WEB APPLICATION SECURITY

Web Development and Design (COMP 1710 / 6780) – Semester 1 2023

Suthagar Seevaratnam (Chief Information Security Officer)

Australian National University

BY THE END OF THIS LECTURE, YOU SHOULD BE ABLE TO:

01	Understand the importance of a security mindset
02	Apply a security mindset to the software development life cycle
03	Understand different classes of threat actor
04	Understand the importance of threat modelling
05	Follow the basics of modelling threats using STRIDE
06	Understand the importance of the OWASP 10 framework to mitigate vulnerabilities

ADOPTING AN INFORMATION SECURITY MINDSET

A quick definition of information security

The protection of information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide confidentiality, integrity, and availability.

NIST 800-209

The CIA Triad

Confidentiality – preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information.

Integrity — guarding against improper information modification or destruction and ensuring information non-repudiation and authenticity.

Availability – ensuring timely and reliable access to and use of information

Think about this as a risk management process..

- Identify, assess, and prioritise risks to an organization's information assets and implementing measures to minimize or mitigate those risks.
- It needs to be systematic, comprehensive and must include an understanding of the business value of the asset.
- Must take into consideration of the threats faced and the potential impact to people, business operations and reputation if this risk is realised.

“Data breaches are not failures of technology, they are failures of organisational doctrine.”

Former National Cyber Director Christopher Inglis speaking at the ANU National Security College, 2022.

There’s no such thing as “cyber risk” there are cyber threats which create business risks.

Ubiquitous and Escalating External Threats

Complex / Competing Legislation and Compliance

Weak Internal Governance

Poor design practices

Threat modelling

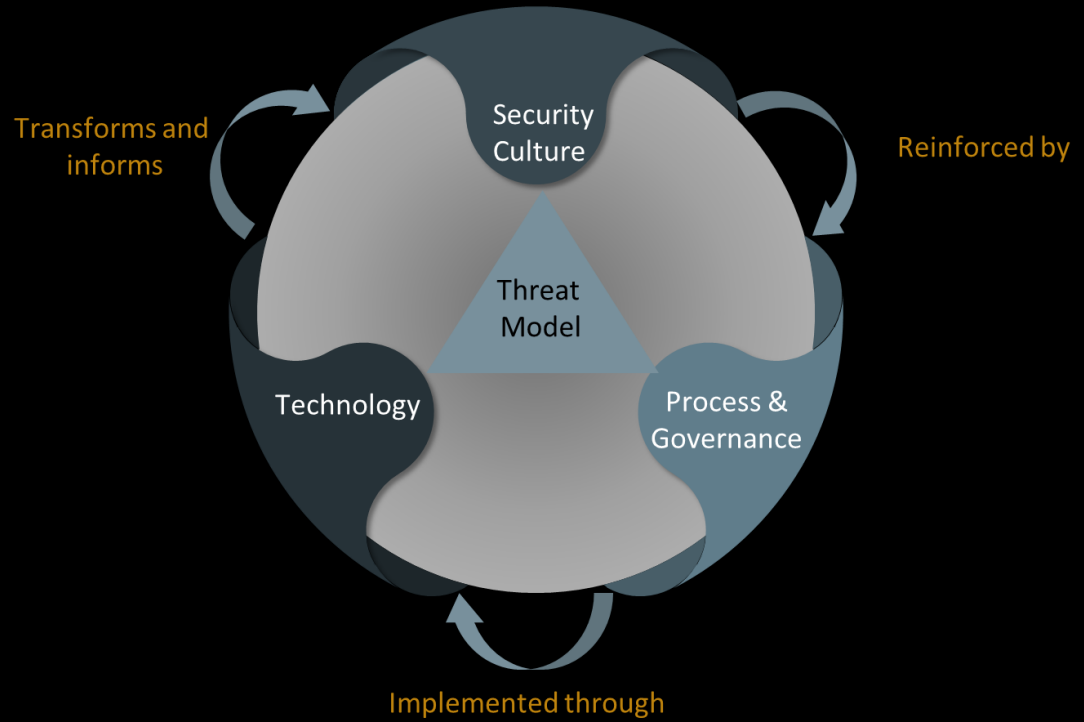
Alignment to regulation and standards

Strong governance

Secure by default

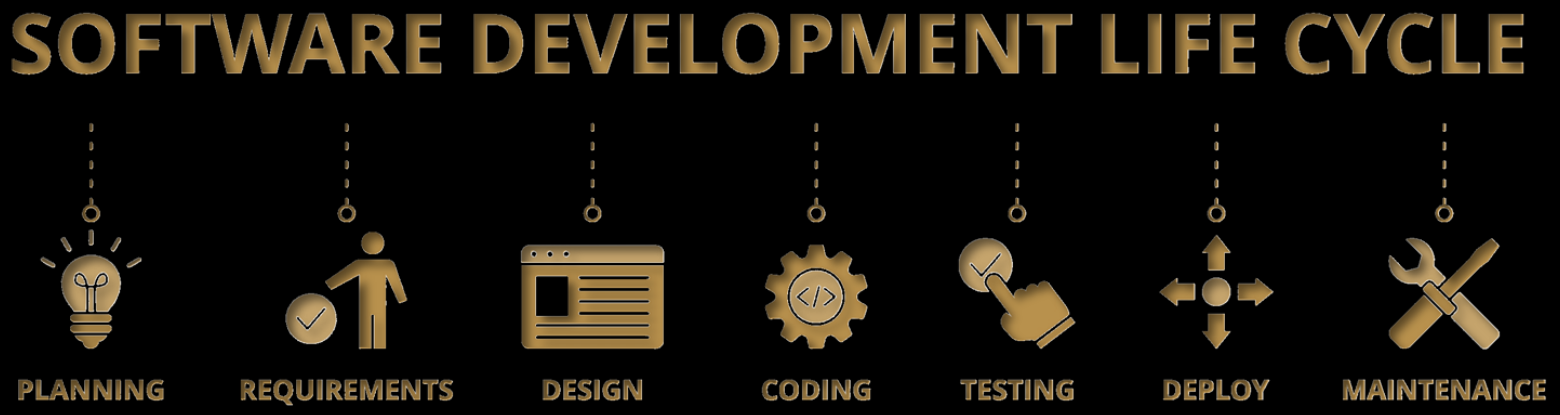
Start with culture end with technology

and be proportionate to threat.

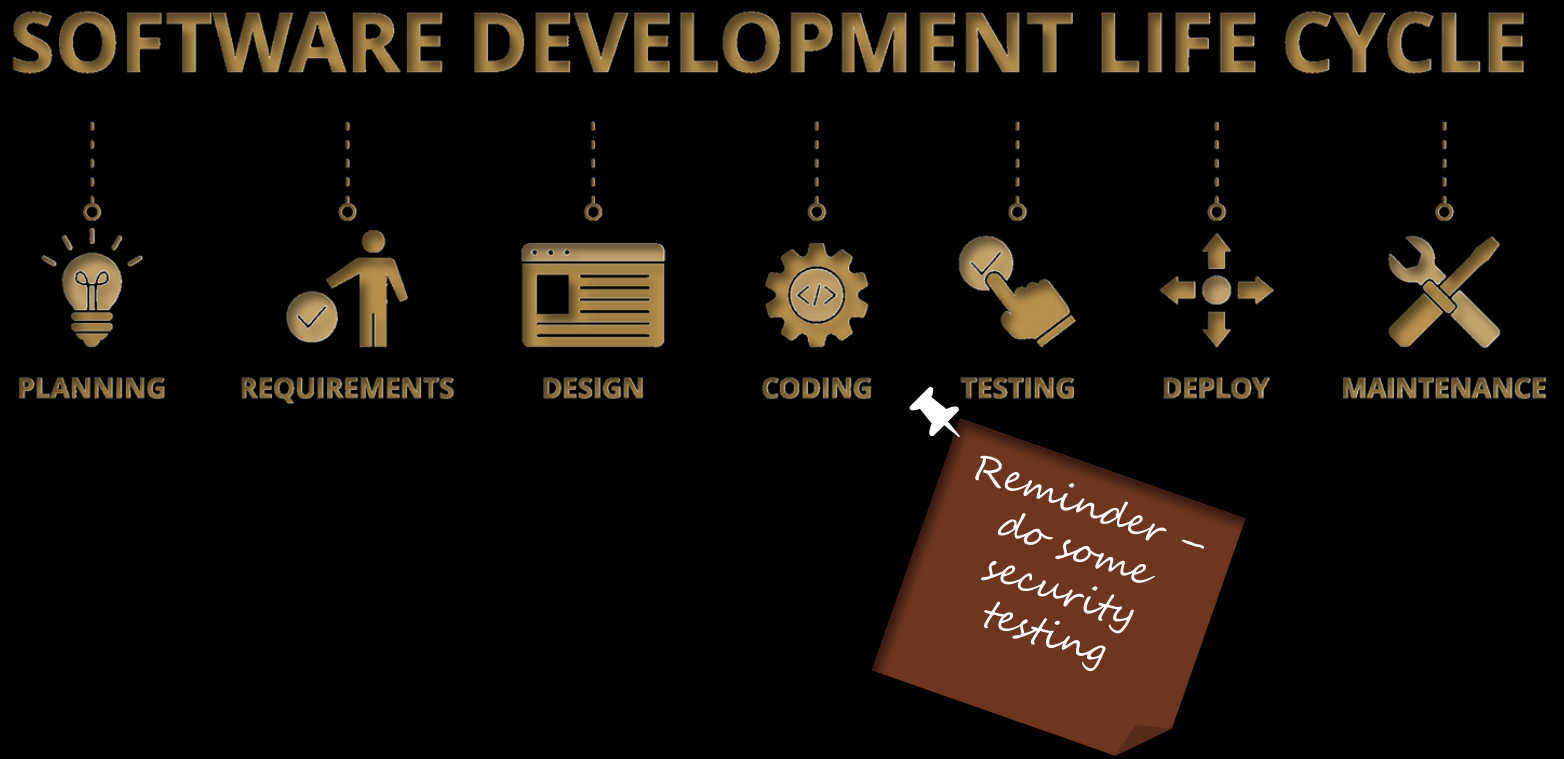


SECURE SOFTWARE DEVELOPMENT LIFECYCLE

In which SDLC phase does security go?

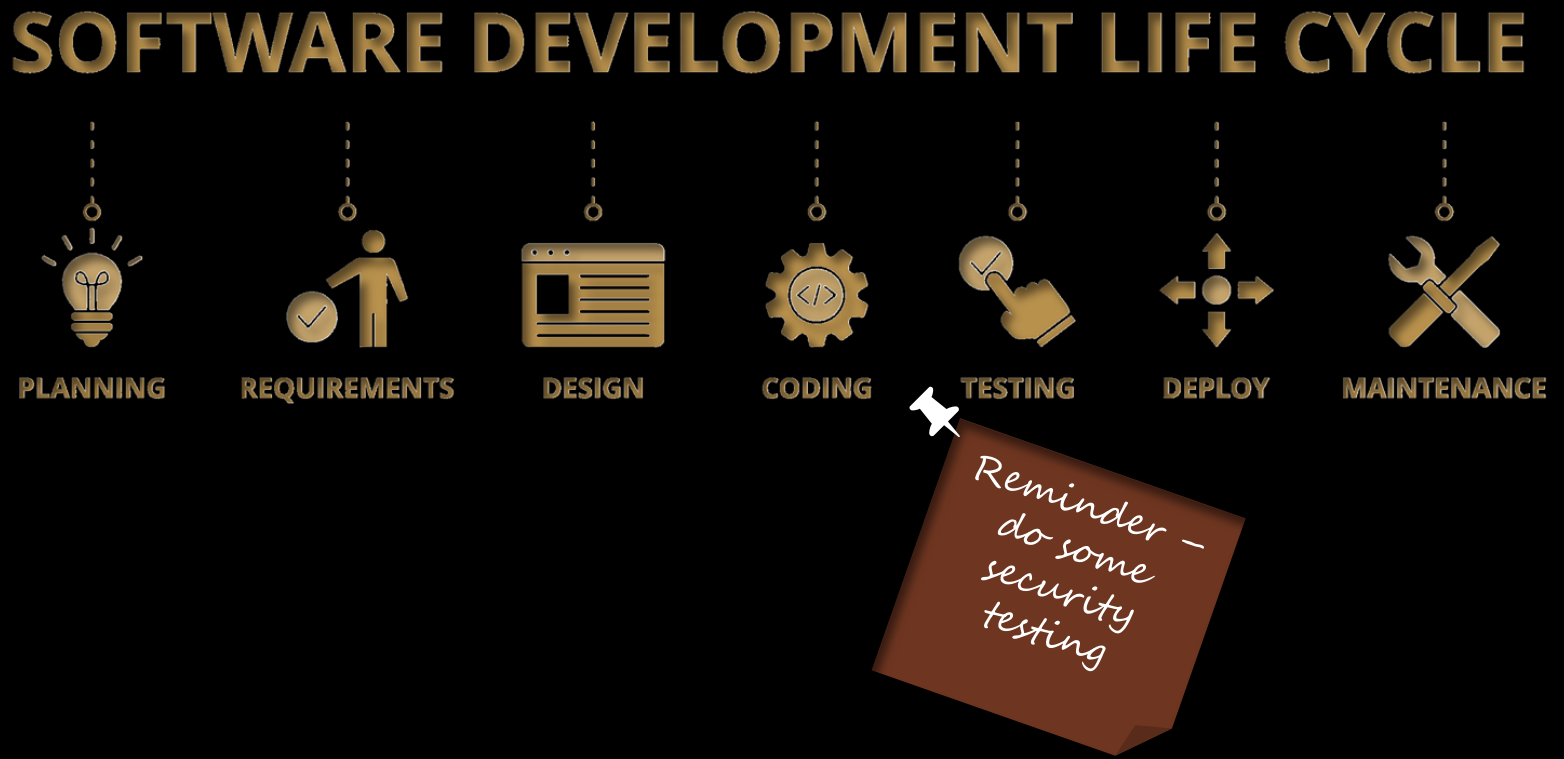


Traditionally…

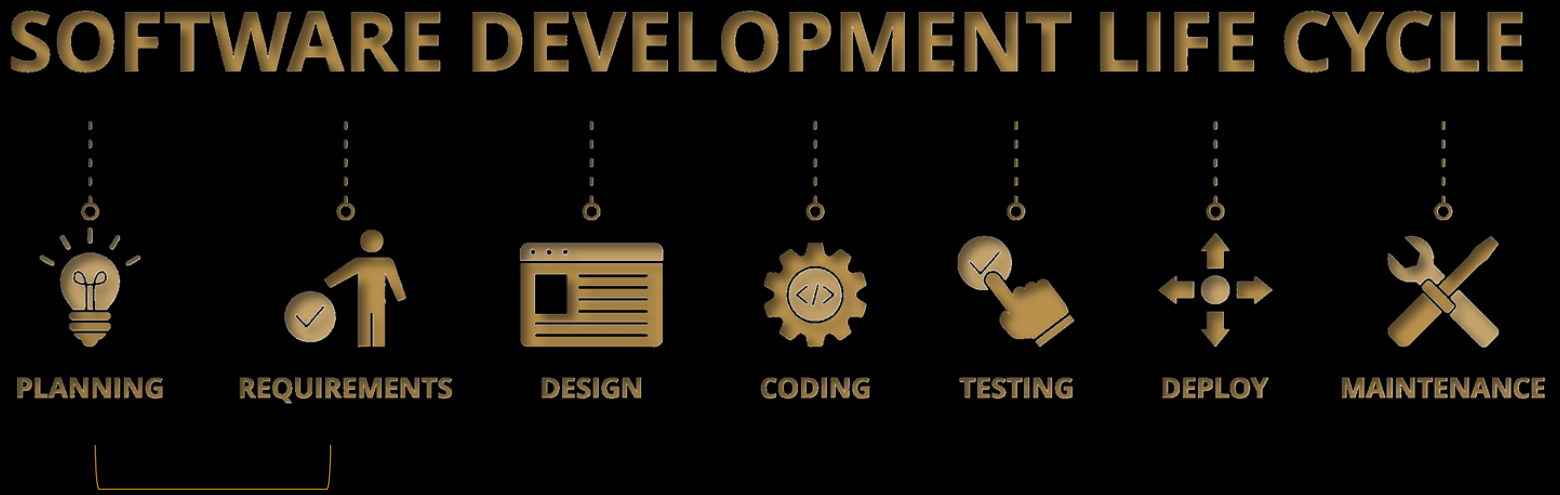


My desk on an average day...

Traditionally…

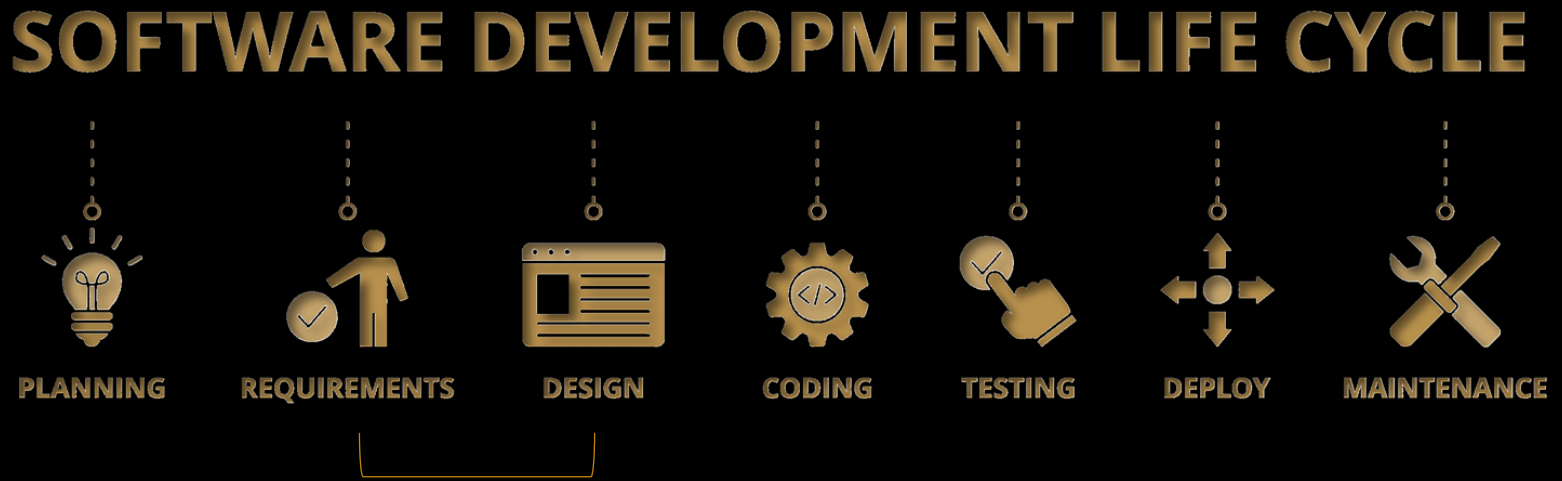


Adopt a security mindset from the beginning



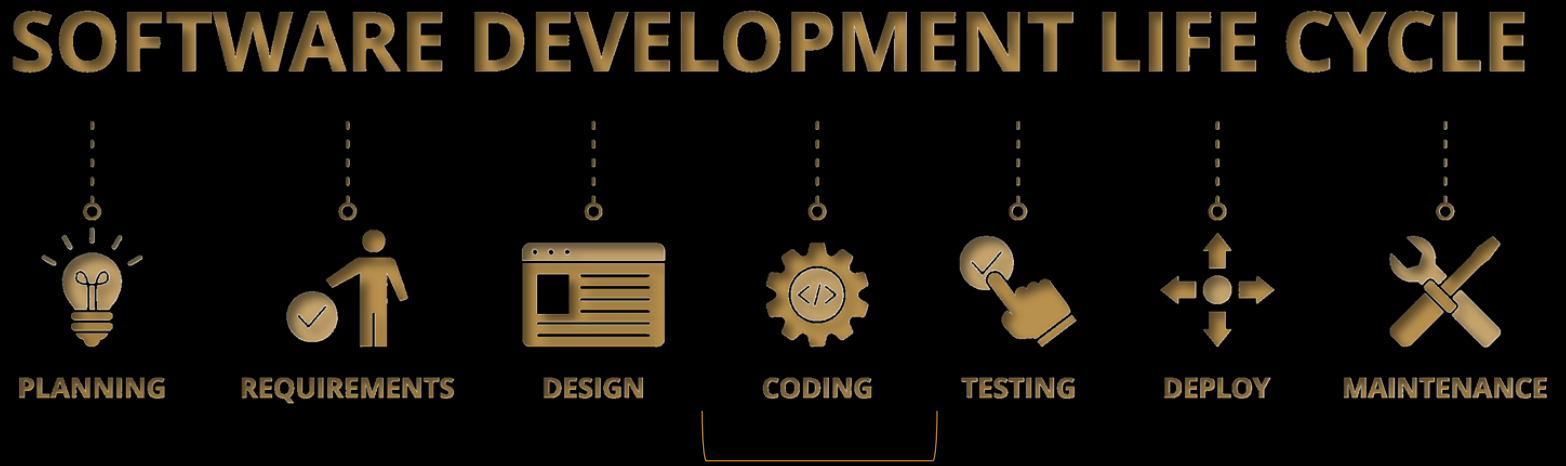
- Security is a requirement: secure-by-default is not a feature, it is a necessity
- Find the gaps, explore where it went wrong before
- Where could it go wrong – anticipate the unexpected
- Assume it will go wrong – use many voices

Model theats



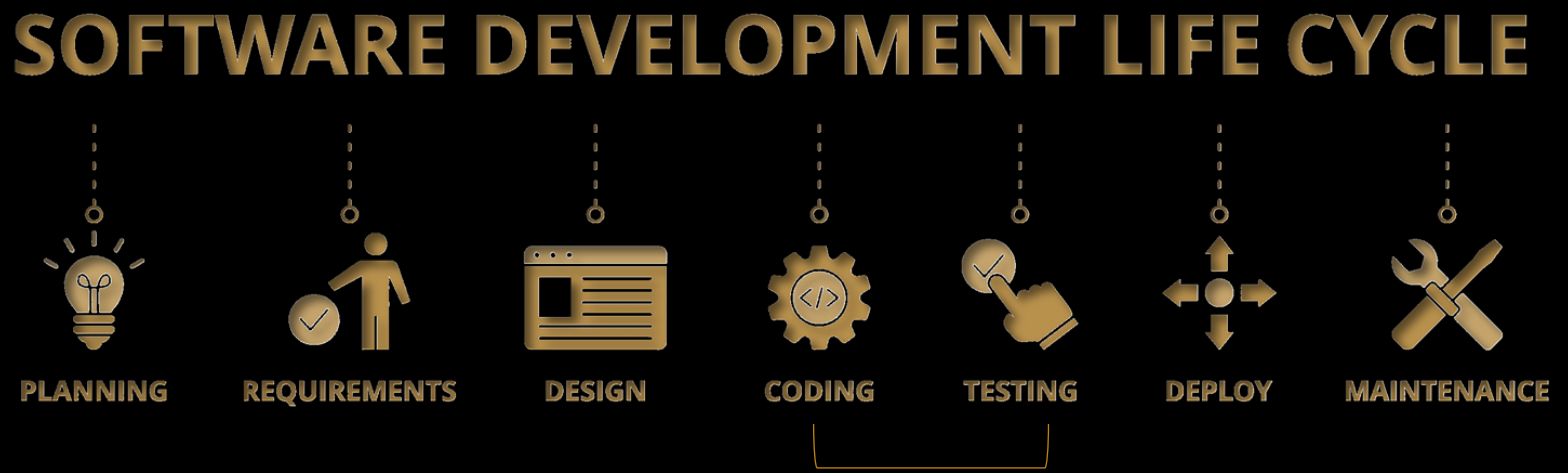
- Threat model, wargame the scenario.
- Prototype early and try and break it – if you don't hackers will
- What data protection regulation do we need to be compliant with?
- Did we design in all security requirements?

Keep code and the practice of coding secure



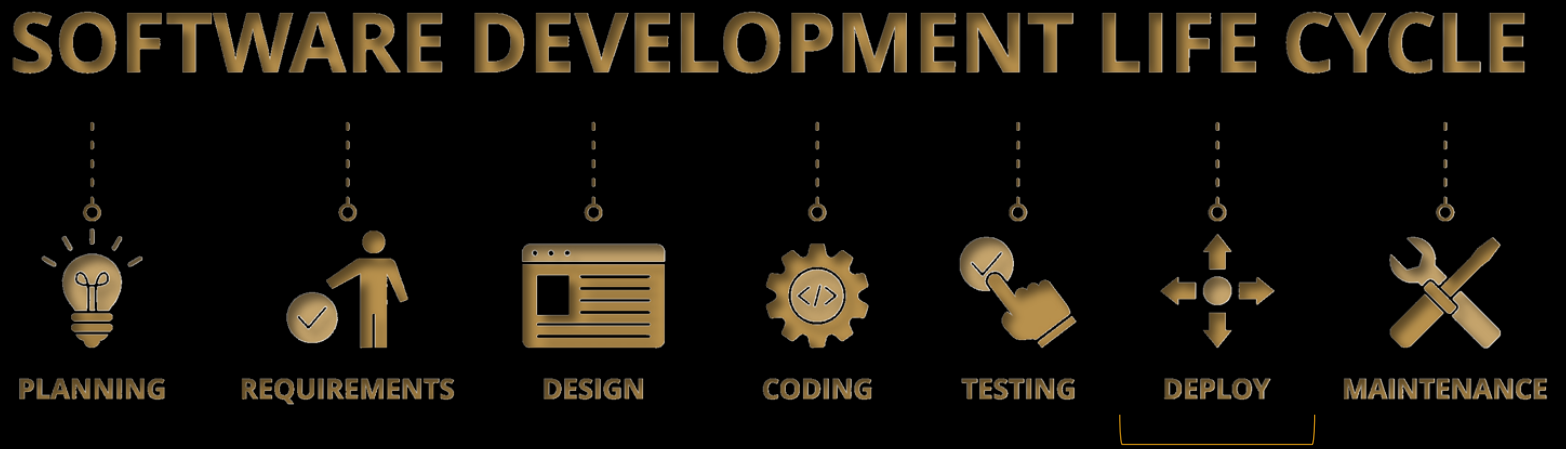
- What secure coding standard / language will we use?
- Are all API calls documented?
- Library checks, code reviews
- Team skills and knowledge
- Detect the vulnerability, mitigate and then confirm.

Add security to your testing



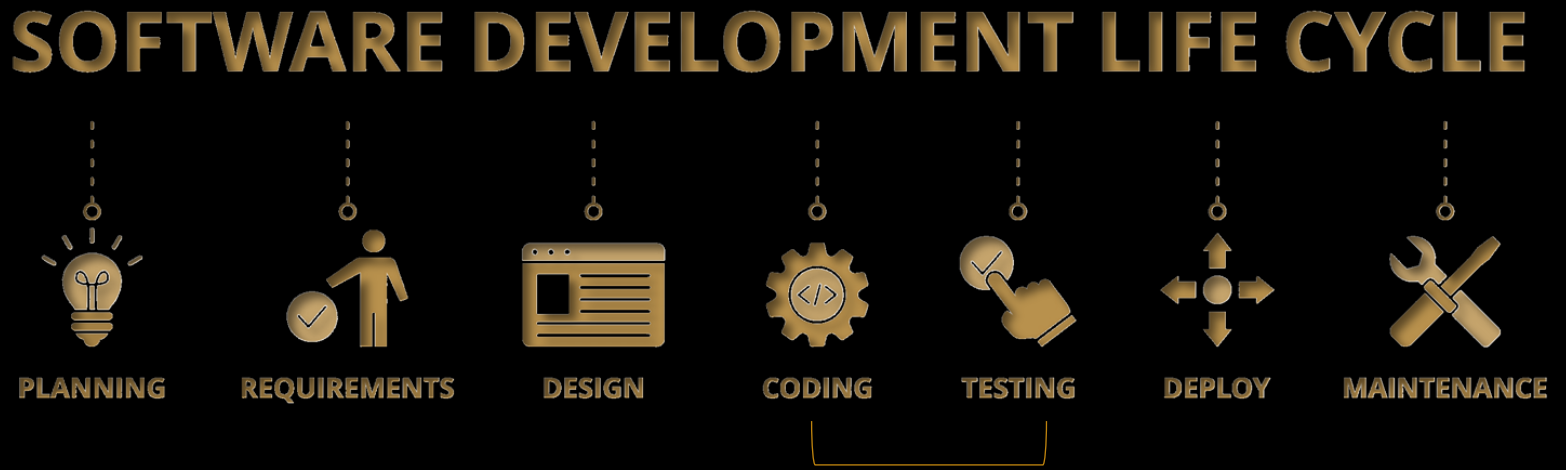
- In addition to regular user and regression testing use
  - Automated unit testing
  - Penetration testing
  - Dynamic and static application security testing
  - Black box and white box approaches

Deploy securely and involve users



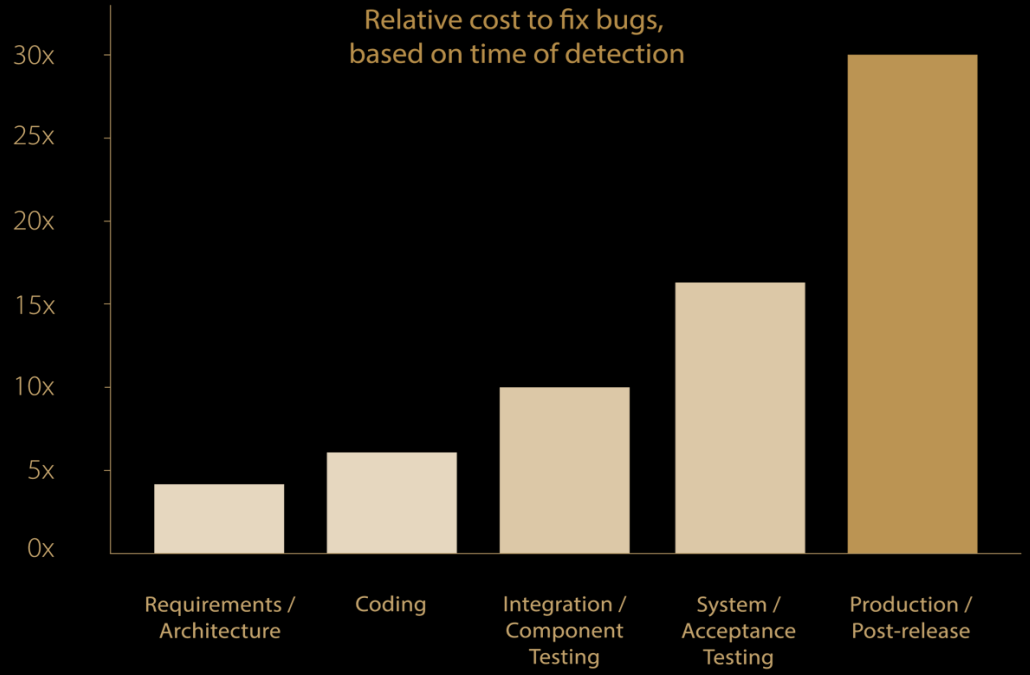
- You can use static and dynamic analysis during deployment
- Lots and lots of user testing – try to get them to break your app.
- Iterate alpha, beta, release candidates – as much as you need
- Signatures, verification – things that ensure your code / app is not tampered with

A developer's job is never done



- Threats evolve all the time, you need to keep monitoring.
- Keep your app and infrastructure fully patched.
- Consider a bug bounty.
- Ensure communication with your user base – they are still "real world testing"

A stitch in time…makes for better security



- There's a time and a place…
- How many of you have put up with bugs after you bought software?
- Secure by default starts at the beginning

National Institute for Science Technology (NIST) *The Economic Impacts of Inadequate Infrastructure for Software Testing*

UNDERSTANDING THREATS

The threat landscape

- High Complexity
  - Rapid evolution
  - Contested space
  - Tools have become “democratised”
- Threat Actor Types
  - Nation state
  - Criminal
  - Issues motivated
  - Opportunistic

State Actors

Motivation	Methods	Capability
▪ Surveillance & reconnaissance	▪ Tailored malware	▪ Extensive resources
▪ Espionage & traversal	▪ Human intelligence enabled	▪ Tailored research and approaches
▪ Detecting criminality	▪ Physical access techniques	▪ Persistent and covert.
▪ Influence / effect	▪ Covert infrastructure	▪ High level of sophistication if required.
▪ Sometimes financial gain		

Criminal Gangs

Motivation	Methods	Capability
▪ Financial gain ▪ Information harvesting ▪ Traversal ▪ Blackmail ▪ Recognition	▪ Ransomware as a service. ▪ Increasingly bespoke malware ▪ Scams, social engineering ▪ Zero days & mainstream exploits.	▪ Can be well financed ▪ Highly organised and discipline ▪ Speed over persistence. ▪ Can be linked other crime types ▪ Often campaign based

Issues motivated *"hacktivists"*

Motivation	Methods	Capability
▪ Recognition ▪ Information harvesting ▪ Political / social statement ▪ Reputational damage / retaliation ▪ Specific issues based effect	▪ Usually "mainstream" exploits. ▪ Sometimes enabled through insider knowledge ▪ social engineering ▪ Defacements, denial of service.	▪ Semi-organised often distributed. ▪ Will purchase malware, exploits, accesses ▪ Global reach ▪ Have their own morality and code

Insider Threat

Motivation	Methods	Capability
▪ Disgruntlement ▪ Financial ▪ Personal statement ▪ Reputational damage / retaliation	▪ Privileged access ▪ Inside knowledge ▪ Deception ▪ May not be malicious – could just be negligent	▪ Very specific ▪ Will use available accesses granted by role. ▪ Limited reach ▪ Might not always be sophisticated*

Threat Landscape

- Threat vectors
  - Phishing
  - Supply chain interdiction
  - Credential harvesting
  - Vulnerability exploitation
  - Insider access

MODELLING THREAT

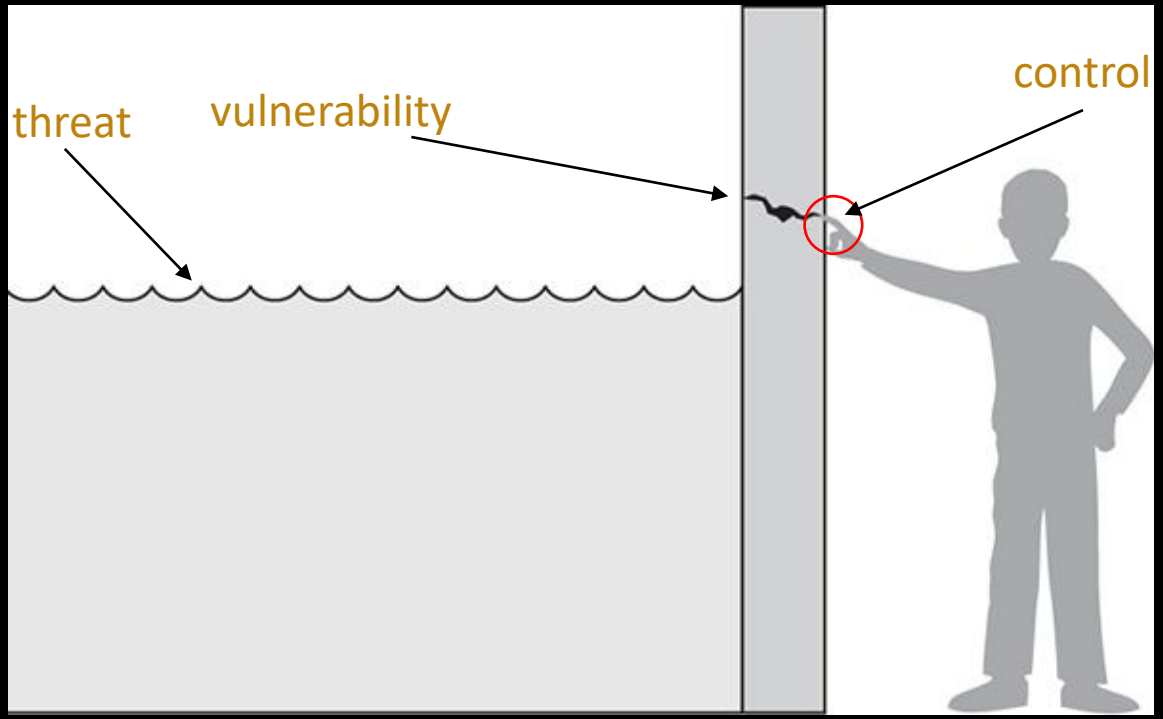
What is Information Security ? <revisited>

"[Information] security is the protection of the items you value, called the **assets** of a computer or computer system.

There are many types of assets, involving hardware, software, data, people, processes, or combinations of these."

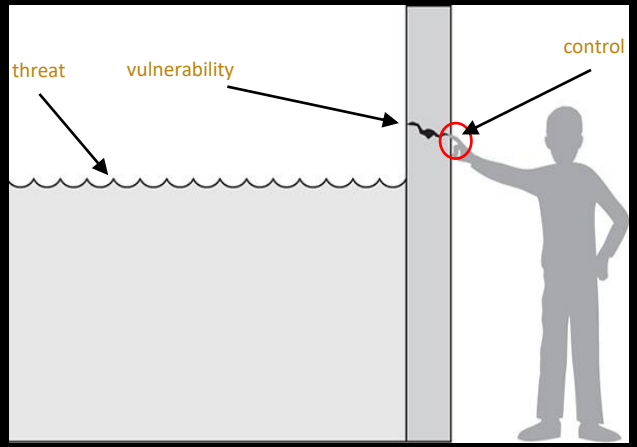
Source: Pfleeger et al., *Security in Computing 5th Ed.*

Threat-Vulnerability-Control Vocabulary



Source: Pfleeger et al., *Security in Computing 5th Ed.*

Threat-Vulnerability-Control Vocabulary



Source: Pfleeger et al., *Security in Computing 5th Ed.*

A **vulnerability** is a *weakness* that *could* be exploited and cause *harm*.

A **threat** is a set of *circumstances* that *exploit* a *vulnerability* that *cause harm*.

Controls *prevent* **threats** from *exercising* **vulnerabilities**.

Source: Pfleeger et al., *Security in Computing 5th Ed.*

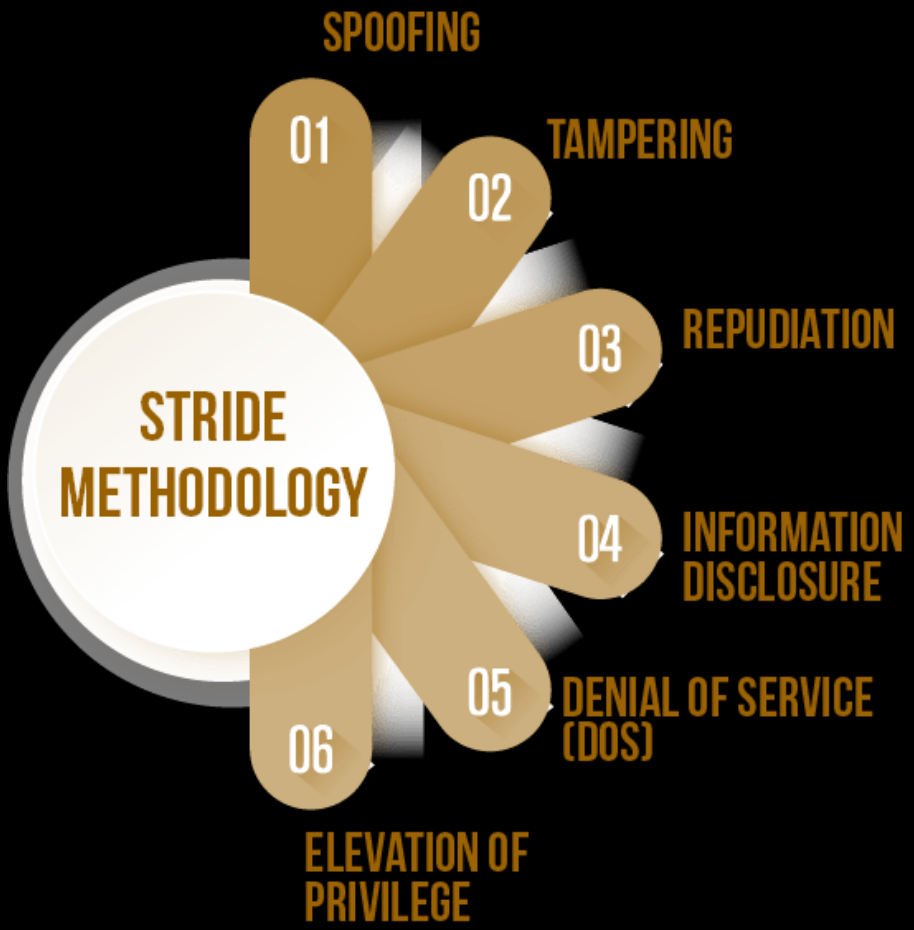
How I model threat

- There are many ways to do this.
- They generally deal with three key elements.
- Each element has its own structure.
- Can be applied at app level all the through to organisation / ecosystem level.

MODELLING WITH STRIDE

STRIDE

- A general purpose threat modelling framework
- Developed by Garg and Kohnfelder at Microsoft
- Asks the basic question of what might go wrong with my application.
-



Graphic source: EC-Council

Spoofing

- This category includes threats where an attacker tries to impersonate a legitimate user, device, or service. Examples include forging digital certificates, manipulating authentication mechanisms, or using social engineering techniques.

- Can my application (and users) be tricked i.e. Loss of authenticity

Tampering

- Tampering threats involve unauthorized modification or alteration of data, code, or configuration settings. Examples include modifying database records, tampering with network packets, or altering the functionality of a program.

- Can the data be altered in an unauthorised way? I.e. loss of integrity

Repudiation

- Repudiation threats involve an attacker denying responsibility for their actions or transactions. Examples include denying that a transaction took place, denying that data was modified or deleted, or denying that an action was taken.

- How can the system be fooled into a transaction that can be denied? I.e. loss of non-repudiability.

Information disclosure

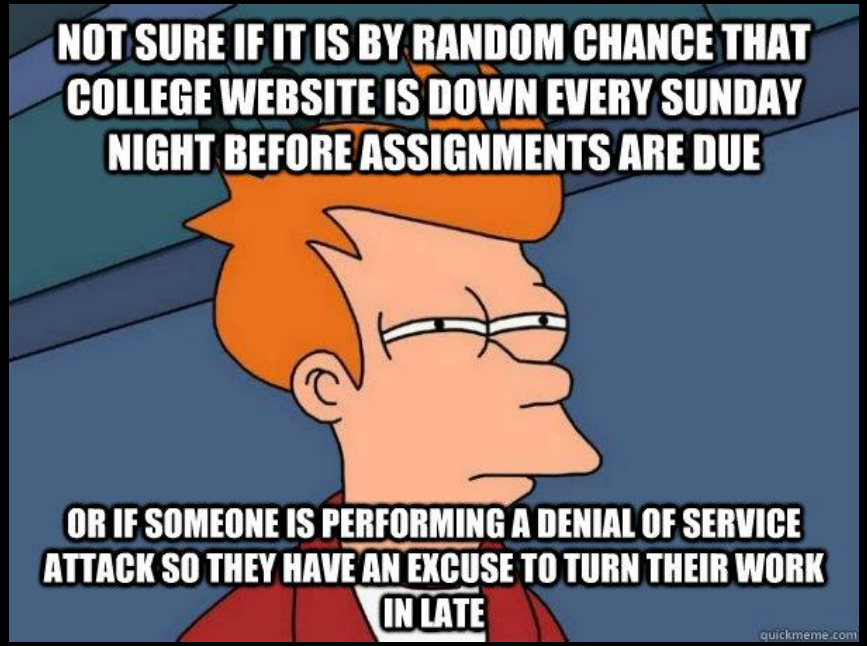
- This category includes threats where an attacker gains access to sensitive or confidential information. Examples include stealing passwords, reading sensitive data from memory, or eavesdropping on network traffic. Or the application inadvertently leaks information

- How can the system be fooled into a revealing information to an unauthorised party? I.e., loss of confidentiality

Denial of service

- Denial of service threats involve disrupting the availability or functionality of a system or application. Examples include flooding the network with traffic, crashing a server, or overloading a database.

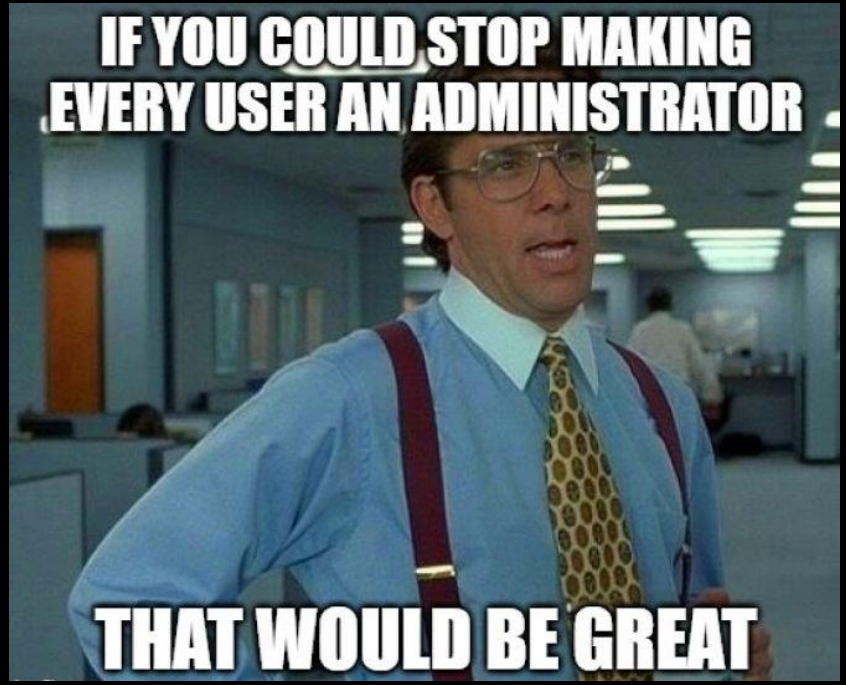
- How can the system be off-lined by a malicious actor? I.e., loss of availability



Elevation of privilege

- This category includes threats where an attacker gains higher privileges or access than they are authorised to have. Examples include exploiting a software vulnerability to gain administrator access, bypassing access controls, or using stolen credentials.

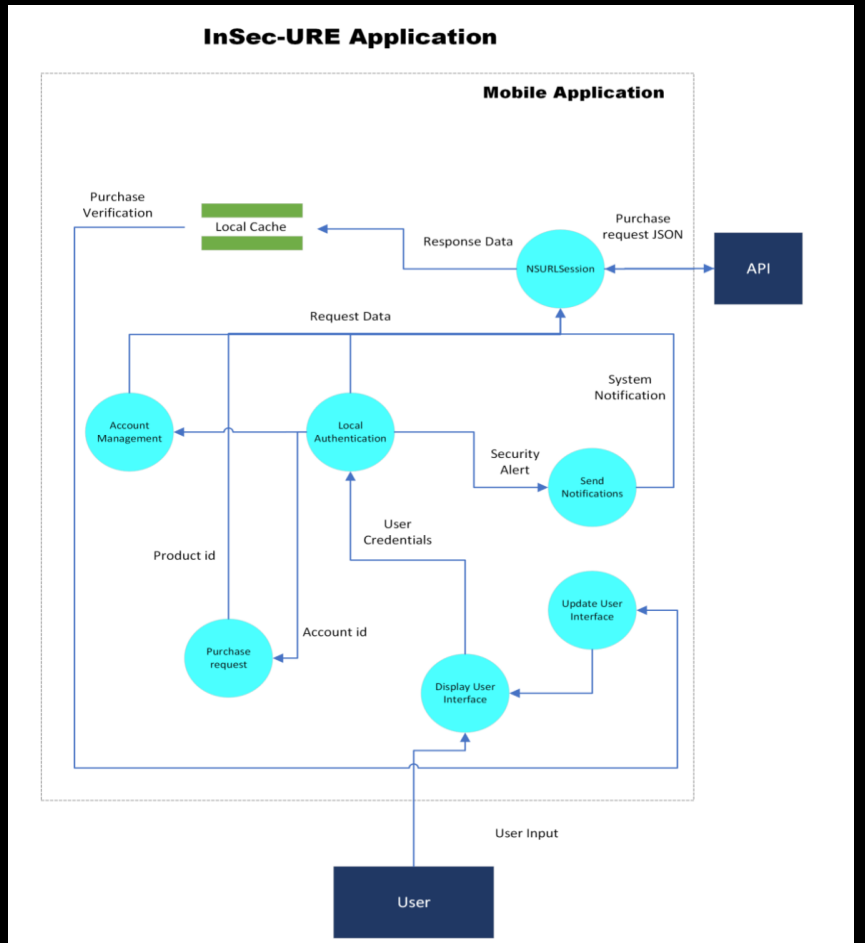
- How might a malicious actor gain admin rights? I.e., loss of authorization



We are going to do an example together

- Using STRIDE lets examine this new mobile app.

	Type of Threat	What Was Violated	How Was It Violated?
S	Spoofing	Authentication	Impersonating something or someone known and trusted.
T	Tampering	Integrity	Modifying data on disk, memory, network, etc.,
R	Repudiation	Non-repudiation	Claim to not be responsible for an action
I	Information Disclosure	Confidentiality	Providing information to someone who is not authorized
D	Denial of Service (DoS)	Availability	Denying or obstructing access to resources required to provide service
E	Elevation of Privilege	Authorization	Allowing access to someone without proper authorization



Source: Christopher Romero and Michael Burch (RSAC 2023)

MITIGATING WEB APP VULNERABILITIES

TOP 10

What is the OWASP Top 10?

- A guidance document for web developers.
- A systematic method of thinking about security issues in web applications.
- Provides guidance on the most commonly exploited vulnerabilities against web applications.
- A great starting point for your secure software development lifecycle.

Start here!

A01:2021-Broken Access Control

A02:2021-Cryptographic Failures

A03:2021-Injection

A04:2021-Insecure Design

A05:2021-Security Misconfiguration

A06:2021-Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures

A10:2021-Server Side Request Forgery

Source: OWASP Foundation

Broken Access Control (A01:2021)

Weak access controls or enforcement policy allows users / threat actors to act outside their intended permissions.

Failures can lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.

[Mitigation] Access control is only effective in trusted server-side code or server-less API, where the attacker cannot modify the access control check or metadata

Source: OWASP Foundation

Cryptographic Failures (A02:2021)

Data at rest and in transit need to be protected with strong encryption. Poorly implemented or missing encryption will result in data being transmitted in an insecure manner i.e using clear text or easily broken cryptographic algorithms. Using insecure protocols like HTTP, SMTP, FTP can also endanger information and potentially people.

[Mitigation] Use established cryptographic algorithms, proven libraries, strong keys and established implementation patterns. Ensure data in transit, rest and use are encrypted.

Source: OWASP Foundation

Injection (A03:2021)

Injection vulnerabilities arise when untrusted data is improperly handled by an application's interpreter. This allows an attacker to manipulate or inject malicious code into the system, which can lead to unintended and potentially harmful consequences.

[Mitigation] Lots of testing (static, dynamic, user). Strong input validation and proper handling of user-supplied data as data not commands. Plus secure coding practices like input/output encoding and output validation.

Source: OWASP Foundation

Insecure design (A04:2021)

These vulnerabilities arise from poor planning, architecture and requirements development. Think back to the first element of the software development lifecycle. This is about not adequately thinking about controls and business risks – not about poor implementation.

[Mitigation] secure-by-design, secure-by-default paradigms. Threat model, prototype, iterate and “shift left” as much as you can into the pre-code stage.

Source: OWASP Foundation

Security misconfiguration (A05:2021)

Applications or their components are misconfigured and deviate from best practice. This is a whole of application issue i.e., application, network, hosting, server side, client side, databases. Other flaws might be a lack of security hardening.

[Mitigation] repeatable hardened pattern or build configuration, least functionality paradigm, automated testing of configuration. Try and remove the human error component.

Source: OWASP Foundation

Vulnerable and Outdated Components (A06:2021)

A software component is unsupported or out of date, or vulnerable to a known exploit – think log4j. Libraries may have flaws or even the underlying OS.

[Mitigation] vulnerability management, continuous monitoring throughout the life of the application. Patching is essential as is a system of being notified of vulnerabilities in your supply chain. CI / CD tools often have this built-in. Keep a software bill of materials.

Source: OWASP Foundation

Identification and Authentication Failures (A07:2021)

Session management or other functions controlling identity or authentication are not implemented well or do not have any safeguards. This will allow attackers to assume the identity of a legitimate user. E.g. credential stuffing.

[Mitigation] Multifactor Authentication (MFA), rate limit authentication or account lockout. Adequate password length, random session IDs, consistent error messages for account enumeration attacks.

Source: OWASP Foundation

Software and Data Integrity Failures (A08:2021)

Code and infrastructure that fail to protect against integrity violations e.g. using untrusted plugins and libraries from unreliable sources. Or auto-update features without proper integrity verification allowing attackers to distribute and execute their own malicious updates on all installations.

[Mitigation] digital signatures, trusted repositories, check all software dependencies, ensure there is a review process for updates.

Source: OWASP Foundation

Security Logging and Monitoring Failures (A09:2021)

A common feature in many incidents as it allows attackers to gain a foothold in your application and achieve their objectives while remaining undetected. Think of this like the lack of security cameras at a bank.

[Mitigation] log, log, log, log…did I mention logging? Protect you logs. Monitor your logs continuously – particularly high-risk areas.

Source: OWASP Foundation

Server-Side Request Forgery (SSRF) (A10:2021)

Occurs when a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list.

[Mitigation] Input validation / URL sanitization, control list Ips / URLs, use safe libraries and APIs, segment network, interpose a proxy between server and external resource, restrict resource access.

Source: OWASP Foundation

A simple example – even big companies get it wrong.

Request :-
POST /<page_id>/userpermissions HTTP/1.1
Host : graph.facebook.com
Content-Length: 245
role=MANAGER&user=<target_user_id>&business=<associated_business_id>&
access_token=<application_access_token>

Response:-
true

Credit : Laxman Muthiyah

One very quick change…we now have access to the user's non-business page.

Request :-
POST /<page_id>/userpermissions HTTP/1.1
Host : graph.facebook.com
Content-Length: 245
role=MANAGER&user=<target_user_id>&access_token=<application_access_token>

Response:-
true

Credit : Laxman Muthiyah

And then we delete the original admin…

Request :-
Delete /<page_id>/userpermissions HTTP/1.1
Host : graph.facebook.com
Content-Length: 245
user=<target_user_id>&access_token=<application_access_token>

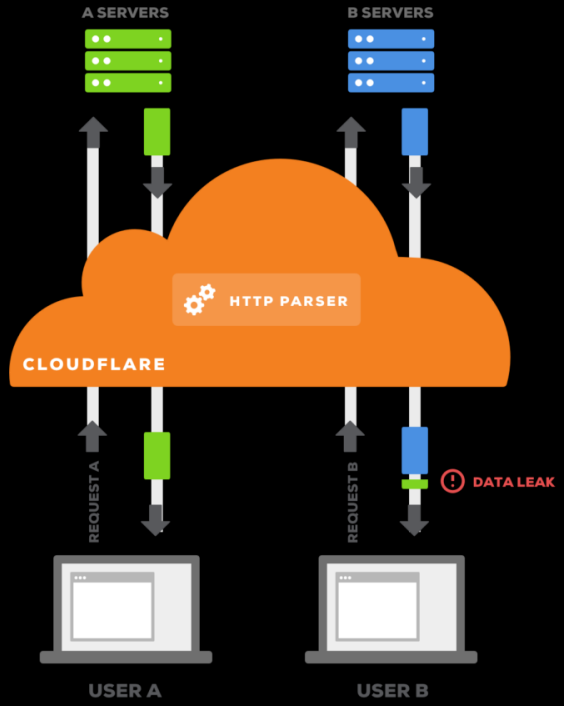
Response:-
true

Credit : Laxman Muthiyah

Cloudbleed, a cautionary tale about poor design

- Discovered in February 2017 that affected the content delivery network (CDN) provider, Cloudflare.

- Due to a software bug in Cloudflare's edge servers, sensitive information from one website could be leaked and mixed with responses from other websites using the same CDN.



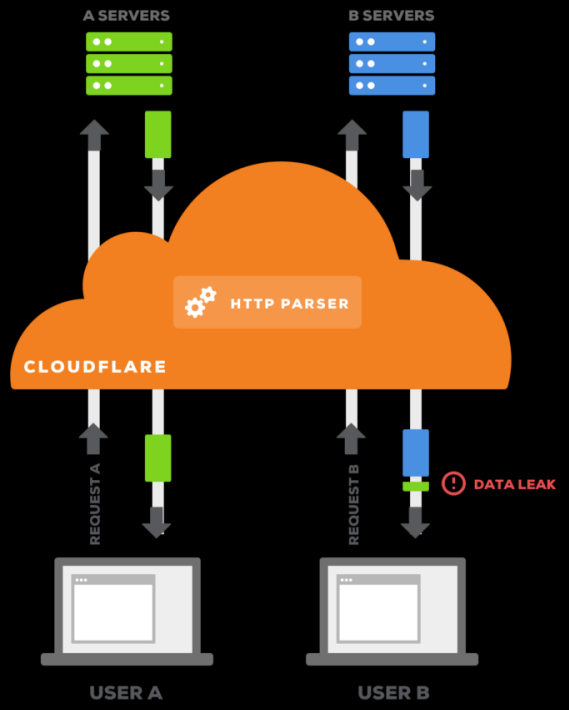
Graphic credit https://blog.sqreen.com

Cloudbleed cont…

- The data leak occurred when the Cloudflare edge servers returned random chunks of memory containing sensitive information in response to certain types of HTTP requests.

- The leaked information included HTTP cookies, login credentials, API keys, personal data, and other sensitive data exchanged between websites and their users.

- Bug existed in the HTML parser (ragel) for years, but was exposed when a new parser (cf-html) was introduced. When a feature used both parsers…buffer overrun and memory leakage.



Graphic credit https://blog.sqreen.com

END OF LECTURE

Australian
National
University