SnowBe Online Secure Software Development Life Cycle Plan

Framework Selection, Team Design & DevSecOps Roadmap

Sean Richard

sean@autom8ionlab.com · linkedin.com/in/a8l-sean-richard · +1.352.818.9694

NIST SSDF | DevSecOps | Secure SDLC | Threat Modeling | Supply-Chain Risk | Team Design

PROJECT SUMMARY

At a glance

Scenario	SnowBe Online grew $100M in three years on AWS-hosted sales, a WordPress storefront, on-premise servers and business apps, with PCI obligations and CEO concern about software supply-chain risk.
Decision	Adopt NIST SSDF as the secure development framework over Microsoft SDL: broad, vendor-neutral and easier for a new three-person team to operationalize.
Design	A project manager plus three specialized developers (architect / security champion, DevSecOps engineer, test and vulnerability response), working in Agile Scrum with DevSecOps.
Outcome	A 90-day implementation roadmap: foundation, design and pilot, operationalize.

SNOWBE CASE

SnowBe today: business success, software gaps

The business is successful, but the operating environment creates clear security and compliance pressure.

Business reality

- $100M growth in three years
- Most sales processed online through AWS
- Multiple stores in the U.S. and Europe

Data and control exposure

- Customer information kept indefinitely
- Broad access across servers and applications
- Logging and device governance need work

Immediate pressure points

- PCI obligations
- WordPress, patching, firmware, anti-virus, and backup gaps
- CEO concern about software supply chain risk

What this means for development

- SnowBe cannot treat software as a side project. It now needs a secure, repeatable development model that supports growth and compliance.
- The framework choice must work across cloud systems, web applications, internal business tools, and third-party components.
- A small new team needs structure first, then speed. Security has to be built in before the first release, not added later.

How I made the framework decision

I compared the options against SnowBe's risk profile, team maturity, and operating environment.

1
Start with risk
Online payments, stored customer data, weak access control, PCI exposure, and supply chain concern.

2
Compare frameworks
SSDF and Microsoft SDL both support secure development, but they organize adoption differently.

3
Check team maturity
Karen is building a three-person team from scratch, so the framework must be easy to operationalize.

4
Test environment fit
SnowBe spans AWS, WordPress, on-prem servers, remote laptops, and business apps.

5
Select long-term fit
The final choice had to support immediate control needs and future software growth.

Decision logic in one sentence

SnowBe needed a framework that was broad, practical, vendor-neutral, and easier to adopt for a small new team. That combination pushed the decision toward SSDF.

SNOWBE CASE

Selected framework: NIST Secure Software Development Framework

SSDF is the best strategic fit for SnowBe's current maturity level and mixed technology environment.

Why SSDF fits

- Vendor-neutral and flexible across cloud, web, and internal systems
- Designed to be integrated into any SDLC rather than forcing a single platform mindset
- Gives Karen a clean operating model for a new team
- Supports secure requirements, code protection, testing, and vulnerability response
- Aligns naturally with the NIST-based control direction already underway at SnowBe

Prepare the organization	Define requirements, roles, training, and toolchain expectations before code starts.
Protect the software	Secure code, releases, repositories, and sensitive components from tampering.
Produce well-secured software	Use threat modeling, secure coding, reviews, and testing to reduce defects early.
Respond to vulnerabilities	Track weaknesses, remediate them, and learn from root causes so the team improves over time.

SNOWBE CASE

Why SSDF beat Microsoft SDL for this specific case

Microsoft SDL is strong, but SSDF is the cleaner fit for SnowBe right now.

CRITERIA	MICROSOFT SDL	WHY SSDF WON
Adoption starting point	Strong for mature engineering practice	Easier for a new team building process from zero
Environment fit	Excellent, but read as more platform-centered	Broad fit across AWS, WordPress, on-prem, and custom tools
Governance alignment	Would require translation into SnowBe's current direction	Naturally matches the NIST language already in use
Small-team practicality	Good, but heavier to socialize from scratch	Simpler to translate into roles, responsibilities, and checkpoints
Decision result	Not rejected as weak	Chosen because it is the better fit now

Bottom line: Microsoft SDL remains a credible framework. It simply was not the best starting framework for SnowBe's current maturity, team size, and operating context.

Team design: Karen plus three specialized developers

The point is not a large team. The point is a small team with clear ownership.

LEAD

Karen

Project Manager and Secure Development Lead

DEVELOPER 1

Application Architect and Security Champion

- Own secure design and trust boundaries
- Guide secure coding standards and dependency review
- Lead authentication, authorization, and data protection choices

DEVELOPER 2

DevSecOps and Build Engineer

- Protect repositories, secrets, and CI/CD
- Run scanning, build controls, and release protections
- Reduce software supply chain risk before deployment

DEVELOPER 3

Test and Vulnerability Response Engineer

- Validate requirements through testing and logging checks
- Track defects and support remediation
- Capture lessons learned and root-cause findings

Methodology: Agile Scrum + DevSecOps

SnowBe needs fast feedback, but not speed without structure.

Why this works

- Agile sprints let Karen prioritize the highest-risk and highest-value work first.
- DevSecOps turns security into a normal sprint activity instead of a late review gate.
- A small team gets frequent feedback and can correct design mistakes early.
- This approach supports secure coding, peer review, dependency checks, and automated testing every cycle.

Sprint loop

Why not Waterfall or RAD alone?

Waterfall is too rigid for an immature process. RAD alone can create speed without enough security discipline.

SNOWBE CASE

Why secure software matters at SnowBe

In this case, secure software is a business protection strategy, not only an engineering preference.

Protect customer data	SnowBe stores customer information and purchase history. Weak software controls turn that into direct exposure risk.
Support compliance	The company handles online payments and needs stronger PCI-related discipline and better audit visibility.
Reduce operational risk	Secure development reduces the chance that bad code, bad dependencies, or poor access design disrupt business operations.
Preserve trust and growth	As SnowBe grows, software quality and security shape brand trust with customers, employees, and stakeholders.

SNOWBE CASE

What I would do first: the first 90 days

The framework decision only matters if it turns into operating habits quickly.

DAYS 1 TO 30

Set the foundation

- Confirm security requirements for the new software
- Assign roles, approval points, and backlog ownership
- Stand up repositories, branch protections, and secure build basics

DAYS 31 TO 60

Design and pilot

- Run initial threat modeling on the first product scope
- Choose approved libraries and document dependency rules
- Begin sprint-based development with review and testing gates

DAYS 61 TO 90

Operationalize

- Add automated scanning, logging checks, and release controls
- Track vulnerabilities and remediation deadlines
- Review what worked and adjust team practices before scale-up

Final recommendation for SnowBe

The most important lesson: secure software is a management decision, not only a technical one.

1. Adopt SSDF as the secure development framework.
2. Use Agile Scrum with DevSecOps so security becomes part of each sprint.
3. Give Karen and the three developers clear, specialized ownership from day one.
4. Treat secure software as a growth enabler that protects compliance, trust, and operations.

If SnowBe builds the framework, the roles, and the sprint discipline correctly now, it will spend less time repairing avoidable security debt later.

Thank you

Portfolio edition based on original Full Sail University coursework. Substantive project content is preserved; classroom-only submission headers were removed.

Sean Richard

sean@autom8ionlab.com · linkedin.com/in/a8l-sean-richard · +1.352.818.9694