COMSEC Consulting
Cyber & Information Security

Delivering Security in Continuous Delivery Environment

OWASP AppSec Israel 2013

Yaniv Simsolo, CISSP

Agenda


Definitions and Background
Challenges
Secured Methodology Concepts
Best Practice Approach

COMSEC Consulting
Cyber & Information Security

Proper Disclosure

COMSEC Consulting
Cyber & Information Security

Definitions and Background

Continuous Delivery - Wikipedia

"Continuous Delivery (CD) is a pattern language used in software development to automate and improve the process of software delivery.

Techniques such as automated testing, continuous integration and continuous deployment allow software to be developed to a high standard and easily packaged and deployed to test environments, resulting in the ability to rapidly, reliably and repeatedly push out enhancements and bug fixes to customers at low risk and with minimal manual overhead.

The technique was one of the assumptions of extreme programming but at an enterprise level has developed into a discipline of its own..."

Continuous Delivery - Wikipedia

"Developers used to a long cycle time may need to change their mindset when working in a CD environment. It is important to understand that any code commit may be released to customers at any point"

Using Feature Toggle: "…The technique allows you to release a version of a product that has unfinished features. "

Continuous Delivery Vs. Agile

- Agile is less “continuous”
  - Sprints are numbered per version, dated and known.
  - Planed releases and security tests
  - Better control of additional or changed code

Traditional System Development & Security

- Traditionally: design, develop, deploy, sanity, QA, Security tests, fix & mitigate, release.
  - QA and Security tests begin after Code Freeze (CF)
- In CD environment, release must be as close to CF as possible.
  - Testing must start before CF.

Traditional System Development Vs. Continuous Delivery

- In CD, multiple code components may be unfinished right up to CF.

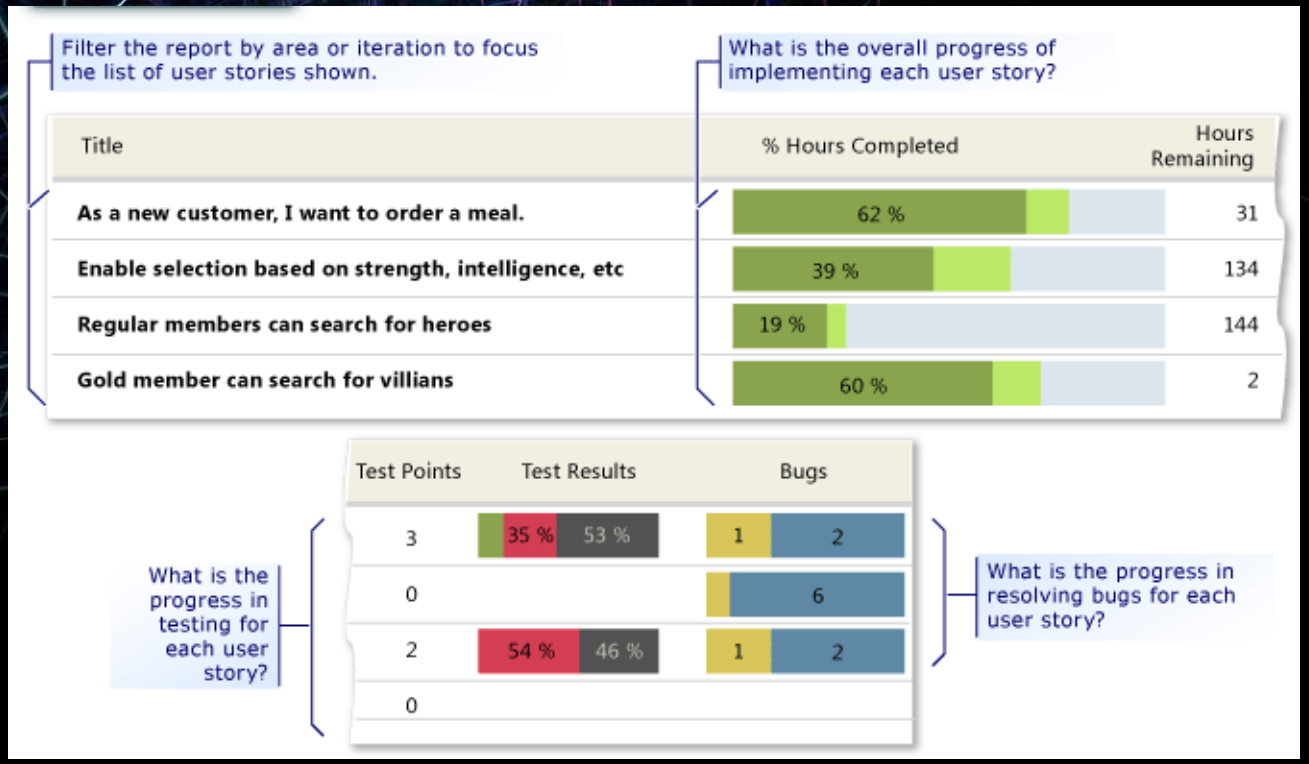
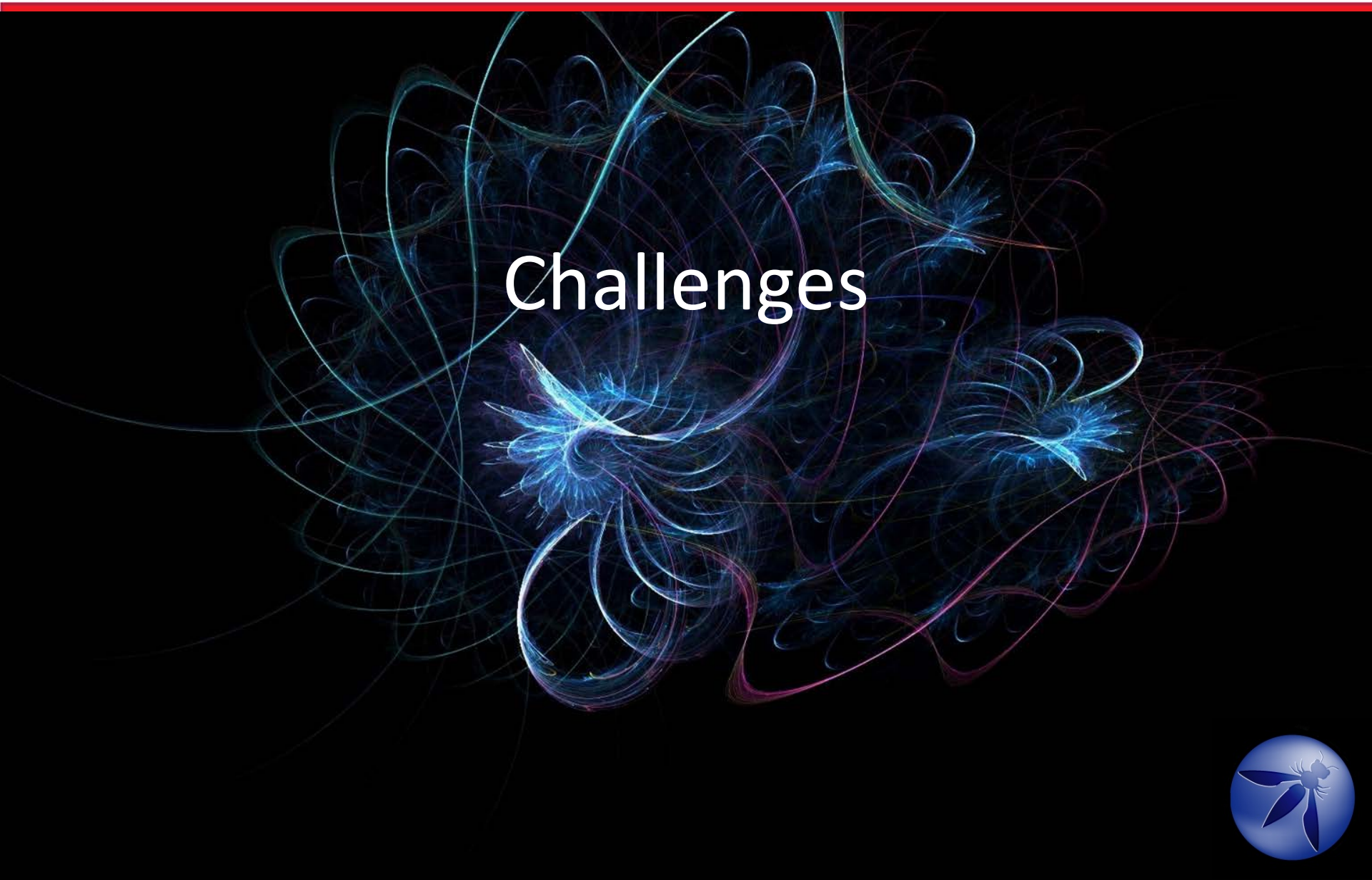


Image: msdn.microsoft.com

COMSEC Consulting
Cyber & Information Security

Challenges



The Time Factor

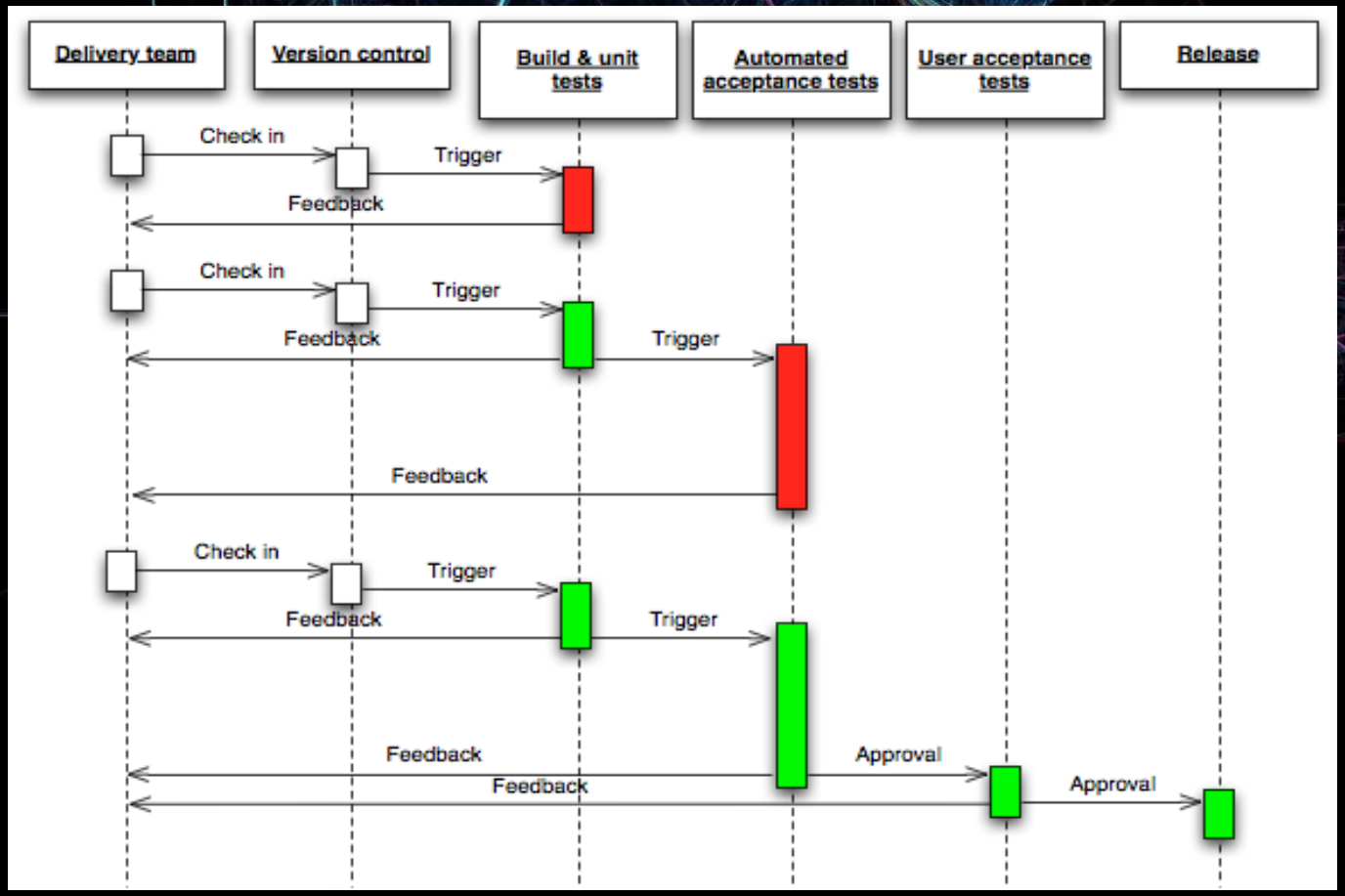
- Security testing a system consumes time
- Release is adjacent to CF

=>

- No timeframe for security testing
- No timeframe for fixing and mitigation

The Time Factor

- Continuous Delivery chart from Wikipedia - Where is the Security timeframe?



The Delta Factor

- Code is Continuously developed
- Wild West “Coding in all directions” is feasible

=>

- What is the Delta to test?
- Is progression of security feasible?

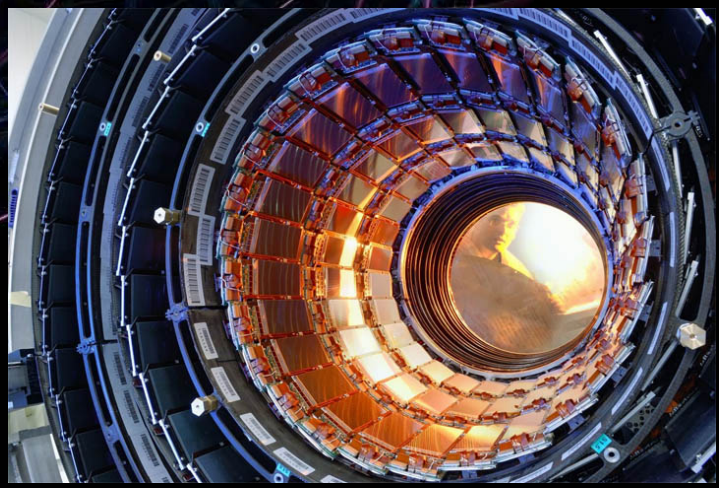


Image: CERN Photo Service

The Delta Factor

- Release Management is accurate to a degree
- Change Management is accurate to a degree

=>

- Extrapolating the Delta be from the RM/CM?
- Is progression of security feasible?

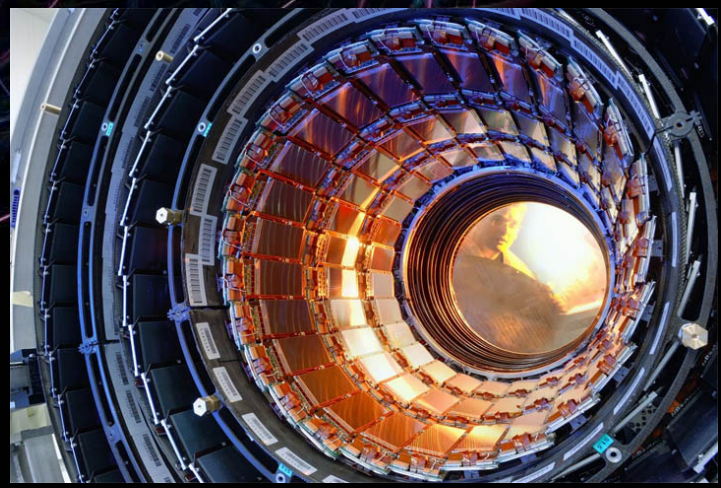


Image: CERN Photo Service

The Code Analysis Challenge

- Code analysis can be automated
- For each testing phase – test all (Delta included)
- Mitigation and fixing during normal development

=>

- Code Analysis does not identify all security bugs
- New technologies coverage
- Lack of Human touch, Schneier on Security, Sep. 1999: "The only way to find security flaws in a piece of code … is to evaluate it. This is true for all code, whether it is open source or proprietary. And you can't just have anyone evaluate the code, you need experts in security software evaluating the code. You need them evaluating it multiple times and from different angles…"

The Automation Challenge

- Security testing can be automated. For each development phase – perform automatic tests.
- Mitigation and fixing during normal development

=>

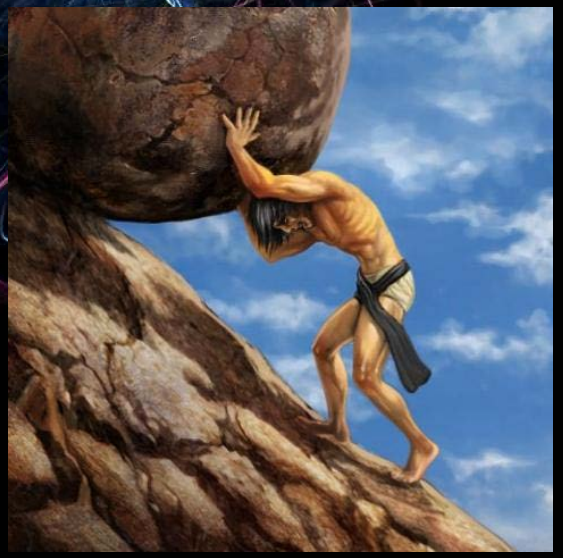
- Automatic Security testing does not identify all security bugs
- Automatic tools are lagging behind - new technologies coverage
- Turing Machine limitations - lack of Human touch

The Sisyphus Challenge

- Code is Continuously changing

=>

- Rendering all previous security tests irrelevant
- For each test phase – restart from square one!
- Right before CF, security retest!

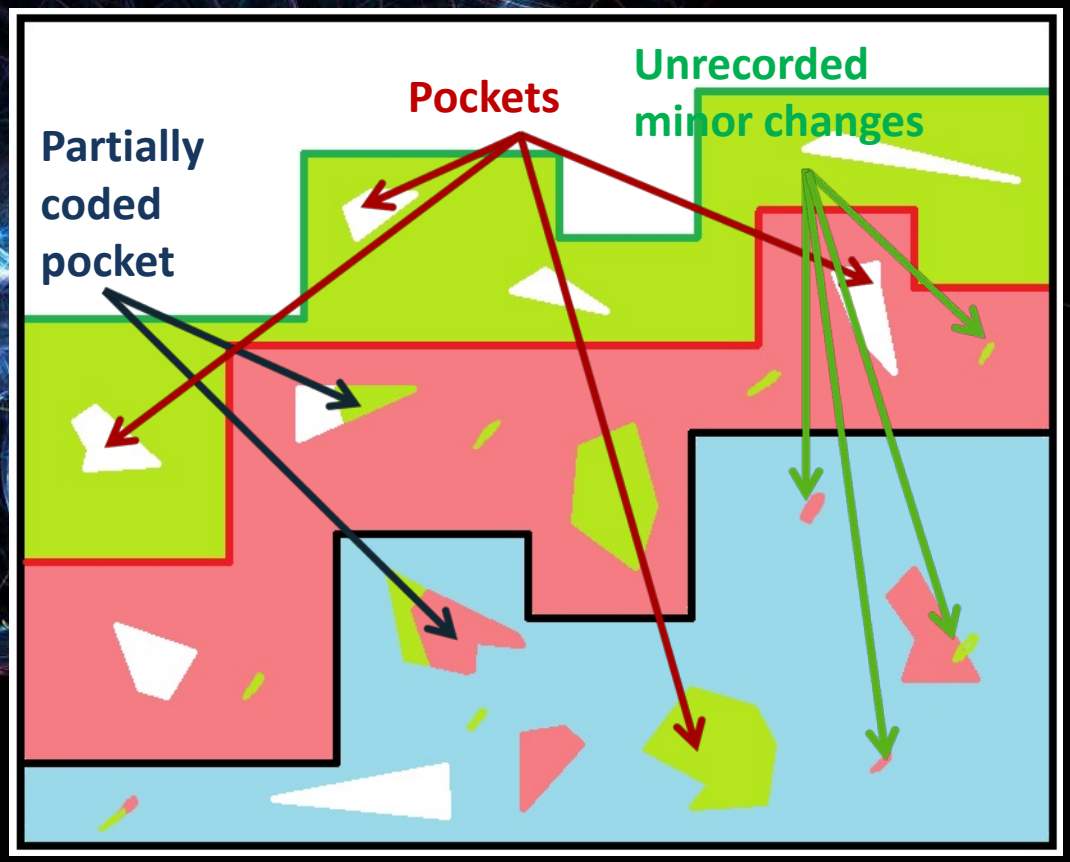


The Scoping Challenge

When comprehensive development controls are absent, accurate scoping is not feasible:

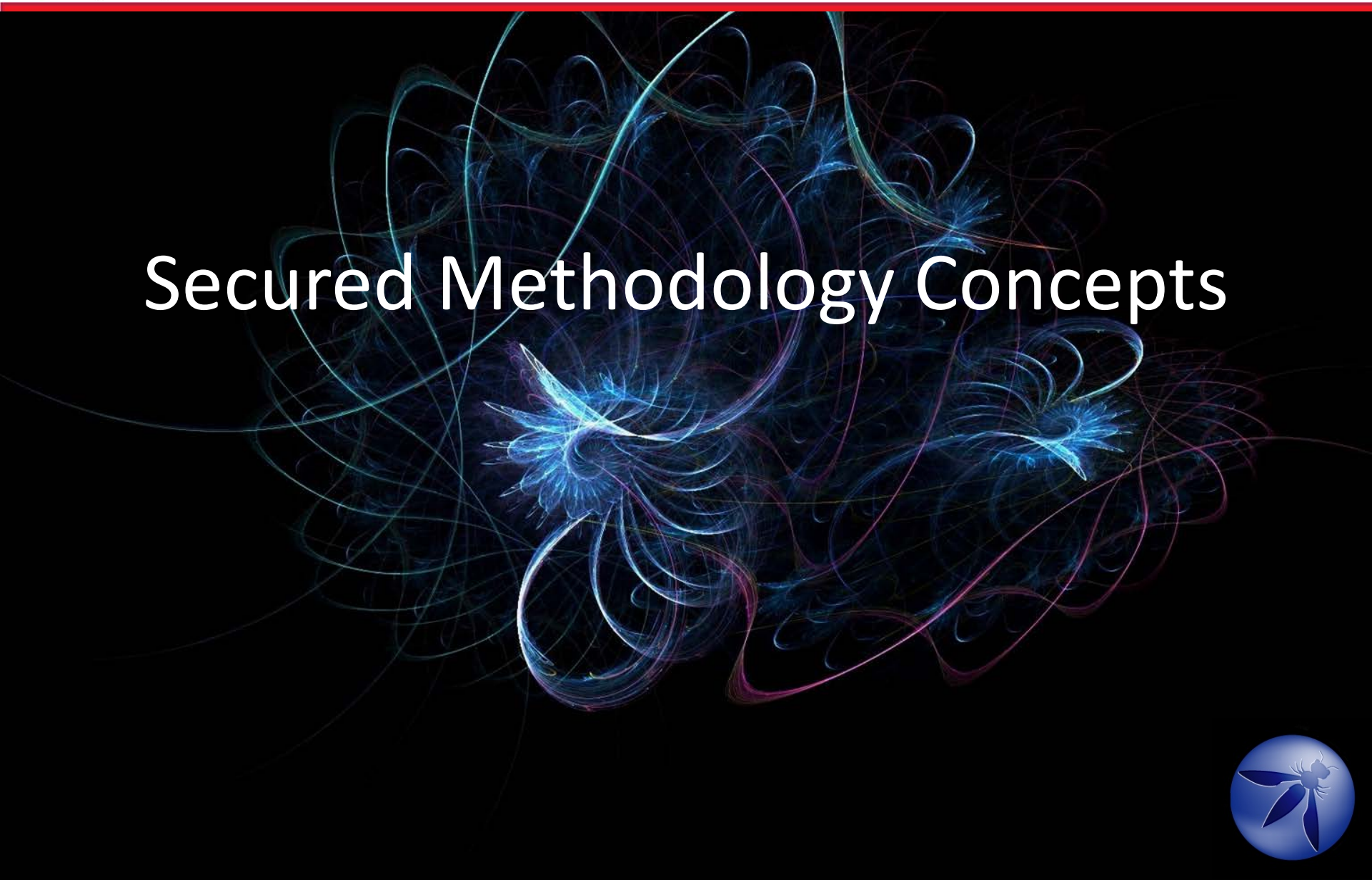
- Testing phase 1
- Testing phase 2
- Testing phase 3
-
- se n

3 colors trilogy, kishlovsky



COMSEC Consulting
Cyber & Information Security

Secured Methodology Concepts



Methodology Concepts

- Several different methodologies are available for approaching the challenges:
  - Partial sampled approach
  - Stepped testing
  - Resource intensive approach
  - Alternative approach

Partial Sampled Methodology

- Test the system partially continuously during development.
- Pros:
  - The partial Sampled approach is highly efficient
  - The “Randomized Algorithm” approach to testing the system is utilized
  - Fast, timely results
- Cons:
  - Partial coverage of the system’s security
  - Efficient?
  - Mitigation and fixing may not be feasible for CURRENT release
  - Mitigation and fixing costs after deployment

Stepped Testing Methodology

- For each development phase: perform a security test that is relevant only to said phase.
- Do not repeat tests.
- Do not retest tested areas.

Stepped Testing Methodology

- Pros:
  - Testing all system components is feasible
  - Timely results
  - Efficient
  - Reduced mitigation and fixing costs before deployment
- Cons:
  - High probability: Partial coverage of the system's security.
  - Accurate comprehensive RM and CM are mandatory
  - Mitigation and fixing may not be feasible for CURRENT release
  - Very large coding peaks in proximity to release

Resource Intensive Methodology

- Before Code Freeze: Perform iterations of the Partial Sampled approach or the Stepped approach.
- After Code Freeze: retest the system, full scope, in white box methodology.

Resource Intensive Methodology

- Pros:
  - COMPLETE coverage
  - Security improvement throughout development phases
  - Ignore RM and CM issues
- Cons:
  - Resource intensive. Duplicate testing for each release.
  - Partial coverage upon release
  - Mitigation and fixing may not be feasible for CURRENT OR NEXT release
  - Mitigation and fixing costs after deployment

Alternative Approach Methodology (The Sarcastic Approach)

- Do not security test the system
  - The CD development environment takes care of any malcoding
  - The QA processes detects all security bugs
  - Anyway: In-Depth Security is dead, Security is dead, hence no need for the extra efforts
  - The +GDP from moving to CD is so large, we can afford security bugs
  - Worst case scenario – mitigate for next release

COMSEC Consulting
Cyber & Information Security

Best Practice Approach

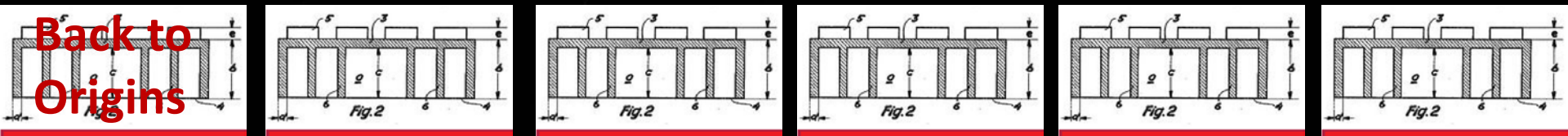
Best Practice Approach

- So, every approach has too many cons…
- Hence, a better approach is required

Best Practice Approach Building Blocks

- Back to the origins:
  - New technologies are evaluated out of band
  - Example, for Message Queue services: opt for the best secured of the available technologies: MQSeries, MSMQ, ActiveMQ, RabbitMQ.
    - One has almost no security for the past decades, other has no built in security configuration options.

Back to Origins

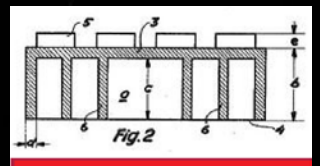


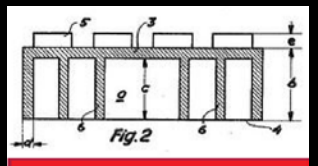
Best Practice Approach Building Blocks

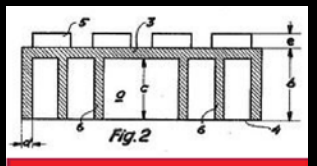
- The holistic approach :
  - Thorough design review of each system component
  - Assimilate secure coding practices
  - System testing of each component
    - Detailed design information extracted out of user stories.

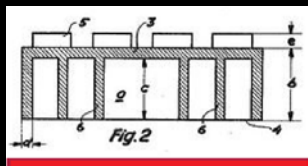
Back to Origins

Holistic Approach





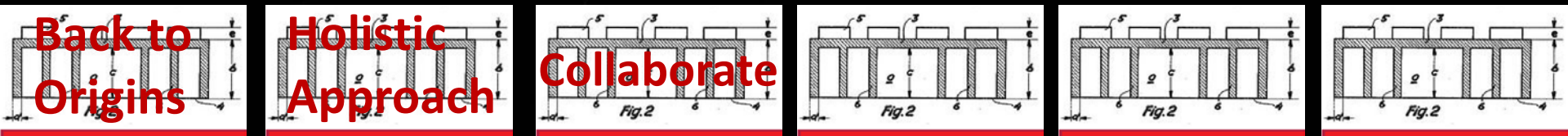




Best Practice Approach Building Blocks

- Collaboration with development teams
  - Essential
  - Create correlations and timely development per component.
  - Maximal effort in identifying Deltas
  - Minimize peak deliveries
- Collaboration level determines much of the **MINUS GDP** derived from security & cyber issues. (can the MINUS GDP be calculated?)

Back to Origins | Holistic Approach | Collaborate



Best Practice Approach Building Blocks

- Automate to the max:
  - Continuously run automated scanning tools on everything available
  - Continuously test the outermost layers (user interfaces)
  - Build automation tools to handle technology gaps
  - Continuously employ code analysis tools on identified Delta And tested code

Best Practice Approach Building Blocks

- Manual testing:
  - Efficient testing is mandatory
  - Concentrate on weak or sensitive spots
  - Seek & identify the shortcuts
  - Employ flexible A+ security experts

Best Practice Approach Building Blocks

- Operational Controls:
  - Not everything is inherently released when ready
  - Control of code versions
  - Control of secured configurations
  - Change management in PROD environments
  - Control the human "soft spot" … Kevin Mitnick: "I get hired to hack into computers now and sometimes it's actually easier than it was years ago"

Best Practice Approach - Conclusion

- New development environment requires new security testing methodologies.
- High level security testing under Continuous Delivery environment IS DOABLE.
- Cooperation with the development AND production teams is essential.

COMSEC Consulting
Cyber & Information Security

Questions?

Yaniv Simsolo, CISSP

Fractal Background: xaxor.com