THREAT MODELING

USING THE SECURE AGILE ARCHITECTURE PRACTICE

Geoffrey Hill

Founder and CEO, Tutamantic_Sec

LinkedIn – www.linkedin.com/in/geoffrey-hill-61b7bb

Founder Tutamantic Ltd (threat modeling)
Email – geoff.hill@Tutamantic.com
Twitter – Tutamantic_Sec

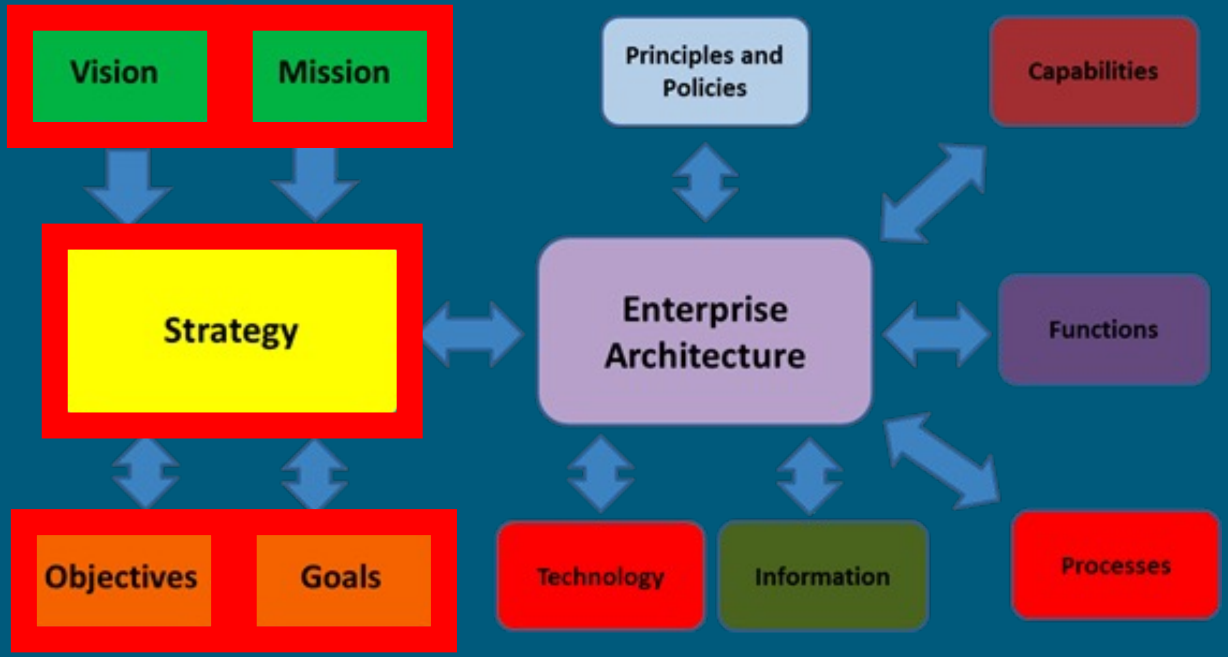
Threat Model prototyping

- Identify *BUSINESS* requirements
  using Agile Architecture – Intentional Architecture
- Conceptual design
  using Agile Architecture – Emergent Design
- Test the prototype
  with zones of trust
- Revise, Enhance and Repeat
  Just enough information… 80/20 rule (Pareto)

Security champions will drive threat model activity

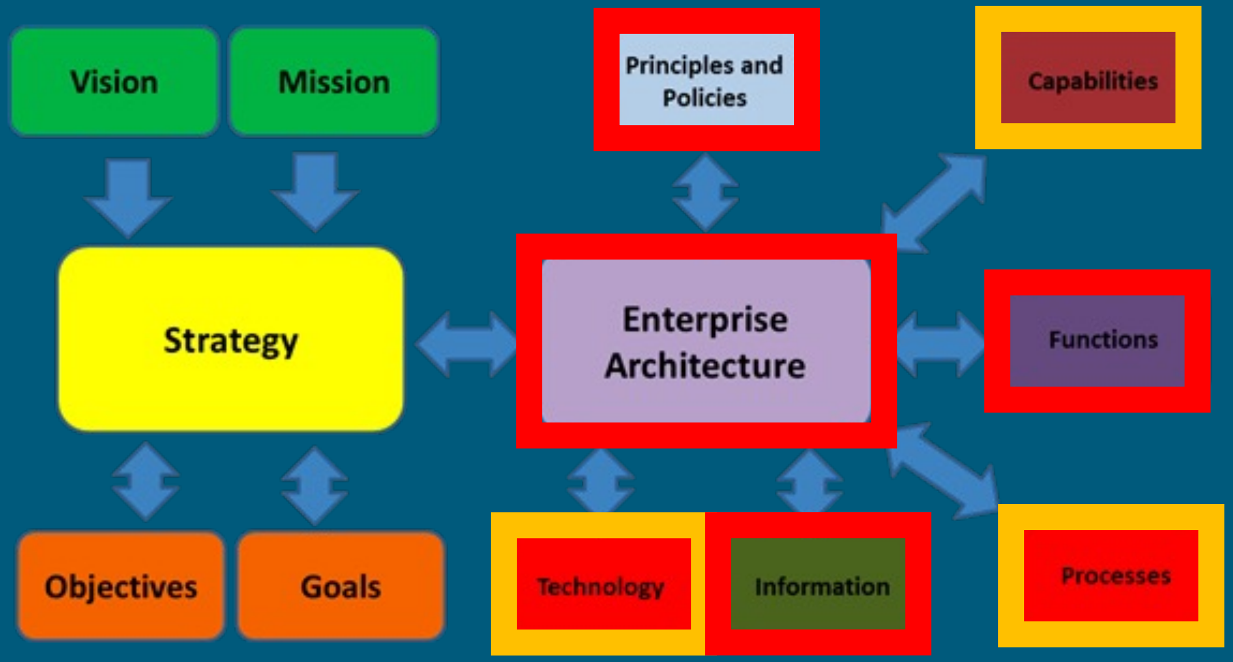
- running **threat model** sessions for security controls pre-sprint
- **triaging** discovered security issues pre-sprint
- providing **security requirements** for business requirements
- **mitigation guidance** is part of sprint Acceptance Criteria
- Providing **security mitigation** help for team

Business security Strategy (overall)



- How the business meets future needs
- Provides a foundation for future business value
- Security needs to understand this to derive <u>security requirements</u>

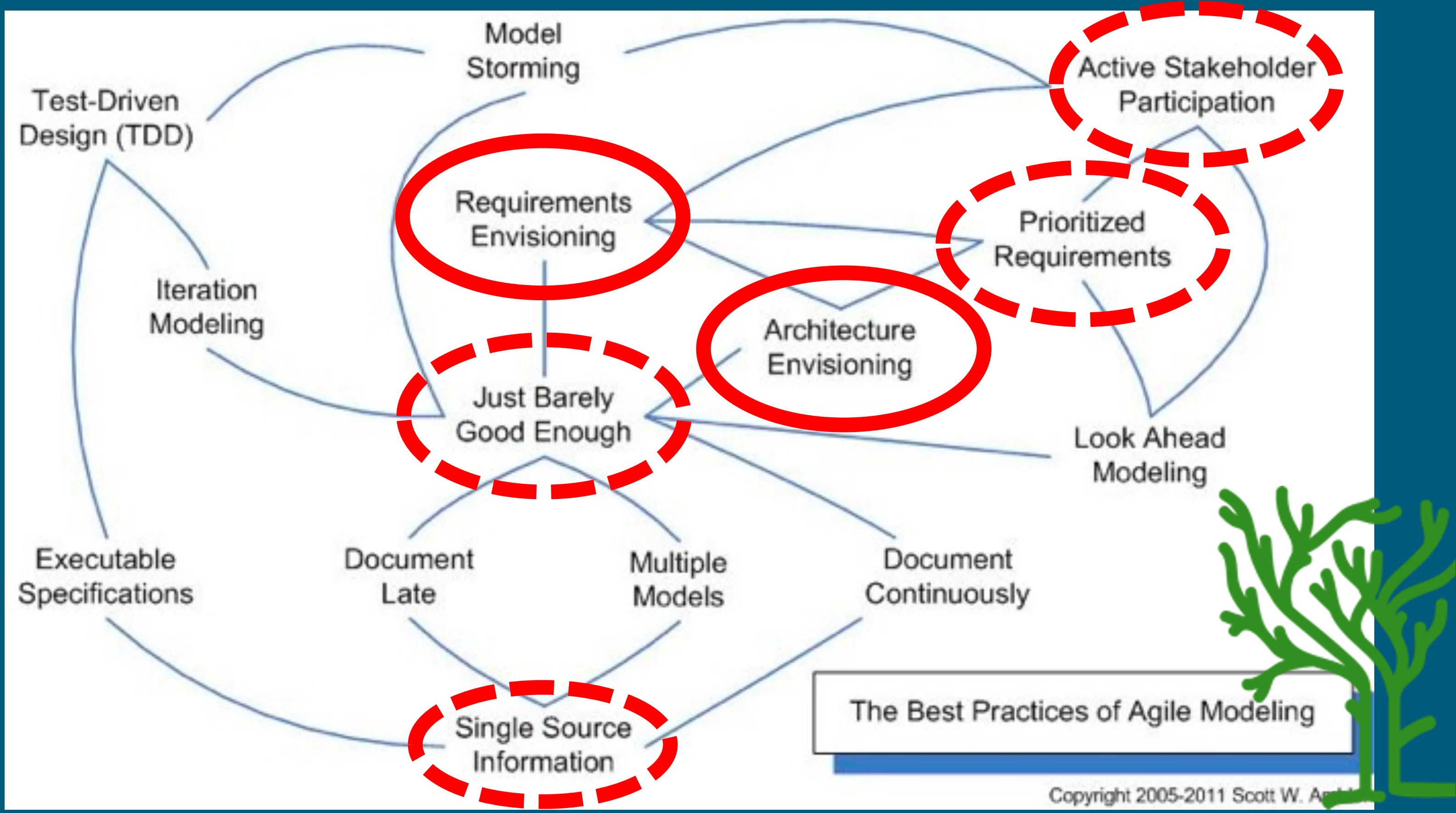
Agile Architecture Engineering



- Active stakeholder participation
- Establish lean security guidance, based on
  - People
  - Process
  - Technology
- Focus on graphic as opposed to traditional documentation. E.G. flowcharts, diagrams, etc.

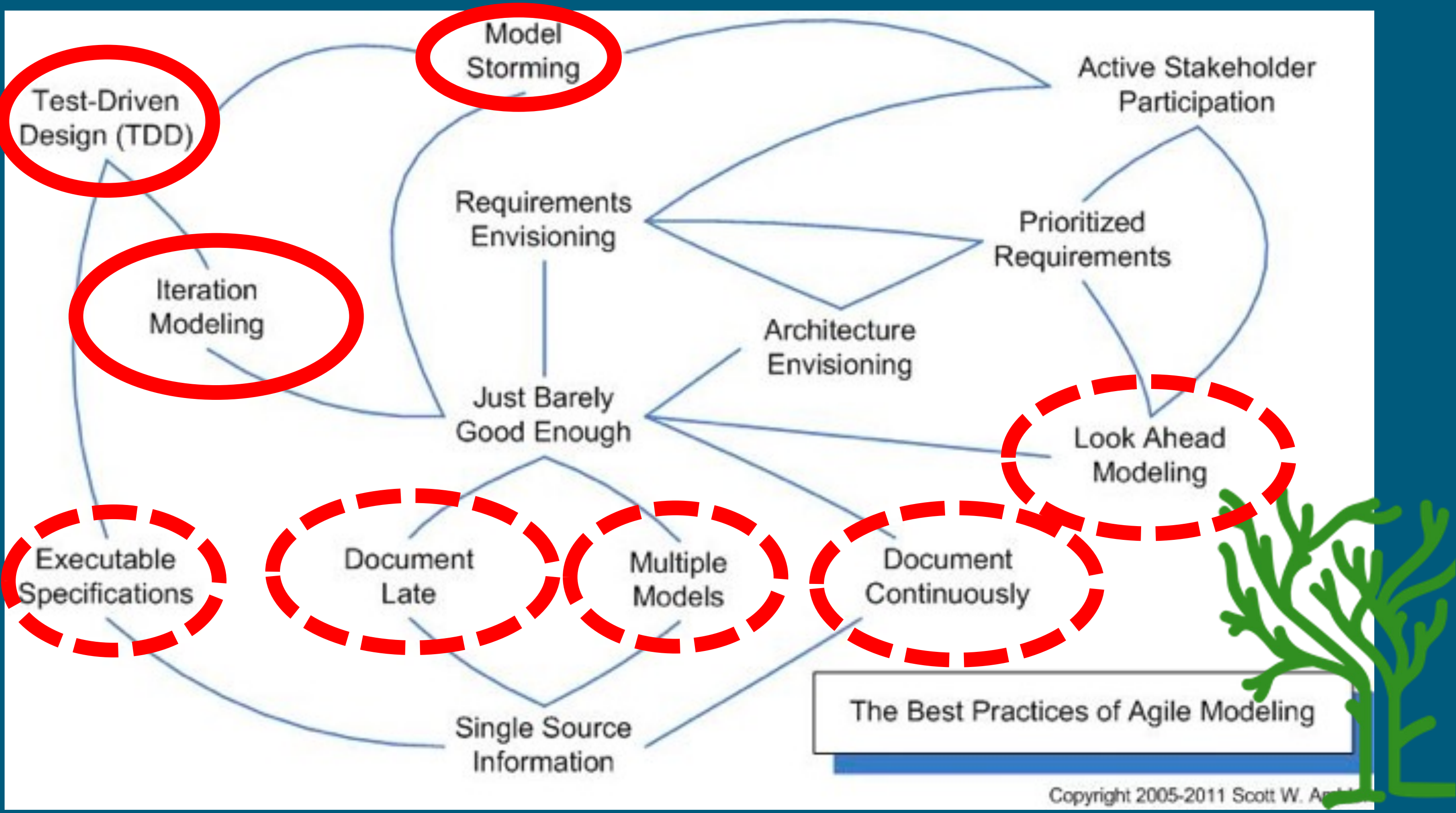
Emergent Design – Pre-Sprint

- put threat modeling into practice during Sprint 0 (scope of system)
- Solid ovals are actual activities
  - Dashed ovals are implicit
- 80/20 (Pareto) rule on all activities
- Security champions to drive the creation / use of threat models for their teams



Emergent Design – In-Sprint

- Team must plan the security work that they will do that iteration
- Solid ovals are actual activities
  - Dashed ovals are implicit
- Dashed ovals represent continuous activity
- Do a little bit of modeling and then coding, iterating back when necessary



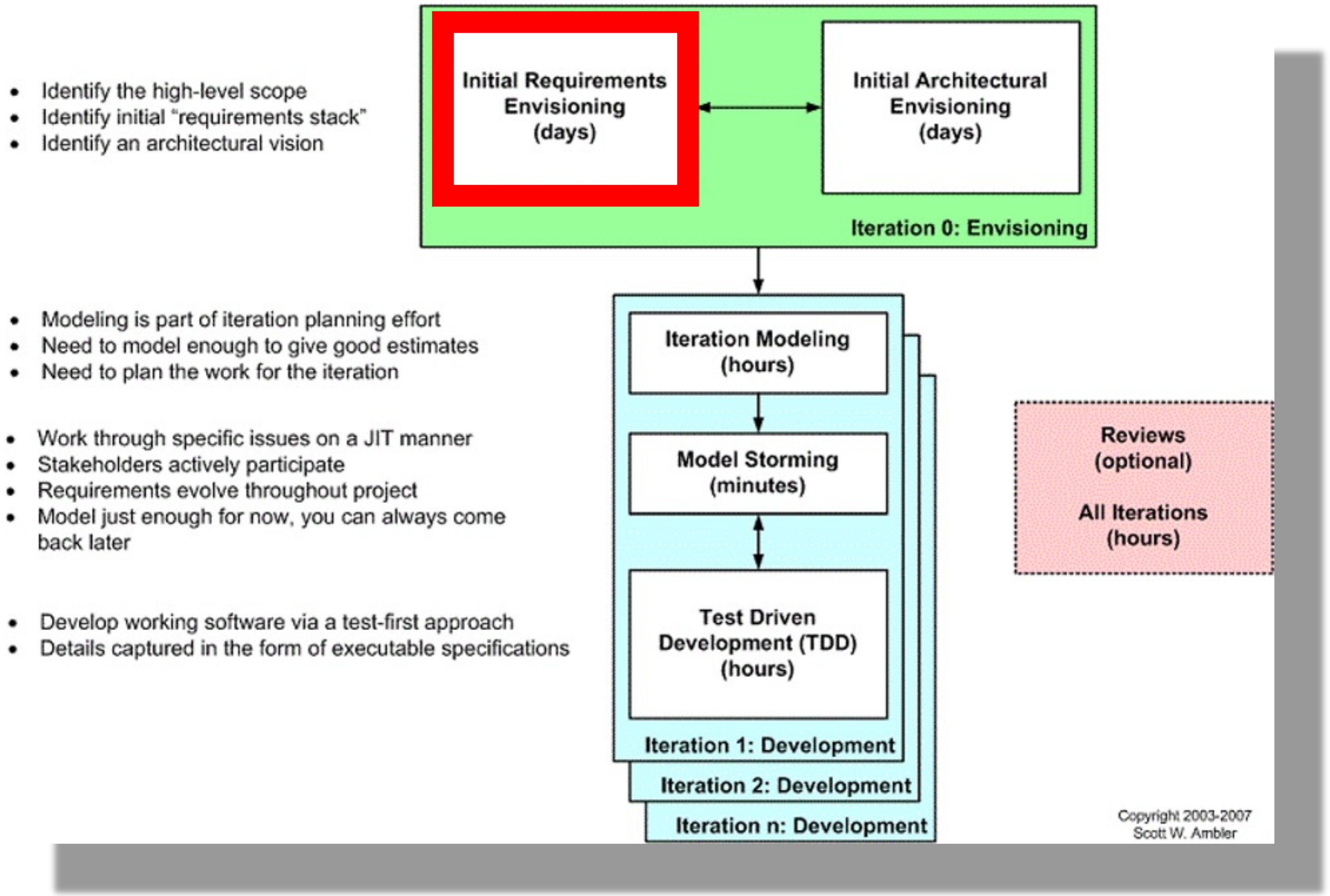
#

Threat Model sprint activities

Where and what to do during a development sprint

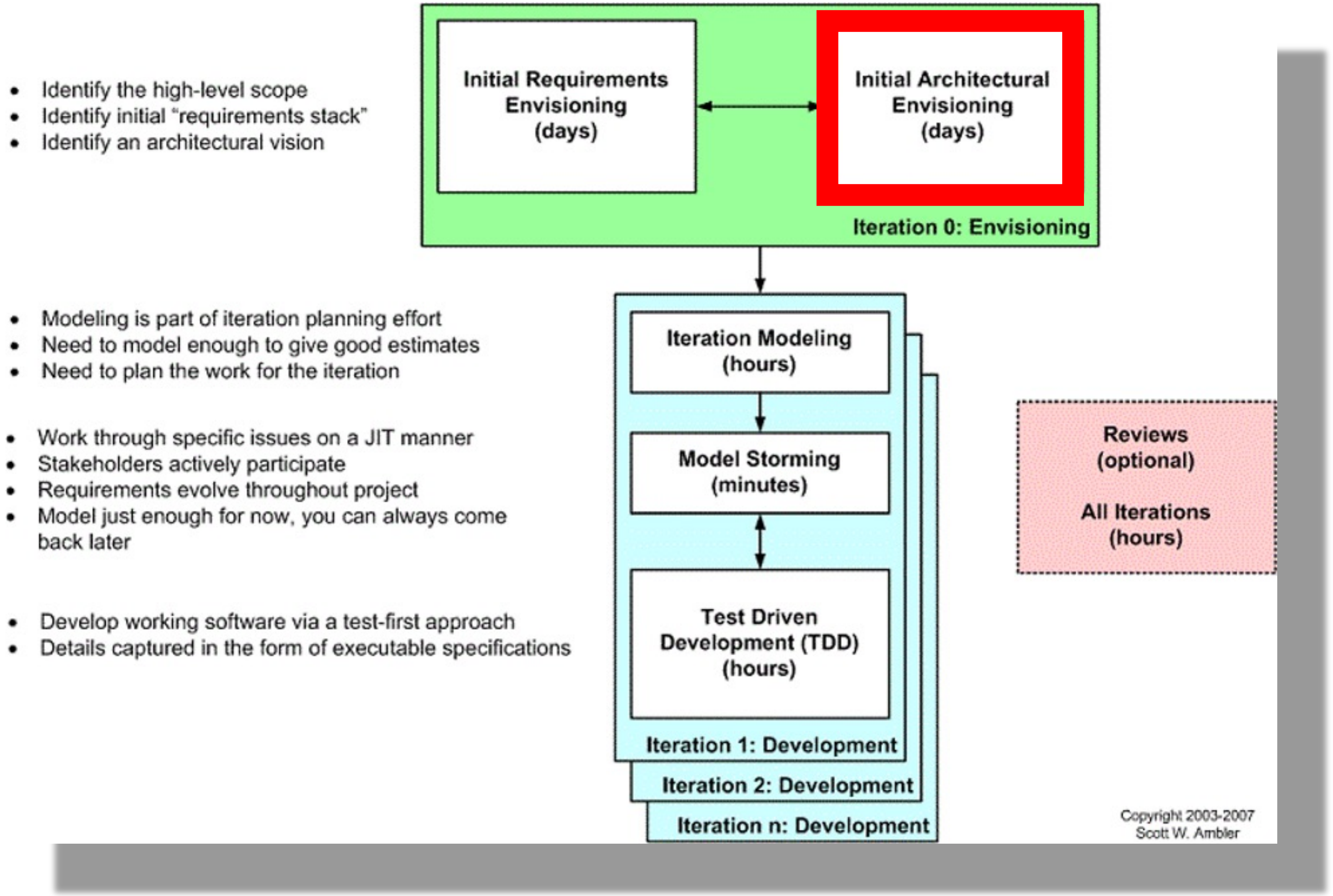
- Security champion will drive a STRIDE analysis (by the team) on all business requirements during Sprint 0

- Security requirements feed into threat model of system



Identify the high-level scope
Identify initial "requirements stack"
Identify an architectural vision
Initial Requirements Envisioning (days)
Initial Architectural Envisioning (days)
Iteration 0: Envisioning
Modeling is part of iteration planning effort
Need to model enough to give good estimates
Need to plan the work for the iteration
Work through specific issues on a JIT manner
Stakeholders actively participate
Requirements evolve throughout project
Model just enough for now, you can always come back later
Develop working software via a test-first approach
Details captured in the form of executable specifications
Iteration Modeling (hours)
Model Storming (minutes)
Test Driven Development (TDD) (hours)
Iteration 1: Development
Iteration 2: Development
Iteration n: Development
Reviews (optional)
All Iterations (hours)
Copyright 2003-2007 Scott W. Ambler

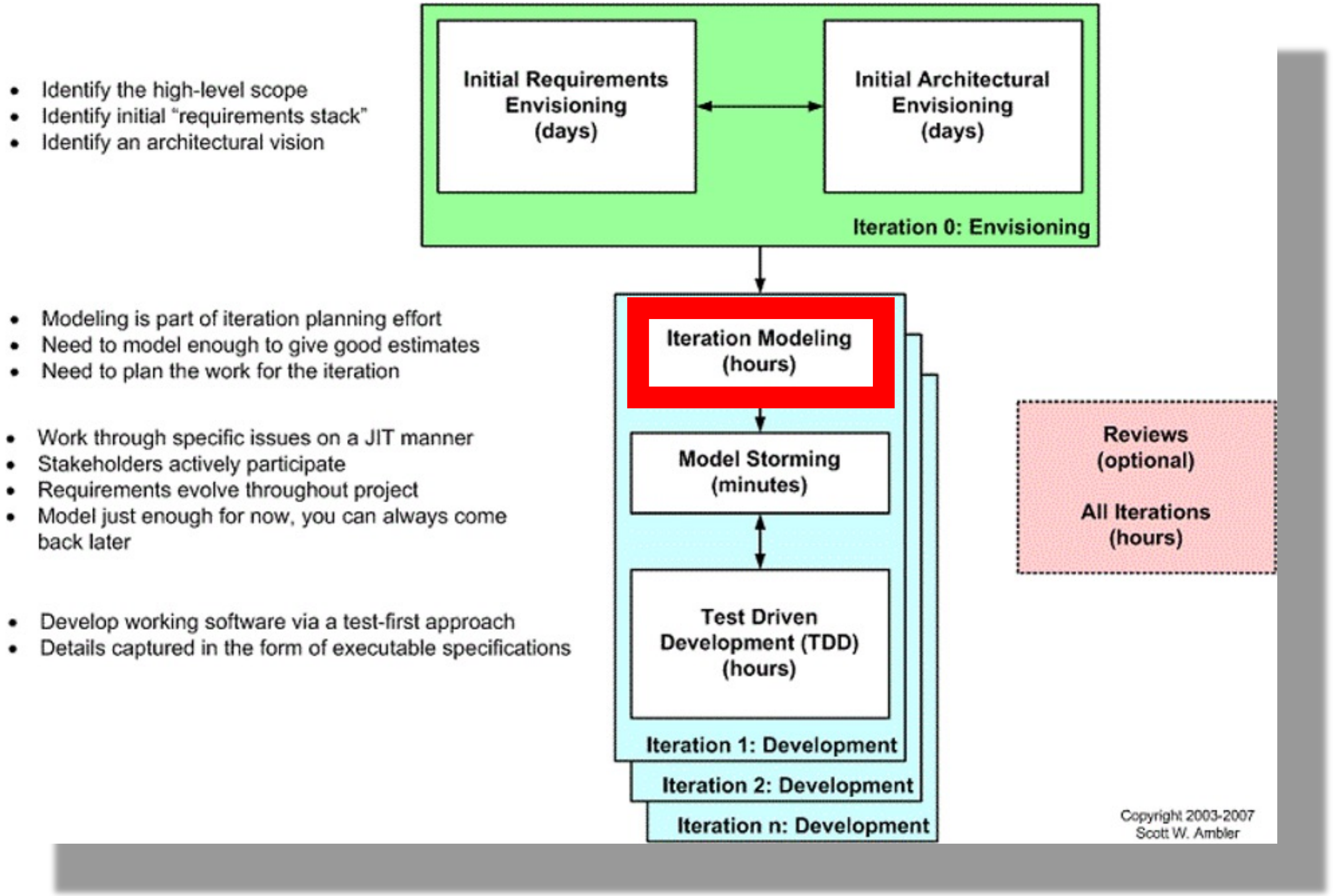
- Initial threat model should be very slim
- Capturing main business entities and security relationships



Identify the high-level scope
Identify initial "requirements stack"
Identify an architectural vision
Initial Requirements Envisioning (days)
Initial Architectural Envisioning (days)
Iteration 0: Envisioning
Modeling is part of iteration planning effort
Need to model enough to give good estimates
Need to plan the work for the iteration
Iteration Modeling (hours)
Work through specific issues on a JIT manner
Stakeholders actively participate
Requirements evolve throughout project
Model just enough for now, you can always come back later
Model Storming (minutes)
Develop working software via a test-first approach
Details captured in the form of executable specifications
Test Driven Development (TDD) (hours)
Iteration 1: Development
Iteration 2: Development
Iteration n: Development
Reviews (optional)
All Iterations (hours)
Copyright 2003-2007 Scott W. Ambler

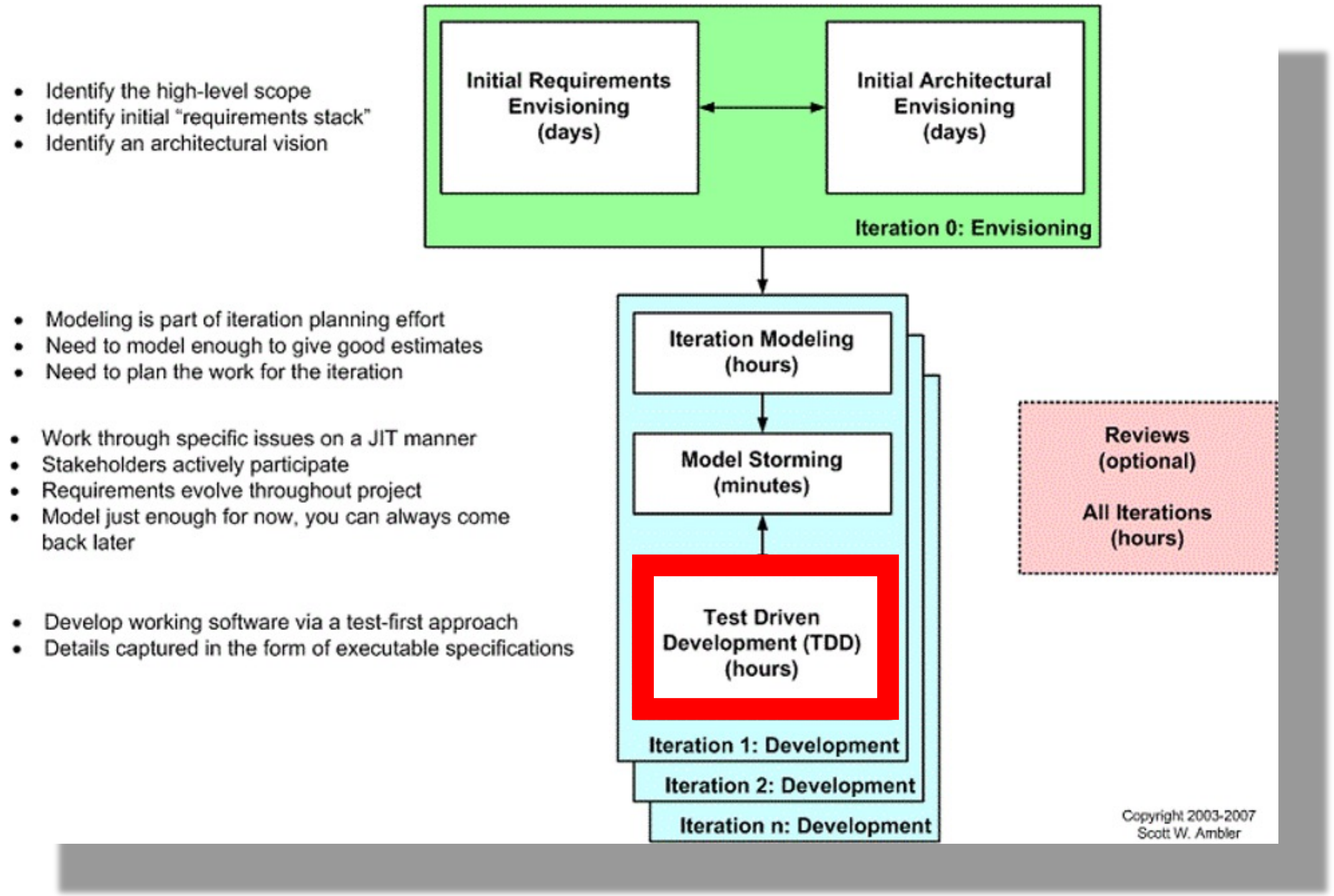
- Security Champions will drive Iteration Modeling

- Initial rough threat model will give good estimates of security pattern integration



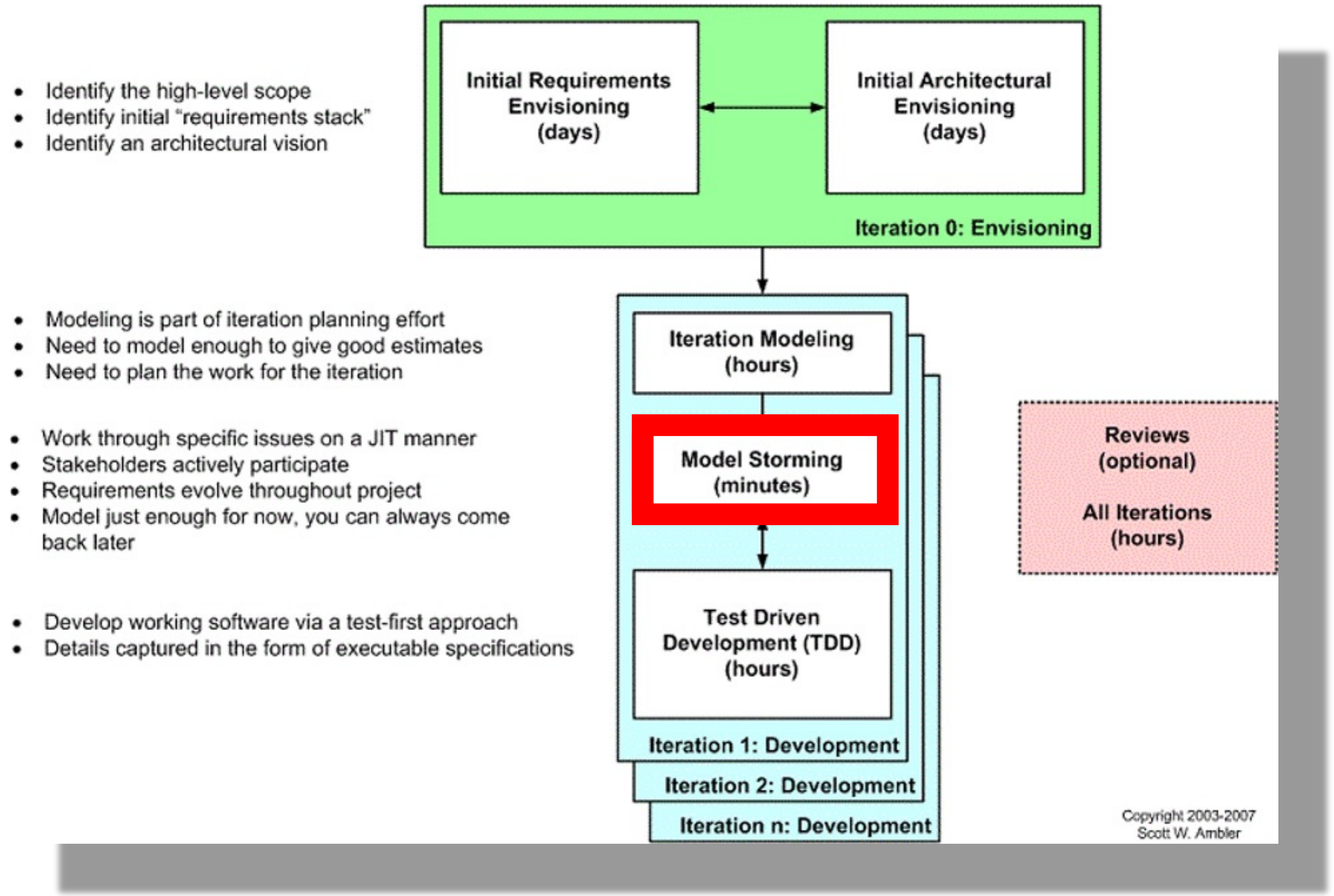
Identify the high-level scope
Identify initial "requirements stack"
Identify an architectural vision
Initial Requirements Envisioning (days)
Initial Architectural Envisioning (days)
Iteration 0: Envisioning
Modeling is part of iteration planning effort
Need to model enough to give good estimates
Need to plan the work for the iteration
Iteration Modeling (hours)
Work through specific issues on a JIT manner
Stakeholders actively participate
Requirements evolve throughout project
Model just enough for now, you can always come back later
Model Storming (minutes)
Develop working software via a test-first approach
Details captured in the form of executable specifications
Test Driven Development (TDD) (hours)
Reviews (optional)
All Iterations (hours)
Iteration 1: Development
Iteration 2: Development
Iteration n: Development
Copyright 2003-2007 Scott W. Ambler

- Tests should verify if the threats have been mitigated



Identify the high-level scope
Identify initial "requirements stack"
Identify an architectural vision
Initial Requirements Envisioning (days)
Initial Architectural Envisioning (days)
Iteration 0: Envisioning
Modeling is part of iteration planning effort
Need to model enough to give good estimates
Need to plan the work for the iteration
Iteration Modeling (hours)
Work through specific issues on a JIT manner
Stakeholders actively participate
Requirements evolve throughout project
Model just enough for now, you can always come back later
Model Storming (minutes)
Develop working software via a test-first approach
Details captured in the form of executable specifications
Test Driven Development (TDD) (hours)
Iteration 1: Development
Iteration 2: Development
Iteration n: Development
Reviews (optional)
All Iterations (hours)
Copyright 2003-2007 Scott W. Ambler

- A good threat model storming ('spike' session) can yield tangible output in 5 to 10 minutes
- Use the model storming to explore the impact of a security requirement or to think through a secure design issue



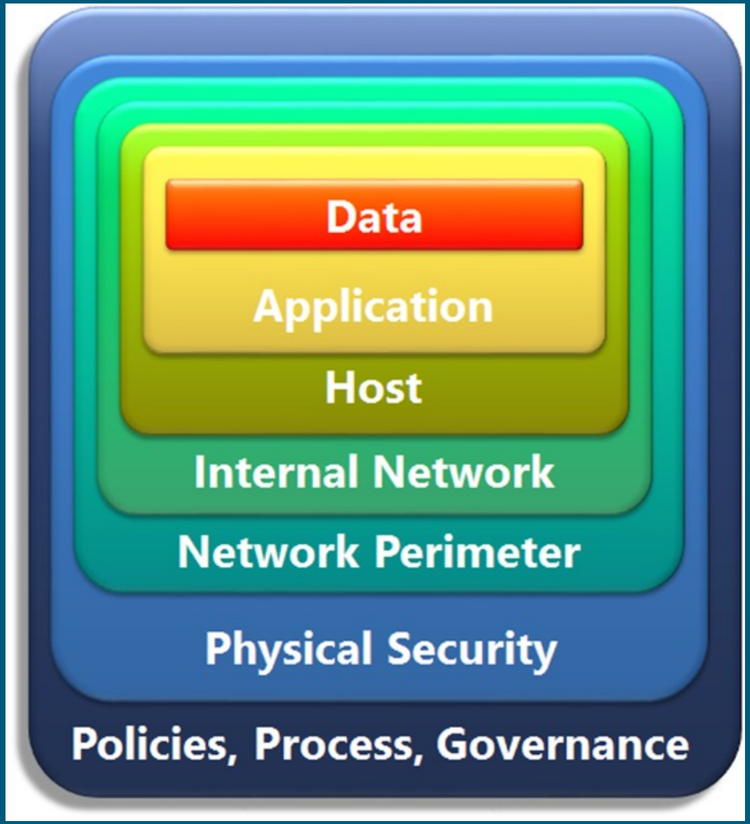
• Identify the high-level scope
• Identify initial "requirements stack"
• Identify an architectural vision
Initial Requirements Envisioning (days)
Initial Architectural Envisioning (days)
Iteration 0: Envisioning
• Modeling is part of iteration planning effort
• Need to model enough to give good estimates
• Need to plan the work for the iteration
Iteration Modeling (hours)
• Work through specific issues on a JIT manner
• Stakeholders actively participate
• Requirements evolve throughout project
• Model just enough for now, you can always come back later
Model Storming (minutes)
Reviews (optional)
All Iterations (hours)
• Develop working software via a test-first approach
• Details captured in the form of executable specifications
Test Driven Development (TDD) (hours)
Iteration 1: Development
Iteration 2: Development
Iteration n: Development
Copyright 2003-2007 Scott W. Ambler

Doing a Threat Model using RTMP

What are the steps?

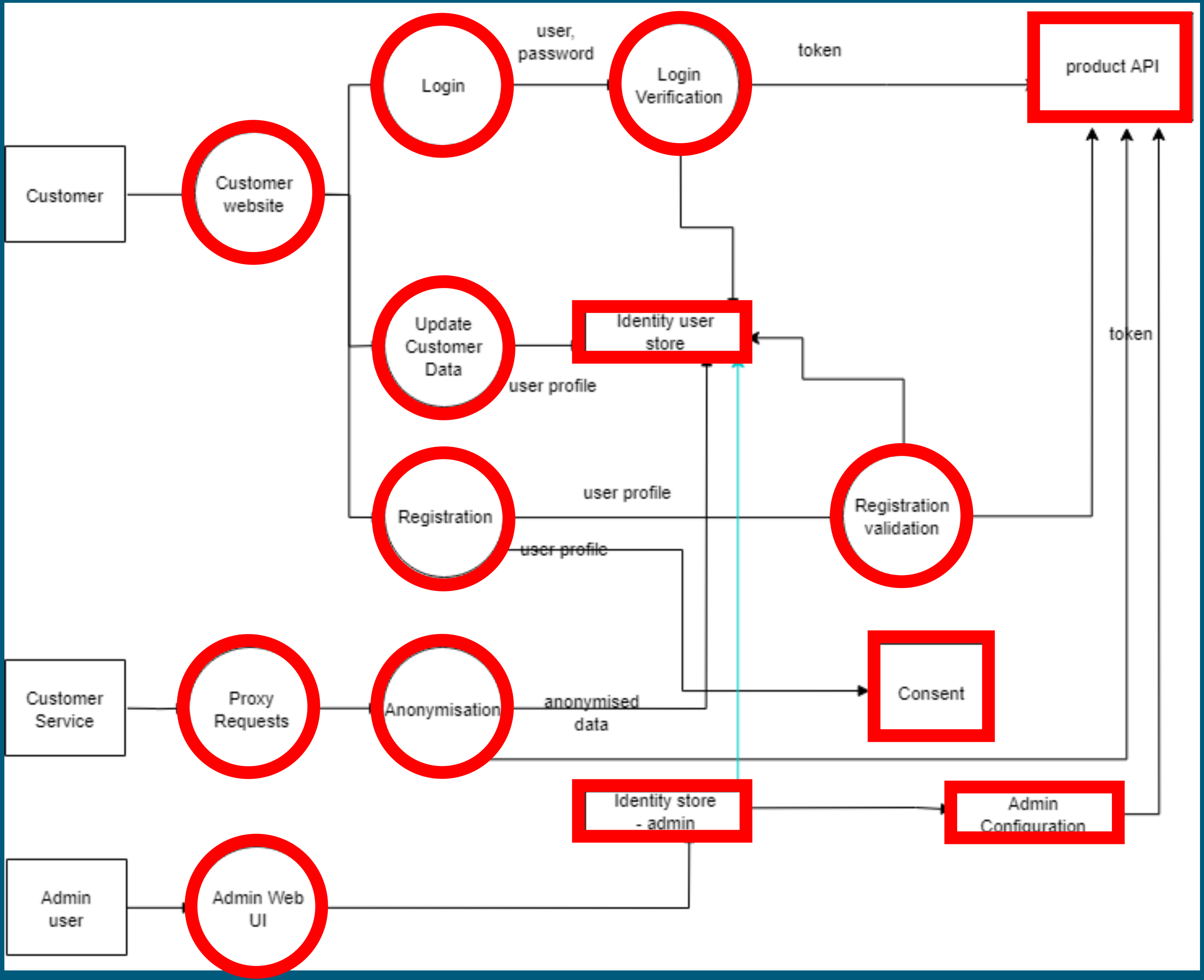
Modern Defense-in-Depth

- Each layer should have independent security
- Data is the most important layer
- Modern Defense-in-Depth is similar to the **Open Systems Interconnection (OSI)** model



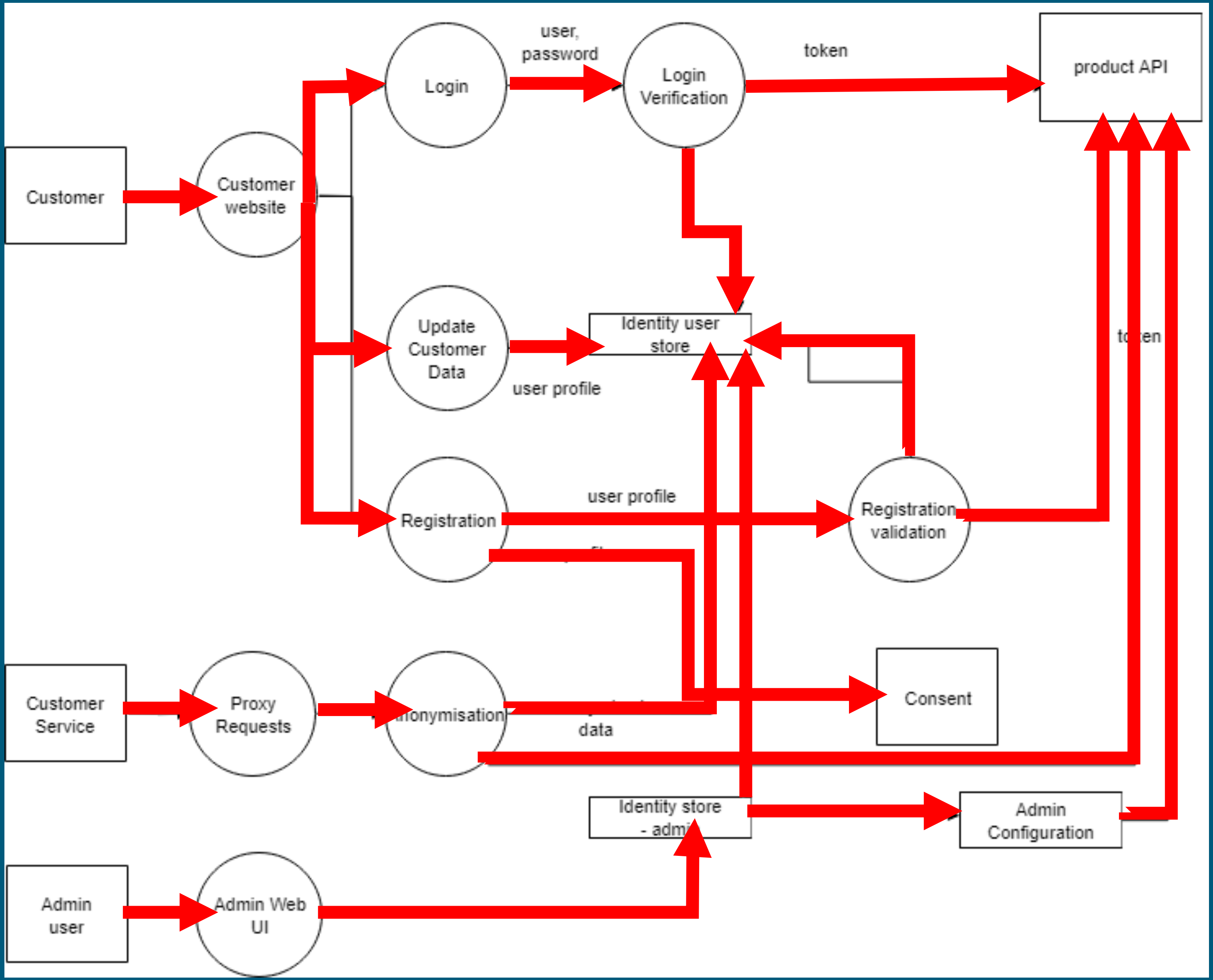
Identify Nodes

- Find out their criticality from knowledgeable team members
- Add relevant metadata to each node to describe the purpose of the node
    - e.g. process
    - Function
    - Sub-system
    - microservice



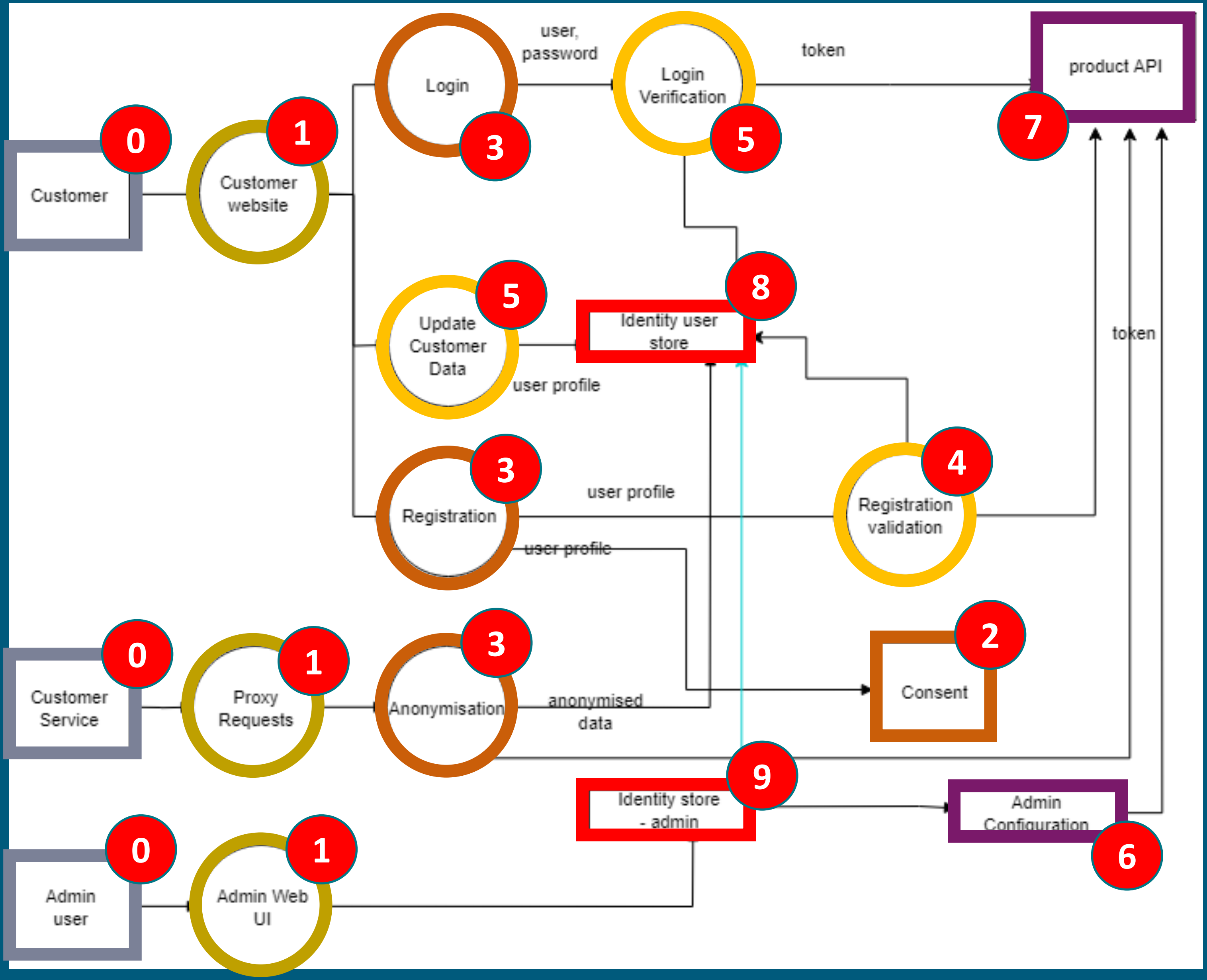
Identify Flows

- Find out the logical communication connections between nodes
- Make the arrows point in the direction of the “request” as opposed to the “response”
- The “request” is usually the command and is more useful to attacker (dangerous)
- The “response” is usually data but not as dangerous as the “request”



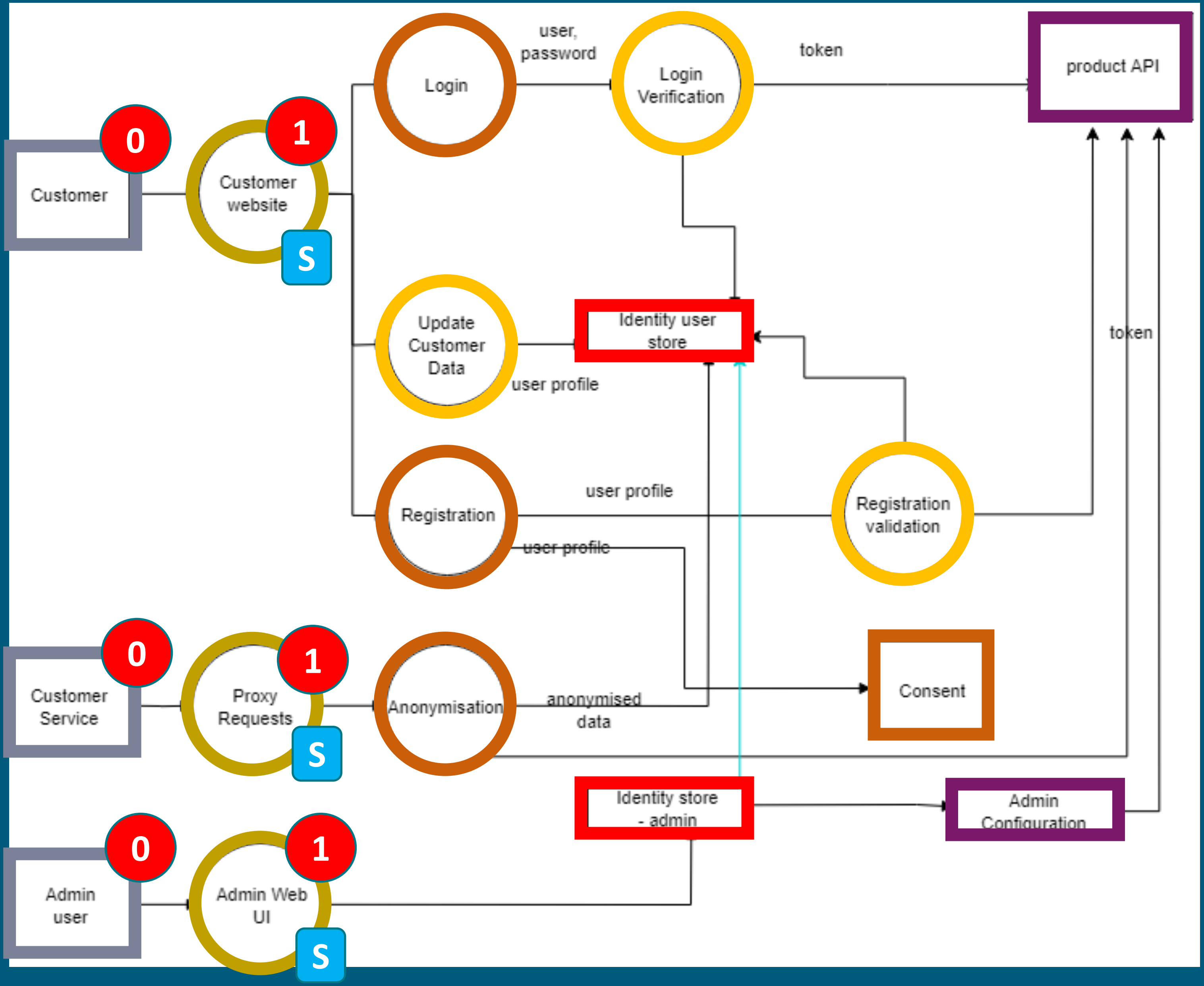
Number the Zones of Trust by criticality

- Not in control of system
- ▬ 0
- Boundary , external communication
- ▬ 1
-
- Low
- ▬ 2 3
-
- Medium
- ▬ 4 5
- High
- ▬ 6 7
- Critical , data hits the disk
- ▬ 8 9



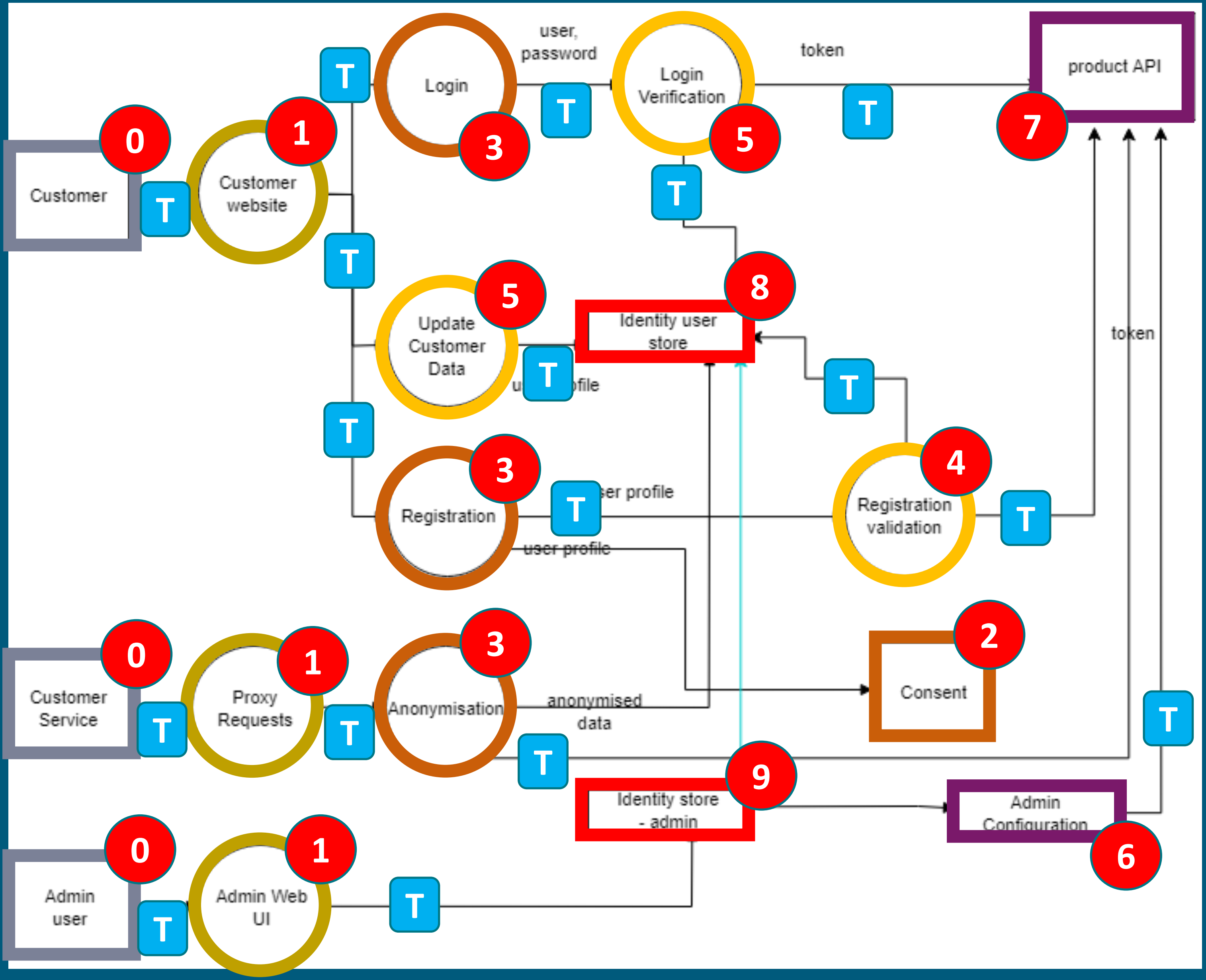
Apply STRIDE by Zone math

- Spoofing (target node)
  - 'Not in control of system' to any other
- Tampering
  - Less critical to more critical
- Repudiation
  - Spoofing +Tampering, or T+S
- Information Disclosure
  - More critical to less critical
- Denial of Service
  - 'Not in control of system' to any other
- Elevation Of Privilege
  - Less critical to more critical



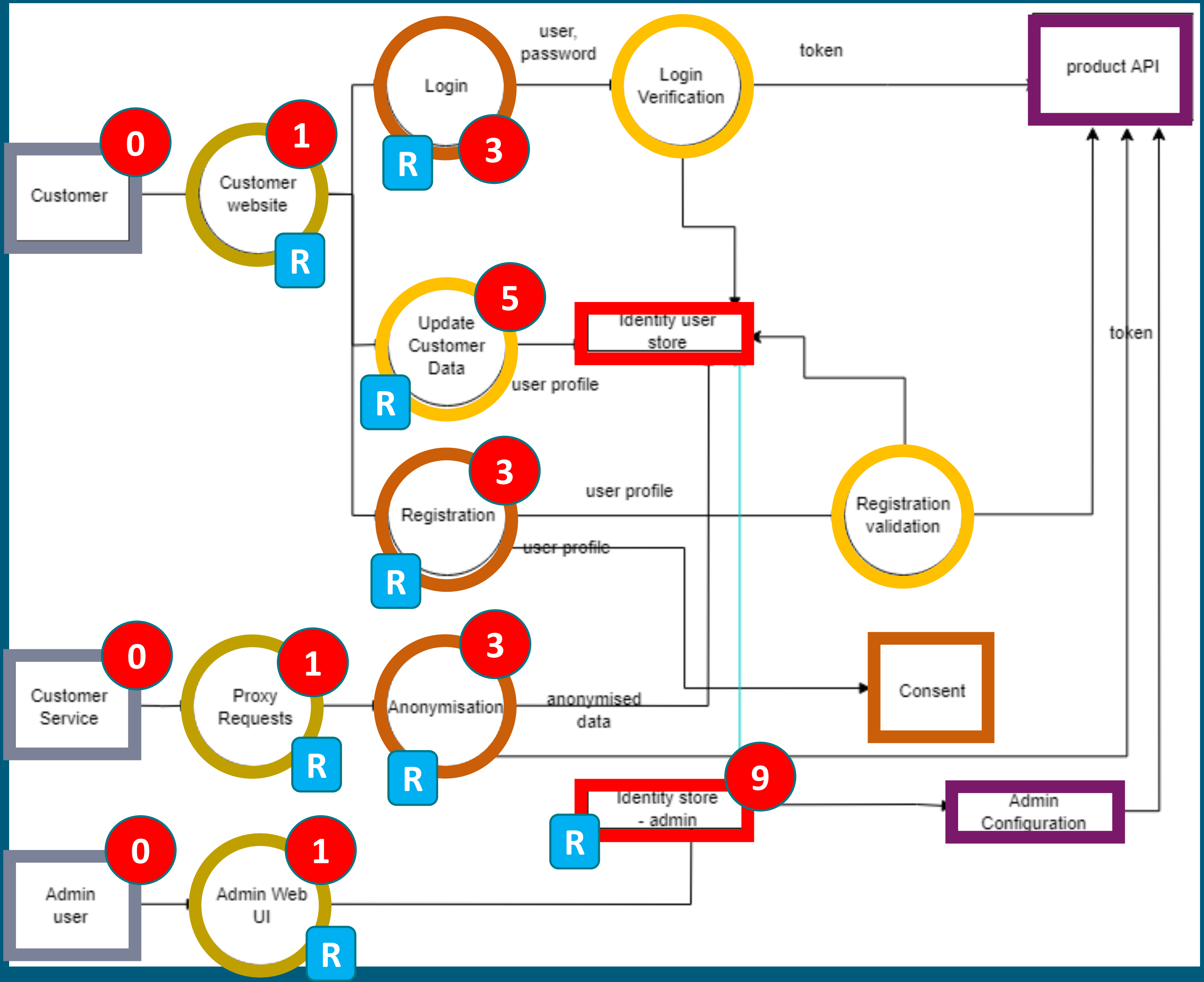
Apply STRIDE by Zone math

- Spoofing
  - 'Not in control of system' to any other
- Tampering (on flow)
  - Less critical to more critical
- Repudiation
  - Spoofing +Tampering, or T+S
- Information Disclosure
  - More critical to less critical
- Denial of Service
  - 'Not in control of system' to any other
- Elevation Of Privilege
  - Less critical to more critical



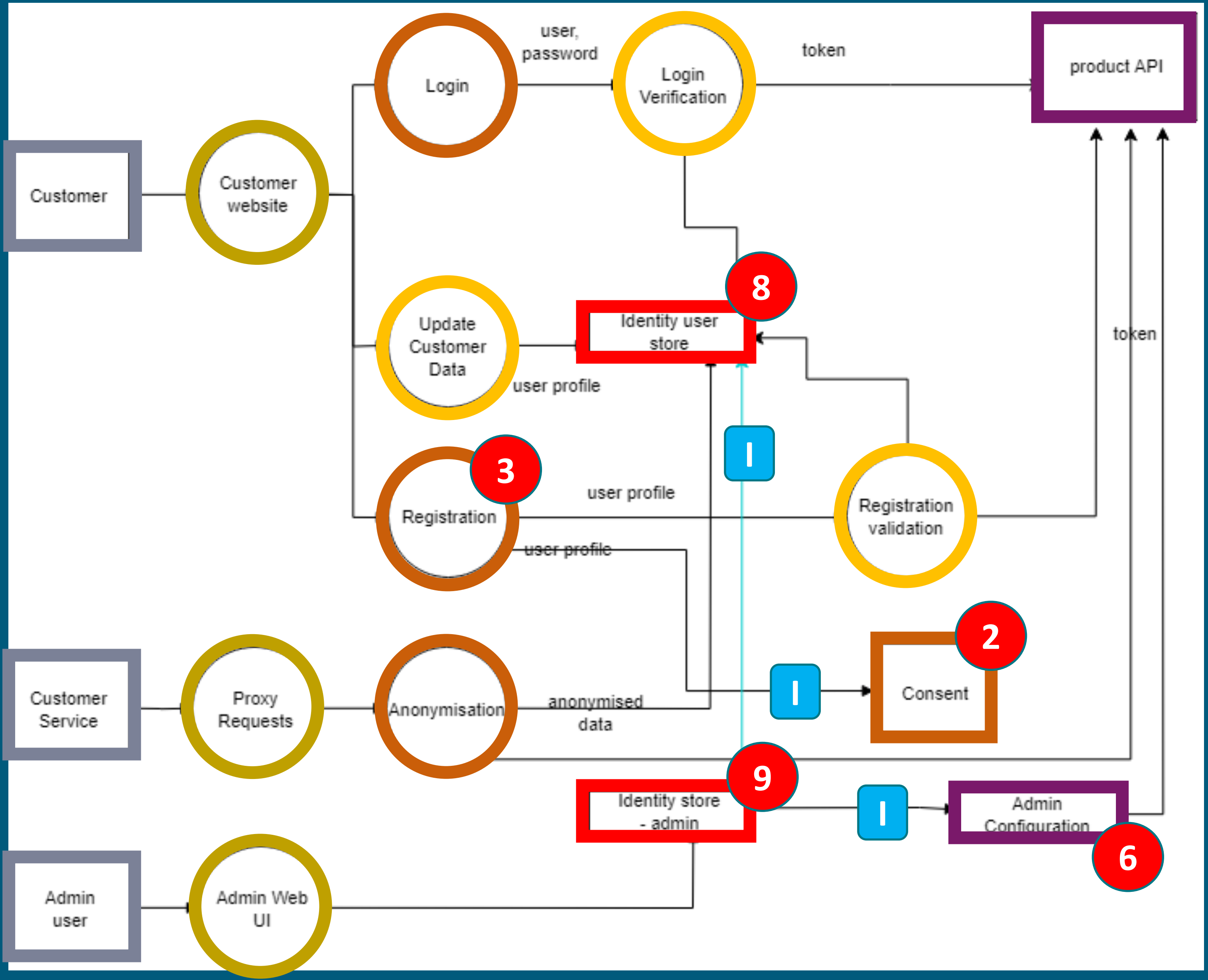
Apply STRIDE by Zone math

- Spoofing
  - 'Not in control of system' to any other
- Tampering
  - Less critical to more critical
- **Repudiation (target node)**
  - Spoofing +Tampering, or T+S
- Information Disclosure
  - More critical to less critical
- Denial of Service
  - 'Not in control of system' to any other
- Elevation Of Privilege
  - Less critical to more critical



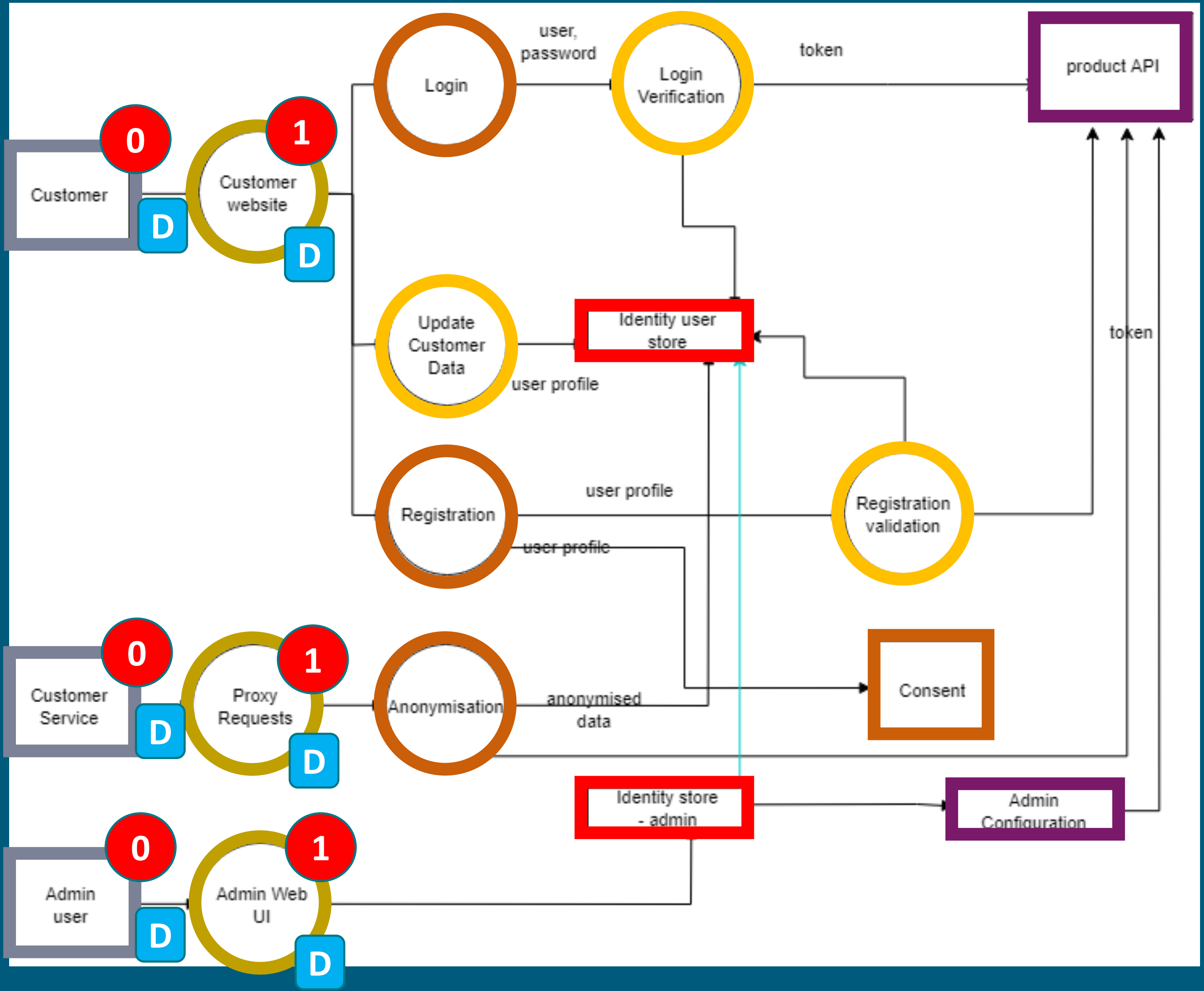
Apply STRIDE by Zone math

- Spoofing
  - 'Not in control of system' to any other
- Tampering
  - Less critical to more critical
- Repudiation
  - Spoofing +Tampering, or T+S
- Information Disclosure (on flow)
  - More critical to less critical
- Denial of Service
  - 'Not in control of system' to any other
- Elevation Of Privilege
  - Less critical to more critical



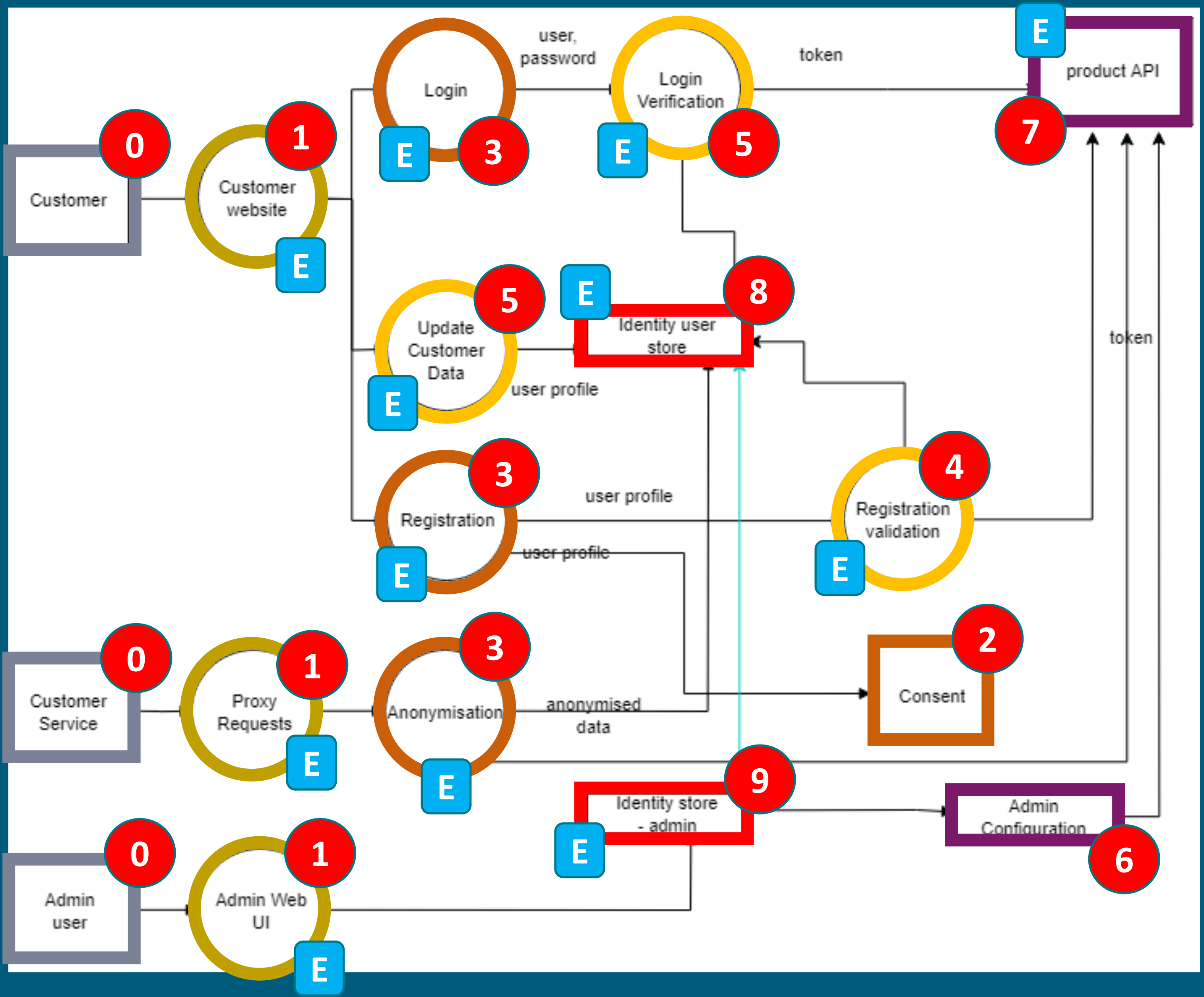
Apply STRIDE by Zone math

- Spoofing
  - 'Not in control of system' to any other
- Tampering
  - Less critical to more critical
- Repudiation
  - Spoofing +Tampering, or T+S
- Information Disclosure
  - More critical to less critical
- Denial of Service (target node and flow)
  - 'Not in control of system' to any other
- Elevation Of Privilege
  - Less critical to more critical



Apply STRIDE by Zone math

- Spoofing
  - 'Not in control of system' to any other
- Tampering
  - Less critical to more critical
- Repudiation
  - Spoofing +Tampering, or T+S
- Information Disclosure
  - More critical to less critical
- Denial of Service
  - 'Not in control of system' to any other
- Elevation Of Privilege (target node)
  - Less critical to more critical



Mitigation mapping

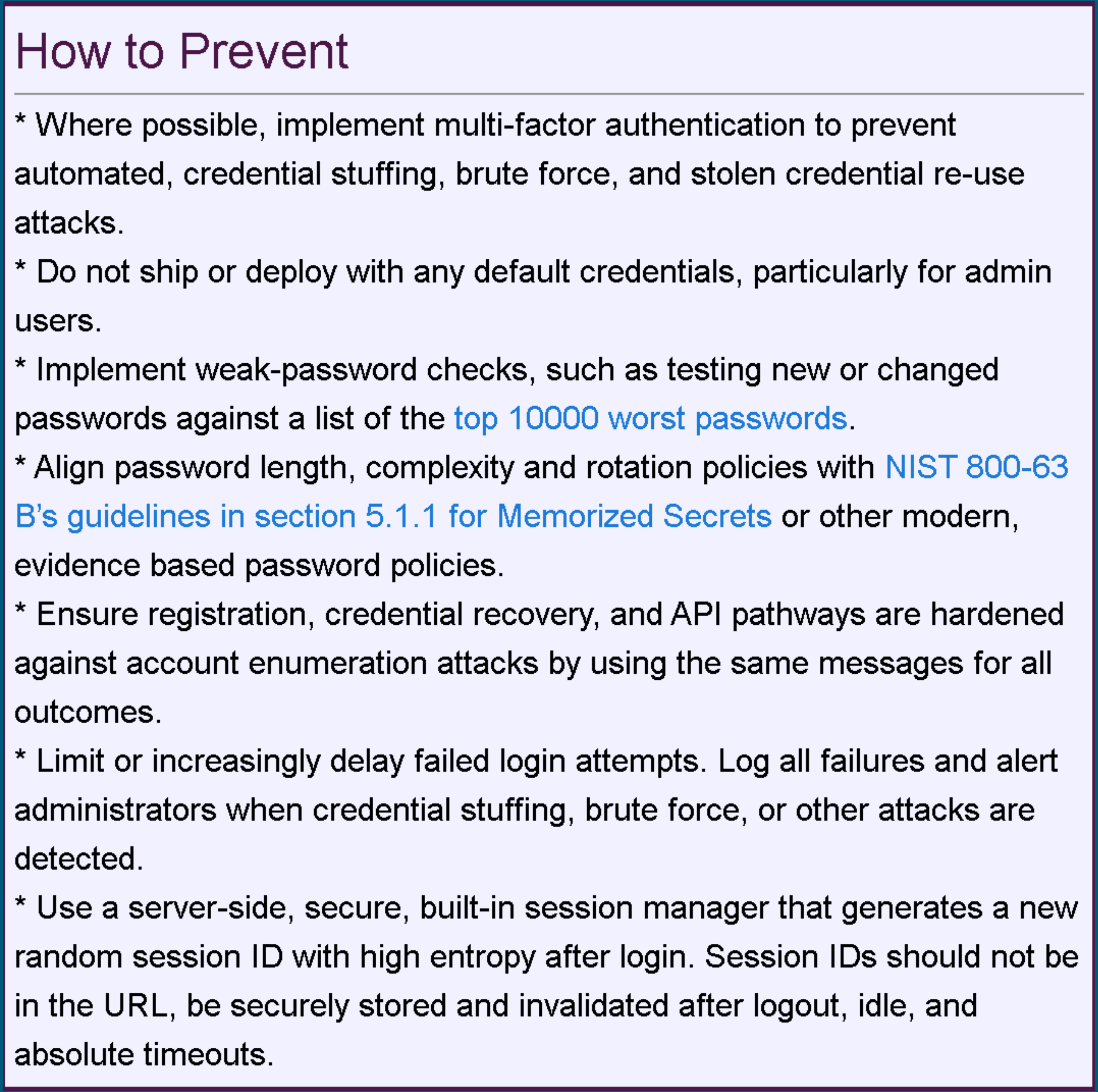
- Find framework mitigation based on STRIDE element mapped to a framework
- For example, find mitigation tactics for A07:2021-Identification and Authentication Failures

STRIDE	OWASP Top 10 (OT10) 2021
L6. Elevation of Privilege	A01:2021-Broken Access Control
L4. Information Disclosure	A02:2021-Cryptographic Failures
L5. Denial-Of-Service	A02:2021-Cryptographic Failures
L2. Tampering	A03:2021-Injection
L2. Tampering	A03:2021-Injection
L2. Tampering	A05:2021-Security Misconfiguration
L4. Information Disclosure	A05:2021-Security Misconfiguration
L5. Denial-Of-Service	A05:2021-Security Misconfiguration
L6. Elevation of Privilege	A05:2021-Security Misconfiguration
L5. Denial-Of-Service	A06:2021-Vulnerable and Outdated Components
L6. Elevation of Privilege	A06:2021-Vulnerable and Outdated Components
L1. Spoofing	A07:2021-Identification and Authentication Failures
L2. Tampering	A08:2021-Software and Data Integrity Failures
L3. Repudiation	A09:2021-Security Logging and Monitoring Failures
L6. Elevation of Privilege	A10:2021-Server-Side Request Forgery

A07:2021-Identification and Authentication Failures

Mitigation Tactics

https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures



How to Prevent

* Where possible, implement multi-factor authentication to prevent automated, credential stuffing, brute force, and stolen credential re-use attacks.
* Do not ship or deploy with any default credentials, particularly for admin users.
* Implement weak-password checks, such as testing new or changed passwords against a list of the top 10000 worst passwords.
* Align password length, complexity and rotation policies with NIST 800-63 B's guidelines in section 5.1.1 for Memorized Secrets or other modern, evidence based password policies.
* Ensure registration, credential recovery, and API pathways are hardened against account enumeration attacks by using the same messages for all outcomes.
* Limit or increasingly delay failed login attempts. Log all failures and alert administrators when credential stuffing, brute force, or other attacks are detected.
* Use a server-side, secure, built-in session manager that generates a new random session ID with high entropy after login. Session IDs should not be in the URL, be securely stored and invalidated after logout, idle, and absolute timeouts.

Thank You!

LinkedIn – www.linkedin.com/in/geoffrey-hill-61b7bb

Founder Tutamantic Ltd (threat modeling)
Email – geoff.hill@Tutamantic.com
Twitter – Tutamantic_Sec