



Risk Based Security Testing

Improving Your Test Strategy to
Expose Security Issues



cigital
Software Confidence. Achieved.

www.cigital.com
paco@cigital.com
+44 7985 419 802

March 2012 ©2012 Cigital, Inc. All Rights Reserved

Course Objectives

- At the end of this course, you will
 - Learn the role of security in the test strategy and test planning process
 - Map tests to risks in a traceability matrix
 - Identify security testing activities that fit into ordinary testing activities



March 2012 ©2012 Cigital, Inc. All Rights Reserved

Agenda

- Part 1: Software Security Testing
- Part 2: Starting Risk-Based Security Testing
- Part 3: Adding Risk-Based Security Testing
- Part 4: Conclusion



March 2012

©2012 Cigital, Inc. All Rights Reserved

Part 1 – Software Security Testing

- An overview of software security testing
- Don't blow up what you do, just build on it

March 2012

©2012 Cigital, Inc. All Rights Reserved

What is Software Security Testing?

- A risk-based, white-box approach to assessing software security
- Inputs are:
 - Business and design objectives
 - The actual requirements
 - Architectural and operational reality
 - The current and near-future capabilities of potential attackers (threat model)
 - The code
- Outputs are:
 - Evidence that software security risks introduced in the software development lifecycle have been effectively mitigated
 - Evidence that software does what it is supposed to do and nothing else
 - Evidence that the software will withstand malicious attack

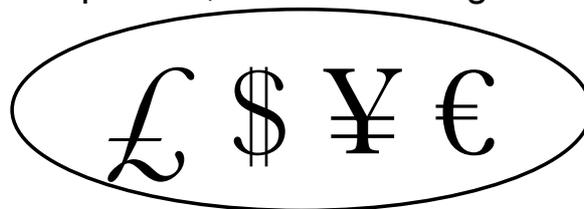


March 2012

©2012 Cigital, Inc. All Rights Reserved

What Are You Trying to Protect?

- The valuable properties of anything considered an asset
 - Data – CIA, privacy, accountability
 - Time – Launch delay, processing delay, etc.
 - Money – can't make sales, can't process transactions
 - Reputation and Brand – loss of trust
 - Legal – compliance, contractual regulation



March 2012

©2012 Cigital, Inc. All Rights Reserved

Two Broad Classes of Security Defects

Implementation Bugs

- Localised to specific bits of code
- SQL Injection
- Buffer overflow
- Cross-site scripting
- Unsafe system calls

Architectural Flaws

- Inappropriate trust of third party systems
- Session management
- Concurrency and transaction issues
- Broken or illogical access control (RBAC over tiers)



March 2012

©2012 Cigital, Inc. All Rights Reserved

Flavors of Software Security Testing

Functional Security Testing

- Test the security-related features of the system
- Ensure they behave in the prescribed manner (e.g., login features)

Risk-Based Security Testing

- Testing non-functional and negative requirements (misuse and abuse cases)
- Ensure security goals are met
- Ensure security risks introduced during software development have been effectively mitigated



March 2012

©2012 Cigital, Inc. All Rights Reserved

Black-box vs. White-box Testing

Black Box

- Treats the system as being opaque
- No knowledge of the internal structure
- Usually focuses on testing functional requirements

White Box

- Allows full internal knowledge
- Uses this knowledge to construct tests and test data
- Uses this knowledge to judge whether something is actually a flaw



March 2012

©2012 Cigital, Inc. All Rights Reserved

Testing Security Functionality

- Often the only type of security testing that QA organizations perform
- Based on written security requirements and associated application security features
- Remember, testing security functionality can be tricky!
 - Add negative test cases
 - Anticipate how attackers might exploit security features

Examples:

- When testing “encrypt file,” also test whether the encryption key is overwritten
- When testing that a “random number” is generated, also test how random it is
- When testing that “add read access” allows a user to read a file, also test whether it allows write access too



March 2012

©2012 Cigital, Inc. All Rights Reserved

Risk-Based Security Testing

- Testing focused on whether identified risks have been appropriately mitigated
 - Concentrate on what you're told “you can't do”
- Identified and prioritized risks come from
 - Architectural risk analysis – artifact analysis usually done by development security architects or external consulting groups
 - Abuse cases, attack patterns, and threat model
 - Informed red-teaming
- Risk-based security testing must use this information and plan, test, and help mitigate these risks



March 2012

©2012 Cigital, Inc. All Rights Reserved

Defining What Security Means for You

- Functional testing requires a definition of what the software must do
 - These are your requirements
- Security testing requires a definition of what “secure” is for your system
 - Allows us to test “secure” or “not secure”
 - These are your security requirements



March 2012

©2012 Cigital, Inc. All Rights Reserved

Non-functional Requirements

- Auditability
- Extensibility
- Maintainability
- Performance
- Portability
- Reliability
- Security
- Testability
- Usability
- etc.

Example Non-Functional Requirements

- The system shall run on Windows XP, Windows Vista, and MacOS X 10.5
- User logins will take at most 20 seconds from submitting credentials to seeing first screen.
- The system will require less than 10 Mbs network speed to handle 100 concurrent users.



March 2012

©2012 Cigital, Inc. All Rights Reserved

New and Old Vocabulary

- Functional security requirement
 - *A condition or capability needed in the system to control or limit the fulfillment of requirements*
- Non-functional security requirement
 - *A property of the system required to ensure fulfillment of requirements in the face of abuse or misuse*



March 2012

©2012 Cigital, Inc. All Rights Reserved

Security non-functional Requirements

- Audit logs shall be verbose enough to support forensics
 - *All account modification events shall be logged. The event log shall contain date, time, user, action, object, prior value, new value*
 - *Audit logs shall have integrity protection...*
- Application use of credit card data shall be PCI compliant. e.g. PCI 3.3:
 - *Mask Primary Account Number (PAN) when displayed (the first six and last four digits are the maximum number of digits to be displayed).*



March 2012

©2012 Cigital, Inc. All Rights Reserved

Deriving Security Requirements

- App Req 1: All accounts have passwords
- App Req 2: 3 bad attempts == account lock
- Implication: Bad guy can DoS the App
 - Try every account 3 times
 - All accounts locked
- Derived requirement:
 - Accounts should unlock after 5 minutes of no attempts
 - eBay attack



March 2012

©2012 Cigital, Inc. All Rights Reserved

Thinking backwards

- Think of abuse cases and misuse cases as “backward” use cases
- Consider grammatical negation
- Start with use cases
 - Think about what a system does
 - Continue at increasing levels of detail
- Once you know what a system does, look at it from the adversary's perspective.
 - How can they disrupt the system?
 - How can they profit from the system?



March 2012

©2012 Cigital, Inc. All Rights Reserved

Anticipating Attacks

- Scenario:
 1. Receive contact info via SMS
 2. Confirm acceptance with handset user
 3. Add contact to address book
- What are some example requirements?
- How about security requirements?



March 2012

©2012 Cigital, Inc. All Rights Reserved

Contact Info via SMS

- Verify standard format (e.g., VCF)
 - Verify required fields (if any)
- Reconstruct multiple SMS into single record
- (optional) Check for duplicate in address book

- It's still not good enough
 - What will a bad guy do?



March 2012

©2012 Cigital, Inc. All Rights Reserved

SMS-based Attacks

- Spam contacts (e.g. "for great deals, call...")
 - Display sender's info in confirmation
 - Allow immediate delete
- Field-based overflows
 - Check length on all strings before importing
 - Truncate long inputs
- Out of order attacks
 - Fragmentation, ordering
- Character-set attacks (UTF-8, UTF-16, etc.)
 - Coerce character sets
 - Discard unsupported letters



March 2012

©2012 Cigital, Inc. All Rights Reserved

Anti-requirements: a useful construct

- Requirements generally have the form:
*The system shall **[do something]** given **[inputs]***
- To develop an anti-requirement:
 - Categorize the possible outcomes
 - Rank in order of severity from perfect to worst
 - Define a threshold – what outcomes are unacceptable
 - Explore the inputs and determine the outcome associated with each
 - Determine which are acceptable and which are not
 - Associate each input and outcome
- This exploration of the requirements from an “anti” perspective allows you to design security requirements to address unacceptable outcomes from the code that implements a requirement



March 2012

©2012 Cigital, Inc. All Rights Reserved

An example of “anti-requirements”

*Requirement: The system **shall** produce a unique identifier valid for N days into the future **given** a time, an integer N, and a valid authorization token where $0 < N \leq 7$*

Consider Undesirable Outcomes

- Non-unique identifier produced
- Identifier with incorrect validity period
- etc.

Address undesirable outcomes in order of business impact

Formulate *Positive REQUIREMENTS* to mitigate unacceptable outcomes

Consider Inputs

- Time is negative
 $N \leq 0$
- $N > 7$ etc.
- N is non-numeric
etc.

Map inputs to outcomes

- Bad N = error
- Bad time = error
- Invalid auth = error
- error = invalidate session



March 2012

©2012 Cigital, Inc. All Rights Reserved

Recognizing Security Requirements

Bad examples:

- “Be secure”
- “Don't allow buffer overflows”

Slightly better

- “XYZ data should be cryptographically protected”
- “Strongly authenticate users”
- “Meet SOX regulatory guidelines for data protection”
- “Do not allow meta-characters in input fields”
- “Phone number fields only accept x, y, z...”

Pretty Good

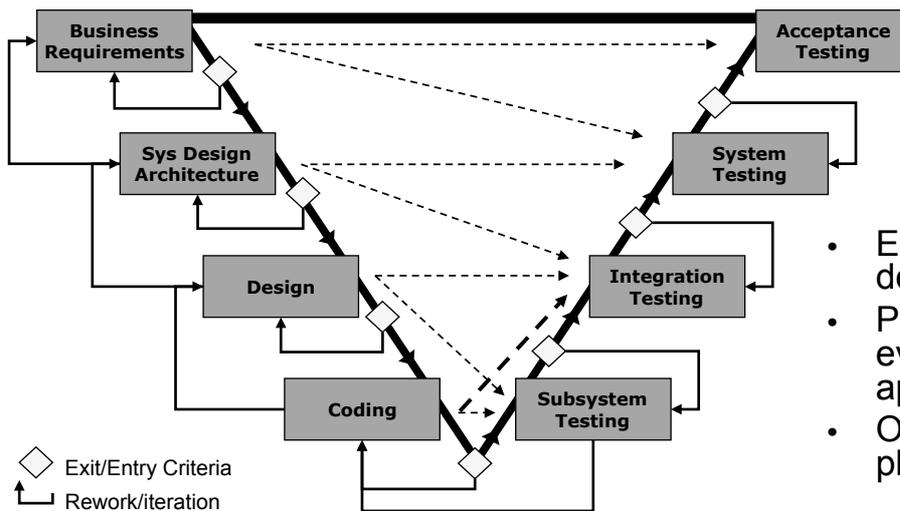
- “All user input fields shall be limited to 100 ASCII characters.”
- “Personally Identifiable Information will not be used as primary keys in databases.”



March 2012

©2012 Cigital, Inc. All Rights Reserved

Artifact Analysis



- Each phase reduces defects
- Provides continuous evaluation of application readiness
- Optimizes test planning

◇ Exit/Entry Criteria
 ↻ Rework/iteration



March 2012

©2012 Cigital, Inc. All Rights Reserved

Part 2: Getting Started with Risk-Based Security Testing

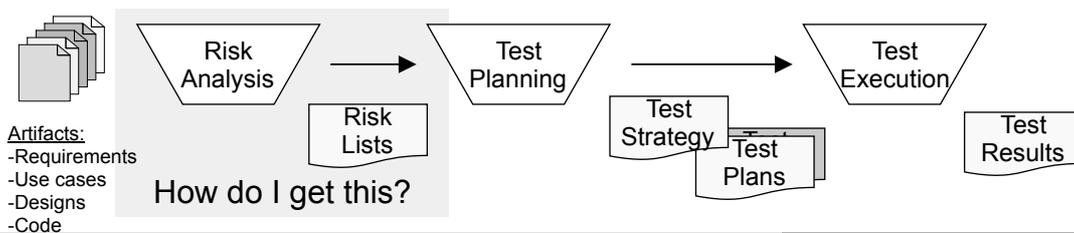
- The risk-based security testing process
- Assuming you're just getting started and you're basically on your own
- We'll cover “if you already have stuff” later

March 2012

©2012 Cigital, Inc. All Rights Reserved

Where Do I Start?

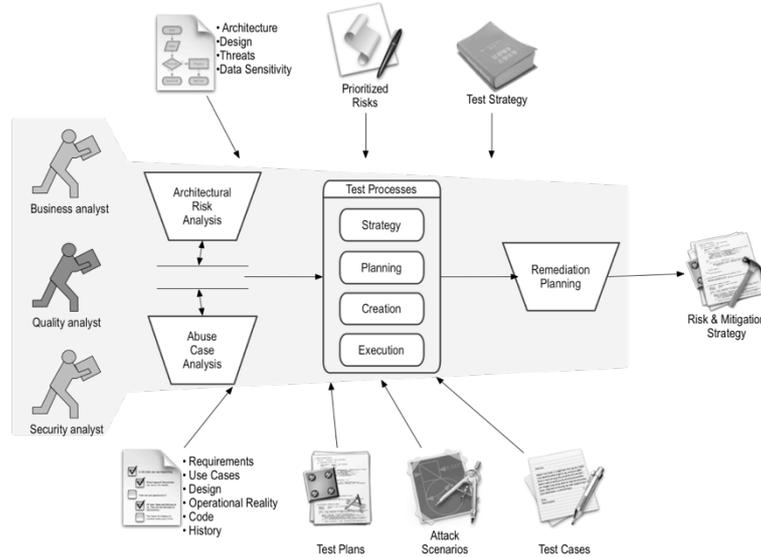
- The basic RBST process:
 1. Use a list of security risks
 - Pull from earlier activities
 - Bootstrap if necessary
 2. Build test plan and strategy
 3. Execute tests



March 2012

©2012 Cigital, Inc. All Rights Reserved

Risk-Based Security Testing Process



March 2012

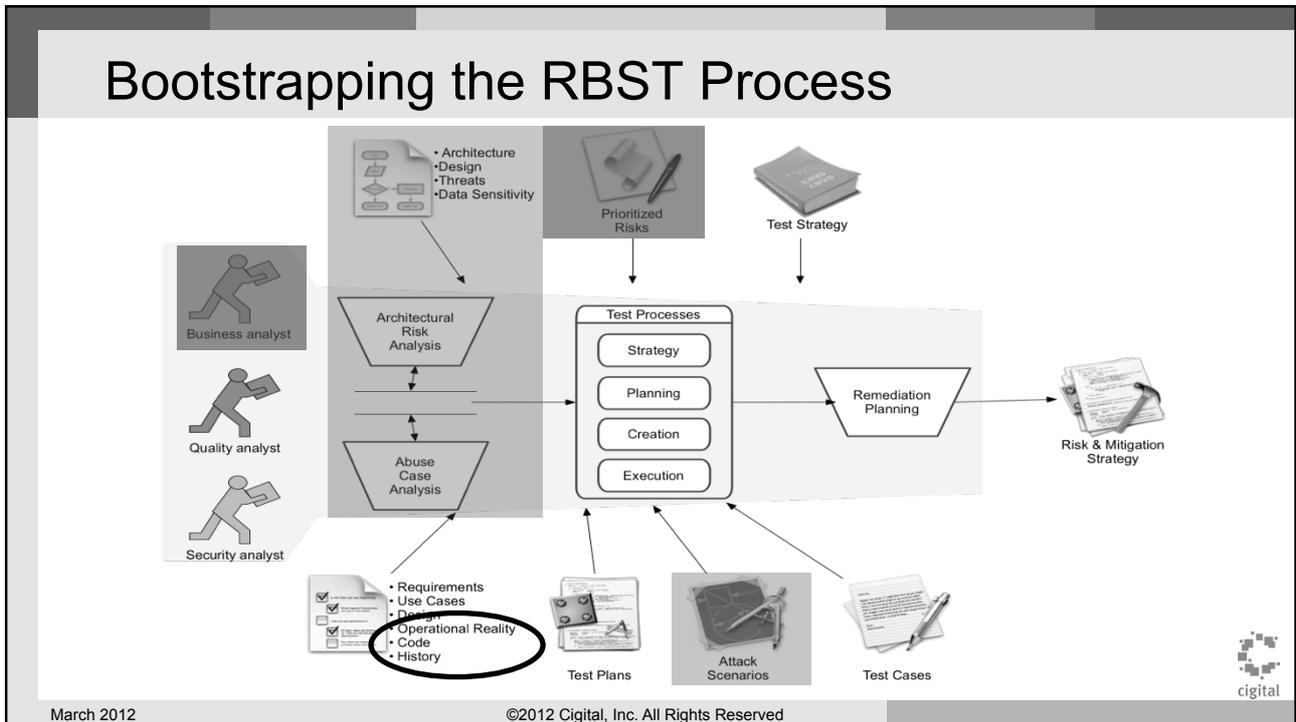
©2012 Cigital, Inc. All Rights Reserved

How To Get Started If You Have Nothing

- Learn from history
 - Use security goals to inspire test cases
 - Use guiding design principles to inspire test cases
 - Design tests to spot common vulnerabilities
 - Common test types and methods
 - Plan to classify identified defects

March 2012

©2012 Cigital, Inc. All Rights Reserved



Good Security Goals

<p>Traditional CIA</p> <p>Confidentiality</p> <ul style="list-style-type: none"> • limiting access and disclosure to “the right people;” • preventing access by or disclosure to “the wrong people” <p>Integrity</p> <ul style="list-style-type: none"> • the trustworthiness of information resources • Authenticity of the origin of information <p>Availability</p> <ul style="list-style-type: none"> • information systems provide access to authorized users 	<p>Additional Concepts</p> <ul style="list-style-type: none"> • Auditability / Accountability • Monitoring / Logging • Privacy • Non-repudiation
---	---

cigital

March 2012 ©2012 Cigital, Inc. All Rights Reserved

Guiding Principles for Secure Design

1. Secure the Weakest Link
2. Practice Defense in Depth
3. Fail Securely
4. Follow the Principle of Least Privilege
5. Compartmentalize
6. Keep It Simple
7. Promote Privacy
8. Remember that Hiding Secrets is Hard
9. Be Reluctant to Trust
10. Assume Nothing



March 2012

©2012 Cigital, Inc. All Rights Reserved

History of Common Mistakes

- Vulnerability Taxonomies
 - Common Vulnerability Enumeration – <http://cve.mitre.org>
 - Common Weakness Enumeration – <http://cwe.mitre.org>
 - United States Computer Emergency Readiness Team (US-CERT)
<http://www.us-cert.gov>
 - Open Web Application Security Project (OWASP) Top 10
http://www.owasp.org/index.php/OWASP_Top_Ten_Project
 - Seven Pernicious Kingdoms: *A Taxonomy of Software Security Errors* (McGraw, Tsipenyuk, Chess) - <http://www.fortify.com/vulncat>
 - 19 Deadly Sins of Software Security, (Howard, LeBlanc, Viega)
- Attack Patterns
 - Common Attack Pattern Enumeration and Classification - <http://capec.mitre.org>
 - Exploiting Software (Hoglund, McGraw)

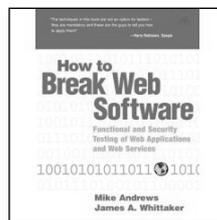
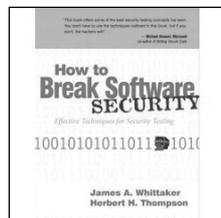
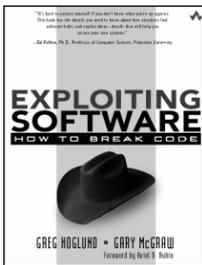


March 2012

©2012 Cigital, Inc. All Rights Reserved

Common Methods For Security Testing

- *Exploiting Software* (Hoglund, McGraw)
- *How To Break Software Security* (Whittaker, Thompson)
- *How To Break Web Software* (Andrews, Whittaker)
- *Web Security Testing Cookbook* (Hope, Walther)



March 2012

©2012 Cigital, Inc. All Rights Reserved



Online Security Mailing Lists

- **Bugtraq** - <http://www.securityfocus.com/archive/1>
 - **Full Disclosure** – <https://lists.grok.org.uk/mailman/listinfo/full-disclosure>
 - **Risks** – <http://www.risks.org>
 - **SC-L** – <http://www.securecoding.org/list>
 - **Security Tracker** – <http://www.securitytracker.com>
-
- Constantly changing horizon
 - Look up your own stuff!

March 2012

©2012 Cigital, Inc. All Rights Reserved



Example Risk Classifications

Security Clasifications

Technology

Will the software product do what it needs to do to fulfill the product requirements? Can we make the vision work?

Schedule

Can the vision be made to work within the market window?

Market

Does the market really want to buy the product? Can the product be sold such that the company is profitable?

Brand

Can the product fail in a way that damages the brand?

Compliance

Does a product failure lead specifically to statutory, regulatory, or other non-compliance

Business Classifications

Disclosure

The dissemination of information to an individual who does not have proper authorization.

Deception

Risks that involve unauthorized change and reception of malicious information stored on a computer system or data exchanged between computer systems.

Disruption

Where access to a computer system is intentionally blocked as a result of an attack or other malicious action. It is important to note that in some cases performance degradation can be as harmful as performance interruption.

Usurpation

Unauthorized access to system control functions.



Putting It All Together



All provide “hints” for thinking about security risks your software will be exposed to



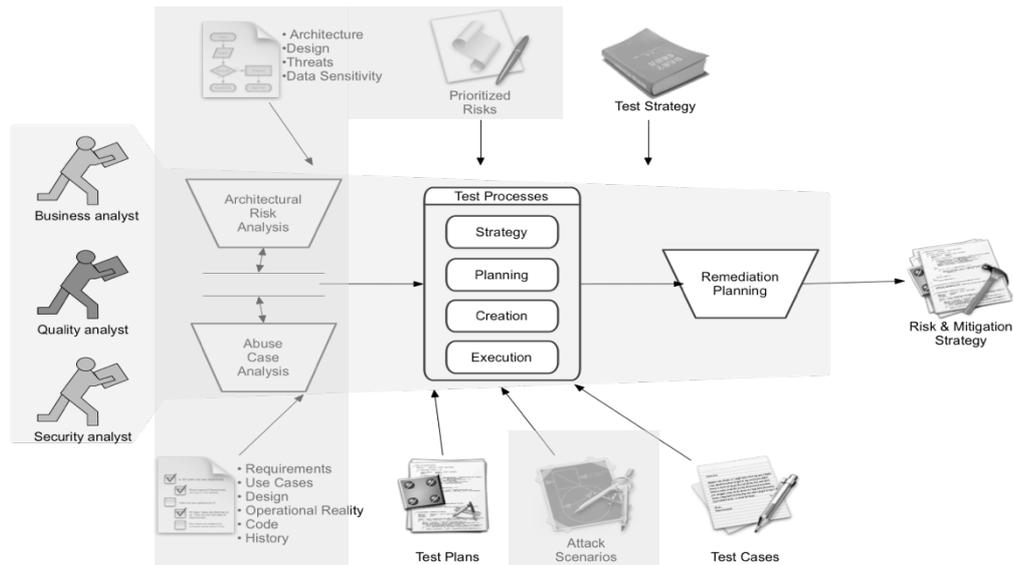
Part 3 – Adding Risk-Based Security Testing

- The risk-based security testing process
- Assuming you have some useful risk artifacts to start with

March 2012

©2012 Cigital, Inc. All Rights Reserved

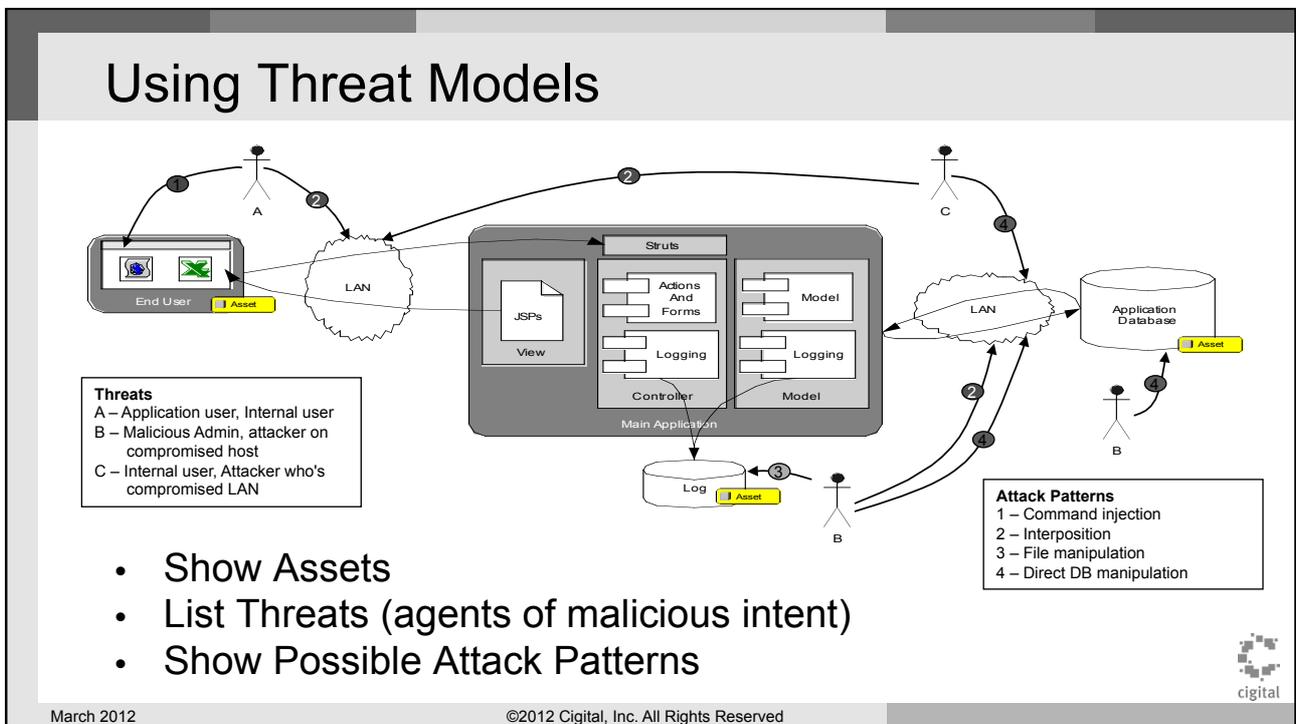
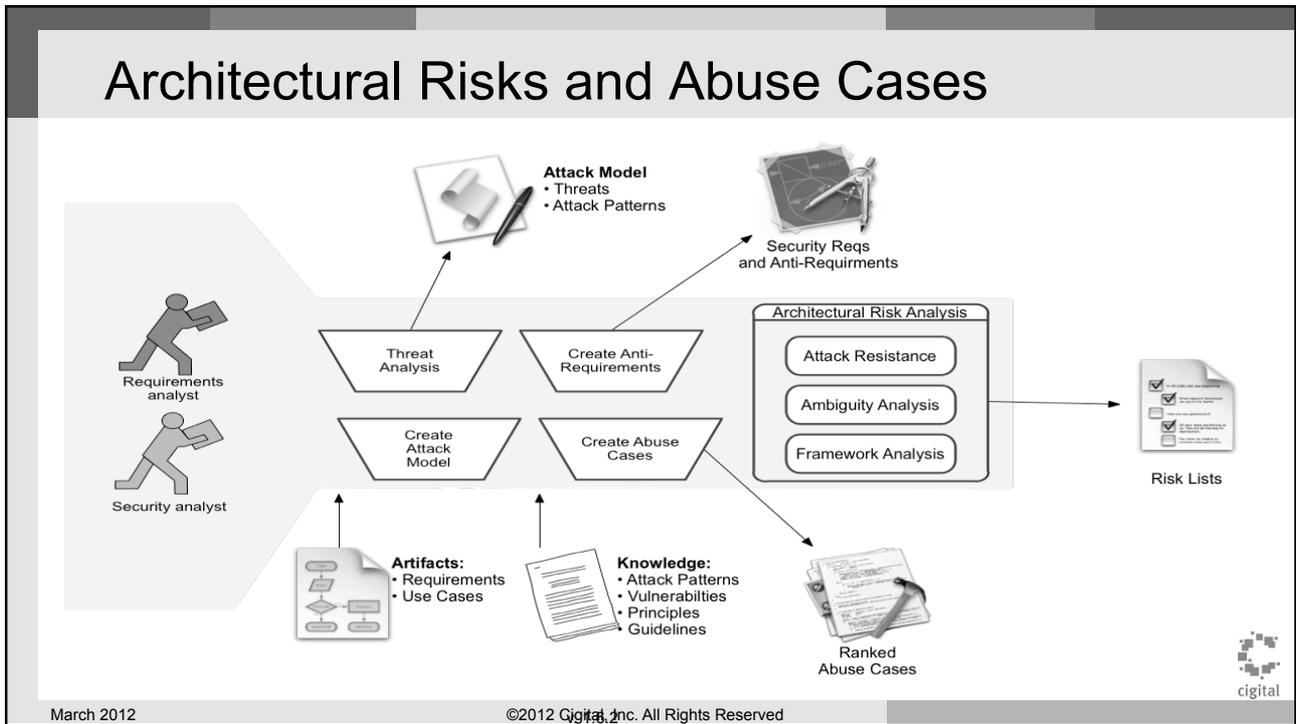
Integrating the RBST Process



March 2012

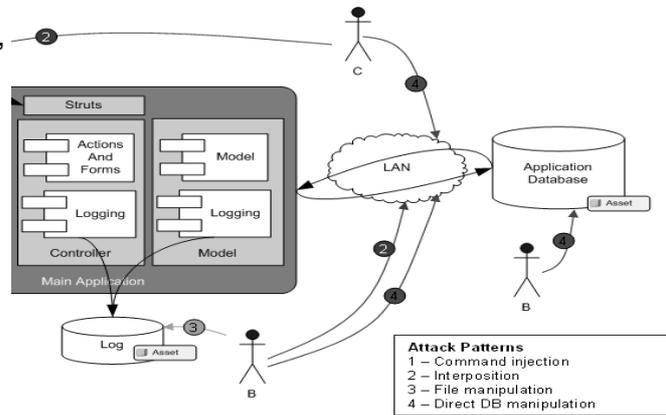
©2012 Cigital, Inc. All Rights Reserved





Using Architectural Risk Analysis Results

- Architectural Risk Analysis identifies, documents, and prioritizes possible defects in code, such as:
 - Misuse of cryptography
 - Compartmentalization problems in design
 - Lack of consistent input validation
 - Invalid assumptions of trust
 - Insecure or lack of auditing
 - Lack of authentication or session management on APIs
 - And so on
- Will likely be narrative text or block diagrams

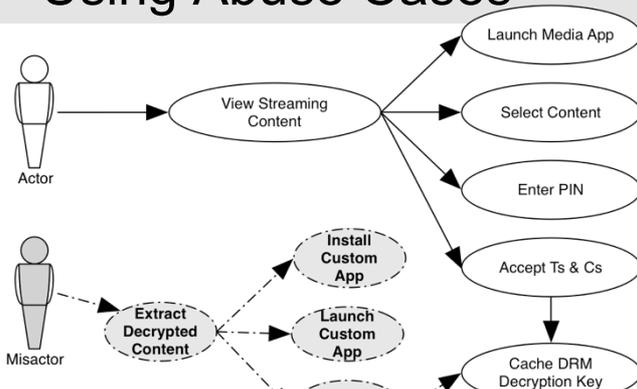


March 2012

©2012 Cigital, Inc. All Rights Reserved



Using Abuse Cases



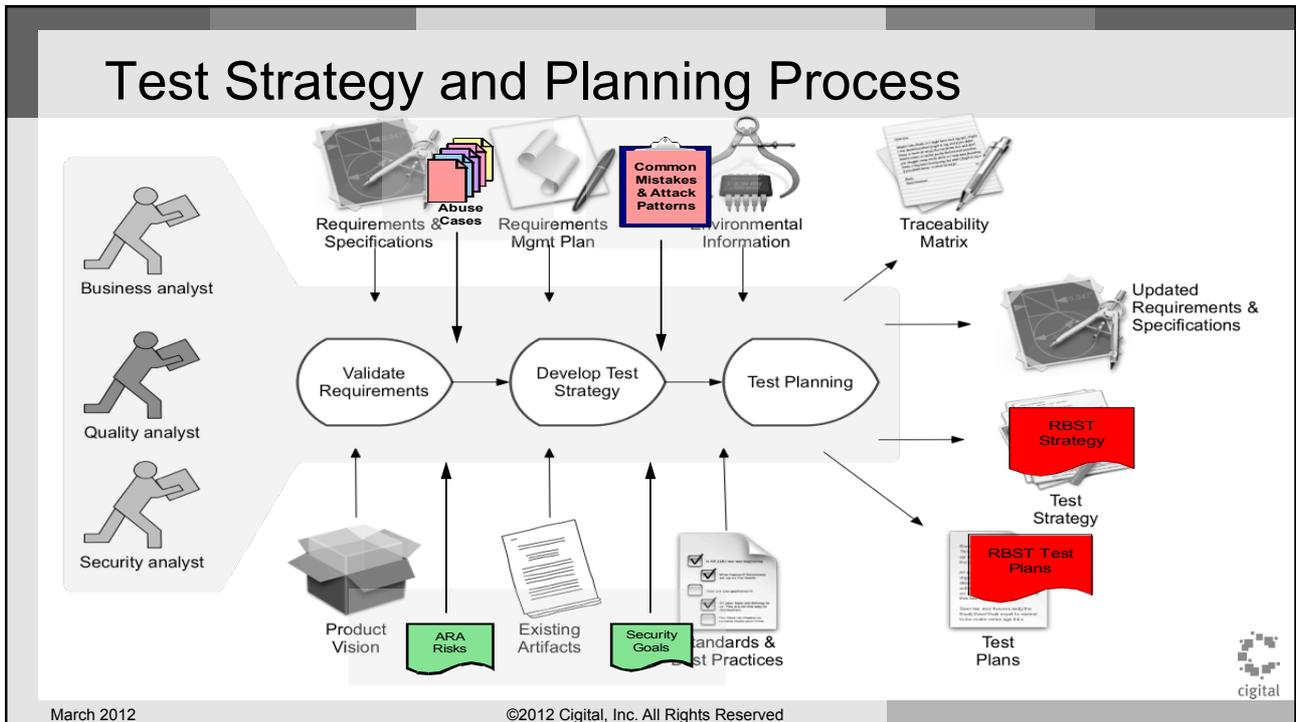
- May be simple diagrams
- May be narrative

		Details / How / What	Conditions	Protections
Delivery	1	Overflow filesystem path	Device uses filesystem and accepts file uploads	
	1.1	Attacker creates a file with filename > 255 chars	Attacker uses operating system that permits malicious filenames	None
	1.2	Attacker performs OBEX push from attack device (e.g., laptop)	Bluetooth is enabled, attacker's device is paired, or security is optional	Disable Bluetooth by default. Don't allow OBEX push from unpaired devices.

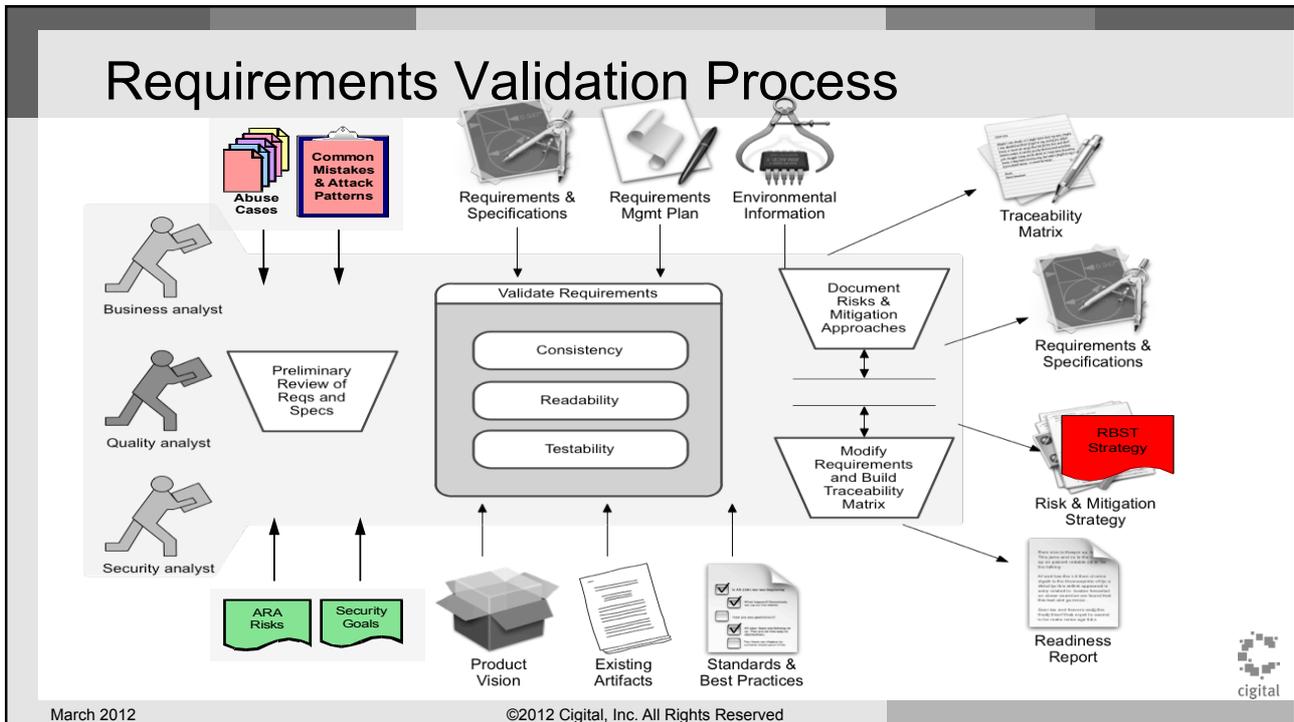
March 2012

©2012 Cigital, Inc. All Rights Reserved





- ## Expanding Test Strategy and Planning
- Test strategy and planning
 - A formalized approach to determining where, when, and how testing should be performed to maximize the impact of software testing
 - A phased approach that includes requirements validation, test strategy, and test planning
 - Collect new artifacts
 - Business and design objectives
 - ARA results, abuse cases, prioritized list of risks, code, etc.
 - Code component map, data flow diagrams, etc.
 - Choose additional testing as driven by risk
 - Basic security issues, security mechanisms, inter-component issues, abuse cases, misuse cases, failure checking, assumptions, design issues, other
 - Building on current testing strategy, identify additional code areas or properties that require testing
 - Augment the existing test plan
 - Build test cases the way you do now, but look at new things
- digital
- March 2012
- ©2012 Cigital, Inc. All Rights Reserved



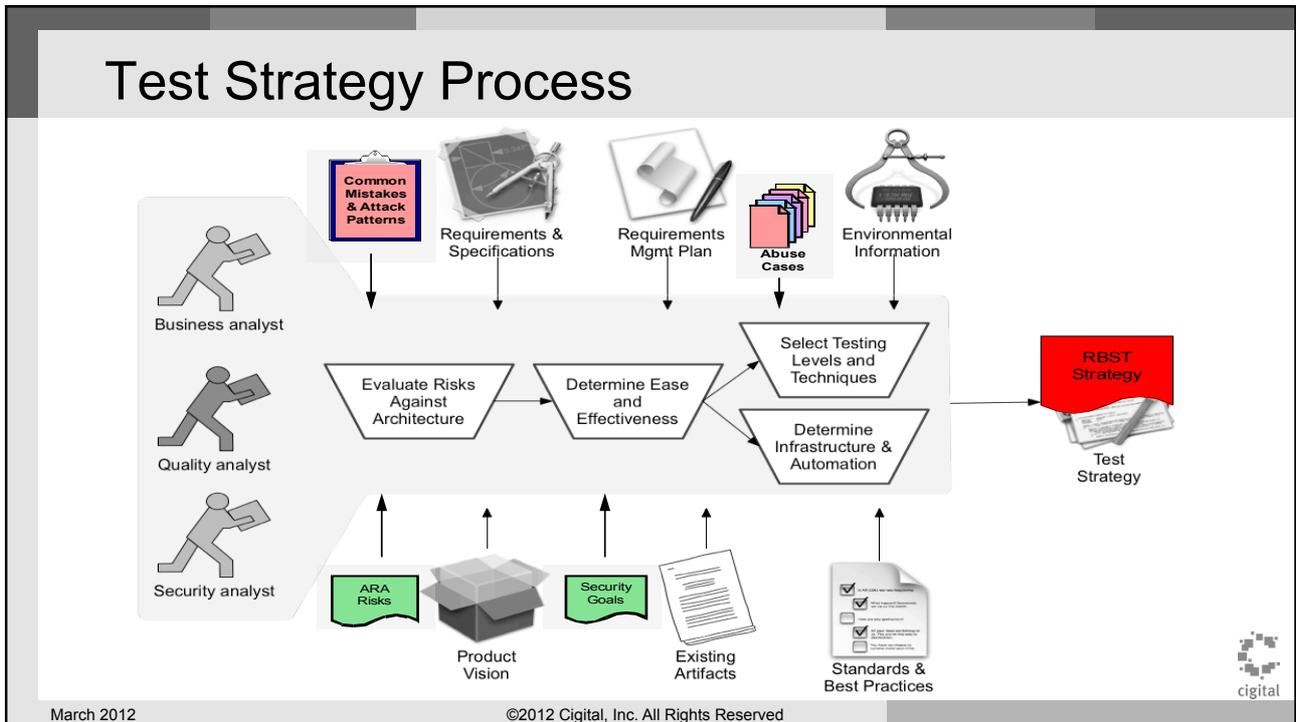
Requirements Validation

- “Do we have the requirements right?”
- Review checklist
 - **Consistency** – verify internal and external consistency between requirements, assumptions and interactions are consistent, and terms and concepts are used consistently
 - **Readability** – verify documentation is easily read and well formatted
 - **Testability** – verify that there is objective acceptance criteria for testing and each requirement is clear, concise, unambiguous
 - **Coverage**: ARA risks, threats, attack patterns, abuse cases, and so on
- **Account for security goals**

cigital

March 2012

©2012 Cigital, Inc. All Rights Reserved

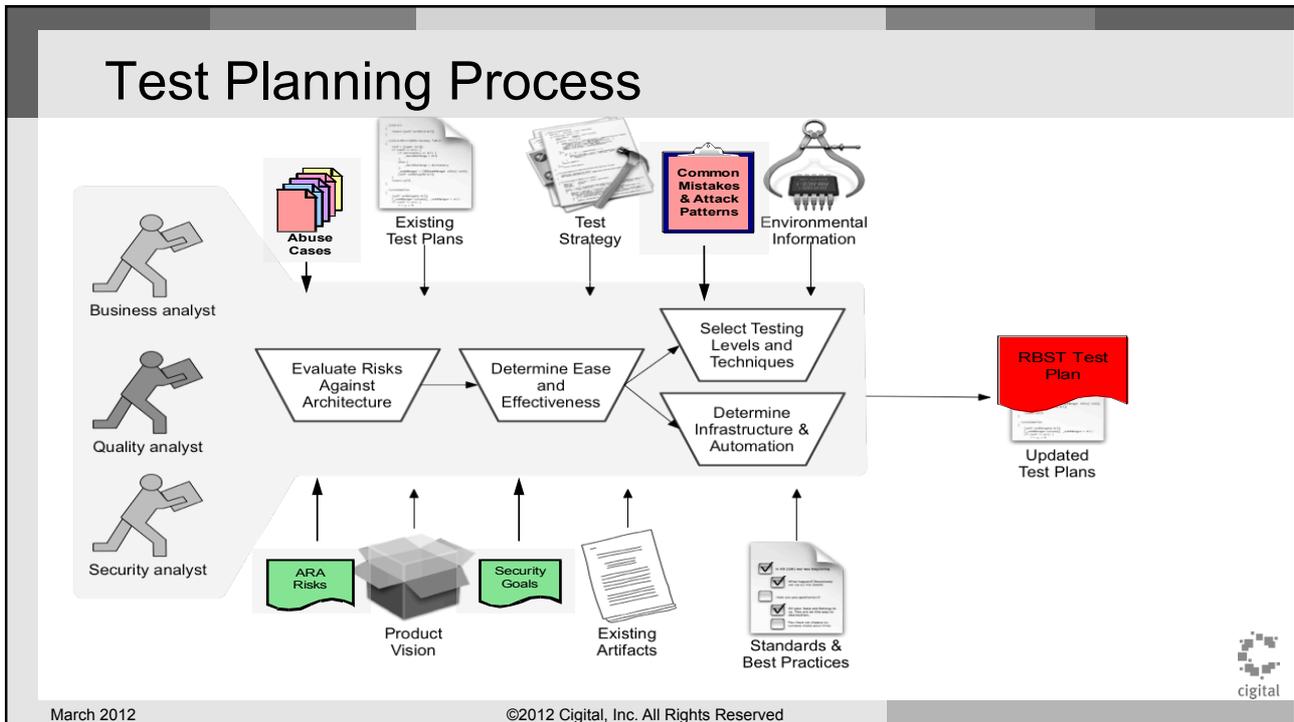


Test Strategy

- A solid test strategy drives an effective and efficient testing process
- Steps in the process
 - Understand application criticality and risks
 - Including ARA prioritized risks and abuse cases
 - Analyze the integrity level necessary for each system component
 - Identify the most effective testing techniques for mitigating the identified risks
 - Determine the acceptance criteria for each type of testing
- Test strategy content
 - Overall description of application to test
 - Identified business priorities/needs and associated risks
 - Definition of specific testing techniques that mitigate risks, meet objectives, and effectively test the application at appropriate test levels (subsystem, integration, system)
 - Associated exit criteria for test completeness based upon risk and coverage
 - Definition of test infrastructure necessary to effectively test
 - Overall test automation strategy

Account for risks and attacks

March 2012 ©2012 Cigital, Inc. All Rights Reserved



Test Planning

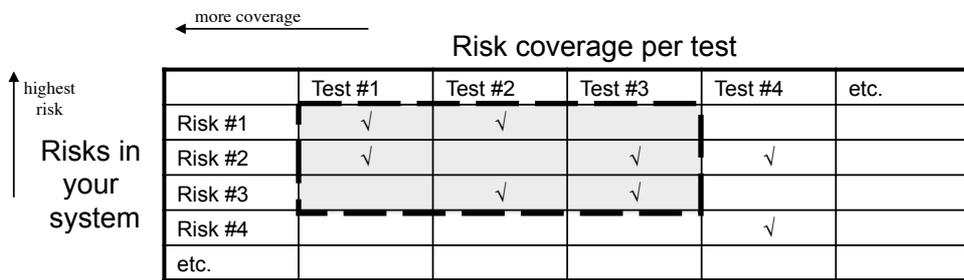
- Is driven by an overall test strategy
- Includes the following information:
 - Overall description of system and objectives
 - Test requirements and cases for each testing technique
 - Information on supporting test infrastructure
 - Information on supported test automation
 - Detailed exit criteria for each testing technique
 - Definition of test oracles for validating results
 - AND: The risks we are trying to validate
- Remember: Test setup, test validation, and test teardown are often effective areas on which to concentrate automation
- **Account for abuse cases and attack patterns**

cigital

March 2012 ©2012 Cigital, Inc. All Rights Reserved

What Are You Accomplishing?

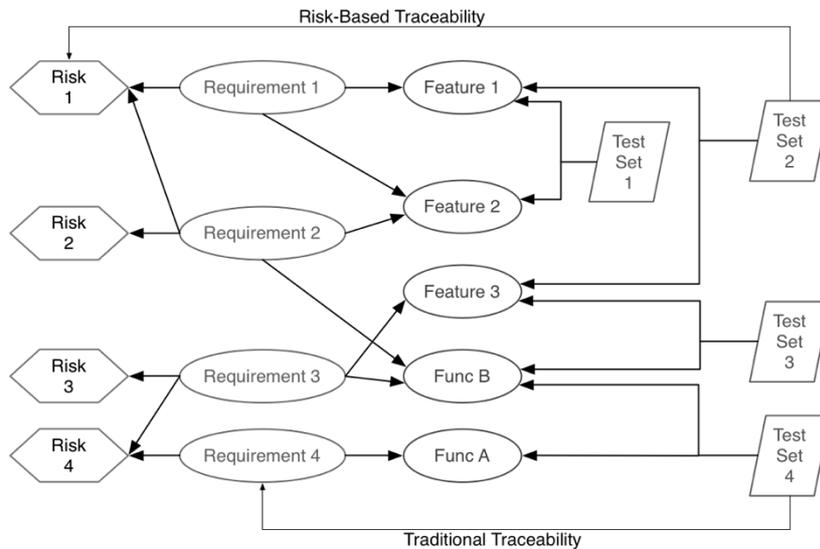
- We exposed you to goals, principles, common errors, test types, architectural risk analysis results, and abuse cases
- We talked about thinking like an attacker and open code, white-box testing
- We talked about the test planning process and how to take advantage of your new knowledge
- It was all so that you could draw that red box below – software security tests that address real risks in your specific code, prioritized by coverage and resources



March 2012

©2012 Cigital, Inc. All Rights Reserved

A New Kind of Traceability



March 2012

©2012 Cigital, Inc. All Rights Reserved

Who Does This?

- YOU!
- Functional security testing can be performed by developers and traditional QA staff
- Risk-based software security testing is performed by those with training
 - Thinking like an attacker
 - Crafting tests that may not result in an easily observable result
 - Crafting a series of tests, each relying on the results of a previous test
 - Follow the shiny object down the rat hole



March 2012

©2012 Cigital, Inc. All Rights Reserved

Think Like a Bad Guy, But Realize...

- Hackers have nothing but time to:
 - Crack expensive testing apps and use them for free
 - Tear apart your entire code base with a debugger, disassembler, or decompiler
 - Examine every register, environment setting, data structure, variable, API, timing, state transition, etc.
 - Crash the application a million times during fault injection or fuzzing to maybe get one useful result
 - Get five friends to help craft a test harness to try out some bright idea
 - Read every string in every binary
- You don't (have infinite time), so you have to:
 - Change your mindset, but be practical
 - Use internal (white-box) knowledge to stay ahead



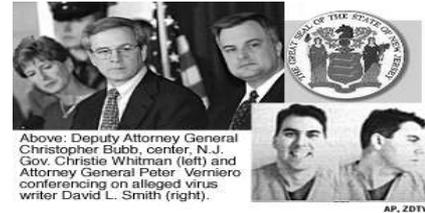
March 2012

©2012 Cigital, Inc. All Rights Reserved

Who Are These “Bad Guys”?

- Hackers
 - “Full disclosure” zealots
 - “Script kiddies”
- Criminals
 - Lone guns or organized
- Malicious insiders
- Competitors
- Police, press, terrorists, intelligence agencies

- Bad guys do not distinguish between bugs, flaws, defects, coding errors, configuration errors, security lapses, network vulnerabilities, or anything else



Above: Deputy Attorney General Christopher Bubb, center, N.J. Gov. Christie Whitman (left) and Attorney General Peter Vernofer conferencing on alleged virus writer David L. Smith (right).



March 2012

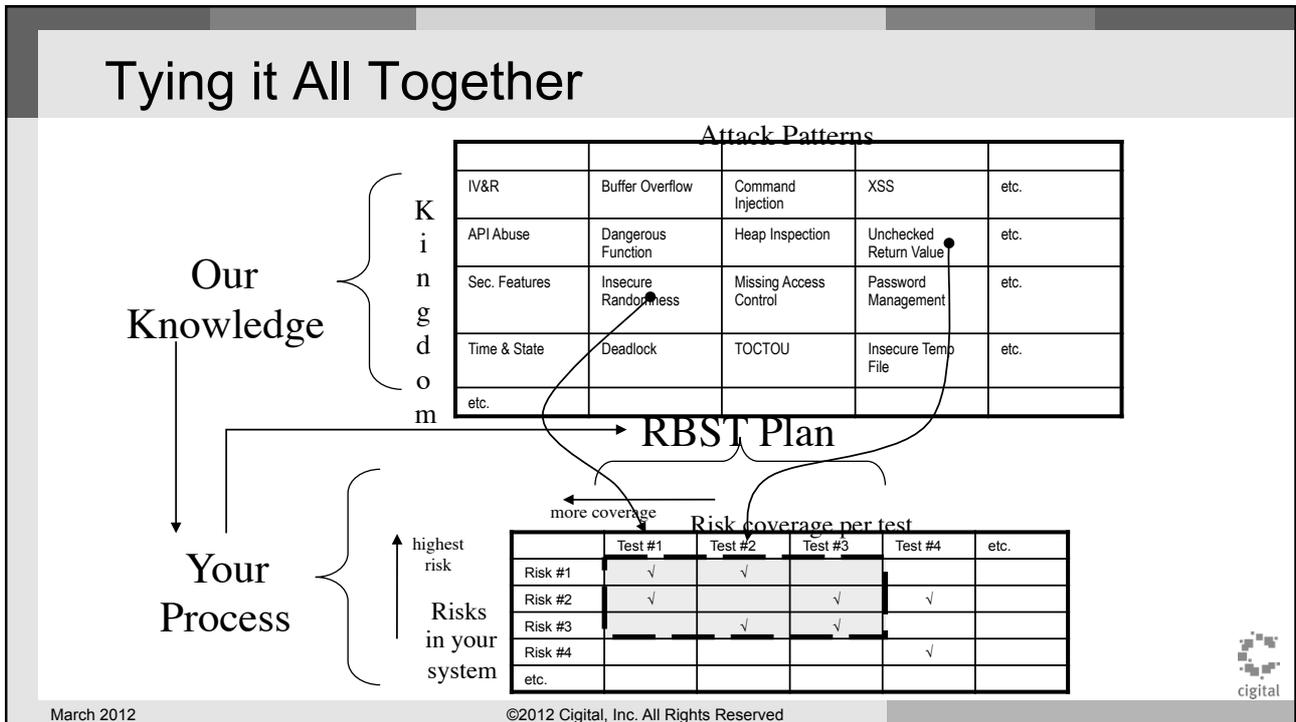
©2012 Cigital, Inc. All Rights Reserved

Part 4 – Conclusion

- What have we learned and what do we do now
 - Bring this knowledge home and help it stick

March 2012

©2012 Cigital, Inc. All Rights Reserved



- ## Challenges in Adopting Software Security Testing
- Software security testing is most effectively performed by QA as part of unit and integration testing, but
 - **People** – may not have the baseline understanding of security risks required to make testing effective
 - **Process** – may not include important steps necessary for determining software risk and security threats as part of the strategy and planning process
 - **Technology** – may not be familiar with or trained on software security testing tools
 - **Integration** – software development and security organizations may not have mechanisms in place to provide QA with the necessary risk information
- March 2012 ©2012 Cigital, Inc. All Rights Reserved

Know There Is Always More To Test

- The proverbial “Hello, World” J2ME application could easily be running in, on, and around 50 million lines of framework, operating system, firmware, and related software
 - And the security posture of your code is likely critically dependent upon all of it in one way or another
 - Expand your testing over time to account for interactions and data flows with other components



March 2012

©2012 Cigital, Inc. All Rights Reserved

Resources

Books

<i>Software Security</i>	McGraw
<i>Exploiting Software</i>	Hoglund and McGraw
<i>Building Secure Software</i>	Viega and McGraw
<i>How to Break Web Software</i>	Andrews and Whittaker
<i>How to Break Software Security</i>	Whittaker and Thompson
<i>Exploiting Software</i>	Hoglund, McGraw
<i>Shellcoder's Handbook</i>	Koziol, Litchfield, Aitel

Web Sites

Security Tracker	http://www.SecurityTracker.com/
Risks Digest	http://www.risks.org/
Phrack	http://www.phrack.org/
Full Disclosure	http://archives.neohapsis.com/archives/fulldisclosure/
US CERT	http://www.us-cert.gov/
OWASP	http://www.owasp.org/
Build Security In	https://buildsecurityin.us-cert.gov/daisy/bsi/home.html
Bugtraq	http://www.securityfocus.com/archive/1

Mailing Lists

Secure Coding <http://www.securecoding.org/list>

Motorola Courses

SCC1360	Introduction to Produce Security
SCC1327	Secure Programming
TSS009	Product Based Security Defense



March 2012

©2012 Cigital, Inc. All Rights Reserved



The best time to plant an oak tree was twenty years ago.



The next best time is now.
—Ancient Proverb



cigital
Software Confidence. Achieved.

Paco Hope <paco@cigital.com>

March 2012 ©2012 Cigital, Inc. All Rights Reserved