



Data Communications

DCF255

Lecture 9 | Application Security

Agenda

- Top 10 Vulnerabilities and Proactive Action
- Typical/Old Insecure SDLC Process: Resulting in Insecure Software
- Cost of Software Defects
- New Paradigm: Building Security In (New Secure SDLC)
- 10 Best Practices for Building Secure Software
- Application Security Testing Techniques

OWASP 2021

Top 10 Web Vulnerabilities

And How to Prevent Them

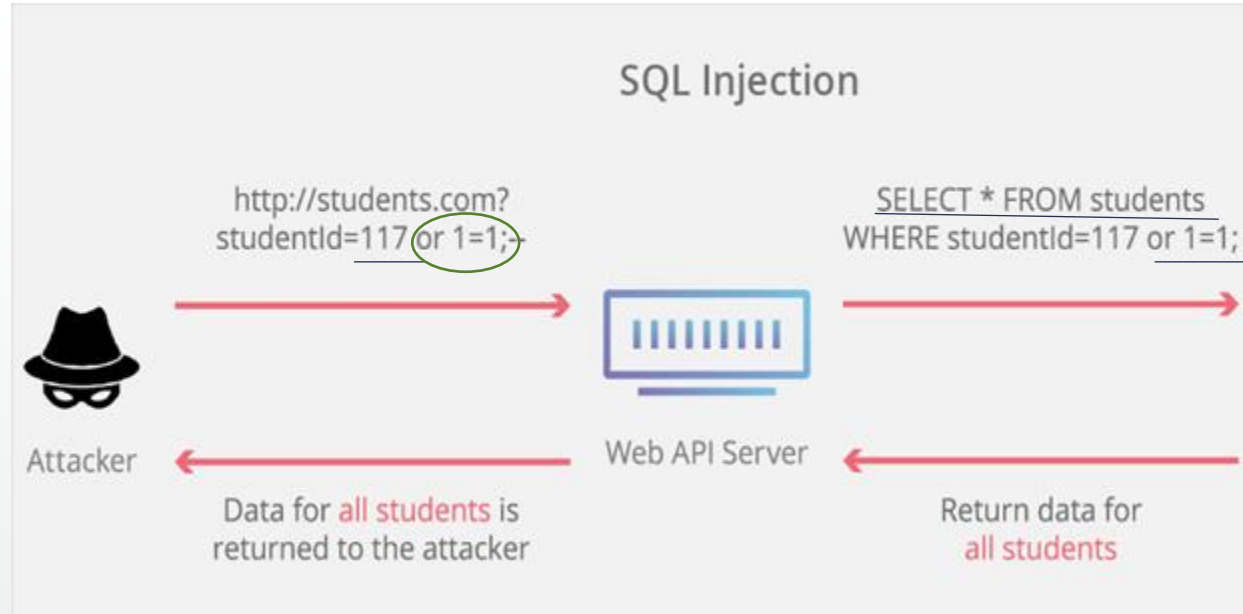
C5: Validate All Inputs

[illegible]

[illegible][illegible]

[illegible]

Vulnerabilities Examples



String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") + "'";

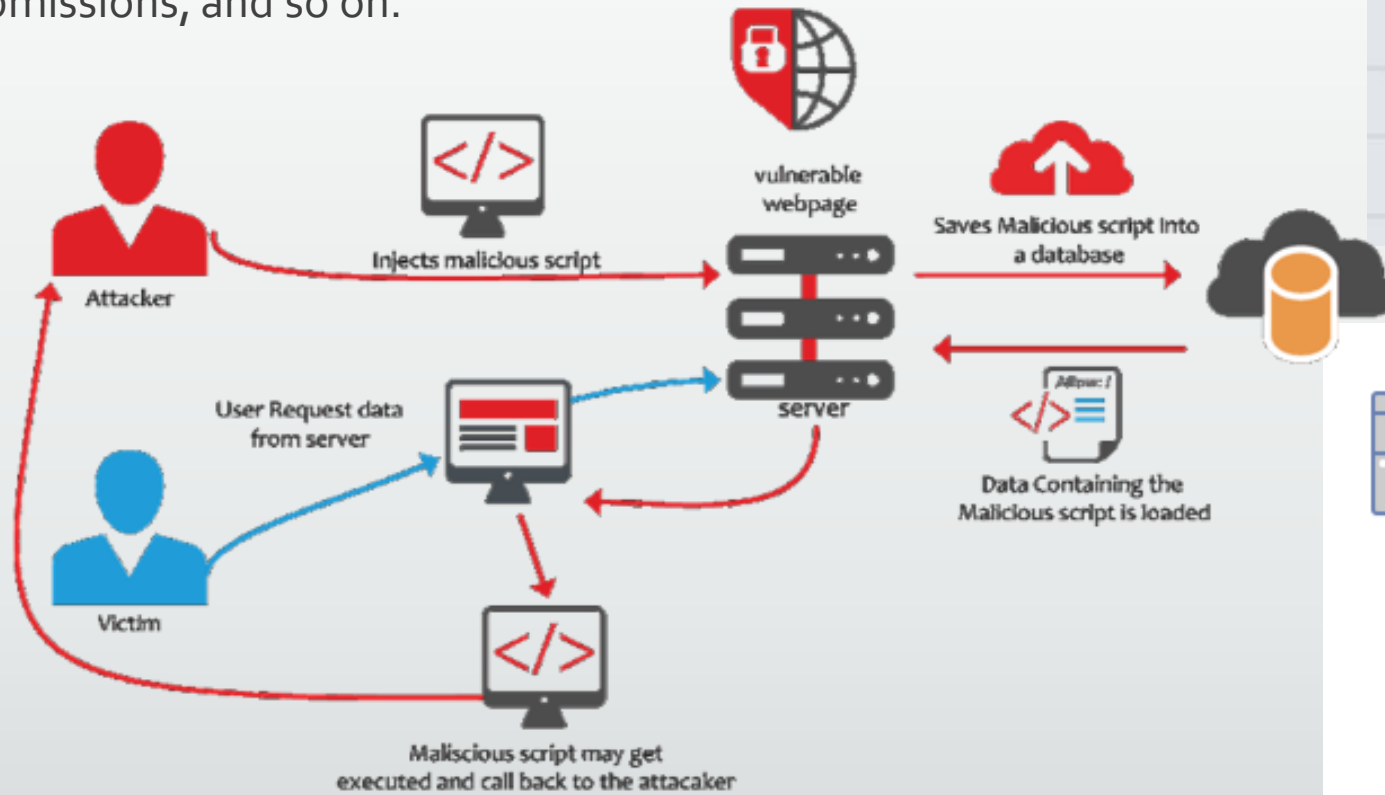
This query can be exploited by modifying the "id" parameter value as follow:

`http://example.com/app/accountView?id=' or '1'='1`

This makes a request to the application to return all records from the account table, other similar and more severe injections can modify the data, and even cause a loss of data.

Vulnerabilities Examples

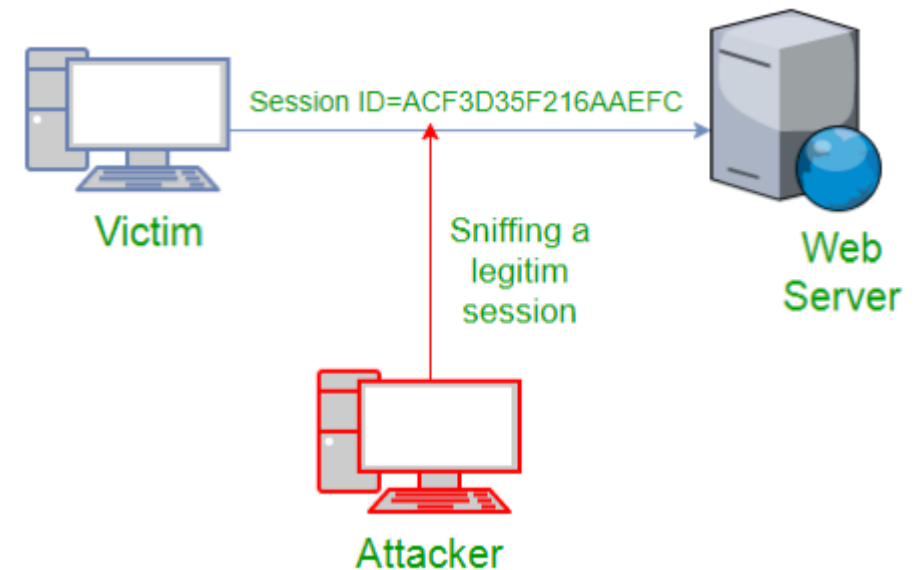
XSS attacks are essentially malicious injections (client-side) that are added to a web page or app through user comments, form submissions, and so on.



Sensitive Data exposure



Broken Authentication



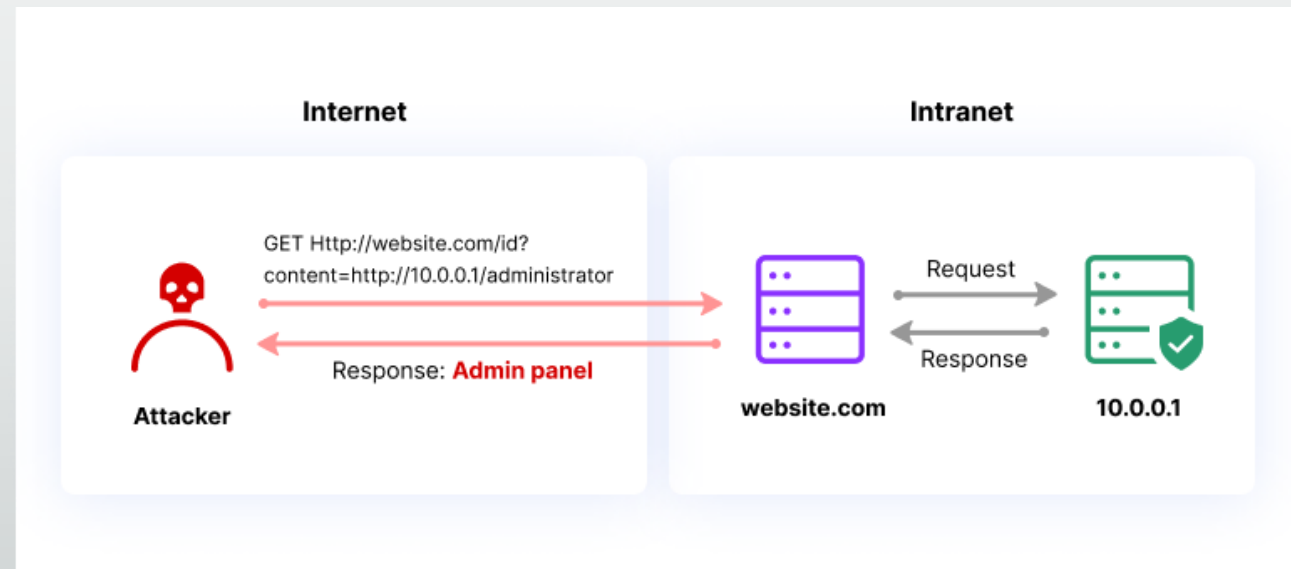
Vulnerabilities

Examples

Server-Side Request Forgery

- Server-side request forgery (also known as SSRF) is a web security vulnerability that allows an attacker to induce the server-side application to make requests to an unintended location.
- In a typical SSRF attack, the attacker might cause the server to make a connection to internal-only services within the organization's infrastructure. In other cases, they may be able to force the server to connect to arbitrary external systems, potentially leaking sensitive data such as authorization credentials.

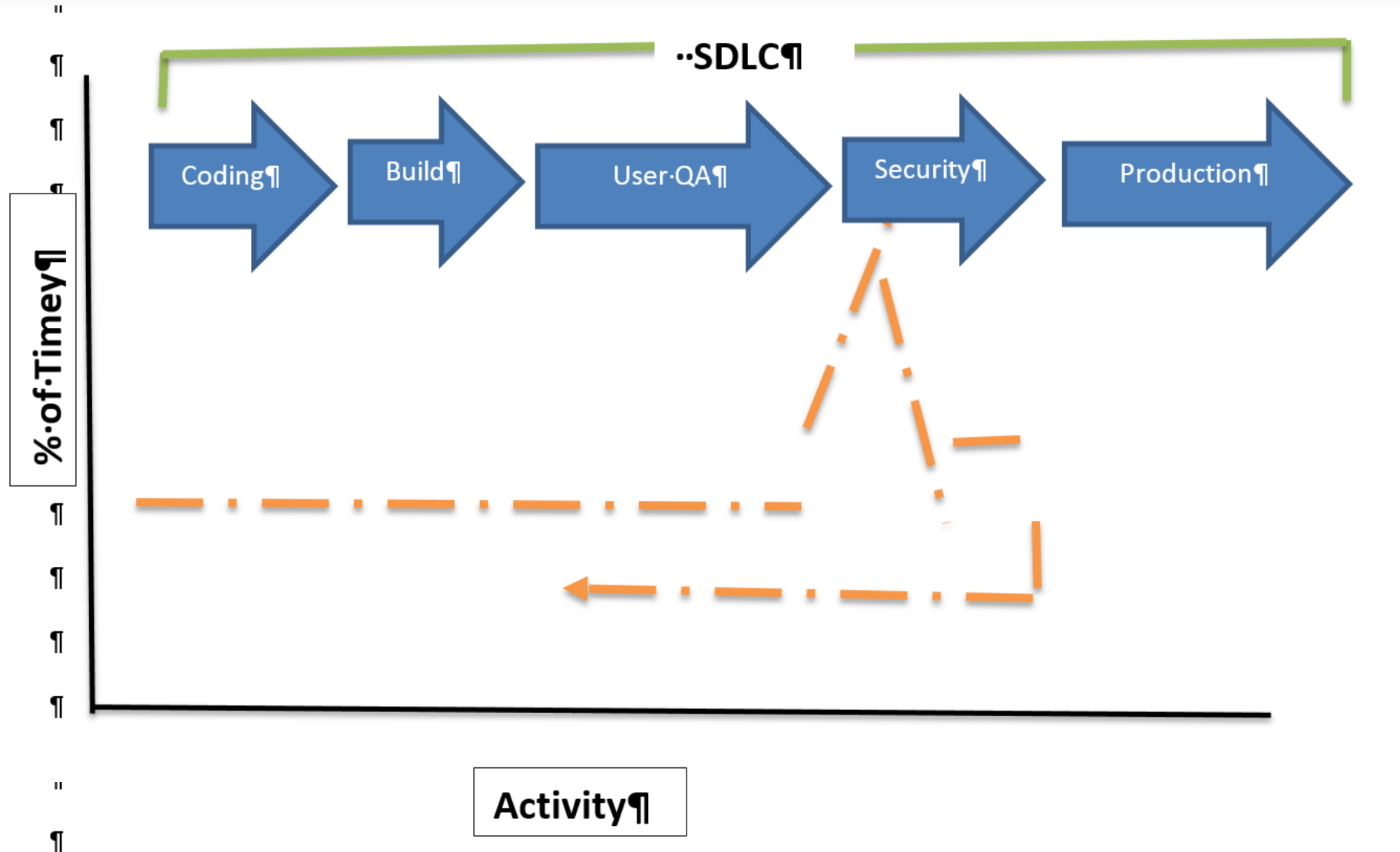
The attacker will replace the original URL with another, typically using the IP 127.0.0.1 or the hostname "localhost", which point to the local file system on the server. Under this hostname, the attacker finds a file path that leads to sensitive data.



Typical/Old Insecure SDLC Process

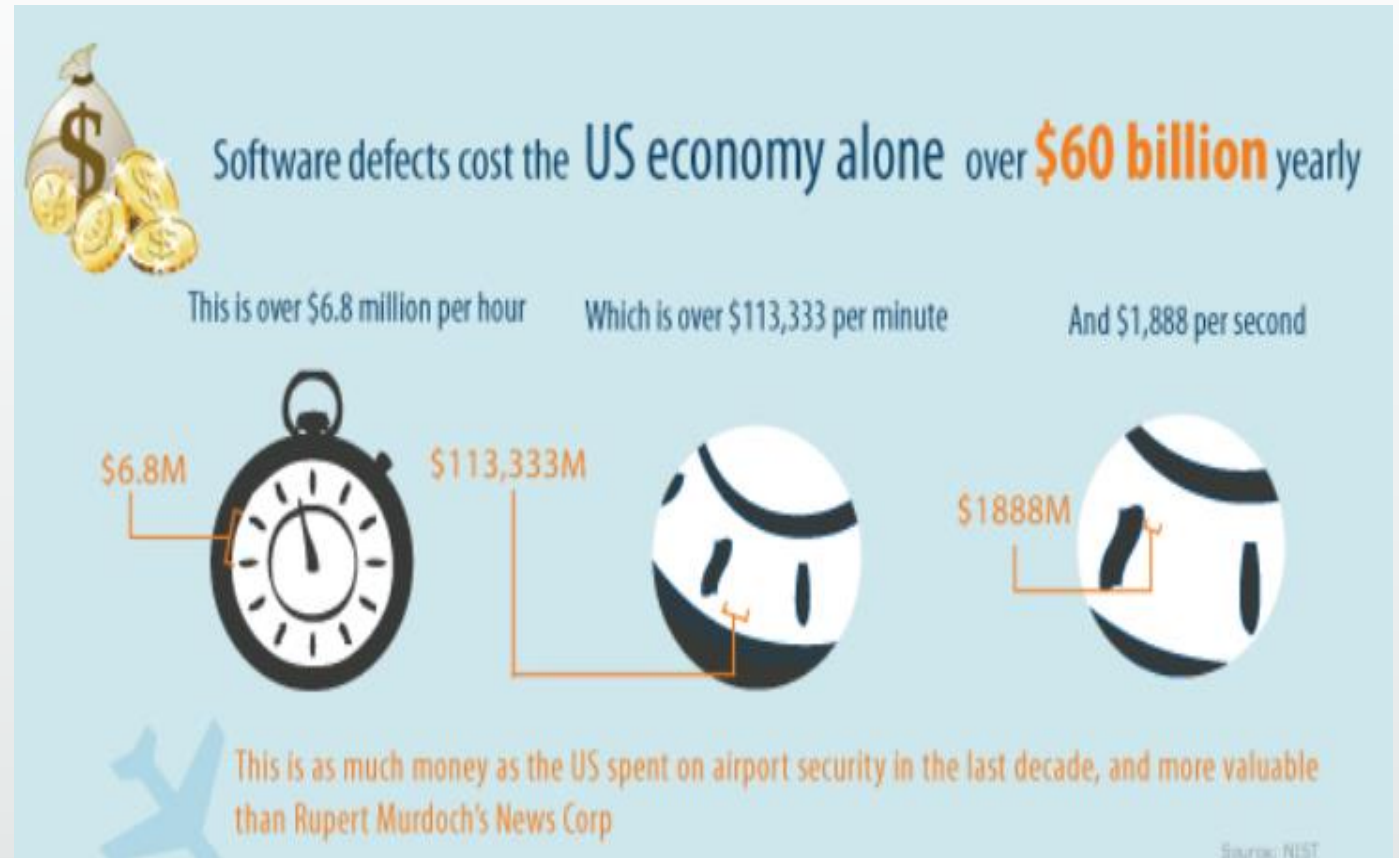
Inconsistently Builds Insecure Software

Typical SDLC Process



Cost of Software Defects

- \$60 Billion Annually
- Due to Bugs Vulnerabilities and Malware
- Results in Fewer Programs, Research
- Increased taxes and fees to consumers



A New Paradigm: A Software Development Framework for Building Secure Software

Building Security In

Security can never be an “after thought”

- In the development process or of management's responsibility
- Software manufactures treated differently than other businesses
- Can't sue for damages due to faulty software. No need to invest capital to build secure software
- Users (customers) are more interested in features than security



Security is Everybody's Business – New Paradigm

- Software vendors should be sued for faulty software and pay damages to businesses
- Developers should form professional association and make security engineering a performance criteria
- Management should see building secure software as an investment in building trust
- Users should make buying secure software a priority over new features



Gary McGraw – CTO of Cigital

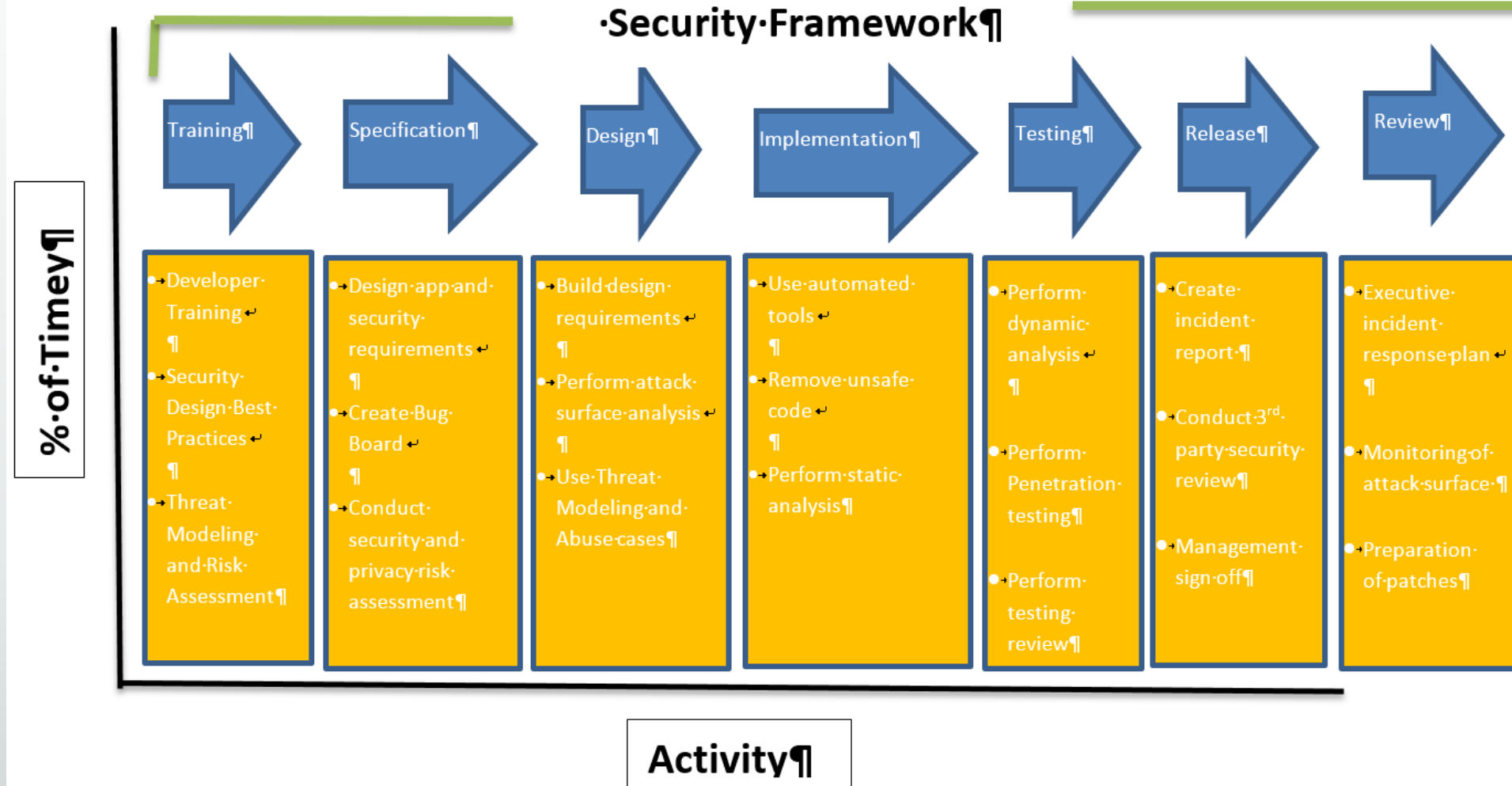
- Builders must practice security engineering, ensuring that the systems [they] build are defensible and not riddled with holes (especially when it comes to the software).
- Operations people must continue to architect reasonable networks, defend them, and keep them up.
- Administrators must understand the distributed nature of modern systems and begin to practice the principle of least privilege.
- Users must understand that software *can* be secure so that they can take their business to software providers who share their values. (Witness the rise of Firefox.) Users must also understand that they are the last bastion of defense in any security design and that they need to make tradeoffs for better security.
- Executives must understand how early investment in security design and security analysis affects the degree to which users will trust their products.



Software Development Framework with Security Build In

New & Secure SDLC Design

New Framework for Building Secure Software (New & Secure SDLC Design)



Training

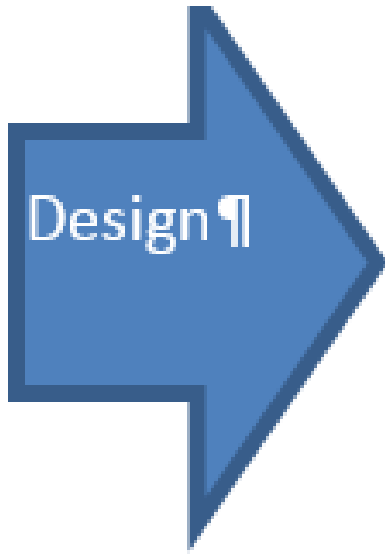
- Developer Training
- Security Design Best Practices
- Threat Modelling and Risk Assessment





- Design Application and Security Requirements
- Create Bug Board
- Conduct Security and Privacy Risk Assessment





- Build Design Requirements
- Perform Attack Surface Analysis
- Use Threat Modelling and Abuse Cases



Implementation¶

- Use automated tools
- Remove Unsafe Code
- Perform Static Analysis (Checkmarx)



Testing

- Perform Dynamic Testing
- Perform Penetration Testing
- Perform Testing Review (VeraCode)





Release

- Management Sign Off
- Conduct 3rd Party Security Review
- Create Incident Report





- Executive Incident Response Plan
- Monitoring of Attack Surface
- Preparation of Patches

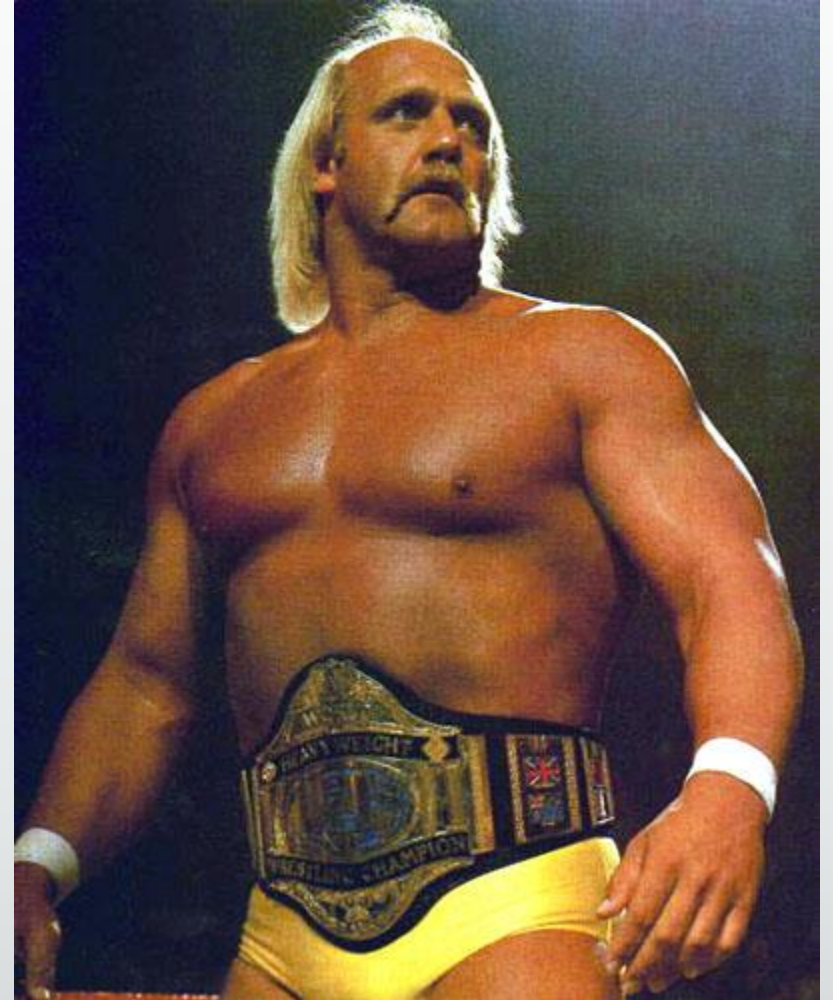


10 Best Practices

For Building Secure Mobile Software

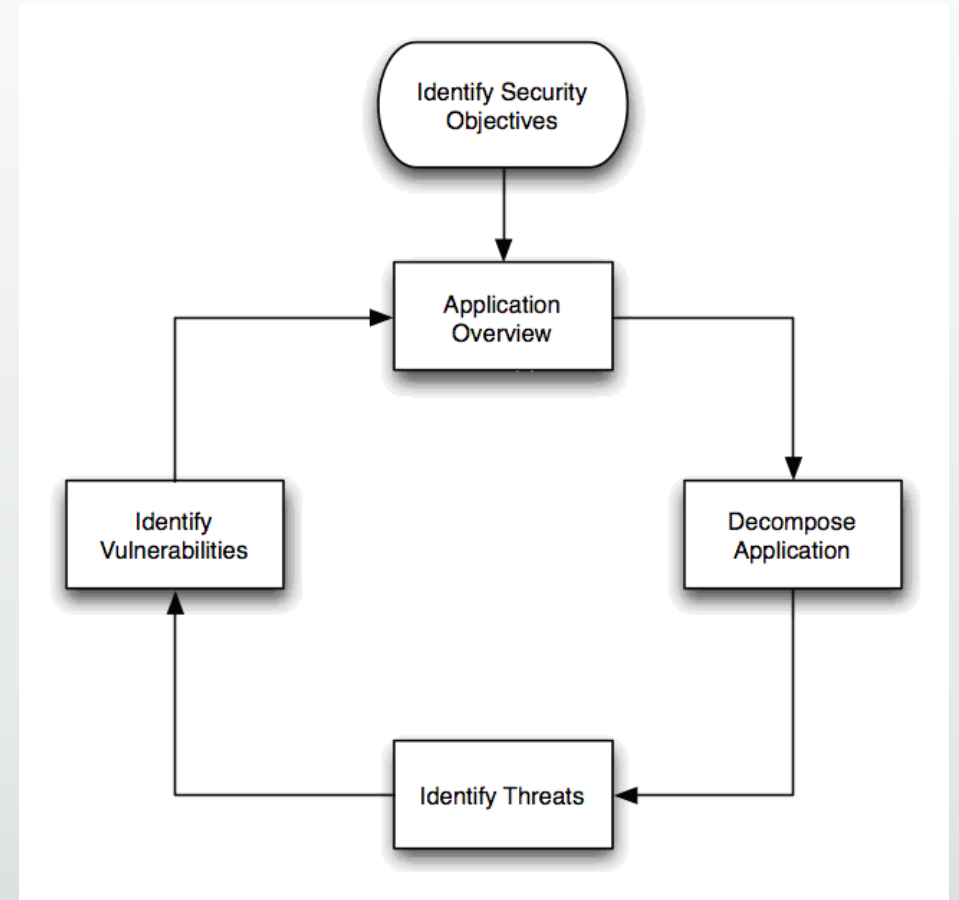
1. Always assign security to a Champion

- Champion is a person who fights for a cause on behalf of someone else
- Top Level Manager with authority and access to budgets
- Small company could be a security trained developer who over sees security
- Large company could be a CISO who is part of the CIO team and responsible for all aspects of security



2. Always model threats during software development

- Security requirements are designed with the application requirements
- Thinking like the hacker and analyzing possible misuses and compromises your application may cause
- Design software to avoid past mistakes



3. Always use modular code to separate parts of the application

- Dividing your code into modules segregates the code
- Makes it easier to reuse the code and provides better security
- “Sandboxing” technique to isolate dangerous actions to restricted areas
- Separate critical functions of the application and require multiple authentications



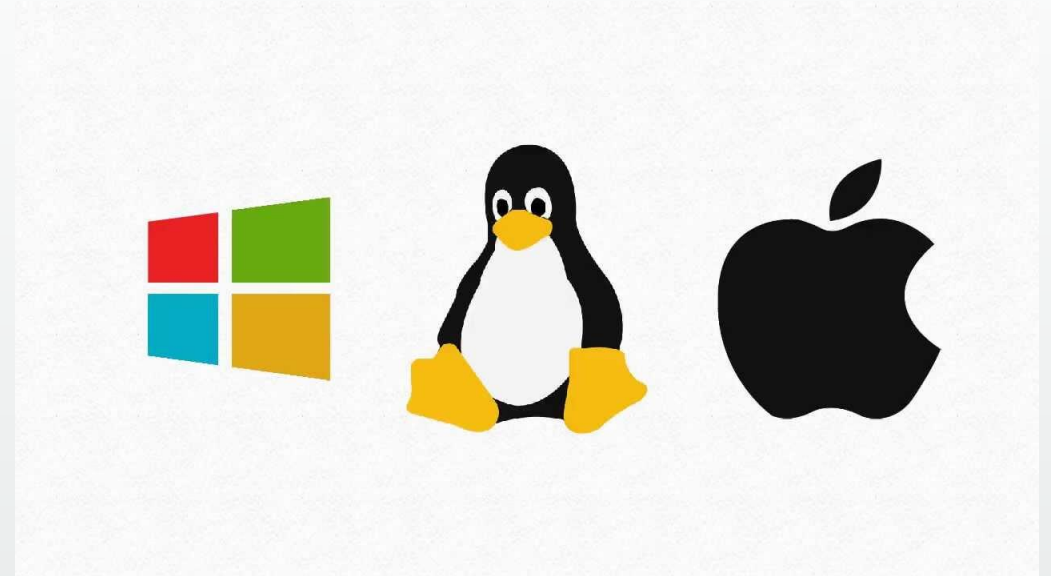
5. Always use tested code libraries

- Use only trusted libraries that have been used for many years
- Research the library before you use –
- Never write your own security code. Use only trusted encryption libraries that have been used for many years
- Good libraries have good documentation. Work and test the to employ it correctly



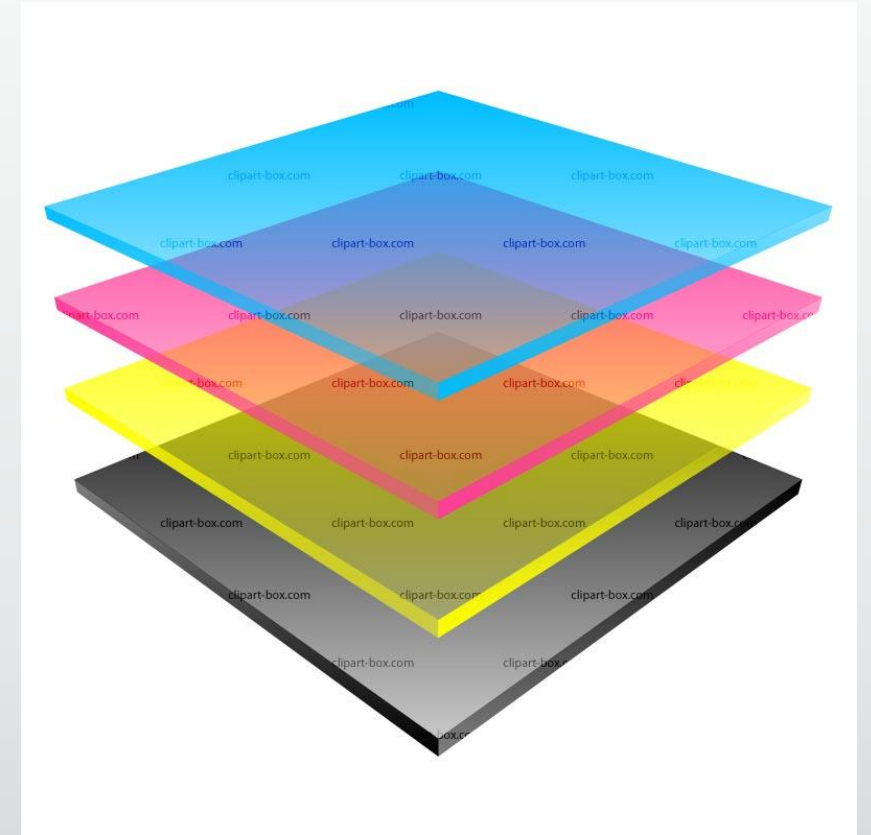
5. Always test application on different mobile platforms and APIs

- Good performance and security one platform does not guarantee the same on another
- Each platform as different APIs and must be independently tested for performance and security
- Research known vulnerabilities and abuse cases and test application on each platform
- Goal is to avoid past mistakes



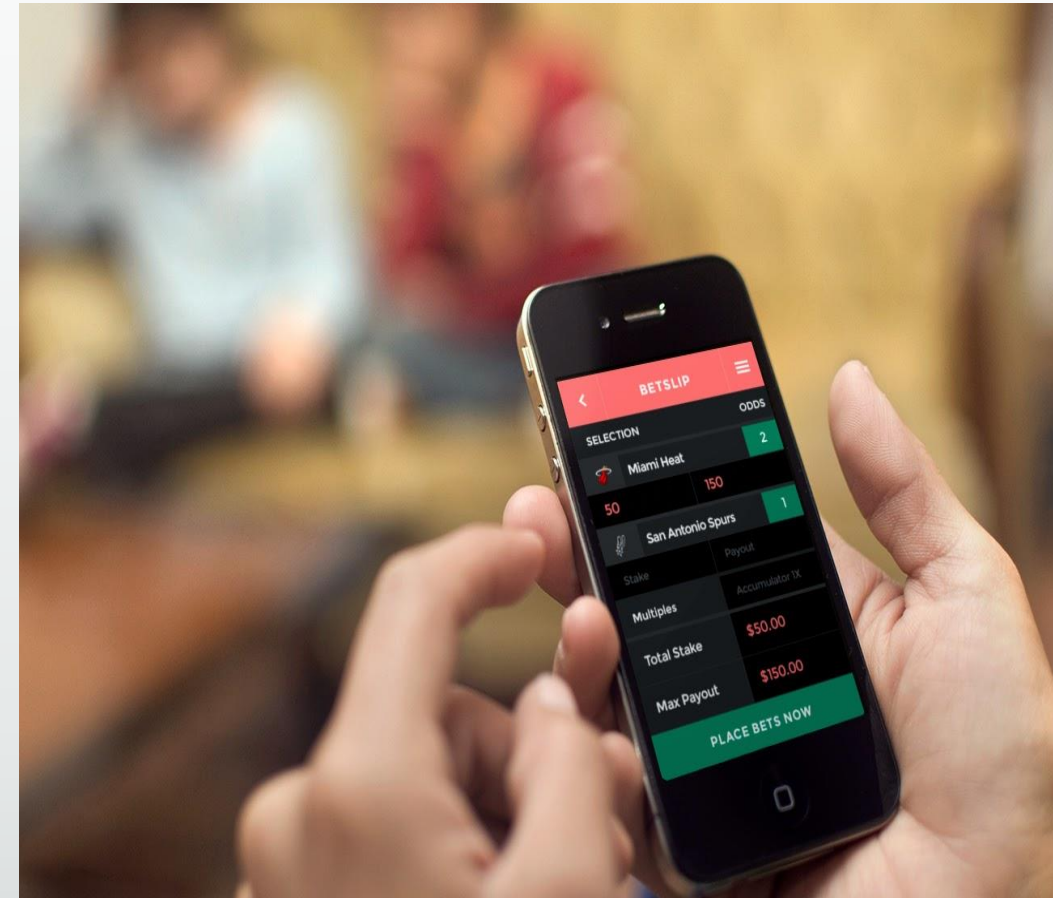
6. Always add multiple levels of authentication

- Consumers like to use static passwords for online transactions.
 - Main reason for security questions
- Passwords with security questions not as good as authentication
- Two factor authentication is better
 - Something I know –password, pin
 - Something I have – smartphone, dongle
- Passwords must be strong. Length more important than complexity



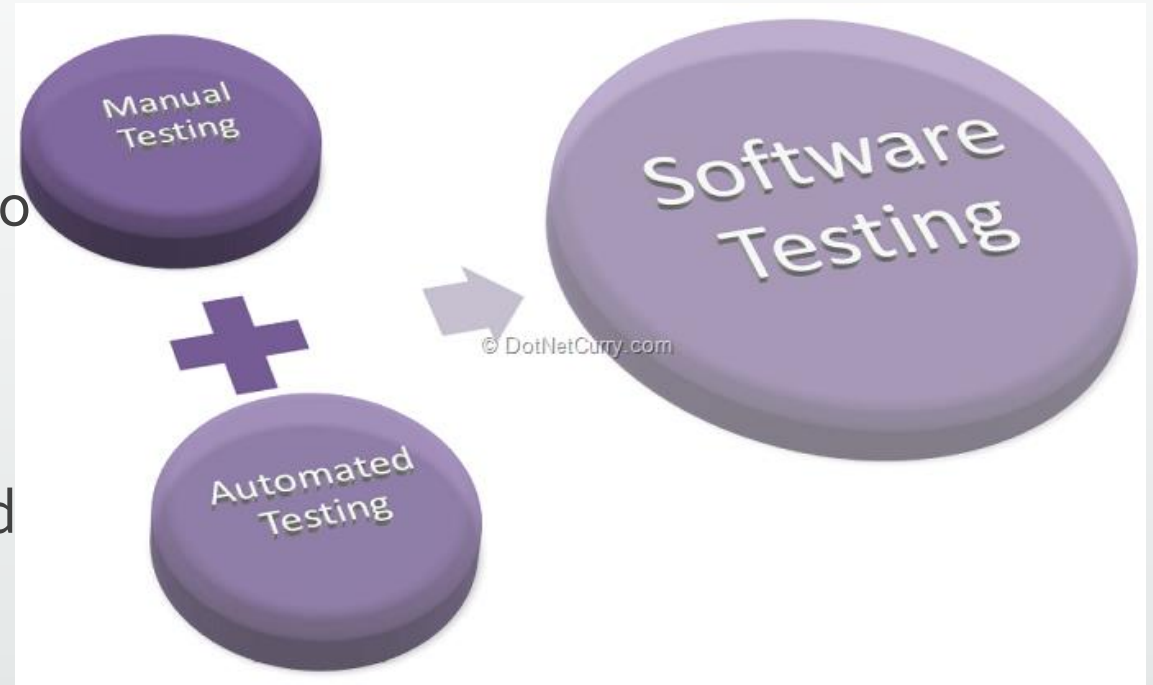
7. Always require minimal data from the user

- Design app to request the least amount of data
 - What information do I need?
- Design a “choke point”
 - Test and sanitize data input for text box, file, variable or library before using the data in the app or storing on the server




8. Always use automated tools and auditors for testing

- Use automated static and dynamic testing tools
- Code require too tedious for humans to maintain concentration during entire process
- Security testing of the software should also be done by independent 3rd party testing companies
- <https://www.veracode.com/blog/2013/12/static-testing-vs-dynamic-testing>
- Another set of eyes independent of the developers who wrote the program is invaluable



9. Always build user trust

- Users trust that their authentication information and emails will be stored securely
- To buy or use your software program users must trust it
 - Especially true for apps that use cameras, GPS or microphones.
- On web site display privacy policy and explain how data is protected and what will happen if a breach occurs.
- Users use number of downloads and user reviews to rate application.

T 
R E L Y
U P O N
S O U R C E
(SOMETHING BIGGER)
T O T A L L Y



10. Always stay updated on latest exploits, vulnerabilities

- Threat environment constantly changing.
- Subscribe to:
 - www.cert.org
 - cve.mitre.org
 - www.owasp.org
- After release developers must test app for security threats and vulnerabilities
- Prepare and distribute patches
- Time Delay in Deploying patches and devices older than 18 months are not supported



Application Security Testing Techniques

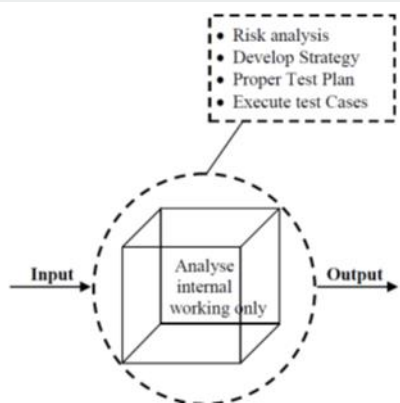
SAST, DAST, & IAST

Application Security Testing Techniques

1. **Static Application Security Testing (SAST)**- used as a Source Code Analysis tool.
 - White box testing approach that examines the application from the inside, searching its source code for conditions that indicate that a security vulnerability prior to the launch of an application and used to strengthen code.
 - It is used at a very early in the software development as it does not require a working application and can take place without code being executed.
2. **Dynamic Application Security Testing (DAST)**- find security vulnerabilities and weaknesses in a running application
 - A black-box testing methodology in which an application is tested from the outside typically web apps.
 - A tester using DAST examines an application when it is running and tries to hack it just like an attacker would.
 - Some of the methods of security testing by DAST are: fault injection techniques on an app, such as feeding malicious data to the software, to identify common security vulnerabilities, such as SQL injection and cross-site scripting.
 - DAST can also cast a spotlight in runtime problems like authentication and server configuration issues, as well as flaws visible only when a known user logs in.

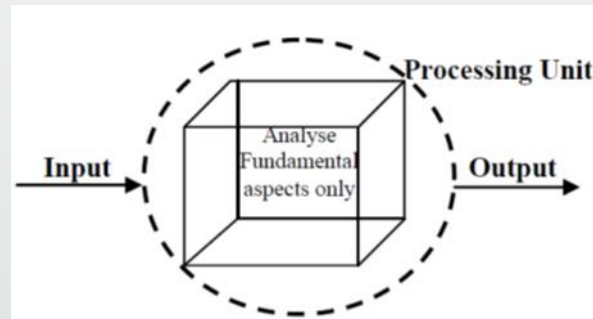
Application Security Testing Techniques

3. **Interactive Application Security Testing (IAST)** -combine the strengths of both SAST and DAST methods as well as providing access to code, HTTP traffic, library information, backend connections and configuration information.
- IAST places an agent within an application and performs all its analysis in the app in real-time and anywhere in the development process IDE, continuous integrated environment, QA or even in production



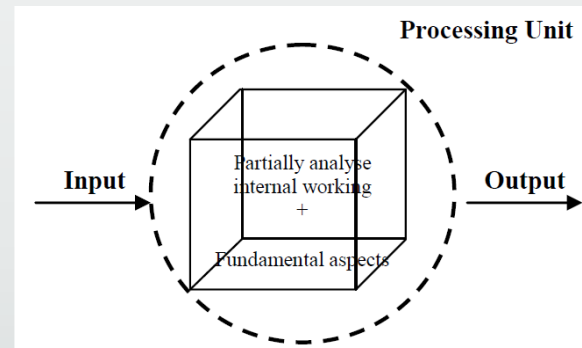
White Box Testing

[1]



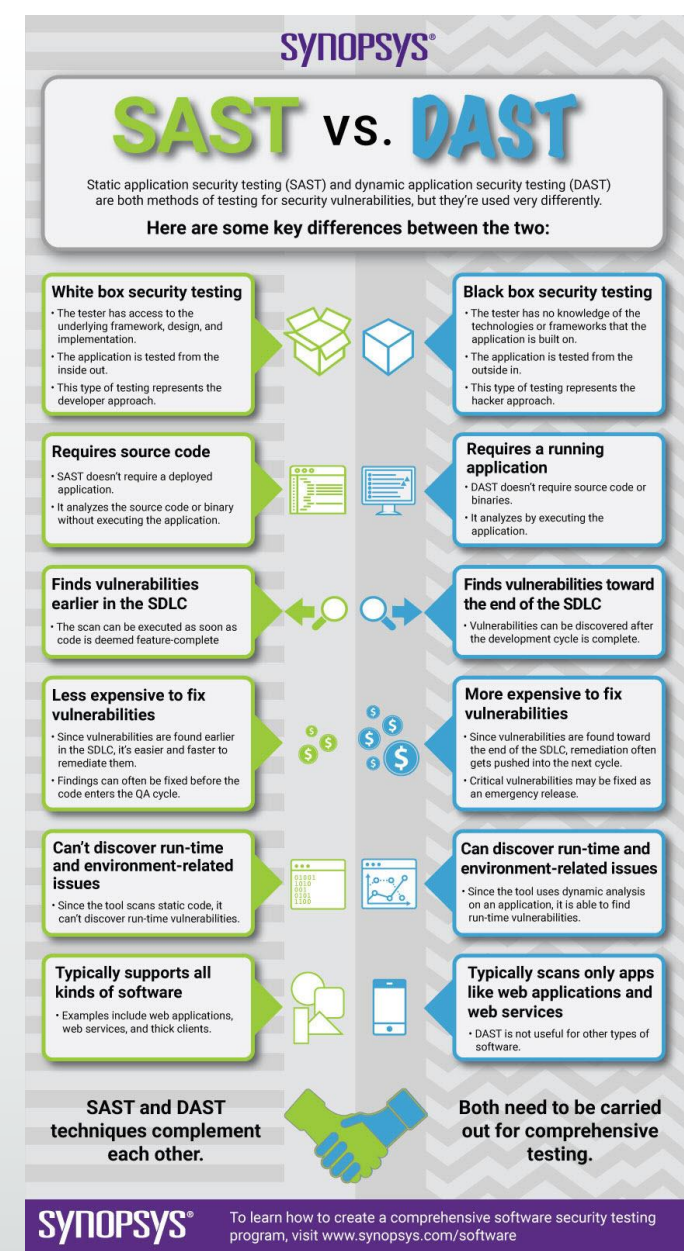
Black Box testing

[1]



Grey Box testing

[1]



[2]

Application Security Testing

Testing Roles

- Hackers - Access computer system or network without authorization
- Crackers - Break into the systems to steal or destroy data
- Ethical Hacker - Performs most of the breaking activities, but with permission from the owner
- Script Kiddies or packet monkeys - Inexperienced Hackers with little/no programming language skills
- Scrip

Security Testing Tool

- Metasploit
- Veracode
- Check marx
- W3af
- Veracode
- CodeScan

Summary

1. The OWASP has conducted a survey of web application vulnerabilities. The most common vulnerability is injection. This can be stopped by creating a choke point and checking and sanitizing all input data
2. The current procedure for developing software results in insecure software because security testing is at the end of the process and the developers building the code may not have had any security training
3. A new paradigm is needed with a change in law that makes software vendors liable for faulty software. Developers must form a professional association and see secure coding as a performance criteria. Security must be built into the development process at every step.
4. Following the 10 best practices for building mobile applications will eliminate all of the 10 Top OWASP vulnerabilities, especially subscribing to web sites which deal with new threats and vulnerabilities. Unfortunately, patches for mobile devices take a long time to be deployed and if the device is old the manufacture may not support the update.

References

- [1] Khan. M.E, Khan.F ,” A Comparative Study of White Box, Black Box and Grey Box Testing Techniques”, (*IJACSA*) *International Journal of Advanced Computer Science and Applications*, Vol. 3, No.6, 2012
- [2] <https://www.synopsys.com/glossary/what-is-sast.html>
- [3] <https://www.guru99.com/what-is-security-testing.html>