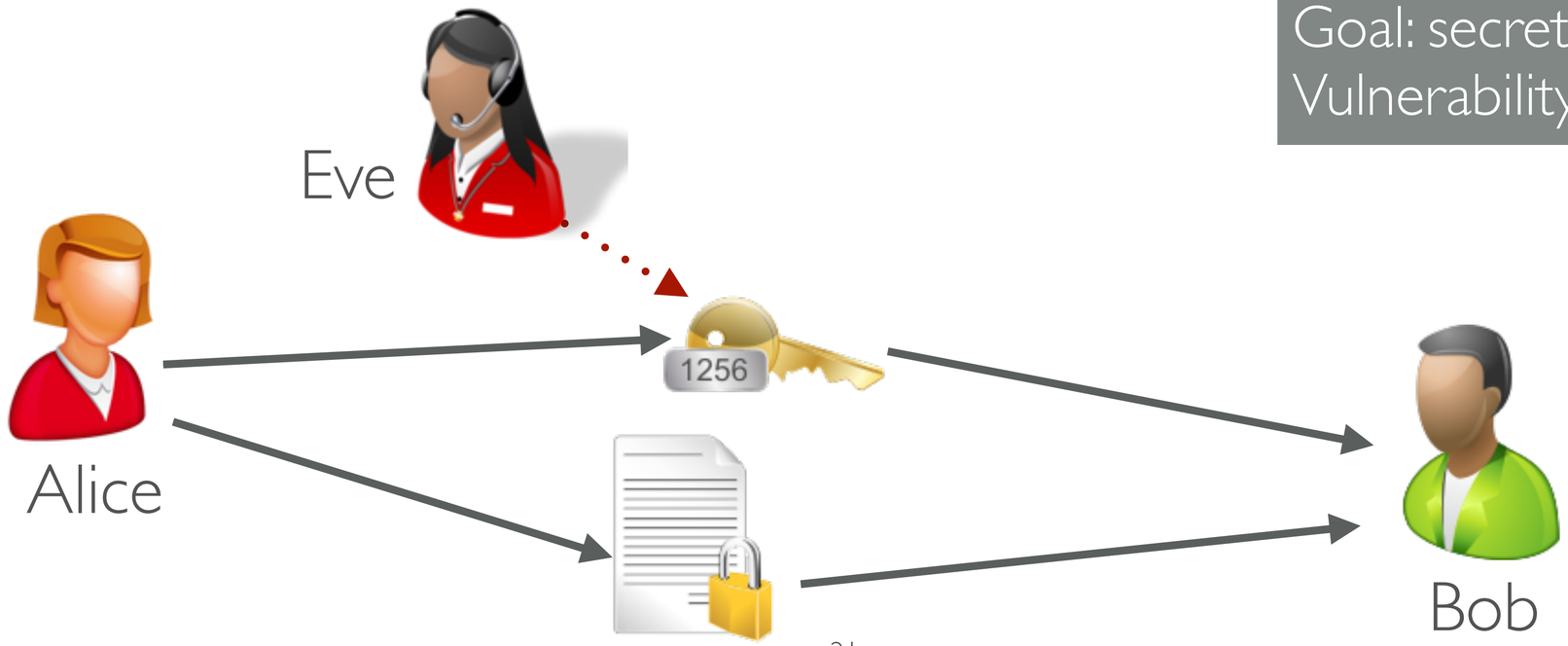
A procedure for optimising security by identifying ***objectives*** *and* ***vulnerabilities****, and then defining* ***countermeasures*** *to prevent, or mitigate the effects of, threats to the system* — OWASP

- Identify the real **risks** to **focus** security **effort**
- A technique all developers can be familiar with

THREAT MODELLING

- Who might attack your system?
- What is their goal?
- Which vulnerabilities might they exploit?

Attacker: opportunist insider
Goal: secret document
Vulnerability: unprotected key



endava

Analyse

FINDING THREATS - STRIDE

Spoofing	Pretending to be someone that you're not
Tampering	Changing information you shouldn't
Repudiation	Being able to deny performing an action
Information Disclosure	Getting access to information illicitly
Denial of Service	Preventing a service being offered
Elevation of Privilege	Gaining privileges you shouldn't have

endava

Analyse

CAPTURE THREAT MODEL

ID	25	26
Threat Type	Tampering	Spoofing
Component	WebUI	WebUI
Threat	Javascript tampering in browser, altering order data	WebUI user spoofing session ID for other user account
Mitigation	OPR-5543 - Add validation and unit tests for incoming order	OPR-5547 - Regenerate session ID and recheck on every request

endava

THINKING POINT

Can you identify a couple of threats in your environment?

who might attack? why?
what vulnerability might allow this?
what mitigations can you use?

endava

Analyse

EXPLORING ATTACKS: ATTACK TREES

Attacker: Professional hacker

Goal: Obtain customer credit card details

Attack: Extract details from the system database.

1. Access the database directly
    1. Crack/guess database passwords
    2. Crack/guess OS passwords to bypass db security
    3. Exploit a known vulnerability in the database software
2. Access the details via a DBA
    1. Bribe a database administrator (DBA)
    2. Social engineering to trick DBA into revealing details
3. …

COMPARE THREATS - DREAD MODEL

- **Risk** = **Damage** (1..10) +
  Reproducibility (1..10) +
  Exploitability (1..10) +
  Affected Users (1..10) +
  Discoverability (1..10)

- Sum values and divide by 5 for the DREAD rating
  - https://www.owasp.org/index.php/Threat_Risk_Modeling
  - Can be criticised for lack of consistency - but still a useful process

DREAD MODEL EXAMPLE

- **Suppose a threat where …**
  - **damage** limited to individual users => 5/10
  - is **reproducible** with a browser => 10/10
  - **needs** malware for the exploit => 5/10
  - **affects** many but not all users => 5/10
  - and can be **discovered** easily => 10/10

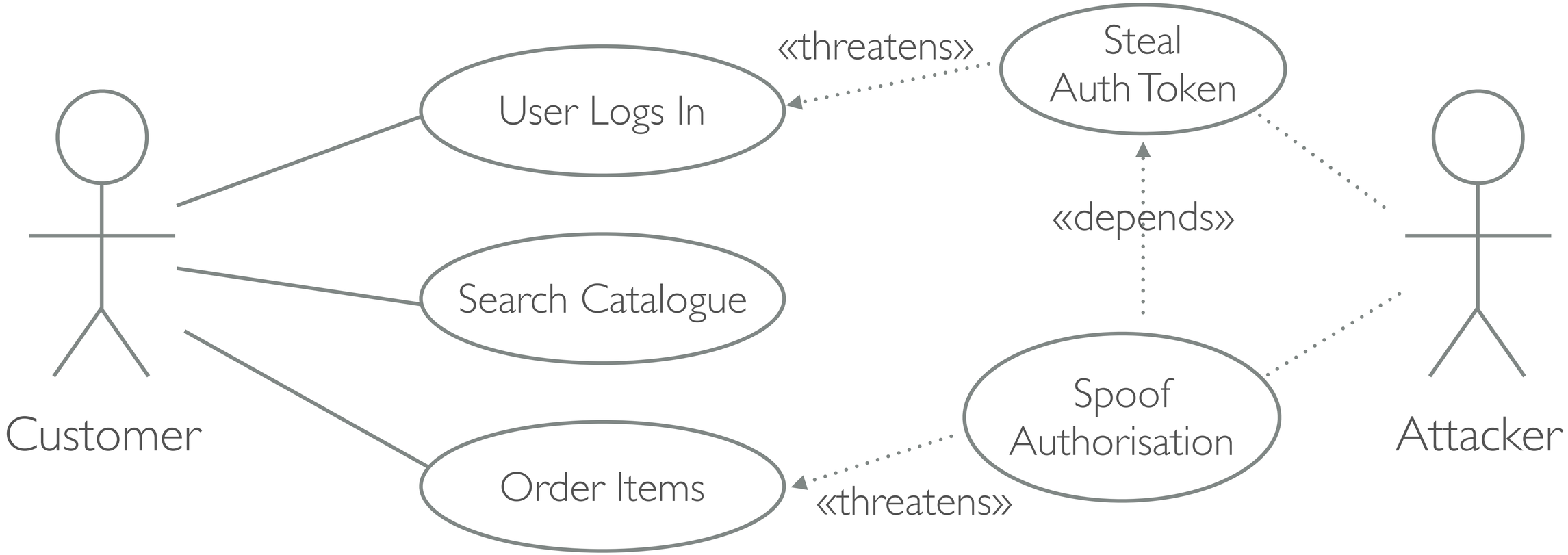
- **DREAD value** = (5+10+5+5+10)/5 = **7/10**

a useful process … but thinking is still required!

LIBRARIES FOR KNOWN PROBLEMS

- **OWASP** Top 10 list
  - https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
- **WASC** threat classification
  - http://projects.webappsec.org/f/WASC-TC-v2_0.pdf
- Mitre's CAPEC & **CWE**
  - Common Attack Pattern Enumeration & Classification
  - Common Weaknesses Enumeration

SECURITY ABUSE CASES



endava

Analyse

ABUSE CASE EXAMPLE

Abuse Case:	Spoofing Authorisation via Valid Authentication
Threat:	The misuser steals an authorisation token and attempts to use it via a valid (other) authenticated identity
Preconditions:	1) The misuser has a valid means of user authentication (e.g. username/password). 2) The misuser has a stolen user authorisation token.
Actions:	1. The system shall request the user's identity and authentication. 2. The misuser authenticates himself correctly. 3. The system shall identify and authenticate the user. 4. The misuser attempts to authorise using the stolen token. 5. The system rejects the authorisation attempt, audits the event, terminates the session and locks the user account.
Postconditions:	1. The system shall have identified and authenticated the misuser 2. The system shall have prevented the misuser from stealing another user's means of authorisation.

endava

Mitigate

MINIMISE THE ATTACK SURFACE

- **Attack surface:** the potentially **vulnerable** system interfaces
  - **smaller** attack surface = **less to attack and secure**
- OWASP definition:
  - all **channels** into and out of the system
  - the **code** securing those **channels**
  - **data of value** within the application (security & domain)
  - the **code** securing this **data**

endava

THINKING POINT

How would you reduce the attack surface for your system?

input and output "channels"
code securing channels
data items of value
code securing data items

endava

Mitigate

SECURITY COUNTERMEASURES

- Once **risks** prioritised then **implement mitigations**
- Some are **well known** and relatively straightforward
  - e.g. use of role based access control
- Some are more **complex** but **well known**
  - e.g. XSS or SQL injection require input validation
- Some need **custom** solutions
  - e.g. attacks based on organisation structure

*Remember **people**, **process** and **technology**!*

INCIDENT RESPONSE

- Despite security a system may **breached**
- Need a **plan** for what you do when it happens
  - an **incident response plan**
  - an **incident response team**
- **Broader** than just technical
  - technical, management, legal & communications
- A **plan** allowing a clear, logical, risk driven response
  - analysis, mitigation, evidence, communication, lessons
- **Practice** your response

endava

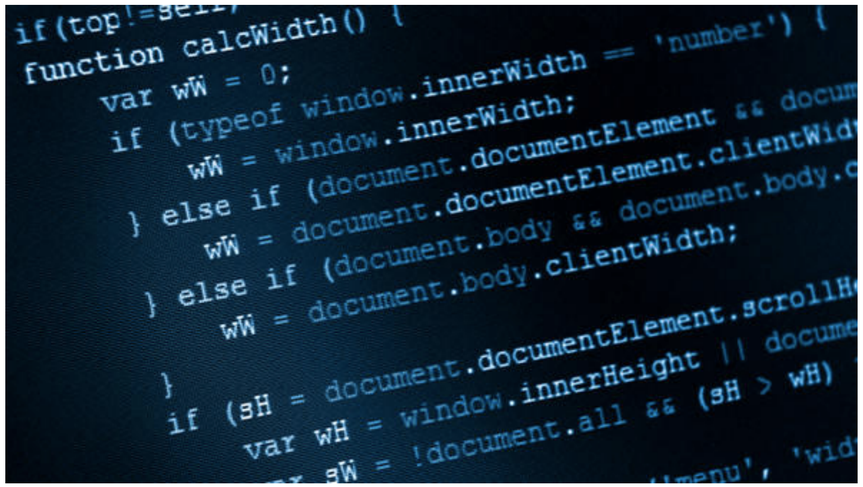
Mitigate

SECURE IMPLEMENTATION

- Secure design is useless if **implemented** insecurely
  - secure implementation outside the scope of this talk
- **Secure implementation** can be **complicated**
  - requires knowledge and care
  - relatively specialist task
- **Static analysis** and expert **code review**
  - FxCop, FindBugs, CodeAnalysis, Coverity, Fortify, …
  - OWASP code review guidelines, Oracle Java security guidelines

TOP SECURITY CODING ERRORS

- Not thoroughly validating input
- Injection attack vulnerabilities
- Insecure randomness
- Using custom cryptography
- Insecure logging
- Careless exception handling
- Lack of security testing



endava

Analyse

TESTING AND VERIFICATION

- As a software quality security needs to be tested
  - security testing largely outside the scope of this talk
- Wide range of security validation activities:
  - static analysis of code
  - functional testing of security features
  - penetration / known vulnerability / fuzz testing
  - manual system security review
  - threat mitigation tests
- Risk driven approach needed to maximise RoI
  - Consider third party assistance

endava

SUMMARY

endava

SUMMARY

- We've looked how to **improve** system **security**
  - we need to be **risk** and **principle** driven
- Security requires: **People**, **Process** and **Technology**
  - the weakest of the three is your security level
- Security needs to be **designed** in
  - its very difficult and expensive to add later

endava

SUMMARY

- Be guided by **risks not** security **technologies**
  - threat risk models (STRIDE and DREAD); attack trees
- Get the **experts** involved for **significant risks**
  - and never invent your own security technology!

endava

SUMMARY (II)

Never stop asking "**why?**" and "**what if?**"

critically important security questions!

Thank you for your attention

Questions?

Eoin Woods
Endava
eoin.woods@endava.com
@eoinwoodz