Secure Requirements

Brenda Larcom <asparagi@hhhh.org>
With significant contributions from Erik Simmons (Intel) & Eleanor Saitta (ISC Project)

Finse Winter School 2014-05-09

BISHOP FOX

Agenda

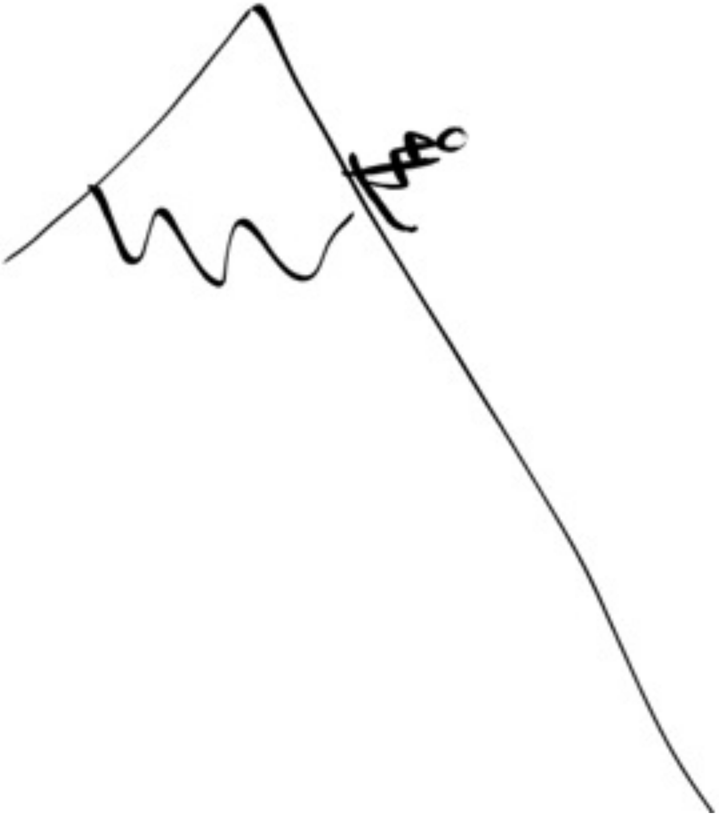
THE PLAN

- Motivation
- Theory
- Security Objectives
- Other Security Requirements

- HAZOP Analysis
- Secure Requirements Checklists
  - Big Requirements up Front
  - Agile

Motivation

WHY REQUIREMENTS MATTER



What is a Requirement?

A requirement is a statement of:

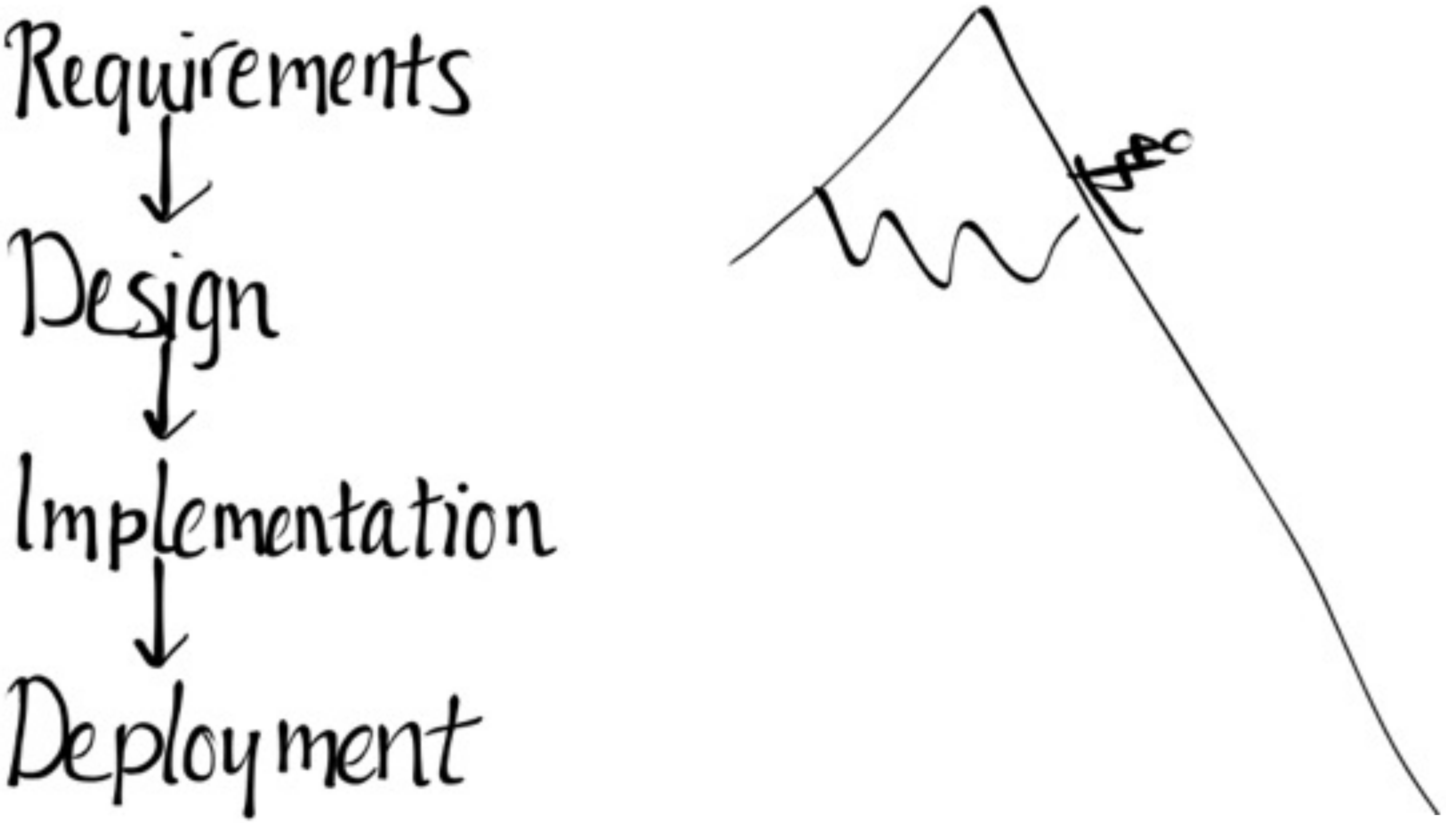
1. What a system must do (a system function)
2. How well the system must do what it does (a system quality or performance level)
3. A known resource or design limitation (a constraint or budget)

More generally,

A requirement is *anything that drives a design choice*

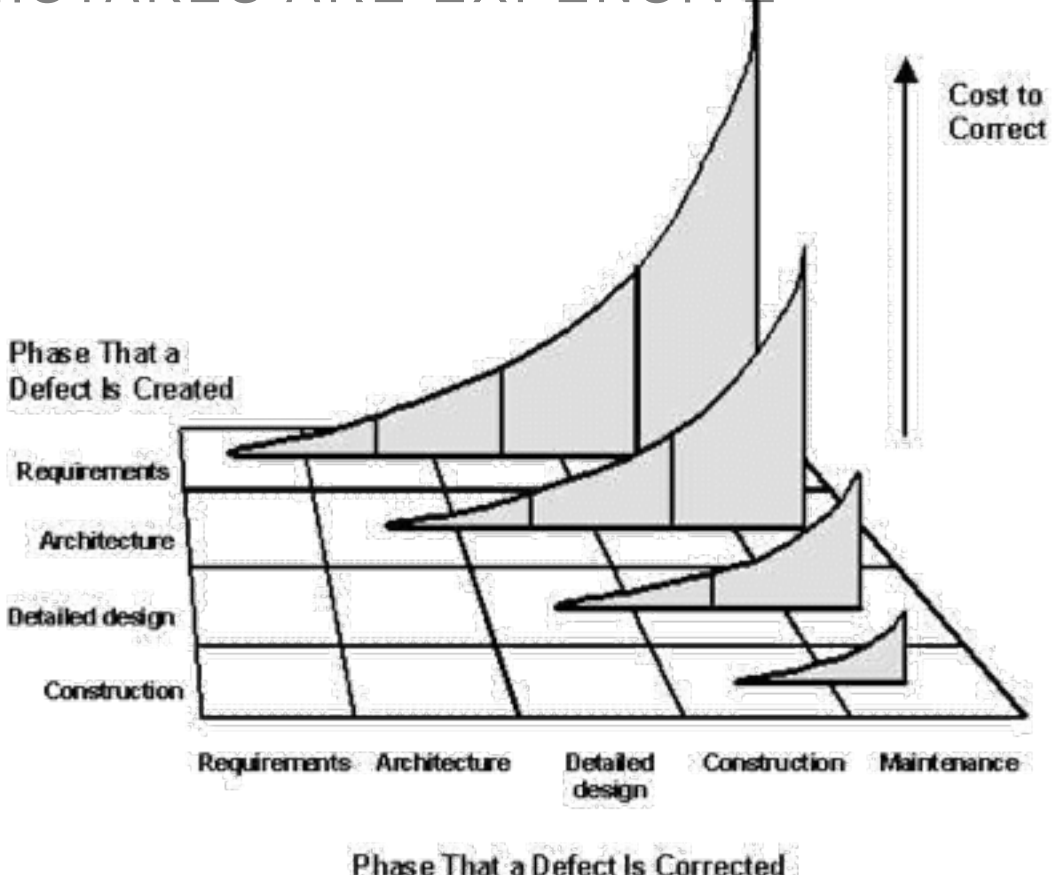
Why should organizations care about requirements?

IT'S ALL DOWNHILL FROM HERE



Why should organizations care about requirements?

MISTAKES ARE EXPENSIVE



Why should security stakeholders care about requirements?

LESS FUSS, MORE THINGS YOU WANT

Security Advocate	Development Team	Business Owner
- New high-risk service or product - Plan to launch next week - No thought to security - I'm supposed to secure this before launch? - We'll have to delay to fix these critical security issues	- Everyone agreed on what to build - We built that - You want a last-minute delay to fix "bugs" none of our users would actually encounter?	- We can't afford to delay launch - Fix what you can this week - Ship it & hope for the best

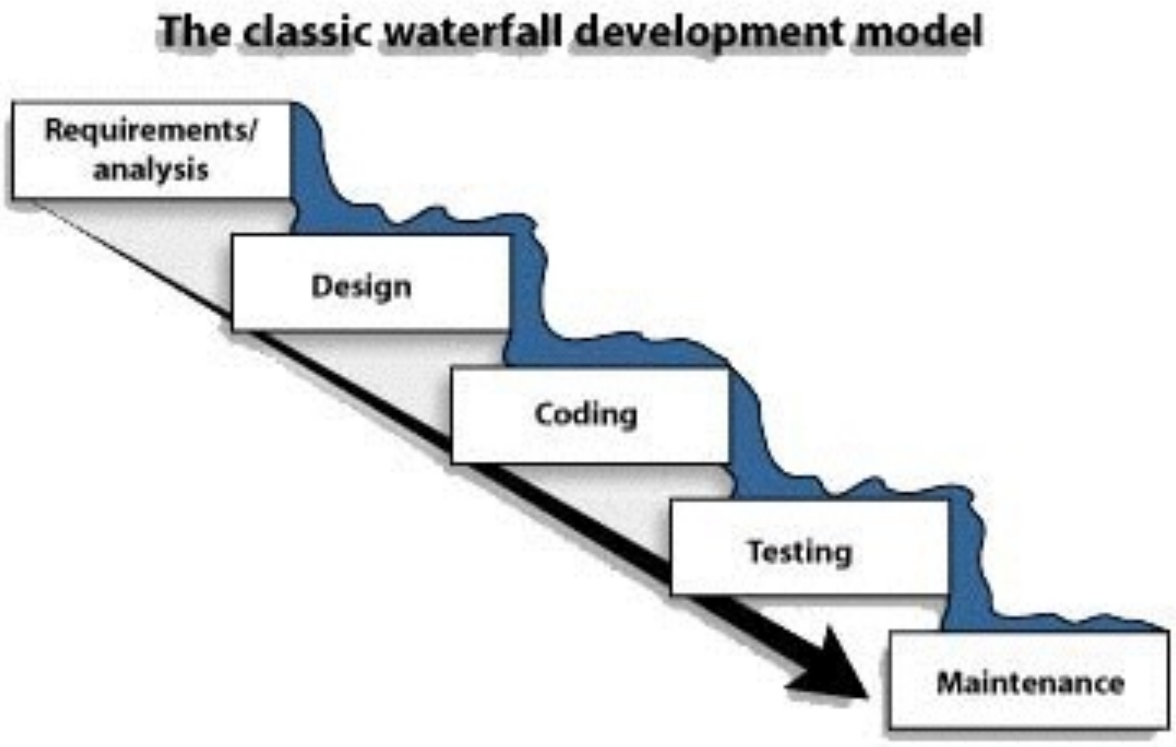
What requirements appeared in other talks this week?

Theory

CONCEPTS

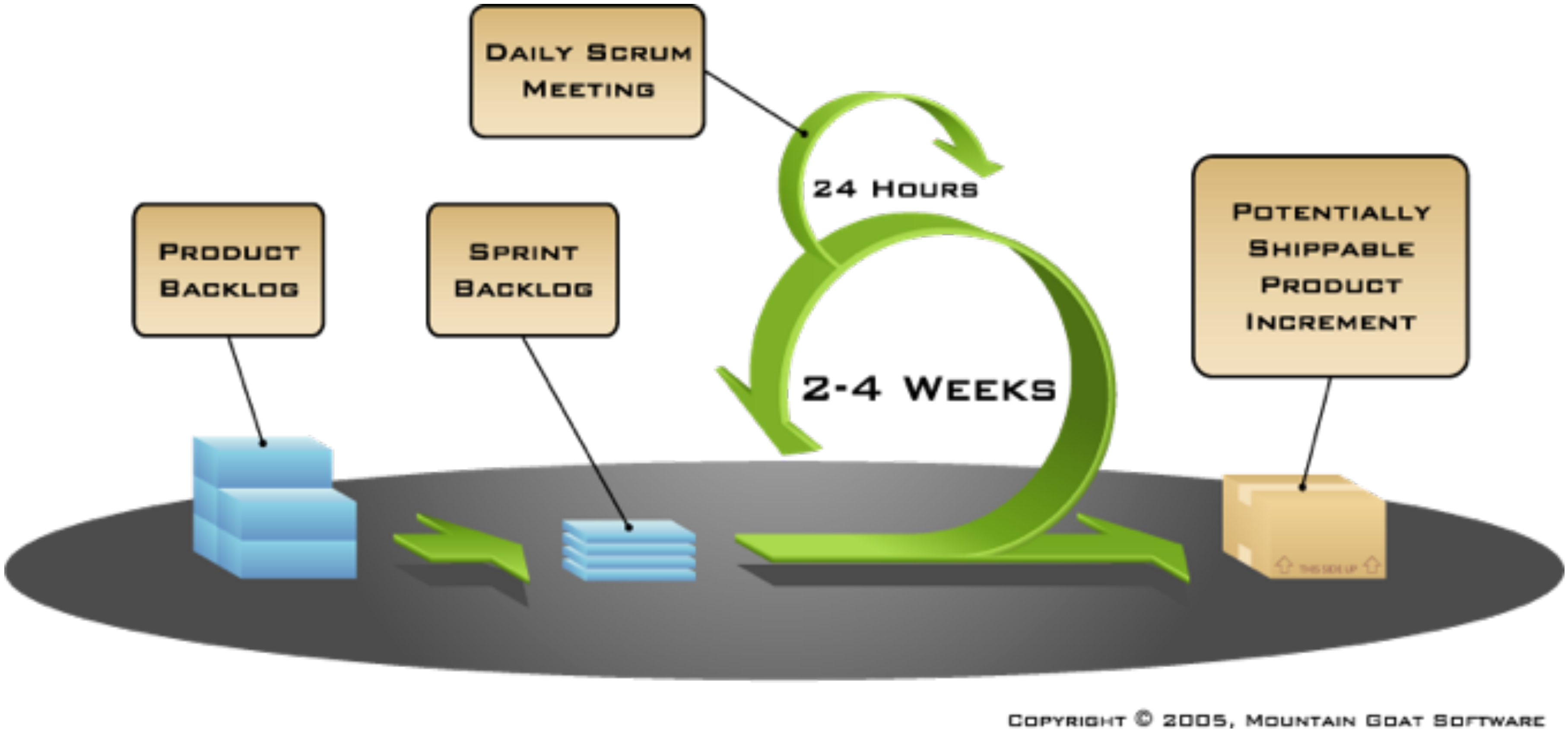
Development Lifecycles

WATERFALL



Development Lifecycles

AGILE



Requirements Engineering

Requirements Engineering is the systematic and repeatable use of techniques for discovering, documenting, and maintaining a set of requirements for a system or service.

Requirements Engineering Activities

Elicitation	Analysis & Validation	Specification	Verification	Management
Gathering requirements from stakeholders	Assessing, negotiating, and ensuring correctness of requirements	Creating the written requirements specification	Assessing requirements for quality	Maintaining the integrity and accuracy of the requirements

Necessary & Sufficient

SPECIFICATION

Necessary

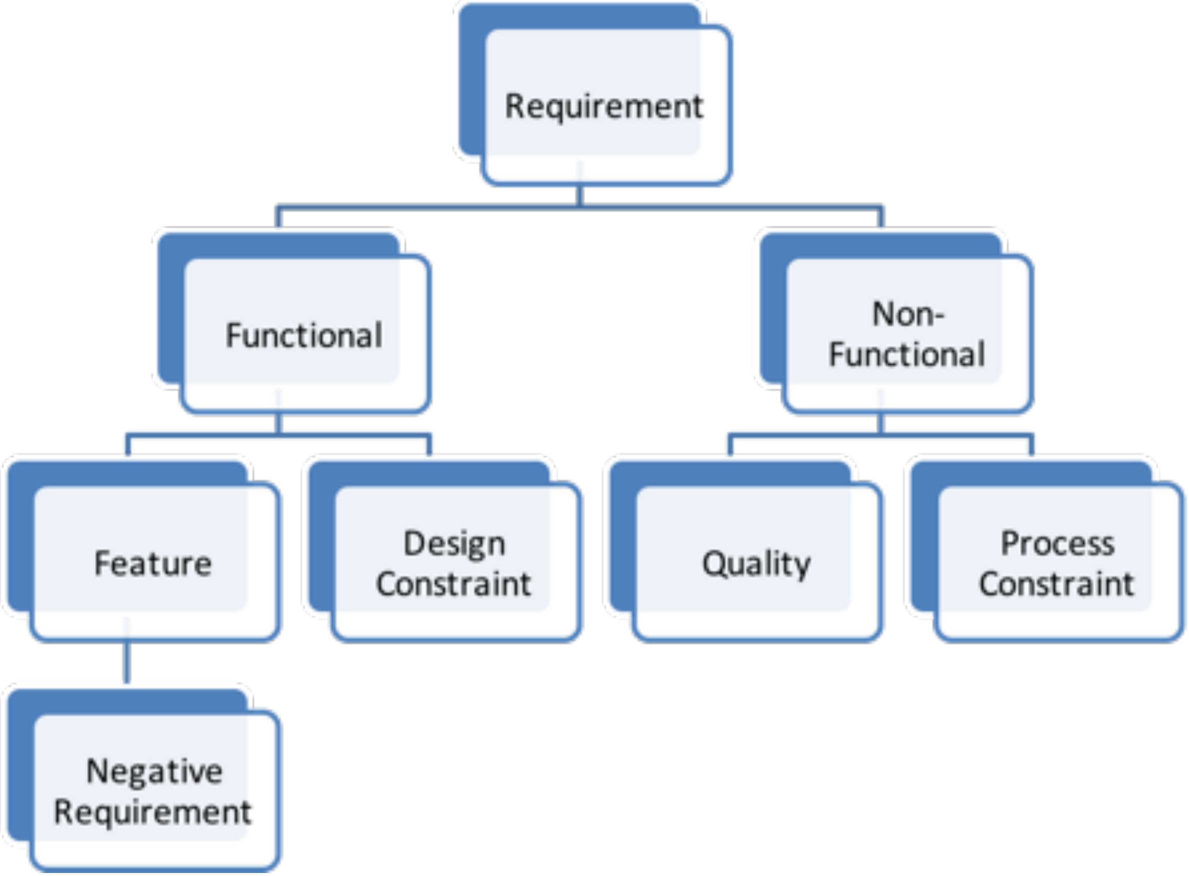
- Problem Domain
  - Objects
  - Relationships
  - Workflow
- Interfaces
- Design Constraints

Sufficient

- Guide the current activities of all team members at an acceptable risk level
- Allow the person downstream as much flexibility as possible

Requirements

TAXONOMY



Functional vs. Non-Functional

SPECIFICATION

- Distinction: How is it measured?
- Functional requirements
  - Boolean
  - Syntax: [Trigger] [Precondition] Actor Action [Object]
- Non-functional requirements
  - Along a scale or interval
  - Typically specify at least scale, meter, minimum, target, maximum
  - Scale - Units of measure, e.g. centimeters
  - Meter - Device or process to measure position on the scale, e.g. measuring tape

Scales

ANALYSIS

- Three types of scales (in order of preference):
  - Natural: Scales with obvious association to the measured quality
  - Constructed: A scale built to directly measure a quality
  - Proxy: An indirect measure of a quality

Natural	Signal to jammer interference (noise) ratio
Constructed	Effort expended in the preparation and execution of the attack, graded as “no rating“, “minimal”, “basic”, "enhanced-basic,” “moderate” or “high”
Proxy	Percentage of developers who have received security training

Are these security requirements?

EXAMPLE

- The equipment cabinet must take at least 10 minutes to disassemble using common household tools.
- The software shall be unable to determine the position of the Logic & Accuracy Testing switch.
- The booth door shall remain locked until the transcript has finished printing.

Are these requirements secure?

EXAMPLE

- The equipment cabinet must take at least 10 minutes to disassemble using common household tools.
- The software shall be unable to determine the position of the Logic & Accuracy Testing switch.
- The booth door shall remain locked until the transcript has finished printing.

Can a single requirement be secure?

Secure Requirements vs. Security Requirements

SUBTLE, BUT CRITICAL

- Security requirement - A requirement that describes a security property, technology, or action
- Secure requirements - The requirements, taken together, will result in a system that meets its stakeholders' security needs
    - E.g. any remaining underspecification will have no effect on system security

Attributes of a Good Requirement

- Complete: A requirement is complete when it contains sufficient detail for those that use it to guide their work
- Correct: A requirement is correct when it is error-free
- Concise: A requirement is concise when it contains just the necessary information, expressed in as few words as possible
- Feasible: A requirement is feasible if there is at least one design and implementation for it
- Necessary: A Requirement is necessary when it:
  - Is included to be market competitive
  - Can be traced to a stakeholder need
  - Establishes a new product differentiator or usage model
  - Is dictated by business strategy, roadmaps, or sustainability

Attributes of a Good Requirement

- Prioritized: A requirement is prioritized when it is ranked or ordered according to its importance
- Unambiguous: A requirement is unambiguous when it possesses a single interpretation
- Verifiable: A requirement is verifiable if it can be proved that the requirement was correctly implemented
- Consistent: A requirement is consistent when it does not conflict with any other requirements at any level
- Traceable: A requirement is traceable if it is uniquely and persistently identified with a Tag

What else do good requirements need to be secure?

Secure Requirements

NOT MY APPROACH I

- Separate process for managing security requirements, a la SQUARE
  - Existing requirements engineering processes seem to work fine
  - Development teams are extremely resistant to process change
- Requiring a specific SSDL
  - Development teams are extremely resistant to process change
  - Frequently, process requirements turn into inefficient & ineffective checkboxes
- Requiring resistance to specific types of attack
  - Fails open if you miss anything
  - Quickly obsolete

Secure Requirements

NOT MY APPROACH II

- Misuse & abuse cases
  - Oriented towards modeling attack goals, vs. defense goals
  - Regression issues with Agile teams (implement & forget)
  - Insufficiently structured to identify enough edge cases to be safe
  - Doesn't address overall high-level goals
- Formal methods for verifying security requirements or validating implementations
  - Don't know how to apply in industry (but maybe y'all do)

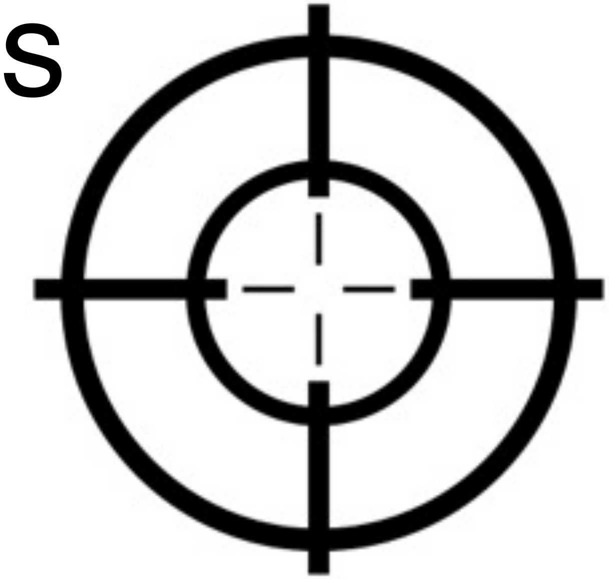
Secure Requirements

MY APPROACH

- Security objectives
  - High-level security goals that express stakeholders' intent to defend
- Review checklists
  - Objective criteria for whether requirements can meet the security objectives
  - No process change required
  - Multiple versions, for Agile vs. waterfall
- HAZOP analysis
  - Structured method for identifying process variations that could affect security objectives

Security Objectives

DEFINING SECURE ENOUGH



BISHOP FOX

Anatomy

DEFINITION

Initial System Configuration	Attacker Starting Privileges	Prohibited Actions	Intended System Response
State of the system before an attack begins, in order for protection to apply	***What the attacker(s) can already do before starting to attack***	***What stakeholders want to prevent the attacker(s) from doing***	***What the system should do if the attacker(s) try to achieve one of the threats***

When the initial configuration holds, the system shall not allow attackers to take the prohibited actions. If such an attack is attempted, the system shall response instead.

Example

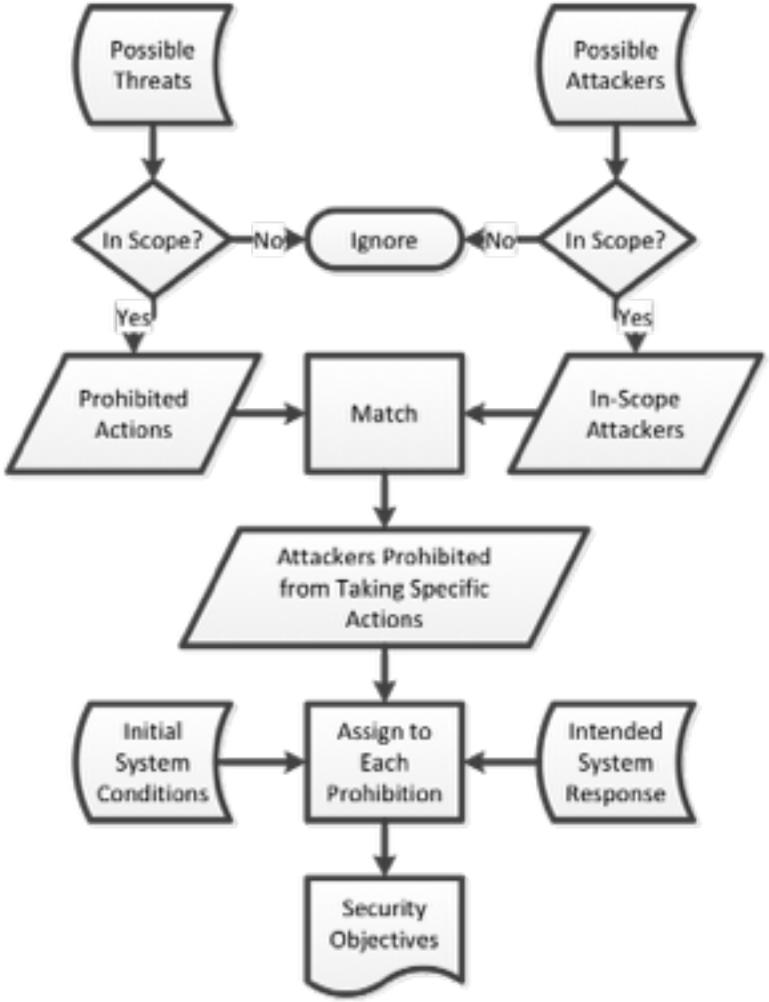
DEFINITION

When implemented as designed, the system shall not allow anonymous internet attackers or mileage plan members to book flights without paying, book flights in violation of fare rules, or grant travel vouchers. If such an attack is attempted, the system shall thwart the attack.

Elicitation, Analysis & Specification

"PROCESS"

- Quick[er]
- Brainstorming-based approach could miss something
- Hard[er] to be sure of semantic agreement
- More formal process is based on Trike



Prohibited Actions

IN-SCOPE THREATS

- Threats
  - In the sense of "what could possibly go wrong?"
- [Action] [Object] [Condition]
- Brainstorm or derive from functional requirements
- Prohibited actions are the subset of threats that would hurt stakeholders most
  - If it doesn't hurt a stakeholder, there's no need to defend against it

In-Scope Attackers

MINIMIZE MODELING

- Focus on attacker starting privileges
  - What can each attacker already do before she launches an attack?
  - Where is she, logically & physically?
  - What does she know?
  - What does she have?
- Assume conspiracy
- Ignore motivation
  - People you don't know are inscrutable
  - The point is not to thwart the attacker, it is to defend yourself
- Resource: Intel's Threat Agent Library

Intended System Response

TAXONOMY

Response	Meaning	Applicability
Prevent	Stop the attack before it gets to the target system	• Implies the existence of a filter (e.g. a firewall) in a different scope from the target
Thwart	Stop the attacker from achieving the threat	• Most common intended response
Detect	Notice the attack is happening	• Not always possible or worth the effort • Implies a next step (e.g. log or alert)
Log	Take notes someone can look into later	• Implies detect • Required if system owners may want to pursue legal action
Alert	Tell someone right now	• Implies detect • Useless unless someone is monitoring • Required if human intervention is the next step
Rate Limit	Let the attack succeed, but not too often or too quickly	• Frequently implies detect • Especially useful to respond to denial of service

Initial System Conditions

CAVEATS

- What state does the system need to be in before it can defend against this attacker trying to reach this threat?

Stakeholders

PARTICIPANT ROLES

Name	Description	Contribution
Product Owner	Individual who contributes business requirements and makes decisions about scope	Do we want that?
Architect	Individual performing high-level design work and large scale system engineering	Can we build that?
Operations	If the project will be deployed in-house or as a service, individual performing or managing deployment and maintenance of the system	Can we do that?
Security	Individual who represents security concerns during development of a project	Would that be safe enough?

Functional or Non-Functional?

EXAMPLE

When implemented as designed, the system shall not allow anonymous internet attackers or mileage plan members to book flights without paying, book flights in violation of fare rules, or grant travel vouchers. If such an attack is attempted, the system shall thwart the attack.

Functional to Non-Functional

GETTING THERE

Initial System Configuration	Attacker Starting Privileges	Prohibited Actions	Intended System Response
State of the system before an attack begins, in order for protection to apply	***What the attacker(s) can already do before starting to attack***	***What stakeholders want to prevent the attacker(s) from doing***	***What the system should do if the attacker(s) try to achieve one of the threats***

When the initial configuration holds, the system shall not allow attackers to take the prohibited actions. If such an attack is attempted, the system shall response instead.

BISHOP FOX

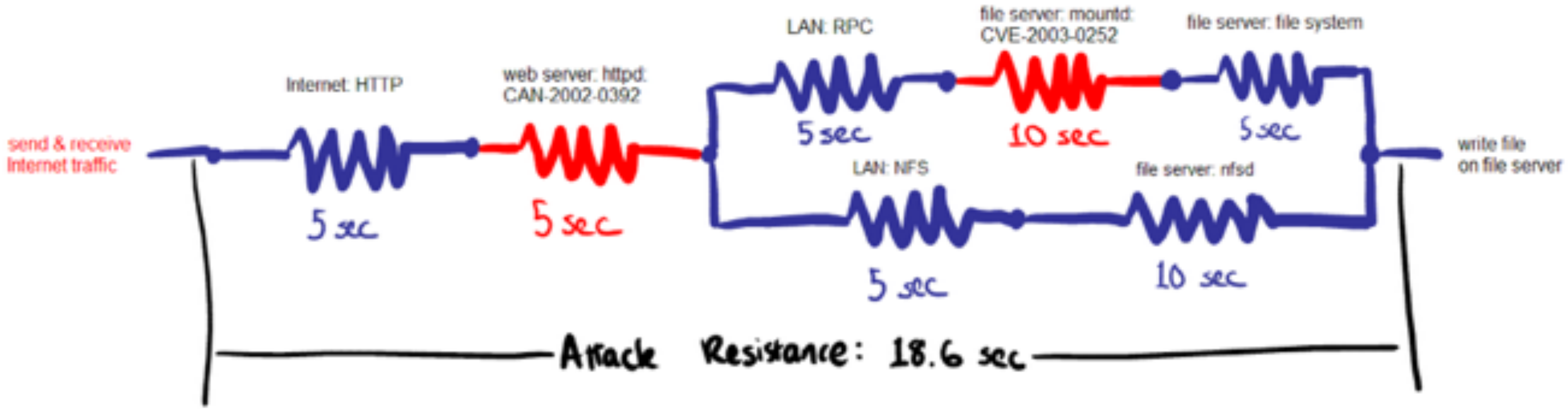
Security Metrics Now

SCALES & METERS

- Number of [known] security defects (probably by severity)
- Number of [known] security incidents
- Percentage of developers who have received security training

- No standard for what counts as a security defect
- Cannot be measured until development, even deployment, is complete
- Measuring secondary indicators

Attack Resistance


send & receive Internet traffic
Internet: HTTP
5 sec
web server: httpd: CAN-2002-0392
5 sec
LAN: RPC
5 sec
file server: mountd: CVE-2003-0252
10 sec
file server: file system
5 sec
LAN: NFS
5 sec
file server: nfsd
10 sec
write file on file server
Attack Resistance: 18.6 sec

- How hard it is to get from starting privileges S to ending privileges E
- Treat functions of system components as resistors
  - Units: seconds to acquire privileges exposed by this function

Why…

… this is useful:

- One number for prevention
  - Architecture, design, implementation or deployment
  - Different analysts
- Uses the right kinds of inputs
  - Attacker & defense goal
  - As-designed behavior
  - Known vulnerabilities
- Matches intuition about badness
- Guesses in design can be measured in implementation

… this is deeply wrong:

- These units don't act this way

… applicability is limited:

- Always worst case
- Ignores
  - Unknown vulnerabilities
  - Non-determinism
  - Non-privilege attacker characteristics

Other Security Requirements

DESIGN CONSTRAINTS

Design Constraints

SOURCES

- Explicit stakeholder requests
  - E.g. required technology, processes
- Standards & regulations
  - If standard is new to devs, or difficult to understand, or not traceable, call out the specific requirements vs. incorporating by reference
- Other requirements for this system
  - Keep derived requirements to a minimum

HAZOP Analysis

DETAILS THAT PROBABLY BITE

BISHOP FOX

What – Overview

What is HAZOP analysis?

- HAZOP = Hazardous Operations
- Systematic method for identifying which variations in a process need to be mitigated for safety
  - Repurposed for security by Thitima Srivatanakul, John A. Clark and Fiona Polack from University of York
- Partially replaces threat and attack trees
  - HAZOP analysis results include threat tree leaf nodes

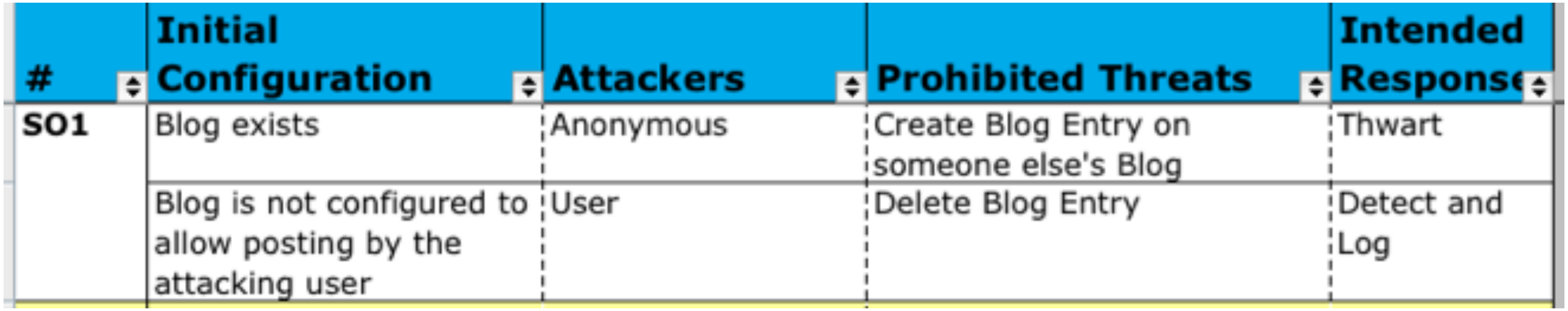
How – Process Overview

What do I do?

- Preparation
  - Set security objectives (Actors, Data Model, Intended Actions, Threats, Security Objectives)
  - Document use case or other sequence of steps (Actors, Data Model, Use Case Details A-L)
- HAZOP analysis
  - Vary each element of each step (Use Case Details M-P)
  - Analyze meaningful variations for security implications (Use Case Details Q-AE)
- Use HAZOP analysis results
  - Plan mitigations (usually, update requirements or design)

What – Example Overview

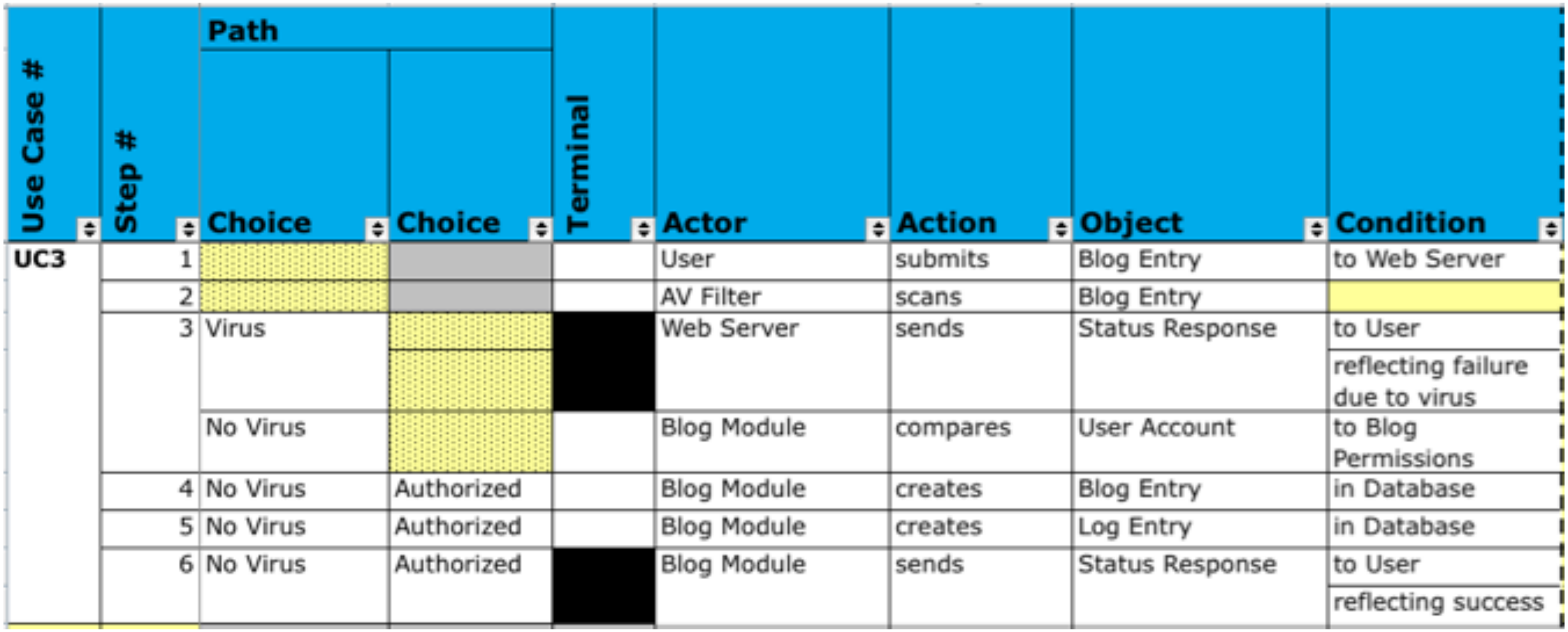
What should it look like before I start?



#	Initial Configuration	Attackers	Prohibited Threats	Intended Response
SO1	Blog exists	Anonymous	Create Blog Entry on someone else's Blog	Thwart
	Blog is not configured to allow posting by the attacking user	User	Delete Blog Entry	Detect and Log

What – Example Overview

What should it look like before I start?



Use Case #	Step #	Path		Terminal	Actor	Action	Object	Condition
		Choice	Choice					
UC3	1				User	submits	Blog Entry	to Web Server
	2				AV Filter	scans	Blog Entry	
	3	Virus		■	Web Server	sends	Status Response	to User reflecting failure due to virus
		No Virus			Blog Module	compares	User Account	to Blog Permissions
	4	No Virus	Authorized		Blog Module	creates	Blog Entry	in Database
	5	No Virus	Authorized		Blog Module	creates	Log Entry	in Database
	6	No Virus	Authorized	■	Blog Module	sends	Status Response	to User reflecting success

How – Varying a Step

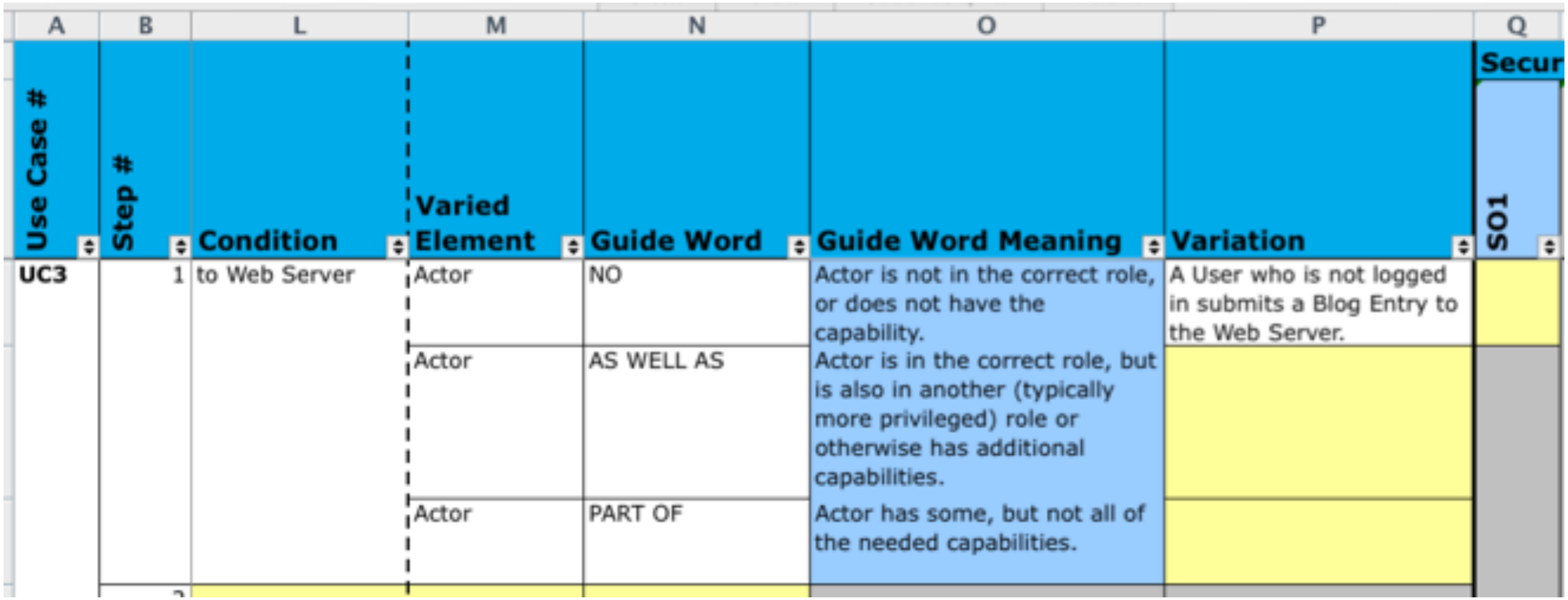
I assembled all that stuff. How do I get started?

- Pick a step
- Pick an element
  - Actor, Action, Object, or Condition? (Use Case Details M)
- Pick a guide word
  - NO, AS WELL AS, PART OF, OTHER THAN, MORE, LESS, BEFORE, AFTER (Use Case Details N)
- Figure out what this variation means, if anything
  - Document all meanings (Use Case Details O and P)

- Productivity tips
  - Vary elements in the same order each time
  - Apply guide words in the same order each time
  - Vary depth-first (all guide words for one element, then all for the next element)

What – Example Variation

How should varying a step turn out?



A	B	L	M	N	O	P	Q
							Secur
Use Case #	Step #	Condition	Varied Element	Guide Word	Guide Word Meaning	Variation	SO1
UC3	1	to Web Server	Actor	NO	Actor is not in the correct role, or does not have the capability.	A User who is not logged in submits a Blog Entry to the Web Server.	
			Actor	AS WELL AS	Actor is in the correct role, but is also in another (typically more privileged) role or otherwise has additional capabilities.		
			Actor	PART OF	Actor has some, but not all of the needed capabilities.		
	2						

How – Analyzing a Variation

I’ve got a variation. Now what?

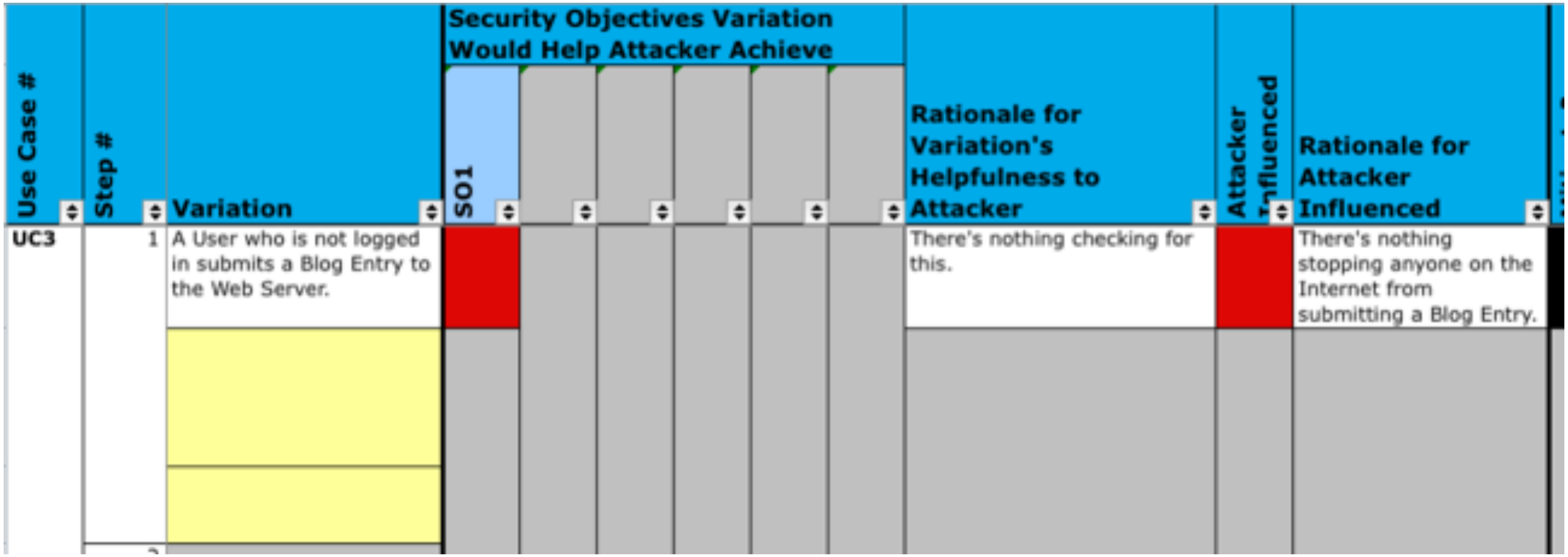
- Can this variation help an attacker?
  - Decide separately for each security objective (Use Case Details Q-AC)
- If the variation can help an attacker, can an attacker* cause or influence it? (Use Case Details AD-AE)

* Not necessarily the same attacker. Assume collaboration!

- Productivity tips
  - Finish all variations for a step (maybe even use case) before analyzing variations

What – Example Analysis

How should analyzing a variation turn out?



Use Case #	Step #	Variation	Security Objectives Variation Would Help Attacker Achieve						Rationale for Variation's Helpfulness to Attacker	Attacker Influenced	Rationale for Attacker Influenced
			SO1								
UC3	1	A User who is not logged in submits a Blog Entry to the Web Server.	🟥						There's nothing checking for this.	🟥	There's nothing stopping anyone on the Internet from submitting a Blog Entry.
	2										

BISHOP FOX

How – Shortcuts

Isn’t that going to be awfully repetitive?

- Factor common portions of use cases out into a separate use case
- Don’t document variations that won’t get you anything
  - Variations that are equivalent to earlier variations
  - Variations you know are not helpful to an attacker, or not controllable by an attacker*
- Claim that variations you know are pure implementation issues are not attacker-influenced

* This has a reviewability penalty and may cause you to miss issues; do 3-10 use cases before you try it.

How – Using the Results

Dude, that’s a lot of data.

- Plan and prioritize mitigations (Use Case Details AF-AI)
- Plan implementation reviews
  - Security code reviews (Use Case Details some from AE, some from AG)
  - Security test coverage (Use Case Details some from AG)
  - Penetration tests (Use Case Details AD-AE, some from AF-AI)

Secure Requirements Checklists

REVIEWING REQUIREMENTS FOR SECURITY

Requirements Security Checklist

BIG REQUIREMENTS UP FRONT

- Use to review all requirements, including security objectives
- Collaboration with Erik Simmons at Intel starting in 2006
- First release to anyone outside Intel
- Hopefully soon to be updated

Requirements Security Checklist

Complete …

- All stakeholders affected by the product's security have reviewed the requirements.
- At least one security subject matter expert has reviewed the requirements.
- All user classes are explicitly documented. Common classes include end users, IT users, administrative users, maintenance users, indirect users, unauthenticated users, anonymous users, and other systems which connect directly to the system.
- All intended system interfaces that cross the system-universe boundary (external interfaces) are explicitly listed. Common examples of interfaces include pins, wiring terminals, buses, user interface hardware, APIs, and network connections.
- Each user class is specified to communicate with the system via one or more of the system's external interfaces and each external interface has at least one user class assigned to it.
- Stakeholders have agreed on the set of assets that the system manipulates, and therefore may require protection.
- All allowed, required, and prohibited relationships among assets are depicted within the requirements. Common relationship types related to security include is a (for example, bicycles are vehicles), has a (for example, bicycles have two wheels) and business rules involving multiple assets (for example, order total is always the sum of the items on the order).

Requirements Security Checklist

… Complete ...

- The functional requirements include all conditions affecting whether each function is allowed from within each system state. Common conditions include permissions, business rules, time restrictions, simultaneous events that are allowed or disallowed, relative position in a sequence of actions, pre-conditions and post-conditions.
- The functional requirements include what must occur when the system encounters invalid data.
- Stakeholders have agreed on the set of security events to which the system must respond. Security events include access violations, many exceptional events, and some events within intended system functionality such as access to critical or confidential assets. For example, all accesses to employee salary data are security events, whether proper or improper.
- Each documented security event includes prescribed system response. Common responses include log the event, deny the event, and alert a user.
- For each event log entry expressed or implied by other functional requirements, the requirements also include conditions affecting whether access to log entries is allowed (even if that access will be implemented using only built-in functionality of underlying subsystems).

Requirements Security Checklist

… Complete

- For each event log entry expressed or implied by other functional requirements, the requirements include a description of the allowed and required contents of the log entries.
- The requirements explicitly specify all privacy and security standards and regulations to which the system must adhere.
- Stakeholders have agreed on the set of attackers, their initial privileges, and the end goals the product must defend against each attacker.

Verifiable

- There are 12 or fewer assets. Even better is 7 plus or minus 2. More than that is too complex for humans to understand and analyze at once and must be considered a risk.
- The scale and meter specified by the non-functional security requirement will yield similar measurements for different security validation teams working on the same product.

Requirements Security Checklist

Feasible

- No attacker's initial privileges as described within the requirements include that attacker's end goals. For example, if all Users shall be able to read all Status_Reports, the requirements cannot also specify that the system shall prevent an attacker with User privileges from reading any Status_Report.
- Based on the conditions affecting each function described in the requirements, no legitimate sequence of actions that can be taken starting from an attacker's initial privileges can reach that attacker's end goals. For example, if any Anonymous user shall be able to immediately create a User account (with no approval, verification of identity, payment, or other business rule), and all Users shall be able to read all Status_Reports, the requirements cannot also specify that the system shall prevent an attacker with Anonymous privileges from reading any Status_Report.
- The non-functional security requirements are feasible based on an analysis of system requirements.
- The initial privileges (including physical access, or lack thereof) assigned to each attacker is plausible in light of expected system usage.

Requirements Security Checklist

Prioritized

- Each requirement which is included or modified for security is clearly marked.

Unambiguous

- Each user class has consistent, explicitly defined duties and privileges. That is, if any member of User Class A can (or must) do something, every other member of User Class A in an identical situation can (or must) also do that.
- When conditions affecting whether a function is allowed are enumerated, the conditions are completely specified. For example, if the action is allowed when $x < 0$ and prohibited when $x > 0$, the requirements must also specify what must happen if $x = 0$.
  - Exception: The context specifies a default (either allow or prohibit the action).
- System states & conditions that purposefully do not require any particular system response are explicitly specified.
- The entire lifecycle of each asset (i.e. what causes it to become present in the system, its initial state, and sequences of user actions that can be performed on it until it is no longer present in the system) has been documented.

Requirements Security Checklist

Consistent

- Each user class is allowed to perform at least one action within the requirements.
- No two user classes have identical duties and privileges.
    - Exception: There is reason to believe the duties and privileges of these user classes will diverge in the lifetime of the system being described.
    - Exception: The two categories of users have different intents within the system.
- The system requirements perform at least one action on each asset. If this is not the case, either something is improperly called an asset, or there are missing functional requirements that affect the asset.
- If one asset class has an is a relationship with another class, the first asset class must be a proper subset of the second. For example, if bicycles are vehicles and bicycles and vehicles are both classes of assets, all bicycles must be vehicles. If this is not the case (for example, the system will consider stationary bicycles to be bicycles, but not vehicles), one of the asset classes may be improperly called an asset.
- The conditions necessary for the system to allow each function do not contain a paradox.
- The conditions necessary for the system to prohibit each function do not contain a paradox.

User Story Security

AGILE

- Sets of criteria that work together
  - Applied at different times during agile development
  - Developed at Bishop Fox for several customers

- User Story Security Risk Levels
- User Story Collection Security Checklist
- User Story Defined Security Checklist

- Security Additions to the Definition of Done
- Security Objectives Checklist
  - Current work in progress

User Story Security Risk Levels

VIOLATES SECURITY OBJECTIVES

This story, if implemented as described, will allow an attacker to violate the application's security objectives.

A story should receive this risk rating if either of the following are true:

- The role, feature, and benefit of the story directly contradict the product's security objectives.
- The user story implies one or more variations that, taken independently of other stories that remain on the product backlog, would allow an attacker to violate one or more security objectives, and nothing in the application, done definition, or user story stops the attacker from receiving this benefit, causing the variation, or waiting for the variation to occur.

User Story Security Risk Levels

INCREASES SECURITY RISK

This story, if implemented as described, could help an attacker to violate the application's security objectives, but something else would also have to go wrong for the attacker to succeed.

A story should receive this risk rating if either of the following are true:

- The user story implies one or more variations that, taken independently of other stories that remain on the product backlog, would allow an attacker to violate one or more security objectives, but the user acceptance criteria prevent the attacker from receiving this benefit, causing the variation, or waiting for the variation to occur.
- The user story implies one or more variations that, if an attacker could cause them to occur, would help an attacker to violate one or more security objectives, and nothing in the application or the definition of done stops an attacker from receiving this benefit, causing the variation, or waiting for the variation to occur.

User Story Security Risk Levels

SECURITY RISK-NEUTRAL

This story, if implemented as described, will neither increase nor reduce attackers' ability to violate the application's security objectives.

A story should receive this risk rating if any of the following are true:

- The user story implies no variation that would help an attacker to violate any security objective.
- The user story implies one or more variations that, if an attacker could cause them to occur, would help an attacker to violate one or more security objectives, but existing mitigations in the application stop an attacker from receiving this benefit, causing the variation, or waiting for the variation to occur.
- The user story implies one or more variations that, if an attacker could cause them to occur, would help an attacker to violate one or more security objectives, but the done definition stops an attacker from receiving this benefit, causing the variation, or waiting for the variation to occur.

User Story Security Risk Levels

REDUCES SECURITY RISK

This story, if implemented as described, will reduce attackers' ability to violate at least one security objective.

A story should receive this risk rating if all of the following are true:

- The user story has a security-related benefit.
- The user story does not violate the security objectives or increase security risk.

User Story Collection Security Checklist ...

By the time a project's initial push to gather user stories is complete, the application's story backlog plus its already-implemented user stories (together, the user story collection) should continuously meet these criteria:

- There is at least one user story from the perspective of each administrator the system is expected to require (e.g. a system administrator, an application administrator).
- There is at least one user story in which an administrator deploys or updates the application.
- For each related, but out of scope, system on which the application depends, or with which the application communicates (e.g. the operating system, third party shared libraries that are expected to be deployed independent of the application_, the database server), there is at least one user story in which an administrator updates the dependency.
- For each external resource used by the application (e.g. URLs, cryptographic keys, service account names), there is at least one user story in which an administrator configures this resource.

… User Story Collection Security Checklist ...

- For every security setting or event of security interest_ that appears in a user story,
  - There is at least one user story that describes discovering problems that are caused by that setting or event (e.g. a network operations center employee getting an alert about excessive connection attempts, an administrator noticing that a certificate has expired, a user entering the wrong username or password and receiving an error message).
  - There is at least one user story that describes debugging a problem that is affected by that setting or event (e.g. an administrator determining who tampered with application data, a user determining that their cryptographic key file or an enclosing directory needs more restrictive permissions).
- There is at least one user story in which a user solves a problem that is affected by that setting or event (e.g. an administrator resetting a user's password).
- Unless all application features and data are intended to be publicly and anonymously accessible, there is at least one user story in which a user allows additional access to the application, or its features, or its data (e.g. an administrator adding a user to a role).
- For every feature or data with a user story in which a user allows access, there is also a user story in which a user restricts access (e.g. if a document owner gives permission to access a document, some user revokes permission to access a document).

… User Story Collection Security Checklist

- For every variation that is both implied by a user story and could help an attacker violate a security objective__, there is a mitigating user story whose user acceptance criteria describe the scenario and refer to the implying user stories and the security objective the unmitigated variation would help violate.
- For every user story that violates the security objectives or increases security risk, there is a corresponding user story that reduces the security risk_.

There is no need to stop with just one of each of these kinds of user stories. Overall, the intent of this checklist is to ensure that user stories that would increase the application's security are available for discussion and prioritization.

User Story Defined Security Checklist ...

Before a user story is pulled into a sprint_, business stakeholders, developers, and testers must agree that the user story meets all of the following criteria:

- The role and acceptance criteria unambiguously communicate the criteria that must be met before the system should allow access to the feature.
- If the feature will affect or refer to objects that feature prominently in the project's domain, the feature and its acceptance criteria describe all such references and effects.
- If the story is intended to provide a security benefit, address known security issues, reduce security-related technical debt, or mitigate specific threats, this motivation is clearly documented in the benefit of the story. Example: "As a system administrator, run the system without obsolete versions of library X so that attackers cannot exploit known vulnerabilities [to violate the security objectives]". Example: "As a registered user, I can log in with my username and password so that I am the only one who can see or manage my bank account".
- The feature described would provide the benefit described. Counter-example: "As a registered user, I can log in with my username and password so that I can trust the system_" fails this test because logging in allows the system to trust the user, not vice versa.
- If the story will mitigate a known vulnerability in a third-party library, the date the vulnerability was publicly announced is included in the user story.

… User Story Defined Security Checklist

- Variations on the user story have been systematically considered for security implications_.
- The acceptance criteria mention all variations of this user story that could help an attacker violate a security objective, and refer to a user story that mitigates each variation that this user story does not.
- Variations that could help an attacker violate a security objective are tagged with whether an attacker could cause or wait for the variation to occur.
- A security risk level_ has been assigned.

Overall, the intent of this checklist is to ensure that all stakeholders have the same understanding of the constraints that are likely to affect the security design of this feature, and that the user story contains enough information about security to accurately inform prioritization decisions.

Thank You