AGILE SECURITY; OR, HOW TO DEFEND APPLICATIONS WITH FIVE-DAY RELEASE CYCLES
What does “Agile” mean, anyway?
The Agile Manifesto

- Individuals and interactions
- Working software
- Customer collaboration
- Responding to change
- Processes and tools
- Comprehensive documentation
- Contract negotiation
- Following a plan
Security Development Lifecycle

The SDL: Microsoft’s industry leading software security assurance process designed to protect customers by reducing the number and severity of software vulnerabilities before release.

- **Training**
  - Core training

- **Requirements**
  - Define quality gates/bug bar
  - Analyze security and privacy risk

- **Design**
  - Attack surface analysis
  - Threat modeling

- **Implementation**
  - Specify tools
  - Enforce banned functions
  - Static analysis

- **Verification**
  - Dynamic/Fuzz testing
  - Verify threat models/attack surface

- **Release**
  - Response plan
  - Final security review
  - Release archive

- **Response**
  - Response execution

**Education**

**Process**

**Accountability**

**Ongoing Process Improvements**
Challenges: Adapting SDL to Agile

- Iterative nature of Agile
- Projects may never end
- Just-in-time planning/YAGNI mentality
- General avoidance of project artifacts
- Emphasis on project/iteration backlogs
- General avoidance of automated tools
Challenges: Adapting SDL to Agile

- Iterative nature of Agile
- Projects may never end
- Just-in-time planning/YAGNI mentality
- General avoidance of project artifacts
- Emphasis on project/iteration backlogs
- General avoidance of automated tools
SDL “Classic” phased approach

- Fits spiral or waterfall…
- …but Agile doesn’t have phases
Idea: Move SDL to product backlog

- Very Agile…
- …but not secure
Idea: Do the full SDL every iteration

- Very secure…
- …but not Agile!
Iterative nature of Agile

- From the Principles Behind the Agile Manifesto:

  “Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.”
Idea: Drop some requirements

- But every requirement is, well, required
- Need to keep all requirements
- Need to reorganize into Agile-friendly form
SDL-Agile process

- Training:
  - Core training
- Requirements:
  - Define quality gates/bug bar
  - Analyze security and privacy risk
- Design:
  - Attack surface analysis
  - Threat modeling
- Implementation:
  - Specify tools
  - Enforce banned functions
  - Static analysis
- Verification:
  - Dynamic/fuzz testing
  - Verify threat models/attack surface
- Release:
  - Response plan
  - Final security review
  - Release archive
- Response:
  - Response execution
Three classes of requirements

- **Every Sprint**
  - Training
  - Threat modeling
  - etc...

- **One-Time Only**
  - Set up tracking
  - Upgrade compilers
  - etc...

- **Bucket**
  - Fuzz parsers
  - Create response plan
  - etc...
Requirements as backlog items

- One-time requirements get added to the Product Backlog (with deadlines)
- So do bucket requirements
- Every-sprint requirements go to the Sprint Backlog directly

**Product Backlog**
- Set up tracking system
- Upgrade to VS2010
- Fuzz image parser
- Fuzz network parser
- ...

**Sprint Backlog**
- Threat model new stored procedures
- Run static analysis
- ...

Agile sashimi

- At the end of every sprint:
  - All every-sprint requirements complete
  - No bucket items more than six months old
  - No expired one-time requirements
  - No open security bugs over the bugbar
Bug bar

- **Critical**
  - EoP: Remote Anonymous
  - Info Disc: HBI/PII

- **Important**
  - EoP: Local Anonymous
  - DoS: Asymmetric Persistent

- **Moderate**
  - Info Disc: LBI
  - DoS: Temporary

- **Low**
  - Info Disc: Random memory
Challenges: Adapting SDL to Agile

- Iterative nature of Agile
- Projects may never end
- Just-in-time planning/YAGNI mentality
- General avoidance of project artifacts
- Emphasis on project/iteration backlogs
- General avoidance of automated tools
Security Incident Response

- Because 2:00 AM Christmas morning is a poor time to hold a Scrum meeting…
Challenges: Adapting SDL to Agile

- Iterative nature of Agile
- Projects may never end
- Just-in-time planning/YAGNI mentality
- General avoidance of project artifacts
- Emphasis on project/iteration backlogs
- General avoidance of automated tools
Security bug tracking

- Must track bug cause
  - Buffer overflow
  - XSS
  - Etc
- And effect
  - STRIDE
- Important for bugbar criteria
Threat modeling

- "The cornerstone of the SDL"

- Data Flow Diagrams (DFDs)
  - STRIDE/element
  - Mitigations
  - Assumptions
  - External dependencies
Sidebar: Exception workflow

Level 1
Level 2
Level 3
Level 4
Level 5
Challenges: Adapting SDL to Agile

- Iterative nature of Agile
- Projects may never end
- Just-in-time planning/YAGNI mentality
- General avoidance of project artifacts
- Emphasis on project/iteration backlogs
- General avoidance of automated tools
Writing secure code

- 10% Writing **Security** Features
  - Cryptography
  - Firewalls
  - ACLs

- 90% Writing **Secure** Features
  - Overflow defense
  - Input validation
  - Output encoding
Secure code does not featurize

- Not a User Story
- Doesn’t go in the Product Backlog
- Can’t get prioritized in or out
- Can’t decide to not do security this sprint
Taskifying the SDL

- Some are straightforward...
  - Enable compiler switches
  - Run static analysis tools

- ...some are tougher (not actionable)
  - Avoid banned APIs
  - Access databases safely
Two strategies

- Verify these things by hand (alone or in pairs)
- Verify these things with tools
Challenges: Adapting SDL to Agile

- Iterative nature of Agile
- Projects may never end
- Just-in-time planning/YAGNI mentality
- General avoidance of project artifacts
- Emphasis on project/iteration backlogs
- General avoidance of automated tools
Static analysis requirements

- FxCop
- CAT.NET
- PREFast (/analyze)
- And/or your alternative tool(s) of choice

- These are “every-sprint” requirements
- Better still: Continuous Integration
Dynamic analysis requirements

- Fuzzers (homegrown)
  - File parsers
  - RPC interfaces
  - ActiveX controls
  - COM objects
- AppVerifier
- Passive HTTP traffic analysis
- And/or your alternative tool(s) of choice

- These are “bucket” requirements
- Or CI...
Secure coding libraries

- AntiXss/Web Protection Library
- StrSafe
- SafeInt

- Use always, check every sprint

<opinion>

This is the future of the SDL

</opinion>
Strengths: Adapting SDL to Agile

- Bucket activities easily move in & out of sprints
- Teams self-select best security activities
- SDL versioning is simpler and more current
- Each iteration is a gate
Strengths: Adapting SDL to Agile

- Bucket activities easily move in & out of sprints
- Teams self-select best security activities
- SDL versioning is simpler and more current
- Each iteration is a gate

“Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.”
Strengths: Adapting SDL to Agile

- Bucket activities easily move in & out of sprints
- Teams self-select best security activities
- SDL versioning is simpler and more current
- Each iteration is a gate

“At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.”
Strengths: Adapting SDL to Agile

- Bucket activities easily move in & out of sprints
- Teams self-select best security activities
- SDL versioning is simpler and more current
- Each iteration is a gate
SDL-Agile “versioning”

**SDL-Classic**
- Updated yearly
- Grandfather clause

**SDL-Agile**
- Updated at any time
- Automatic updating
Strengths: Adapting SDL to Agile

- Bucket activities easily move in & out of sprints
- Teams self-select best security activities
- SDL versioning is simpler and more current
- Each iteration is a gate
Each iteration is a gate

“Security and privacy are most effective when ‘built-in’ throughout the entire development lifecycle”

“Security is most effective when it is ‘baked-in’ from the start”

- This fits Agile perfectly
The Agile Manifesto

- Individuals and interactions
- Working software
- Customer collaboration
- Responding to change

- Processes and tools
- Comprehensive documentation
- Contract negotiation
- Following a plan
The *SDL-Agile* Manifesto

- Continuous, incremental effort
- Automated tasks
- Planned incident response
- Heroic pushes
- Manual processes
- Ad-hoc response
More Resources

- http://www.microsoft.com/sdl
- http://blogs.msdn.com/sdl

- My alias: bryansul