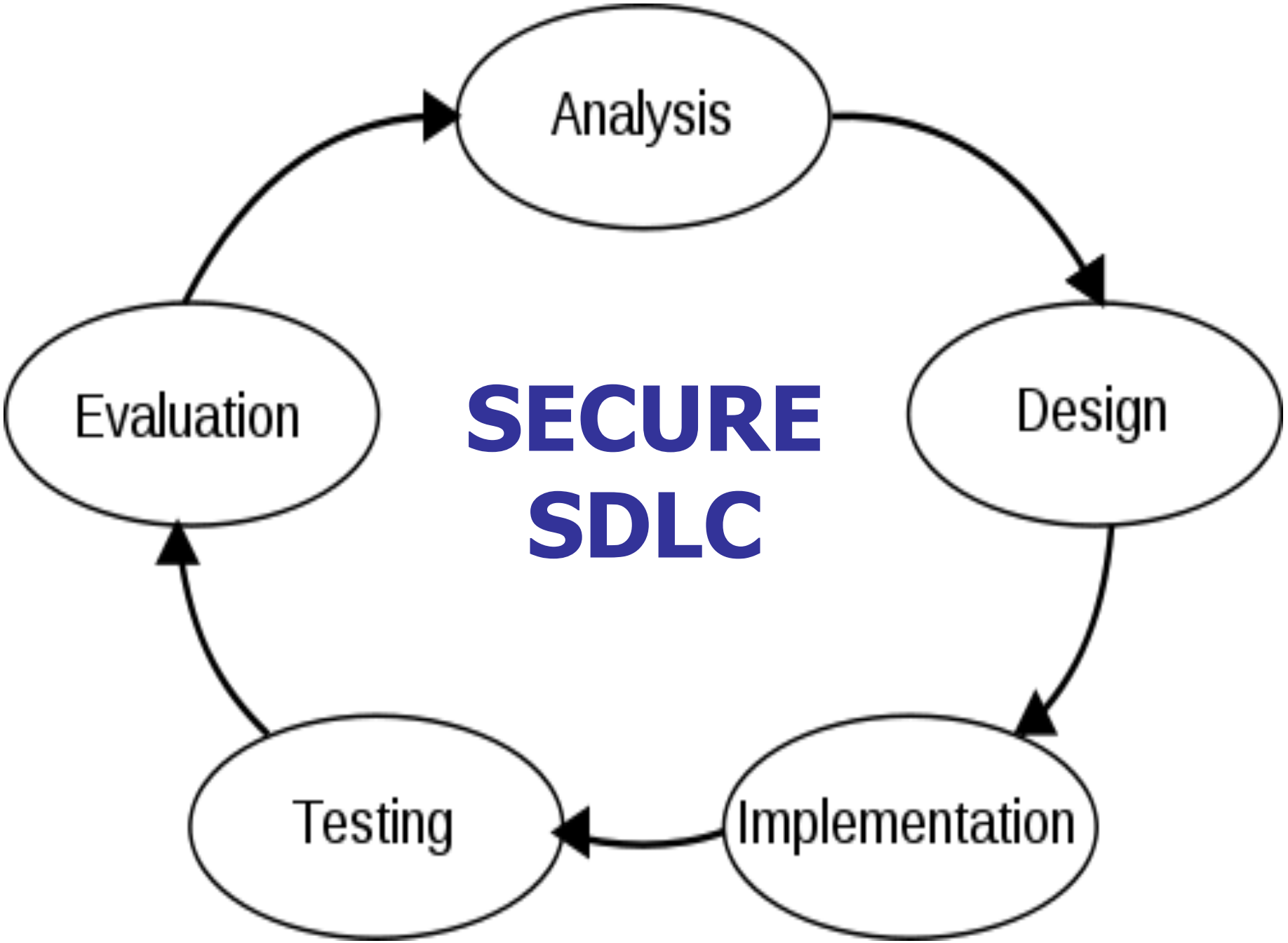
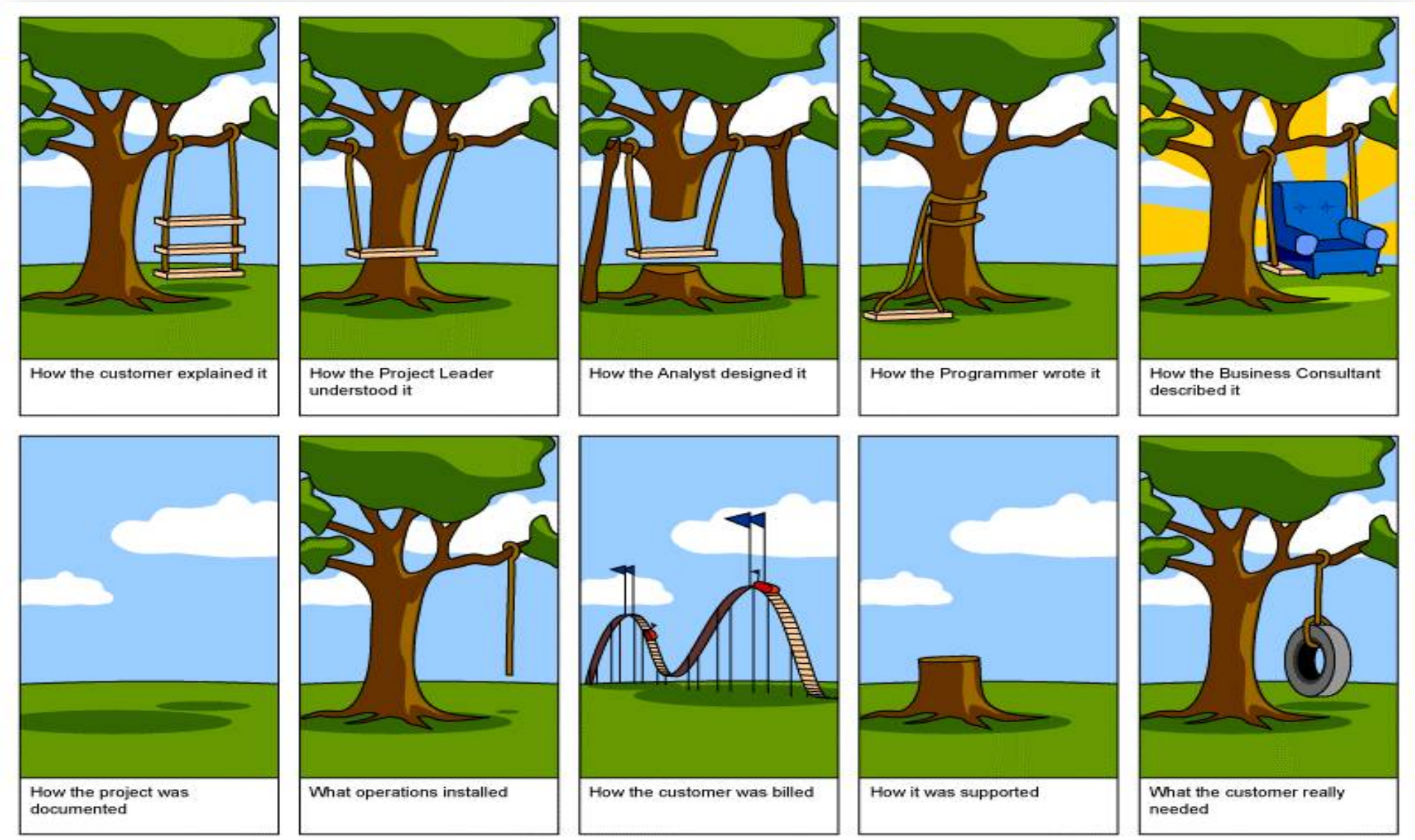

Analysis
Design
Implementation
Testing
Evaluation
SECURE
SDLC

Jim Manico @manicode

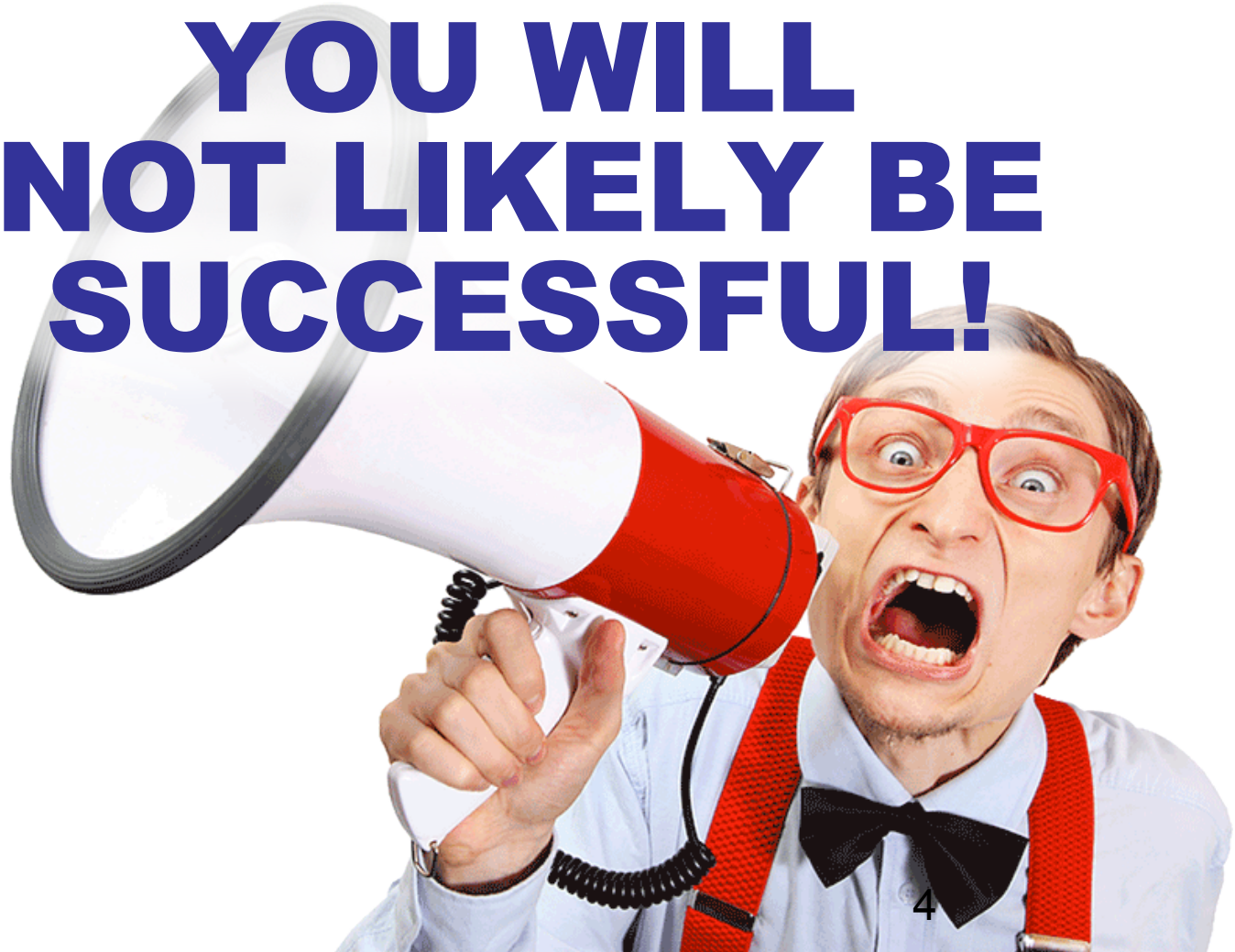
- **OWASP Volunteer**
  - Global OWASP Board Member
  - Manager of several OWASP secure coding projects
- **Security Instructor, Author**
  - 17 years of web-based, database-driven software development and analysis experience
  - Author of "Iron Clad Java" from Oracle Press and McGraw Hill
- **Resident of Kauai, Hawaii**
  - Aloha!

Poor Communication = Epic Software Dev Failure



If you don’t have a published SDLC……

YOU WILL NOT LIKELY BE SUCCESSFUL!



Security in the SDLC

Essential that security is embedded in all stages of the SDLC

- Business Requirements
- Technical Requirements
- Development
- Testing
- Deployment
- Operations

"The cost of removing an application security vulnerability during the design phase ranges from 30-60 times less than if removed during production."

- NIST, IBM, and Gartner Group

SDLC building blocks

Published SDLC (++)

Secure Coding Guidelines (-)

Secure Coding Checklist (+)

Non Functional Requirements (++)

Static Code Analysis (+)

Manual Code Review (+)

Dynamic Code Analysis (+)

Giving Raw Scanner Data to Dev (-)

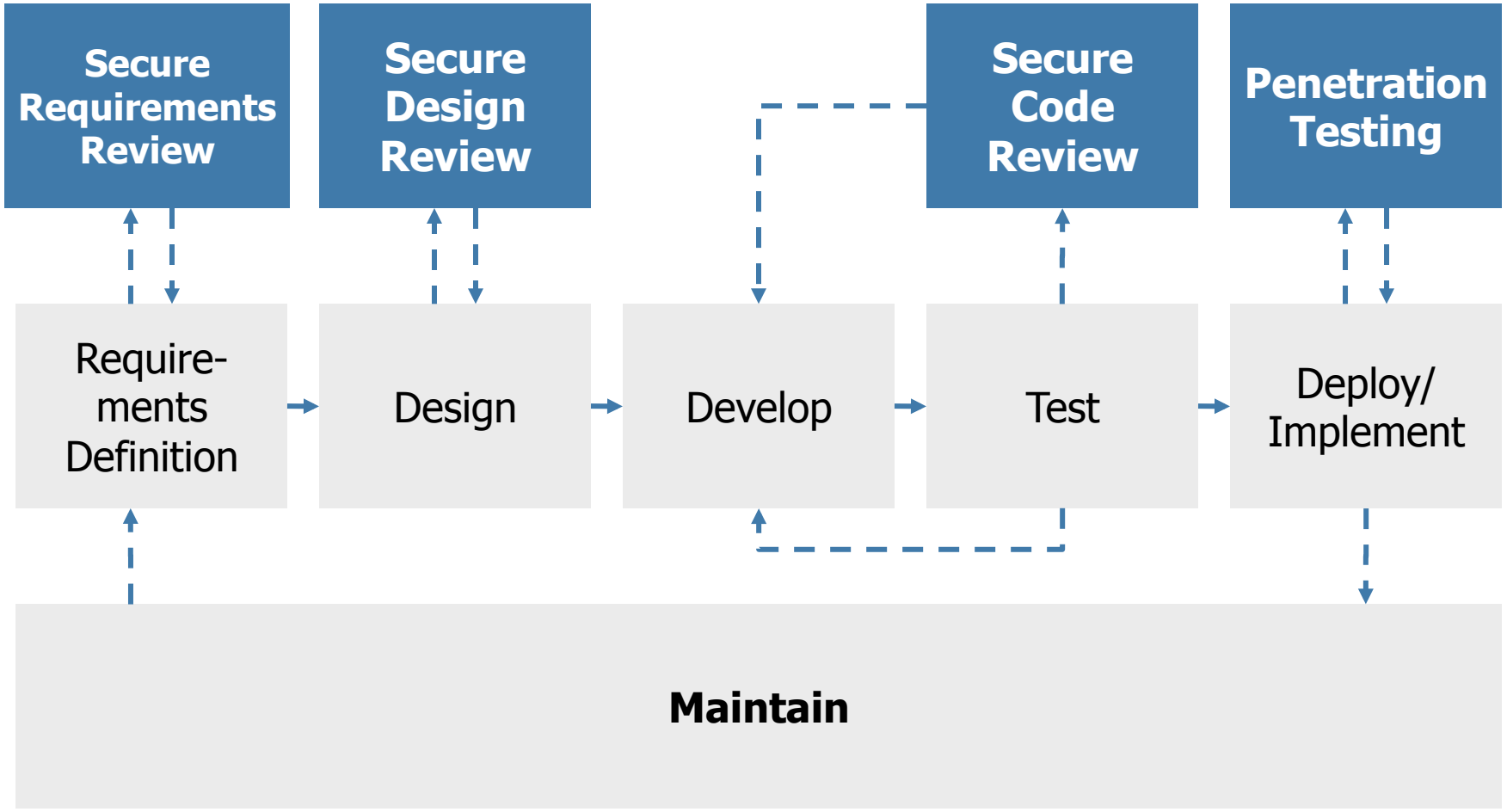
Security Awareness Training (+++)

Threat Modeling (+/-)

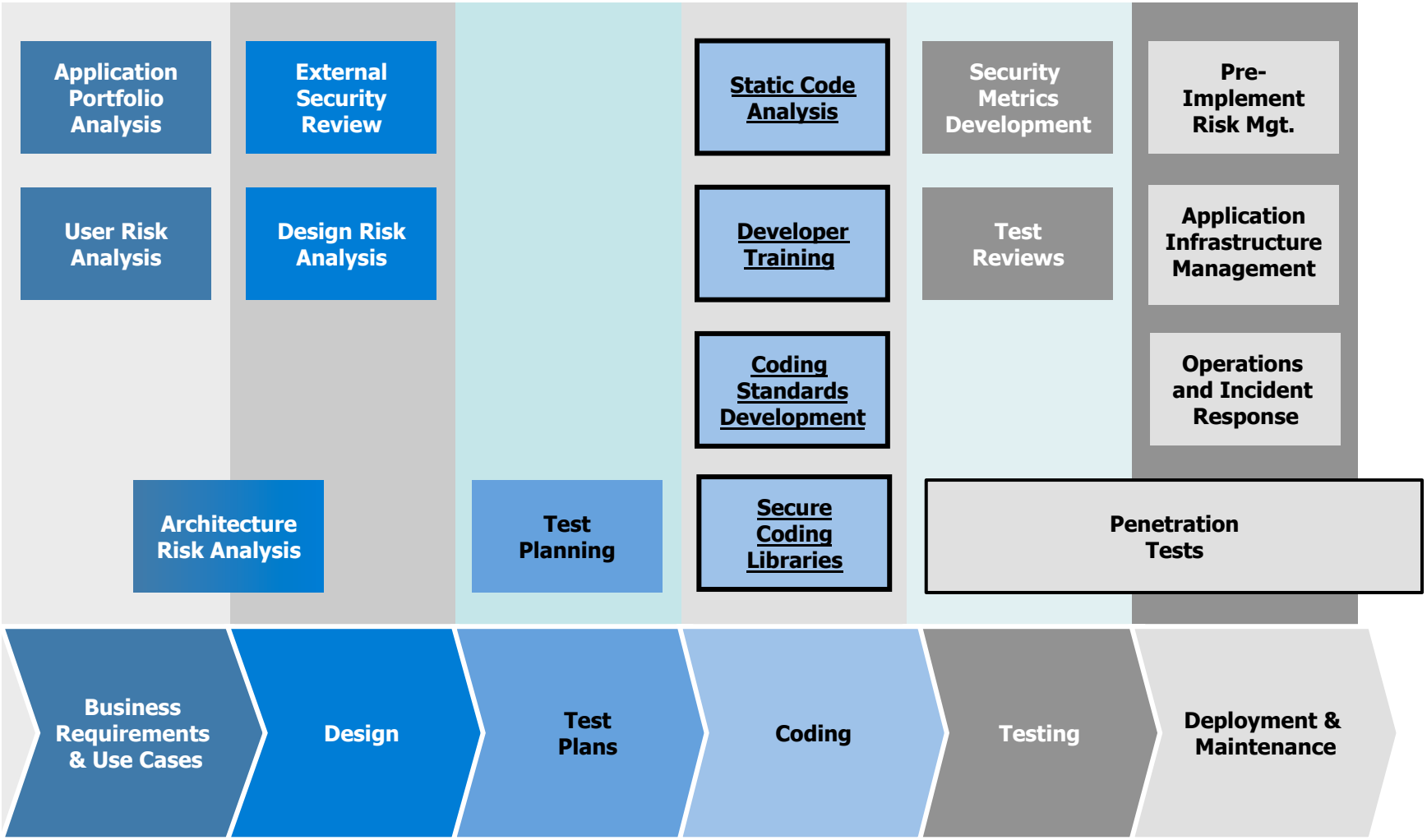
Application Security Risk Matrix (++)

Center of Excellence (++)

Security in the SDLC



Security in the SDLC with more beef



Security requirements

Establishing the security requirements for the application

What are the key security risks within the application? It depends.

- Type of information is the application processing
- Size of company, number of users, danger of features
- Financial and other business impact of potential incidents

Involve risk group and/or internal audit to avoid later conflict

Identify organizations standards (e.g. password lengths, security schemes), legal and regulatory security requirements!

Are these requirements actionable by the development, testing and operations teams?

OWASP Application Security Verification Standard (ASVS)

OWASP standard for security verification of applications

Built in security area sections

- Authentication
- Session Management
- Etc...

Excellent way to manage pentesting and other assessment teams

Excellent standard to help developers understand secure coding controls from a high level

ASVS 2 Authentication Requirements (Easy to Discover)

- V2.1 Verify all pages and resources require authentication except those specifically intended to be public (Principle of complete mediation).
- V2.2 Verify all password fields do not echo the user's password when it is entered.
- V2.4 Verify all authentication controls are enforced on the server side.
- V2.6 Verify all authentication controls fail securely to ensure attackers cannot log in.
- V2.16 Verify that credentials, and all other identity information handled by the application(s), do not traverse unencrypted or weakly encrypted links.
- V2.17 Verify that the forgotten password function and other recovery paths do not reveal the current password and that the new password is not sent in clear text to the user.
- V2.18 Verify that username enumeration is not possible via login, password reset, or forgot account functionality.
- V2.19 Verify there are no default passwords in use for the application framework or any components used by the application (such as "admin/password").

ASVS 2 Authentication Requirements (Intermediate, p1)

- V2.7 Verify password entry fields allow or encourage the use of passphrases, and do not prevent long passphrases or highly complex passwords being entered, and provide a sufficient minimum strength to protect against the use of commonly chosen passwords.
- V2.8 Verify all account identity authentication functions (such as registration, update profile, forgot username, forgot password, disabled / lost token, help desk or IVR) that might regain access to the account are at least as resistant to attack as the primary authentication mechanism.
- V2.9 Verify users can safely change their credentials using a mechanism that is at least as resistant to attack as the primary authentication mechanism.
- V2.12 Verify that all authentication decisions are logged. This should include requests with missing required information, needed for security investigations.
- V2.13 Verify that account passwords are salted using a salt that is unique to that account (e.g., internal user ID, account creation) and use bcrypt, scrypt or PBKDF2 before storing the password.

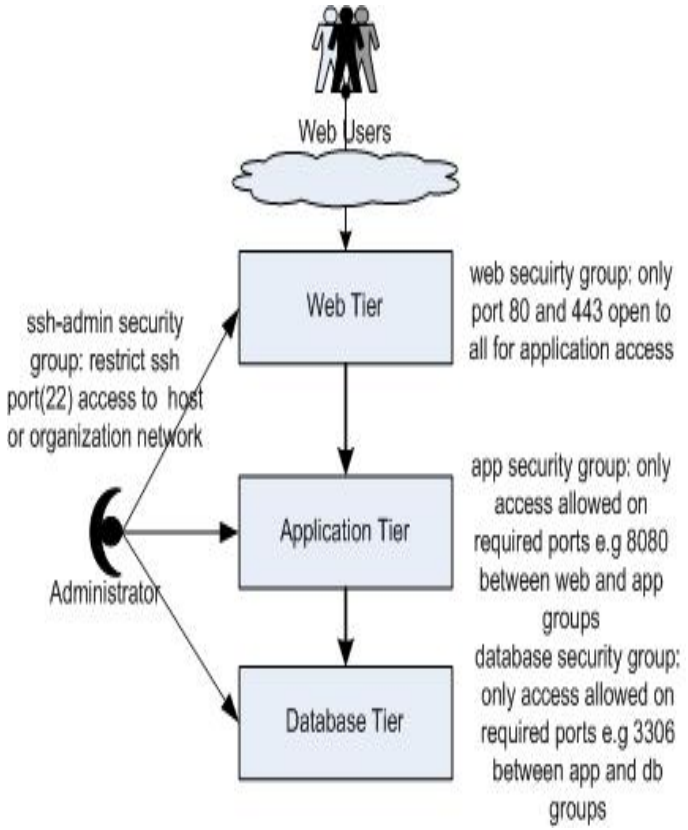
ASVS 2 Authentication Requirements (Intermediate, p2)

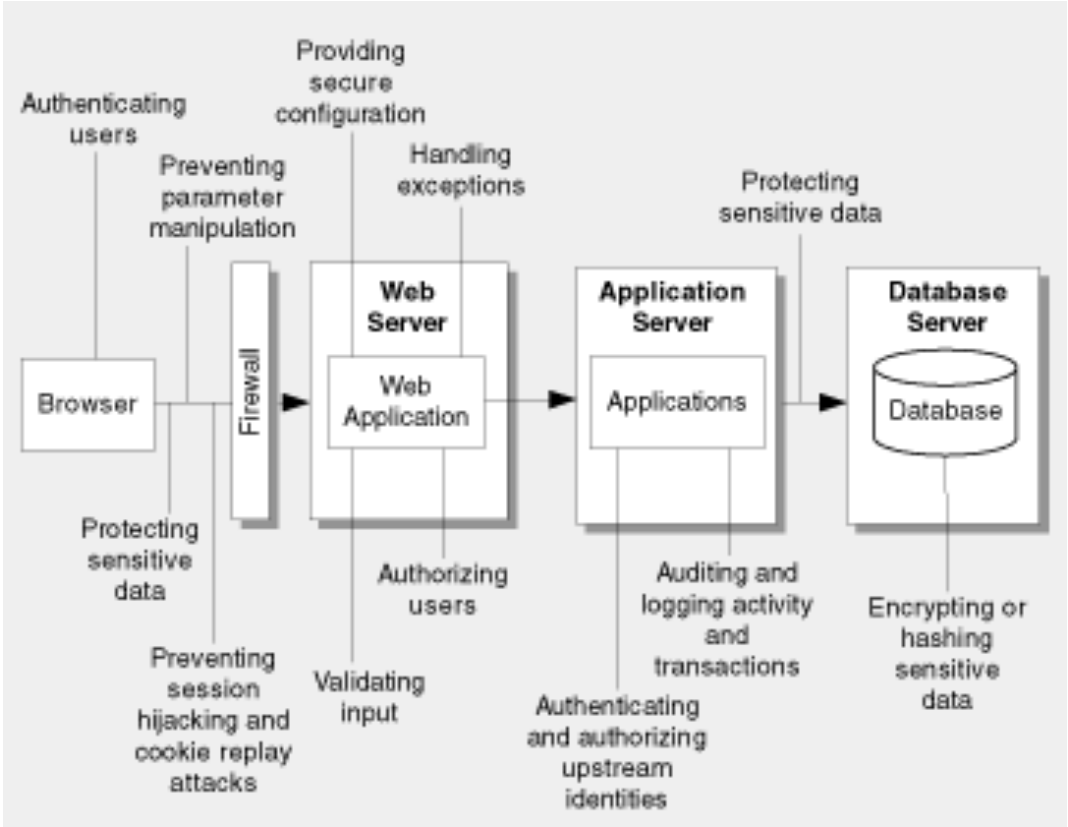
- V2.20 Verify that a resource governor is in place to protect against vertical (a single account tested against all possible passwords) and horizontal brute forcing (all accounts tested with the same password e.g. “Password1”). A correct credential entry should incur no delay. Both these governor mechanisms should be active simultaneously to protect against diagonal and distributed attacks.
- V2.21 Verify that all authentication credentials for accessing services external to the application are encrypted and stored in a protected location (not in source code).
- V2.22 Verify that forgot password and other recovery paths send a link including a time-limited activation token rather than the password itself. Additional authentication based on soft-tokens (e.g. SMS token, native mobile applications, etc.) can be required as well before the link is sent over.
- V2.23 Verify that forgot password functionality does not lock or otherwise disable the account until after the user has successfully changed their password. This is to prevent valid users from being locked out.
- V2.24 Verify that there are no shared knowledge questions/answers (so called "secret" questions and answers).
- V2.25 Verify that the system can be configured to disallow the use of a configurable number of previous passwords.

ASVS 2 Authentication Requirements (Advanced)

- V2.5 Verify all authentication controls (including libraries that call external authentication services) have a centralized implementation.
- V2.26 Verify re-authentication, step up or adaptive authentication, SMS or other two factor authentication, or transaction signing is required before any application-specific sensitive operations are permitted as per the risk profile of the application.

Gratuitious slide with pictures to break up all the text slides

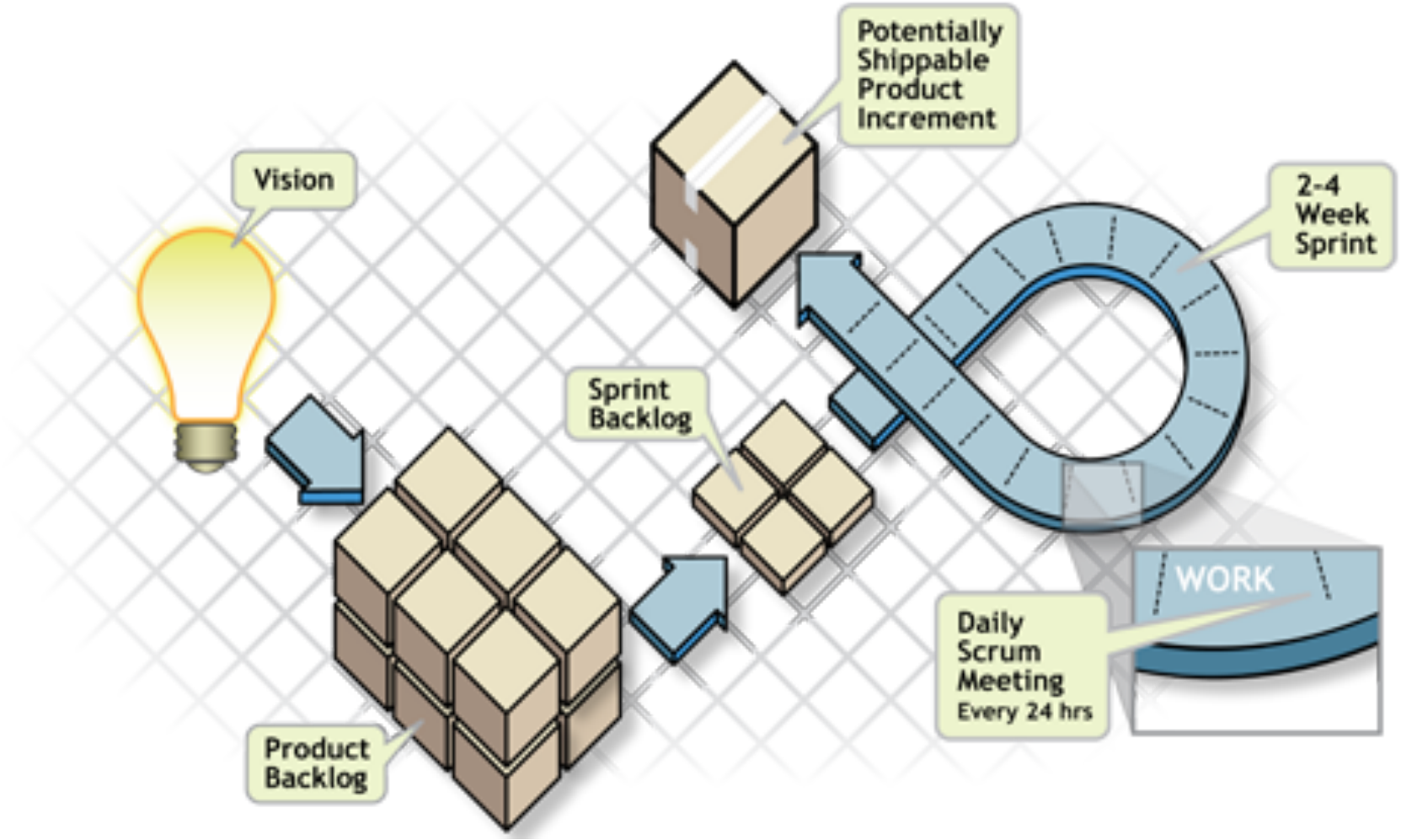




Agile Principles

- Welcome changing requirements, even late in development.
- Deliver working software frequently. Working software is the primary measure of progress.
- Business people and developers must work together daily.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile Workflow



Agile Security

Security Sprint Approach

Security Sprint Approach:

- Dedicated sprint focusing on application security.
- Stories implemented are security related.
- Code is reviewed.
- Stories may include:
  - Input validation story
  - Logging story
  - Authentication story
  - Authorization
  - XSS Story
  - SQLi Story

Every Sprint Approach

Every Sprint Approach:

- Similar to Microsoft Security Development Lifecycle (SDL).
- Consists of the requirements and stories essential to security.
- No software should ever be released without requirements being met.
- Sprint is two weeks or two months long.
- Every security requirement in the every-Sprint category must be completed in each and every Sprint.
  - Or the Sprint is deemed incomplete, and the software cannot be released.

Many organizations that claim to do "agile development" are far from agile!

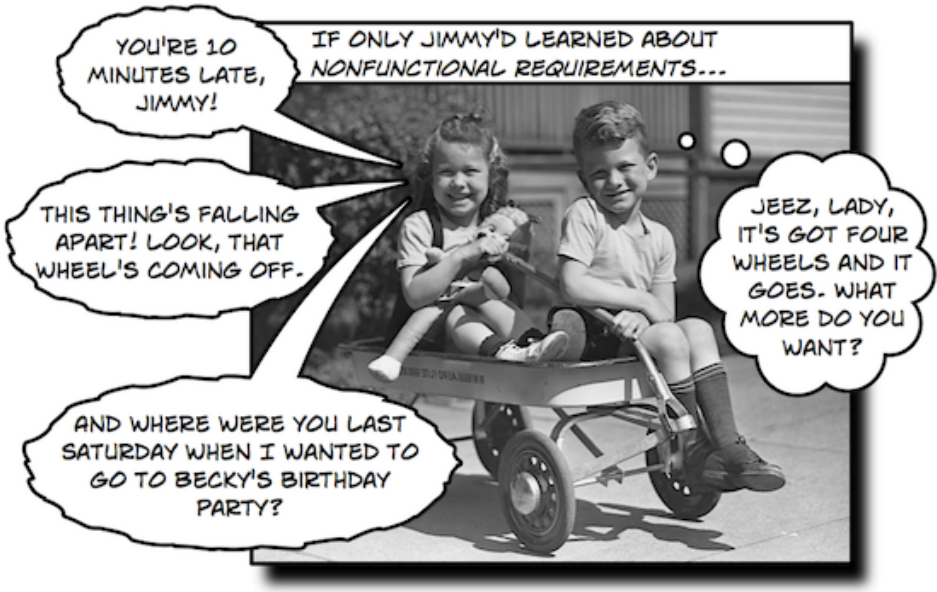
Non-Functional Requirements (++)

Most effective of all building blocks

‘Container’ for other SDLC building blocks.

Can include application security guidelines, secure coding checklist, security policies, etc.

Effective NFRs will document the requirement *and* explain why the requirement is necessary.



Application Security Requirements Tailoring

- Get the security requirements and policy right

- Start with a generic security requirements
  - Must include all security mechanisms
  - Must address all common vulnerabilities
  - Can be use (or misuse) cases

- Tailoring examples…
  - Specify how authentication will work
  - Detail the access control policy
  - Define the input validation rules
  - Choose a logging approach

Developers are almost NEVER given clear security requirements and this is a absolute massive SDLC failure!!!

Threat modeling (+/-)

Hit or miss at most locations

Can be informal process

Combines nicely with NFRs

Discussing NFR often leads to threat modeling discussion

Application Security Risk Matrix (++)

External facing	Data: Non sensitive	Data: sensitive
Internal facing	Data: Non sensitive	Data: sensitive

Development

Ensuring that code is developed securely and implementing the security controls identified during the design phase

Developer security awareness programs

Unit testing of security features of the application

Security audit and code reviews

- Secure coding standards
- Automated code review tools
- Independent code review by third party or IT security

Secure Coding Libraries (+++)

Reusable Security Controls

These are the software centric building blocks that defend software

These components are the heart of application security defense

Dedicate your top developer resources into vetting, building and standardizing on these components

Build training, testing and other activities around these artifacts

Security awareness training (+++)

Instructor-led training

Course curriculum for each job responsibility

Very useful for educating on attack techniques and unexpected behavior

Rewards for training

Your goal should be to provide anyone that can influence application security, e.g. project managers, development managers, application developers, server configuration, release management, QA, etc. with the training, awareness and resources they need to be successful.

Secure Coding Guidelines (-)

Overlooked by developers

“Static and not helpful”

100+ pages that can be language specific

Most surprising discovery over the last 5 years

Secure Coding Checklist (+)

Simple 1-2 page document

All checklist items must be relevant

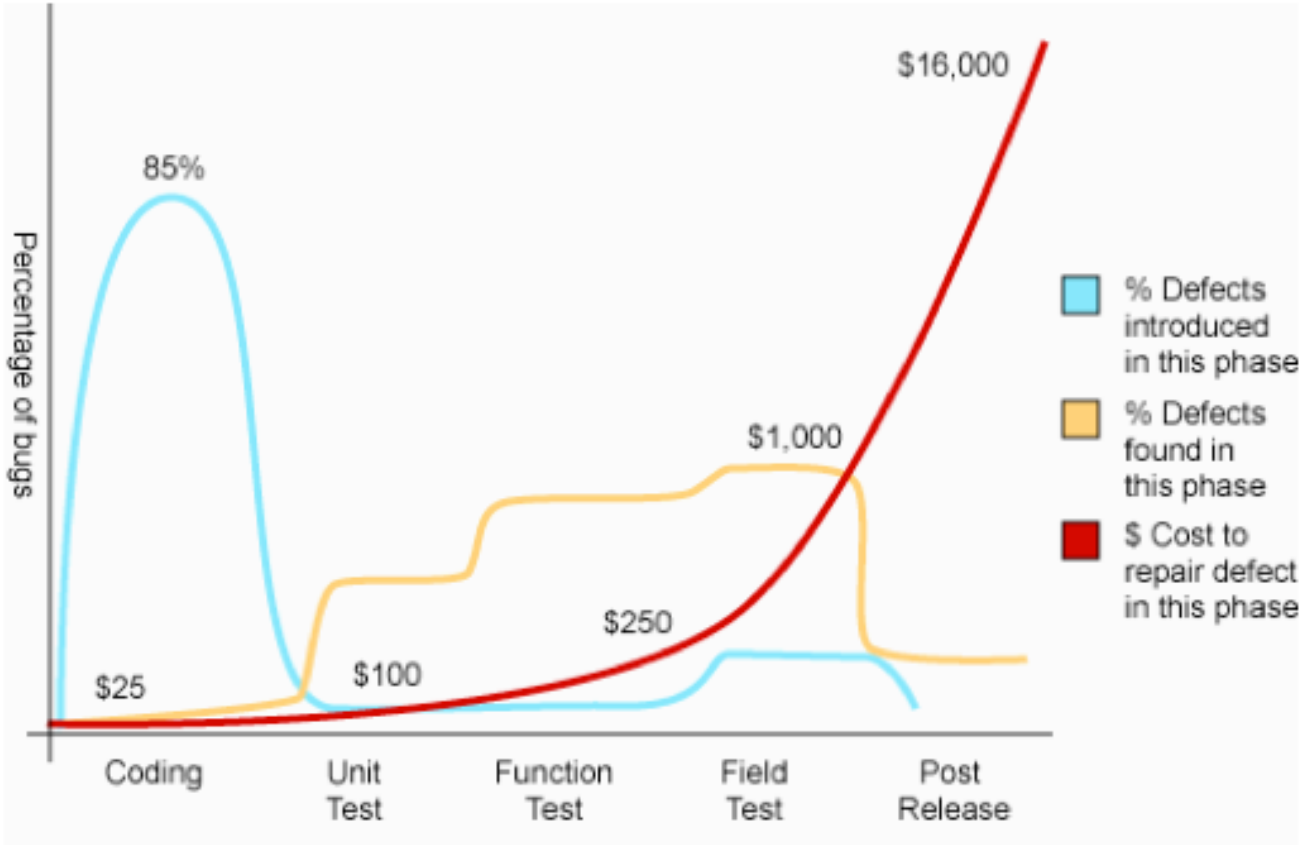
Brief document must be backed up with deeper resources and code samples

Testing

- Ensure the application meets the security standards and security testing is performed
- Has the security design been implemented correctly in the application components?
- Execution of test plans created during the design phase
- Independent penetration testing, including infrastructure assessment
- Security release and sign off before deployment to the production environment

Why Code review

The Cost of Software Bugs



"We cant hack our-selves secure, and if we could it would cost too much"

Source: Applied Software Measurement, Capers Jones, 1996

Static Code Analysis (SCA) (+)

SDLC requires SCA

Must be baked into acceptance criteria for code to leave the SDLC.

Assurance to QA that code is ready for testing

SCA can be integrated into the build process (each automated build spawns Static Code Analysis)

Dynamic Code Analysis (+)

Looks for unexpected application behavior within the interface

Dynamic analysis can happen multiple times during each iteration

Assurance to QA that code is ready for testing

Dynamic analysis can offer 24/7 monitoring

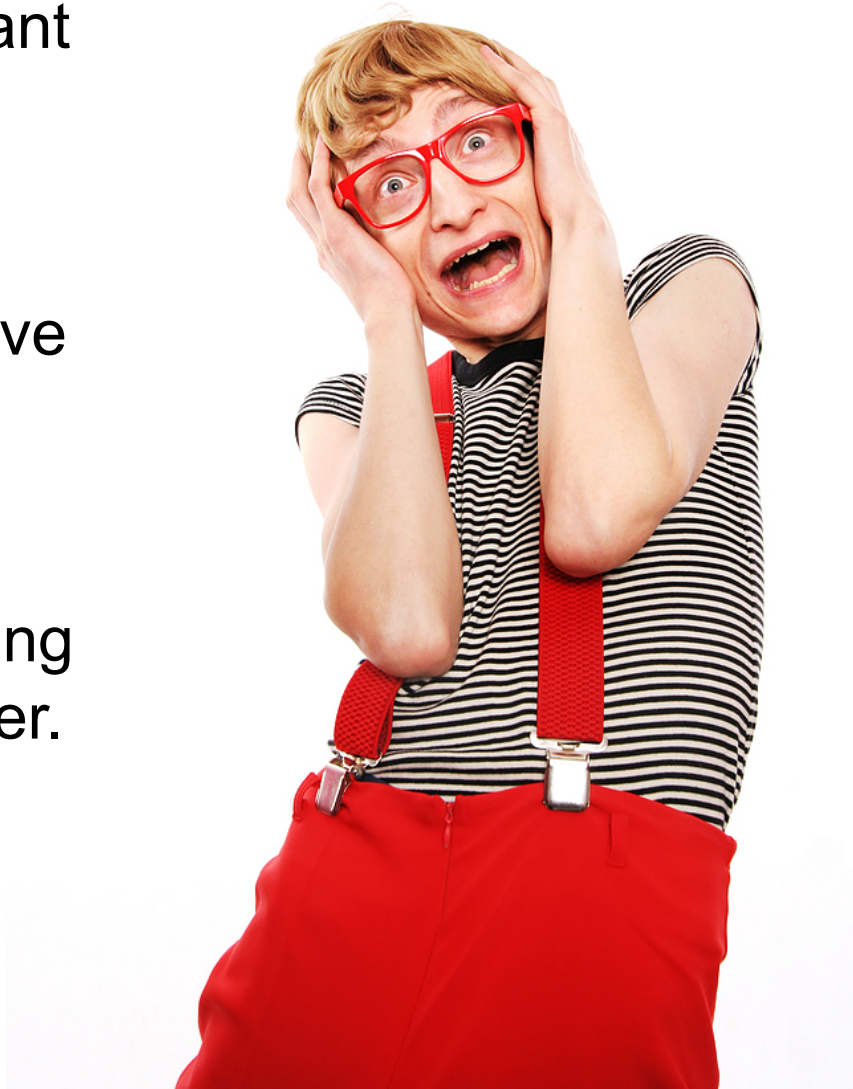
Be tied to incident management process

Giving Raw Scanner Reports to Developers (----------)

Most scanner reports have significant false positives

If you give a false positive laden report to developers who do not have deep security training then **please tell me what are you thinking**?

The first time you give a false reading to a developer you lose them forever.



Center of Excellence (++)

COE Steering Committee

COE Drivers

COE Members

Remove barriers between departments

Positively impact change

If developers have application security questions, where do they go?

Dashboards (++)

Monitor key application security behaviour

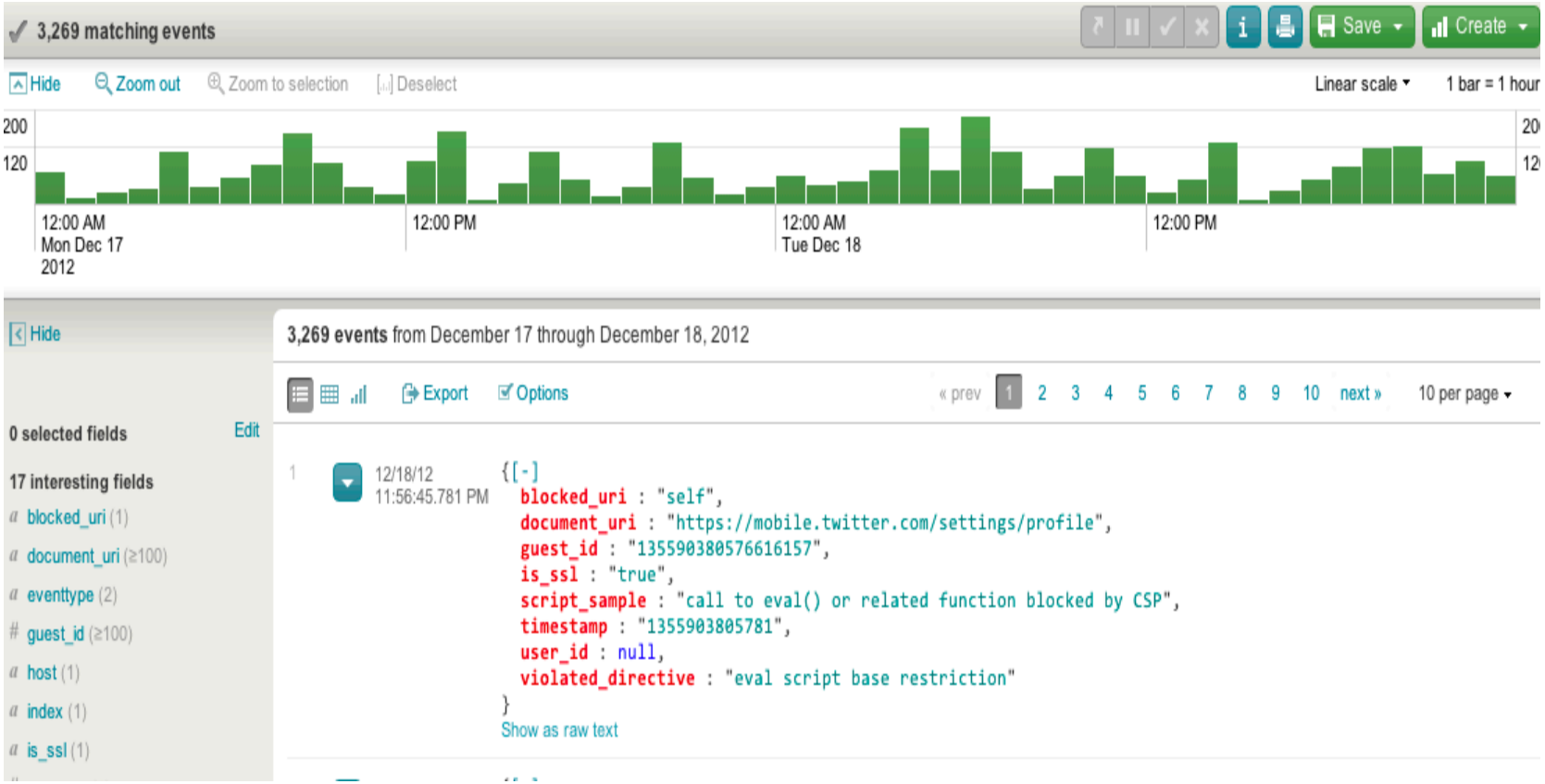
Build a framework where adding more monitoring points is easy to add

Build live dashboards for monitoring teams

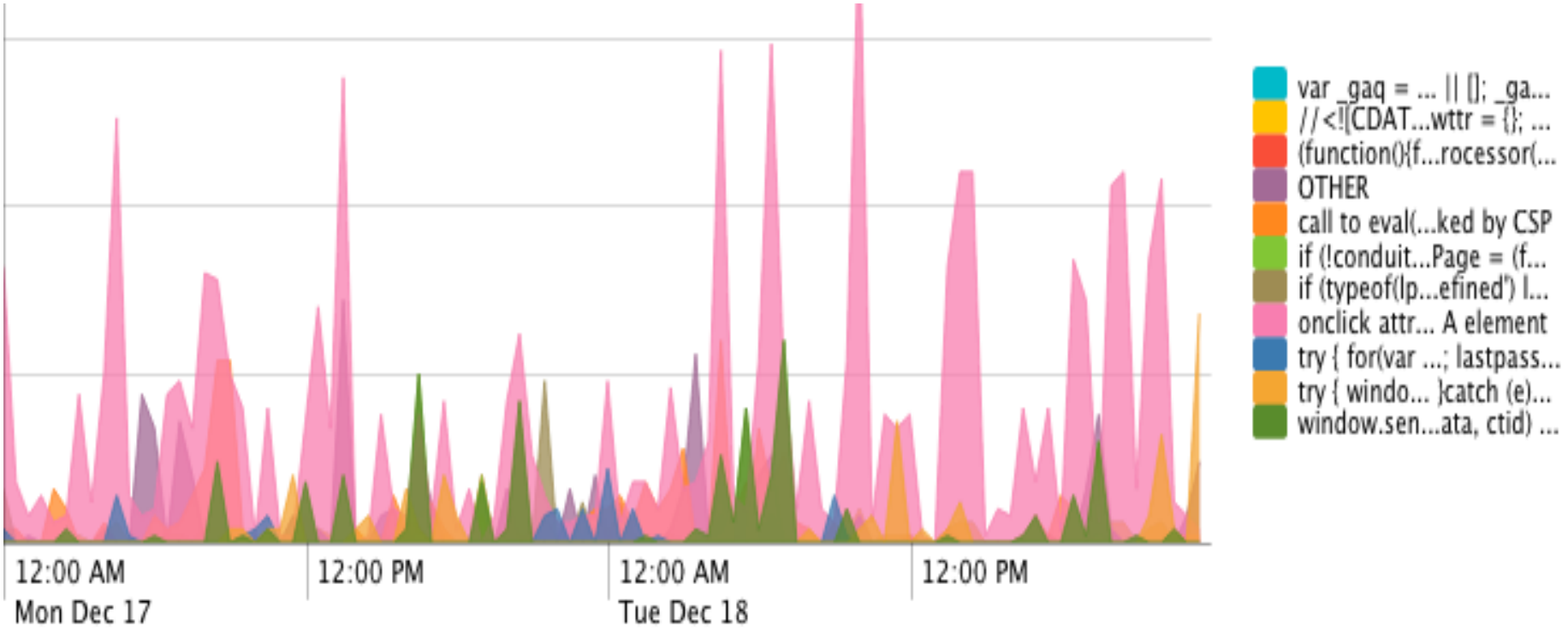
Detect **early** when anomolies occur

Monitor and Tune ALL the things

Splunk


3,269 matching events
Save
Create
Hide
Zoom out
Zoom to selection
Deselect
Linear scale
1 bar = 1 hour
200
120
12:00 AM
Mon Dec 17
2012
12:00 PM
12:00 AM
Tue Dec 18
12:00 PM
Hide
3,269 events from December 17 through December 18, 2012
Export
Options
« prev
1
2
3
4
5
6
7
8
9
10
next »
10 per page
0 selected fields
Edit
17 interesting fields
a blocked_uri (1)
a document_uri (≥100)
a eventtype (2)
guest_id (≥100)
a host (1)
a index (1)
a is_ssl (1)
1
12/18/12
11:56:45.781 PM
{[-]
blocked_uri : "self",
document_uri : "https://mobile.twitter.com/settings/profile",
guest_id : "135590380576616157",
is_ssl : "true",
script_sample : "call to eval() or related function blocked by CSP",
timestamp : "1355903805781",
user_id : null,
violated_directive : "eval script base restriction"
}
Show as raw text

Trending and anomalies



Tracking (++)

It's hard to know if you are getting better unless you can measure it

Track vulnerability data in some form of security CMS

Track trends. Be able to answer which teams and which applications are getting better or worse over time and adjust.

ThreadFix

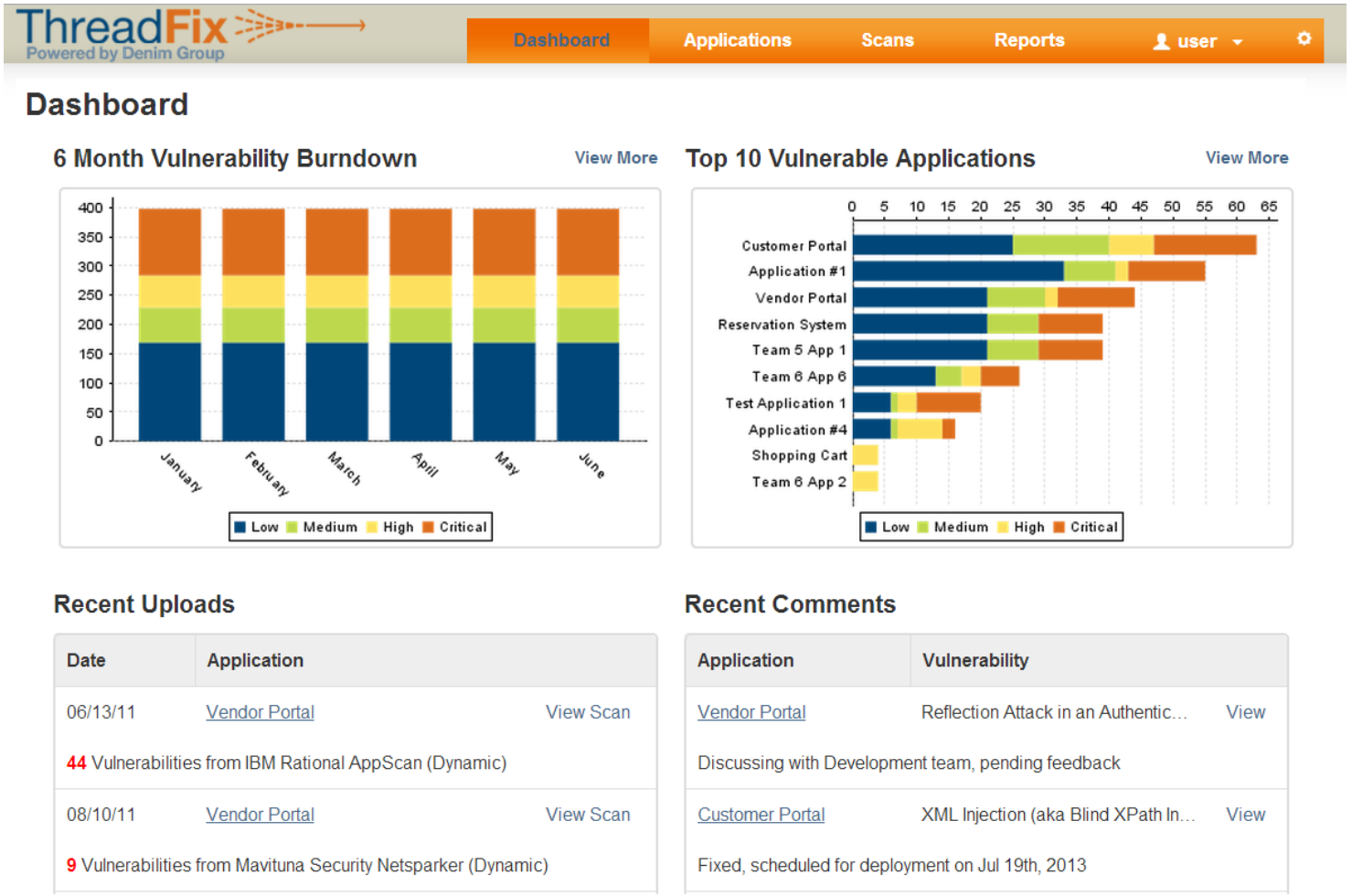
- An open source vulnerability management and aggregation platform that allows software security teams to reduce the time it takes to fix software vulnerabilities
- Freely available under the Mozilla Public License (MPL)
- Download available at: www.denimgroup.com/threadfix

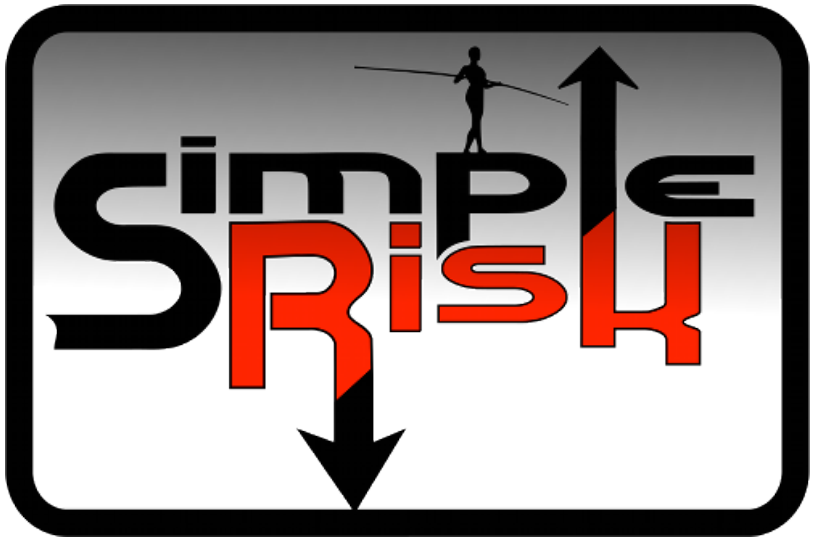
ThreadFix

Powered by Denim Group

Threadfix Risk Tracking

- **Heads up view of vulnerability burn down, most vulnerable applications, and recent files uploads and comments.**





- Mozilla Public License 2.0
- Open source PHP code
- Open source MySQL database

http://www.simplerisk.org

Define Your Risk Rating System

My Classic Risk Formula Is:

RISK = Likelihood x Impact

I consider HIGH risk to be anything greater than: 7

I consider MEDIUM risk to be less than above, but greater than: 4

I consider LOW risk to be less than above, but greater than: 0

Update

High Risk | Medium Risk | Low Risk | Irrelevant

Impact						
Extreme/Catastrophic	5	2	4	6	8	10
Major	4	1.6	3.2	4.8	6.4	8
Moderate	3	1.2	2.4	3.6	4.8	6
Minor	2	0.8	1.6	2.4	3.2	4
Insignificant	1	0.4	0.8	1.2	1.6	2
		1 Remote	2 Unlikely	3 Credible	4 Likely	5 Almost Certain

Likelihood

Plan Mitigations & Perform Reviews

Below is the list of submitted risks that require mitigation planning.

ID	Status	Subject	Risk	Submitted	Mitigation Planned	Management Review
1001	New	Unencrypted backup tapes are lost/stolen	3.2	2013-07-22 11:25:45	No	No
1002	Mgmt Reviewed	A train crash takes out our datacenters	2	2013-07-22 11:27:38	No	Yes

Below is the list of submitted risks that require a management review.

ID	Status	Subject	Risk	Submitted	Mitigation Planned	Management Review
1003	Mitigation Planned	Use of 2z Project Leads to SQL Injection exploit	4.6	2013-07-22 11:32:03	Yes	No
1001	New	Unencrypted backup tapes are lost/stolen	3.2	2013-07-22 11:25:45	No	No

ID	Status	Subject	Risk	Days Open	Next Review Date
1003	Mitigation Planned	Use of 2z Project Leads to SQL Injection exploit	4.6	3	UNREVIEWED
1001	New	Unencrypted backup tapes are lost/stolen	3.2	3	UNREVIEWED
1002	Mgmt Reviewed	A train crash takes out our datacenters	2	3	2014-07-17

Prioritize for Project Planning

1) Add and Remove Projects

Add and remove projects in order to associate multiple risks together for prioritization.

Add new project named [ ] Add

Delete current project named -- Delete

2) Add Unassigned Risks to Projects

Drag and drop unassigned risks marked for consideration as a project into the appropriate project tab.

Unassigned Risks | DR & BCP

A train crash takes out our datacenters

Save Risks to Projects

3) Prioritize Projects

Move projects around and change the order of prioritization. Once finished, don't forget to press the "Update" button to save your changes.

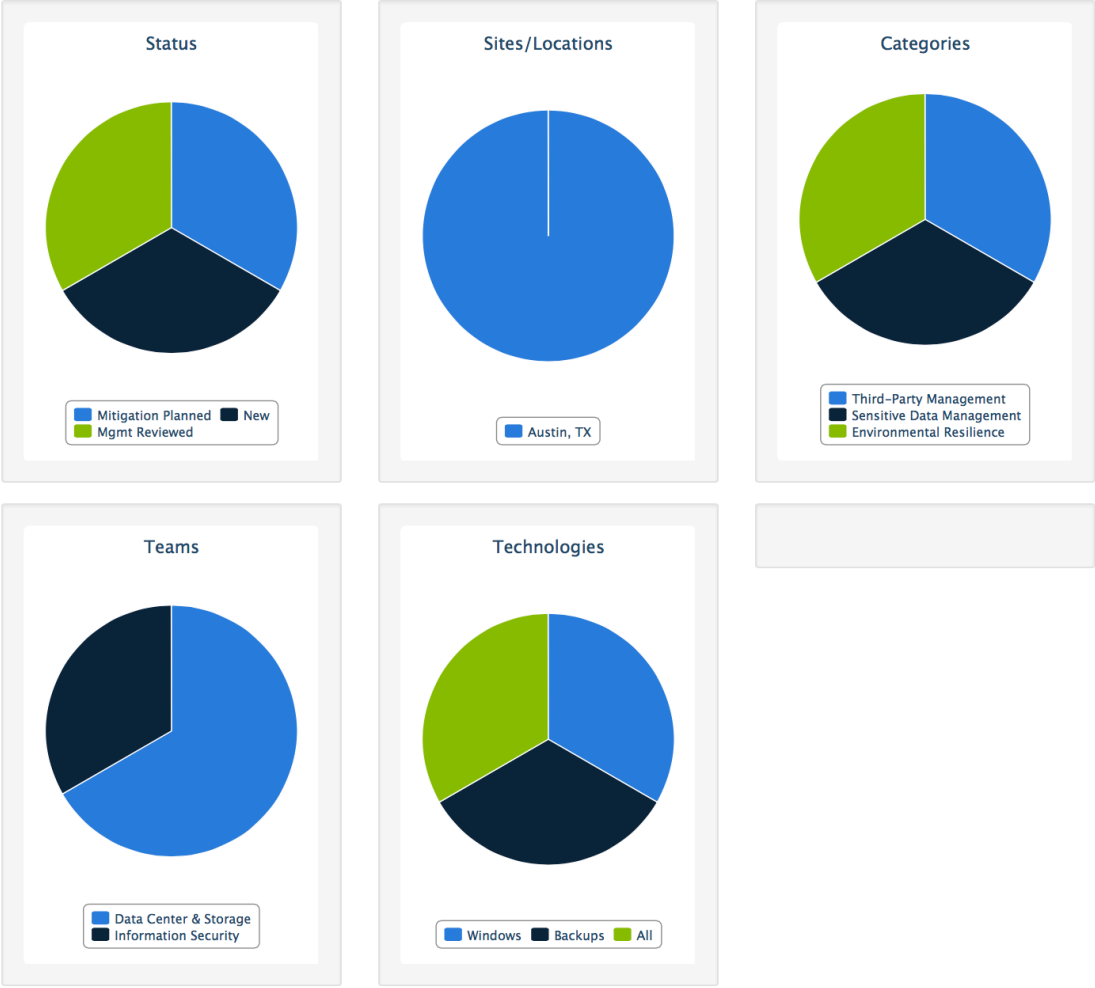
Update

⇅ DR & BCP

Update

Risk Dashboard & Reporting

Open Risks (3)



@manicode

jim@owasp.org

jim@manico.net