Risk Management Policy

OverSiteAI, LLC

Document Version:	2.0
Effective Date:	January 1, 2025
Last Updated:	June 25, 2025
Last Reviewed:	June 27, 2025
Classification:	Restricted
Owner:	Chief Technology Officer
Approved By:	Chief Executive Officer

Table of Contents

Risk Management Policy

OversiteAI, LLC

Document Version: 2.0

Effective Date: January 1, 2025

Last Updated: June 25, 2025

Last Reviewed: June 27, 2025

Classification: Restricted

Owner: Chief Technology Officer

Approved By: Chief Executive Officer

1. Purpose and Objectives

1.1 Purpose

At OversiteAI, we recognize that effective risk management doesn't require a massive bureaucracy or expensive tools. This policy establishes our pragmatic approach to managing risks that could impact our ability to deliver secure, reliable software solutions to our customers. We've designed this framework to be practical for our team of under 20 people while still meeting SOC2 requirements and protecting our business.

Our unique architecture—where our software runs entirely on customer premises without any access to their data—eliminates many traditional security risks. This policy focuses on the risks that actually matter to our business model and leverages our architectural advantages as a primary risk mitigation strategy.

SOC2 Mapping: CC1.1, CC1.2 (Control Environment - Demonstrates Commitment)

1.2 Objectives

We've structured our risk management approach to achieve realistic goals that align with our size and resources:

Focus on What Matters: We concentrate on risks that could genuinely impact our ability to develop, deliver, and support our software. We don't waste time on theoretical risks that our architecture already prevents.

Leverage Built-in Controls: Rather than purchasing expensive risk management tools, we maximize the security features already included in our Azure infrastructure and development platforms. Microsoft invests billions in security so we don't have to.

Integrate with Daily Work: Risk management isn't a separate activity—it's part of how we develop software, support customers, and run our business. Every team member contributes to risk identification and mitigation through their regular activities.

Scale Appropriately: Our controls and processes are designed to work effectively with our current team size while providing a foundation for growth. We focus on automation over manual processes wherever practical.

Demonstrate Competence: While we're a small company, we show auditors and customers that we understand security risks and address them intelligently, emphasizing smart design choices over expensive controls.

SOC2 Mapping: CC1.3, CC1.4 (Control Environment - Structure and Competence)

1.3 Scope

This policy applies to all aspects of our business that could introduce meaningful risk:

Our Team: All employees, contractors, and anyone with access to our systems or source code. With our small team, everyone plays a role in risk management—it's not delegated to a non-existent security department.

Our Technology: The Azure infrastructure we operate from, our development environments, our source code repositories, and the software we deliver to customers. We focus particularly on protecting our intellectual property since we don't handle customer data.

Our Relationships: Vendors who provide critical services (like Azure), customers who rely on our software, and any third parties we work with. We keep these relationships simple and well-documented.

What's Explicitly Out of Scope: Customer data risks remain the responsibility of our customers since our architecture prevents us from accessing their data. This architectural decision eliminates entire categories of risk that other software companies must manage.

SOC2 Mapping: CC3.1 (Risk Assessment - Identifies and Assesses Risk)

2. Risk Management Principles

2.1 Core Principles

Our risk management approach is built on practical principles that reflect both our size and our security-first architecture:

Smart Design Over Complex Controls: We've architected our solution to eliminate risks rather than manage them. By running entirely on customer infrastructure with no data access, we've removed the need for complex data protection controls that would be impractical for our size. When we can't eliminate a risk through design, we prefer simple, automated controls over complex manual processes.

Risk-Based Resource Allocation: We openly acknowledge that we can't address every possible risk with our current resources. Instead, we focus our limited time and budget on the risks that could actually hurt our business or our customers. Low-probability, low-impact risks are documented but not actively managed until we have more resources.

Automation Before Administration: Wherever possible, we use automated tools and Azure's built-in security features rather than creating manual processes that require dedicated staff. For example, Azure Security Center continuously monitors our infrastructure without requiring a security operations team.

Transparency with Stakeholders: We're honest with auditors, customers, and partners about our size and approach. We don't pretend to have enterprise-scale security operations. Instead, we demonstrate how our architectural choices and automated controls provide appropriate security for our business model.

Continuous but Practical Improvement: We review and improve our risk management practices, but at a frequency that makes sense for our size. Quarterly reviews strike the right balance between staying current and not overwhelming our team with administrative tasks.

SOC2 Mapping: CC5.1, CC5.2 (Control Activities - Selection and Development)

2.2 Risk Appetite Statement

Our risk appetite reflects both our business goals and our resource constraints. We're explicit about what risks we will and won't accept:

Zero Tolerance—We Never Compromise On:

- **Intellectual Property Protection**: Our source code is our primary asset. We implement strong controls here regardless of cost or effort.
- **Architectural Integrity**: We never create backdoors or methods to access customer data, even if requested. This principle protects both us and our customers.
- **Legal and Ethical Standards**: We won't cut corners on legal compliance or ethical business practices, no matter the business pressure.
- **Employee Trust**: We maintain zero tolerance for insider threats through careful hiring and access controls.

Low Tolerance—We Invest Significantly to Prevent:

- **Service Availability**: Our customers depend on our software. We target 99.5% uptime through Azure's reliable infrastructure and careful deployment practices.

- **Critical Security Vulnerabilities**: We patch critical vulnerabilities immediately, even if it means interrupting other work.
- **Key Person Dependencies**: With our small team, we actively work to ensure no single person can cripple our operations if they leave.

Moderate Tolerance—We Manage Within Resource Constraints:

- **Process Inefficiencies**: Some of our processes could be more efficient, but optimization takes a back seat to security and delivery.
- **Competitive Pressures**: We accept that larger competitors might move faster in some areas while we focus on security and reliability.
- **Non-Critical Technical Debt**: We maintain a backlog of improvements we'd like to make when resources permit.

Higher Tolerance—We Accept as Part of Growth:

- **Innovation Risks**: We try new technologies and approaches, accepting that some won't work out.
- **Market Timing**: We might miss some market opportunities while ensuring our security foundation is solid.
- **Resource Constraints**: We accept that some desirable controls must wait until we have more staff or budget.

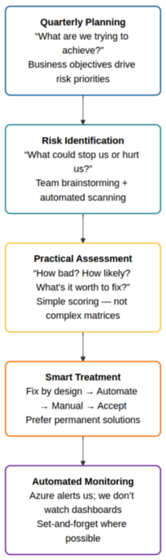
This honest appetite statement helps auditors understand our decision-making and demonstrates mature risk thinking despite our size.

SOC2 Mapping: CC3.2, CC3.3 (Risk Assessment - Fraud Risk and Changes)

3. Risk Management Framework

3.1 Framework Components

Our risk management framework is intentionally simple—complex frameworks require dedicated staff we don't have. We follow a straightforward cycle that integrates with our quarterly business planning:


Quarterly Planning
"What are we trying to achieve?"
Business objectives drive risk priorities
Risk Identification
"What could stop us or hurt us?"
Team brainstorming + automated scanning
Practical Assessment
"How bad? How likely? What's it worth to fix?"
Simple scoring — not complex matrices
Smart Treatment
Fix by design → Automate → Manual → Accept
Prefer permanent solutions
Automated Monitoring
Azure alerts us; we don't watch dashboards
Set-and-forget where possible

Risk Management Flow

3.2 Risk Categories

We organize risks into categories that match how our business actually operates, not theoretical frameworks:

Product & Technology Risks

These are risks to our core product and the technology it depends on:

- Source code theft or corruption (our nightmare scenario)
- Critical bugs that affect all customers simultaneously
- Azure platform failures or changes that break our software
- Supply chain attacks through development dependencies
- Technology choices that become obsolete or unsupported

Customer Success Risks

Risks that could damage our relationship with customers:

- Our software causing issues in customer environments
- Inability to provide timely support due to team size
- Security vulnerabilities that customers must patch
- Documentation gaps that lead to misconfigurations
- SLA violations that trigger penalties

Business Operations Risks

Risks to our ability to function as a company:

- Key employee departure (bus factor)
- Cash flow interruptions affecting operations
- Vendor lock-in with critical services
- Compliance failures (like SOC2) affecting sales
- Intellectual property disputes

Architecture-Prevented Risks

We explicitly document risks that our architecture eliminates, as this helps auditors understand our approach:

- Customer data breaches (we can't access what we can't see)
- Data residency violations (data stays with customer)
- Privacy regulation violations for customer data
- Cross-customer data leakage
- Insider threats to customer data

This categorization helps us focus our limited resources on risks we can actually face while getting credit for risks we've eliminated by design.

3.3 Integration with Business Processes

Risk management isn't a separate activity at OversiteAI—it's woven into how we work:

Development Process: Every pull request considers security implications. Our code review checklist includes security considerations, and Azure DevOps scanning automatically checks for vulnerabilities. We don't need a separate security review process because security is part of development.

Customer Onboarding: When bringing on new customers, we assess any unique risks they might introduce (unusual deployment requirements, compliance needs, etc.). This assessment is part of our standard onboarding checklist, not a separate risk process.

Vendor Selection: Before committing to any new service or tool, we evaluate the risk it introduces. Can we easily migrate away? What happens if they have an outage? This evaluation happens naturally as part of procurement, documented in our decision logs.

Incident Response: Every incident automatically becomes a risk learning opportunity. Our post-incident reviews always ask: "What risk did we miss?" and "How do we prevent this category of issue?" This turns reactive incidents into proactive risk mitigation.

By embedding risk thinking into existing processes, we ensure it actually happens without creating additional overhead for our small team.

SOC2 Mapping: CC1.4, CC2.1, CC3.1 (Integration with Entity Operations)

4. Roles and Responsibilities

4.1 Simple Structure for a Small Team

We don't have layers of management or dedicated risk committees. Instead, we've assigned risk responsibilities to existing roles in ways that make sense for our size. Everyone wears multiple hats, and that's okay—we just need to be clear about who does what.

CEO (Ultimate Risk Owner)

Our CEO sets the tone and makes the big calls on risk. They don't manage day-to-day risk activities but they:

- Approve our overall risk appetite and any changes to it
- Make final decisions on accepting high-impact risks
- Ensure we have adequate resources for critical risk mitigation (within our means)
- Review our risk status quarterly with the CTO
- Sign off on this policy annually

SOC2 Mapping: CC1.1, CC1.2 (Control Environment)

CTO (Operational Risk Manager)

Our CTO handles the practical side of risk management as part of running technology and operations:

- Maintains our simple risk register (a spreadsheet is fine for our size)
- Coordinates quarterly risk reviews with the team
- Ensures Azure Security Center alerts are configured and monitored
- Makes day-to-day decisions on risk treatment for technical risks
- Reports significant risks to the CEO when they need attention

SOC2 Mapping: CC1.3, CC1.4, CC3.1 (Risk Assessment)

Development Team (Front-Line Risk Identifiers)

Our developers are our first line of defense since they touch the code and infrastructure daily:

- Follow secure coding practices in every pull request
- Report security vulnerabilities or concerns immediately
- Participate in quarterly risk brainstorming sessions
- Implement technical risk mitigations as part of normal development
- Keep dependencies updated and monitor for vulnerabilities

SOC2 Mapping: CC2.1, CC2.2 (Information and Communication)

All Team Members (Risk Awareness)

In a company our size, everyone needs basic risk awareness:

- Understand that security is everyone's responsibility
- Report anything suspicious or concerning
- Follow our security policies (like password requirements)
- Participate in annual security training
- Suggest improvements when they spot inefficiencies

4.2 What We Don't Have (And Why That's Okay)

We're transparent about what we don't have because it helps set realistic expectations:

No Risk Management Committee: With fewer than 20 people, a formal committee would be half our company. Instead, risk discussions happen in our regular leadership meetings and quarterly all-hands reviews.

No Dedicated Risk Manager: The CTO handles risk management as part of their role. When we grow beyond 50 people, we might reconsider, but for now, this integrated approach works.

No Complex RACI Matrices: Everyone knows their job. We don't need complicated responsibility charts that nobody reads.

5. Risk Assessment Process

5.1 How We Actually Identify Risks

We keep risk identification simple and integrated with our regular work:

Quarterly Team Sessions (2 hours max)

Every quarter during planning, we spend time as a team asking: "What could go wrong?" We use simple prompts:

- What keeps you up at night about our product?
- What would happen if [key service/person/tool] disappeared tomorrow?
- What are customers complaining about that could get worse?
- What are we putting off that could bite us later?

SOC2 Mapping: CC3.1, CC3.2 (Risk Assessment Process)

Continuous Automated Scanning

We let tools do the heavy lifting for technical risks:

- Azure Security Center continuously scans our infrastructure
- GitHub Dependabot alerts us to vulnerable dependencies
- Azure Monitor tracks availability and performance metrics
- These tools feed our quarterly reviews with real data

Learning from Reality

Every incident or near-miss gets logged and reviewed:

- Customer-reported issues reveal risks we missed
- Failed deployments show process weaknesses
- Support tickets highlight documentation gaps
- Post-incident reviews always ask: "Is this a one-off or a pattern?"

5.2 Our Practical Risk Scoring

We use simple 1-3 scoring because 1-5 scales create false precision for a small company:

Impact Levels (What It Means for OversiteAI)

Level	Operational Reality	Customer Impact
High (3)	Can't deliver core services	Multiple customers affected or one enterprise customer furious
Medium (2)	Significant slowdown but still functioning	Individual customers annoyed but not leaving

Low (1)	Internal inefficiency only	Customers don't notice

Likelihood Levels (Our Simplified View)

Level	What It Means	Rough Probability
Likely (3)	We expect this to happen	>50% chance this year
Possible (2)	Could reasonably happen	10-50% chance this year
Unlikely (1)	Would be surprised if it happened	<10% chance this year

Risk Priority = Impact × Likelihood

- 7-9: High Priority (address immediately)
- 4-6: Medium Priority (plan for next quarter)
- 1-3: Low Priority (document and monitor)

SOC2 Mapping: CC3.3, CC3.4 (Risk Analysis and Evaluation)

5.3 Keeping It Real with Examples

Here's how we score real risks to show the system in action:

Example 1: Azure Region Outage

- Impact: High (3) - We can't serve any customers
- Likelihood: Unlikely (1) - Azure is reliable
- Score: 3 × 1 = 3 (Low Priority)
- Treatment: Accept (Azure's SLA is our compensating control)

Example 2: Senior Developer Leaves

- Impact: Medium (2) - Development slows significantly
- Likelihood: Possible (2) - People change jobs
- Score: 2 × 2 = 4 (Medium Priority)
- Treatment: Mitigate with documentation and cross-training

Example 3: Customer Data Breach

- Impact: High (3) - If it could happen, it would be catastrophic
- Likelihood: N/A - Our architecture prevents this
- Score: N/A

- Treatment: Architectural elimination (our best control!)

6. Risk Treatment That Actually Works

6.1 Our Treatment Hierarchy

We prioritize risk treatments based on what's practical for our size and most effective long-term:

1. Eliminate by Design (Our Favorite)

When possible, we architect away risks entirely:

- No customer data access = no data breach risk
- Serverless functions = no server patching
- Managed services = no infrastructure maintenance
- Cloud-only = no physical security needs

SOC2 Mapping: CC6.1, CC7.1 (System Operations)

2. Automate the Control (Our Second Choice)

If we can't eliminate it, we automate it:

- Azure Security Center for vulnerability scanning
- Automated dependency updates via Dependabot
- Azure AD for access control (no manual user management)
- Automated backups and disaster recovery

SOC2 Mapping: CC6.6, CC7.2 (Automated Controls)

3. Build It Into Process (When Manual is Necessary)

Some things require human judgment, so we embed them in existing workflows:

- Code review includes security checklist
- Customer onboarding includes risk assessment
- Quarterly planning includes risk review
- Incident response includes risk learning

4. Accept with Transparency (Sometimes the Right Answer)

We're honest when accepting risks makes business sense:

- Document why we're accepting it

- Set a review trigger (revenue threshold, team size, etc.)
- Communicate to stakeholders if material
- Monitor for changes that affect the decision

SOC2 Mapping: CC3.4, CC9.1 (Risk Mitigation and Acceptance)

6.2 Making Treatment Decisions

For each medium or high priority risk, we document:

The Business Decision

- What's the real cost of the risk occurring?
- What's the real cost of preventing it?
- Which option moves the business forward?

The Practical Plan

- Who's doing what by when?
- What existing process will include this?
- How will we know it's working?
- When will we check if it's still relevant?

The Success Metrics

We keep metrics simple and measurable:

- Did the risk occur? (Yes/No)
- If yes, was impact higher or lower than expected?
- Is the control actually being followed?
- Should we adjust our approach?

6.3 Resource Reality Check

Before committing to any risk treatment, we ask:

- Can we actually do this with current staff?
- Will this prevent us from delivering features?
- Is there a cheaper automation available?
- What breaks if we don't do this?

If the answer suggests we're overcommitting, we either:

- Find a simpler approach
- Delay until we have resources

- Accept the risk with documentation
- Never pretend we'll do something we won't

SOC2 Mapping: CC3.3, CC4.1 (Monitoring and Resources)

7. Our Risk Register (Keeping It Simple)

7.1 What We Track and Why

We maintain our risk register in a simple spreadsheet—no expensive GRC tools needed. For each risk that scores 4 or higher (medium/high priority), we track:

The Basics

- Simple ID (RISK-2024-001, etc.)
- Plain English description (what could go wrong?)
- Who owns it (actual person, not a committee)
- Current priority score (1-9)
- What we're doing about it

The Reality Check

- Are our controls actually working? (Yes/No/Partially)
- When did we last verify this? (Date)
- Has anything changed? (Brief note)
- When do we check again? (Next quarter unless something changes)

SOC2 Mapping: CC3.2, CC5.1 (Risk Identification and Management)

7.2 Our Actual Risk Register (Living Examples)

Here are real risks we track, showing how practical risk management works for a small software company:

RISK-2024-001: GitHub/Azure DevOps Compromise

- What Could Go Wrong: Someone gains unauthorized access to our source code
- Owner: CTO
- Current Score: 6 (High Priority - Impact: 3, Likelihood: 2)
- What We Do About It:
    - Enforce MFA on all accounts (no exceptions)
    - Use Azure AD SSO where possible
    - Review access quarterly (takes 30 minutes)

  - Alert on any new user or permission change
- Reality Check: Controls working? Yes. Last checked? [Date]. Next review? [Next Quarter]

RISK-2024-002: We Lose Our Best Developer

- What Could Go Wrong: Senior developer leaves, taking critical knowledge
- Owner: CEO
- Current Score: 4 (Medium Priority - Impact: 2, Likelihood: 2)
- What We Do About It:
  - Require documentation for all complex code
  - Pair programming on critical features
  - Record architecture decision records (ADRs)
  - Competitive compensation reviews
- Reality Check: Controls working? Partially - documentation always lags. Last checked? [Date]

RISK-2024-003: Customer Thinks We Can See Their Data

- What Could Go Wrong: Reputation damage from misunderstanding our architecture
- Owner: Head of Customer Success
- Current Score: 3 (Low Priority - Impact: 3, Likelihood: 1)
- What We Do About It:
  - Clear architecture diagrams in all sales materials
  - Explicit "no data access" in contracts
  - Customer education during onboarding
  - Public architecture documentation
- Reality Check: No incidents to date. Architecture = best control.

RISK-2024-004: We Fail SOC2 Audit

- What Could Go Wrong: Lose enterprise deals without SOC2
- Owner: CTO
- Current Score: 4 (Medium Priority - Impact: 2, Likelihood: 2)
- What We Do About It:
  - This documentation project!
  - Quarterly control reviews
  - Annual audit prep assessment
  - Fix findings immediately
- Reality Check: First audit pending. Controls being implemented.

7.3 Maintaining Our Register Without Burning Out

Quarterly Review Process (2 hours maximum)

1. Pull up the spreadsheet during quarterly planning
2. For each risk: "Is this still real? Has the score changed?"
3. Add any new risks identified this quarter
4. Archive risks that no longer apply
5. Assign any new actions to specific people
6. Save and version the spreadsheet

What Triggers Immediate Updates

- A risk actually happens (incident occurs)
- Major business change (new product, big customer, etc.)
- External change (new regulation, Azure announces changes)
- Someone says "Hey, what if..." and it's actually scary

We don't update the register just because it's Tuesday. Updates happen when something meaningful changes.

SOC2 Mapping: CC3.3, CC4.1, CC5.2 (Risk Monitoring and Updates)

8. Monitoring Without a Security Operations Center

8.1 Our Pragmatic Monitoring Approach

We can't afford a 24/7 SOC, and honestly, we don't need one. Here's how we maintain security visibility appropriate to our size and risk:

Let Azure Do the Heavy Lifting

- Azure Security Center provides continuous security monitoring
- Azure Monitor alerts us to availability or performance issues
- GitHub/Azure DevOps tracks all code and infrastructure changes
- These platforms invest more in monitoring than we ever could

Key Indicators We Actually Track

Security Health Indicators:

- Critical vulnerabilities open >72 hours: Target = 0 (Azure Security Center dashboard)
- Failed login attempts spike: Alert if >50/hour (Azure AD alerts)

- Unexpected privilege escalations: Alert immediately (Azure AD PIM)
- Dependency vulnerabilities: Check weekly via Dependabot

Operational Health Indicators:

- Service availability: Target >99.5% (Azure Monitor)
- Customer support tickets about bugs: Trend should be flat or down
- Failed deployments: Target <10% (Azure DevOps tracking)
- Time to resolve critical issues: Target <4 hours

Compliance Health Indicators:

- Overdue security reviews: Target = 0
- Team members without recent training: Target = 0
- Unaddressed audit findings: Target = 0
- Policy exceptions past expiry: Target = 0

8.2 Reporting That People Actually Read

For the CEO (Quarterly, 1 page max)

- Simple risk heat map (3x3 grid, dots for each risk)
- What changed since last quarter (2-3 bullets)
- What needs CEO attention (usually nothing)
- Resource needs if any (be specific)

For the Team (Quarterly All-Hands, 10 minutes)

- "Here are our top 3 risks and what we're doing"
- "Here's what everyone can do to help"
- "Any new risks we haven't thought of?"
- Recognition for risk prevention wins

For the Board (Annually or as needed)

- Executive summary of risk posture
- Comparison to similar-sized companies
- Audit results and remediation status
- Resource requirements for next level

For Auditors (On demand)

- This policy document
- Current risk register spreadsheet

- Evidence of quarterly reviews
- Incident logs showing response

SOC2 Mapping: CC2.1, CC2.3, CC4.1 (Communication and Monitoring)

8.3 Making Alerts Actionable

We configure alerts to avoid fatigue—every alert should require action:

Critical Alerts (Immediate action required):

- Security: Critical vulnerability detected
- Availability: Service down for any region
- Access: Privilege escalation or new admin
- Response: Drop everything and fix

Warning Alerts (Action within 24 hours):

- Security: High vulnerability detected
- Performance: Response time degraded >50%
- Capacity: Resource usage >80%
- Response: Plan fix for next day

Informational Alerts (Review weekly):

- Updates available for dependencies
- Failed login attempts within normal range
- Configuration drift detected
- Response: Address during scheduled maintenance

9. Realistic Assessment Schedule

9.1 What We Actually Do (And When)

We align risk activities with our existing business rhythm rather than creating separate risk management events:

Quarterly Business Planning (Includes Risk Review)

- When: First week of each quarter
- Duration: 2 hours of the planning session
- Who: Entire team
- What Happens:
    - Review risk register (30 minutes)

  - Identify new risks (30 minutes)
  - Update treatments and priorities (30 minutes)
  - Assign any new actions (30 minutes)
- Output: Updated risk register, actions in project backlog

Annual Deep Dive (Part of Strategic Planning)

- When: Q4 during annual planning
- Duration: Half-day session
- Who: Leadership team + key technical staff
- What Happens:
  - Review all risks, not just high priority
  - Assess risk management effectiveness
  - Update risk appetite if needed
  - Plan for next year's growth impact on risk
- Output: Annual risk report, updated policy if needed

Continuous Activities (Built into daily work)

- Vulnerability scanning: Automated daily via Azure
- Dependency checking: Automated weekly via GitHub
- Access reviews: Quarterly with IT audit (30 minutes)
- Incident learning: Immediately after any incident

SOC2 Mapping: CC3.1, CC4.2, CC5.3 (Risk Assessment Timing)

9.2 Triggered Assessments (When Normal Schedule Isn't Enough)

Some events require immediate risk assessment outside our quarterly cycle:

Major Incidents

- What happened and could it happen again?
- Did our controls work as expected?
- What new risk does this reveal?
- Complete within 1 week of incident

Big Business Changes

- New enterprise customer with unique requirements
- Launching a new product component
- Major vendor change (like leaving Azure)
- Assess before committing to change

External Triggers

- New regulation affecting our industry
- Major vulnerability in our tech stack
- Competitor has publicized breach
- Assess within 1 week of awareness

9.3 Keeping It Sustainable

The key to our schedule is integration and automation:

We Don't Have:

- Separate monthly risk committee meetings
- Weekly risk status reports
- Daily vulnerability review meetings
- Quarterly third-party assessments

We Do Have:

- Risk as agenda item in existing meetings
- Automated alerts that matter
- Quarterly rhythm that's predictable
- Annual third-party pen test (SOC2 requirement)

This schedule ensures risk management happens without overwhelming our small team. As we grow, we'll add complexity only when the risk justifies the overhead.

SOC2 Mapping: CC1.4, CC5.1 (Organizational Commitment and Changes)

10. Integration with What We Already Do

10.1 No Separate Risk Universe

Risk management at OversiteAI isn't a parallel process—it's baked into how we run the business. Here's how risk thinking shows up in our daily work without adding overhead:

Product Development

Every feature starts with a simple question: "What could go wrong?"

- Security review is part of our PR checklist, not a separate gate
- Architecture decisions include risk trade-offs in our ADRs
- Sprint retrospectives ask "What risks did we discover?"

- Technical debt is tracked as operational risk

Example: When adding a new API endpoint, the PR template prompts: "Authentication method? Rate limiting? Input validation? Logging?" These aren't separate security reviews—they're just good development.

Customer Success

Customer interactions reveal risks before they become incidents:

- Support tickets highlight documentation gaps
- Deployment issues show product risks
- Feature requests reveal competitive risks
- Customer complaints are early warning signals

Example: Three customers confused about the same feature = documentation risk. We don't need a formal risk assessment to know we should fix the docs.

Vendor and Tool Selection

Before we commit to any new service, we ask three risk questions:

1. What happens if they disappear tomorrow? (Can we migrate?)
2. What happens if they get breached? (What's our exposure?)
3. What happens if they 10x their prices? (Are we locked in?)

Example: We use Azure because Microsoft won't disappear, their security is better than ours, and we could migrate to AWS if needed (painful but possible).

SOC2 Mapping: CC1.4, CC3.1, CC5.1 (Integration with Entity Processes)

10.2 Making Risk Visible Without Extra Meetings

In Our Planning Board

- High-risk items get an orange font in Azure DevOps
- Risk mitigation tasks are labeled "risk-reduction"
- Post-incident fixes are labeled "incident-prevention"
- This makes risk work visible without special reports

In Our Code

- Security-critical code has extra comments explaining why
- Architecture Decision Records document risk trade-offs
- TODO comments include risk markers: `// TODO: RISK: No rate limiting`

- Git commits reference risks: "Fix RISK-2024-001: Add MFA enforcement"

In Our Communications

- Slack #security channel for quick risk discussions
- Weekly standup includes "Any new risks spotted?"
- Customer emails that mention security go to shared inbox
- No separate risk management communications needed

The goal is simple: risk management happens where the work happens, not in separate processes that everyone ignores.

11. Tools That Actually Help (Not Hinder)

11.1 Our Minimalist Risk Toolkit

We use tools we already own rather than buying specialized risk management software:

For Risk Tracking

- **What**: Simple Excel/Google Sheets for risk register
- **Why**: Everyone knows how to use it, easy to share, free
- **Alternative when we grow**: Azure DevOps work items can track risks

For Vulnerability Management

- **What**: Azure Security Center + GitHub Security
- **Why**: Included in services we already pay for, automated scanning
- **What we don't need**: Separate vulnerability management platform

For Monitoring

- **What**: Azure Monitor + Application Insights
- **Why**: Built into our platform, good enough alerting
- **What we avoid**: Expensive SIEM we'd never fully use

For Documentation

- **What**: Markdown files in Git (like this one!)
- **Why**: Version controlled, searchable, everyone can edit
- **What we skip**: Expensive GRC platforms

SOC2 Mapping: CC7.1, CC7.2 (System Operations and Monitoring)

11.2 Information Sources (Free and Useful)

We stay informed without expensive threat intelligence subscriptions:

Microsoft Security Blog: Since we're on Azure, their security updates directly affect us. RSS feed to Slack.

GitHub Security Advisories: Automated alerts for dependencies. Already integrated, no extra work.

CISA Alerts: Free government cybersecurity alerts. Good enough for our needs.

Stack Overflow: Where our actual security questions get answered by people who've faced the same issues.

We don't need APT threat intelligence—we need to know about the vulnerabilities that actually affect our stack.

11.3 Templates That Save Time

Instead of complex forms, we use simple templates that get filled out:

Risk Quick Assessment (for new risks):

```
What could go wrong: [1 sentence]
How bad (1-3): [ ]
How likely (1-3): [ ]
What we're doing now: [Current controls if any]
What we should do: [Proposed action]
Who owns this: [Name]
```

Exception Request (see Section 13 for full process):

```
What policy requirement: [Specific requirement]
Why we need exception: [Business reason]
Risk if we do this: [Honest assessment]
How we'll compensate: [Alternative controls]
When we'll fix properly: [Date or trigger]
Who approves: [CTO or CEO]
```

These templates are in our shared docs. Simple enough that people actually use them.

12. Training That Sticks (Without Death by PowerPoint)

12.1 Security Awareness for Everyone

We keep training practical and relevant to what people actually do:

Onboarding Security Basics (First Week)

- How our architecture protects customers (and us)
- Password manager setup and MFA enrollment
- Recognizing phishing (with real examples we've seen)
- How to report security concerns
- Where to find security policies (like this doc)

Format: 1-hour discussion with CTO, not slides. New hires ask questions about real scenarios.

Annual Refresh (All Hands)

- What changed in security landscape this year
- Incidents or near-misses we experienced
- Updated attack techniques relevant to us
- Quick phishing simulation
- Q&A; on security concerns

Format: 30-minute segment in quarterly all-hands. Interactive, not lecture.

Just-in-Time Training

- New tool? Quick security walkthrough before access
- Customer data question? Architecture review
- Suspicious email? Team discussion of indicators
- Incident occurs? Immediate lesson learned

Format: 5-minute huddles when issues arise. Learning from real events sticks better than theory.

SOC2 Mapping: CC1.4, CC2.2 (Commitment and Communication)

12.2 Role-Specific Skills

Different roles need different risk knowledge:

Developers

- Secure coding practices for our tech stack
- How to use security scanning tools
- Threat modeling for new features
- Incident response procedures

How: Code review feedback, pair programming, security champions program when we're bigger

Customer Success

- Architecture explanation for customers
- How to handle security questionnaires
- Recognizing customer security concerns
- Escalation procedures

How: Customer scenario role-play, shared response templates

Leadership

- Risk-based decision making
- Resource allocation for security
- Incident command basics
- Board reporting on risk

How: Quarterly risk reviews, external advisor sessions annually

12.3 Building Security Culture (Without Being the Security Police)

What Works for Us:

- Celebrate finding vulnerabilities before they're exploited
- Share interesting security articles in Slack
- "Security win of the week" in all-hands
- Blameless post-mortems for all incidents
- Security considerations in every architecture discussion

What We Avoid:

- Mandatory multi-hour training videos
- Security "pop quizzes" that annoy people
- Blame for clicking phishing tests
- Complex security policies nobody reads
- Acting like security trumps all business needs

The goal: Everyone thinks about security as part of doing good work, not as an annoying add-on.

13. Policy Exceptions (Because Reality Happens)

13.1 When and How to Request an Exception

Sometimes business needs conflict with ideal security. We handle this transparently:

Valid Reasons for Exceptions:

- Customer requirement we can't meet any other way
- Temporary technical limitation we're actively fixing
- Cost of control exceeds risk for our current size
- Time-critical business need with planned remediation

Invalid Reasons:

- "It's too hard" (without trying alternatives)
- "We've always done it this way"
- "Customer asked us to weaken security"
- "We don't have time" (for critical controls)

The Exception Process:

1. **Document the Request** (Use our simple template)
   - Specific requirement you can't meet
   - Business impact if we say no
   - Honest risk assessment
   - Proposed compensating controls
   - When this exception would end
2. **Get the Right Approval**
   - Low Risk (score 1-3): CTO can approve
   - Medium Risk (score 4-6): CEO must approve
   - High Risk (score 7-9): CEO approves + board notification
   - Critical Risk: No exceptions, find another way
3. **Implement Compensating Controls**
   - Must be in place before exception starts
   - Documented and verifiable
   - Monitored for effectiveness
4. **Set Expiration**
   - All exceptions expire (max 1 year)
   - Specific trigger for review (e.g., "When we hire developer #10")
   - Calendar reminder for renewal review

SOC2 Mapping: CC3.3, CC3.4 (Risk Assessment and Response)

13.2 Tracking and Reviewing Exceptions

Exception Register (Part of risk register)

- Exception ID and description
- Approval date and approver
- Expiration date
- Compensating controls
- Review status

Quarterly Review

- Are compensating controls working?
- Is the business need still valid?
- Can we implement the full control yet?
- Should we extend, modify, or close?

Example Exception:

```
EX-2024-001: MFA Exception for Legacy Customer API
Business Need: BigCustomer Corp can't update their integration yet
Risk: API access without MFA (Medium - 4)
Compensating Controls:
  - IP allowlist for their servers only
  - Extra logging on all API calls
  - Rate limiting reduced by 50%
Approved by: CEO
Expires: March 31, 2024 or when customer updates
Status: Active, reviewed quarterly
```

13.3 Exception Accountability

Who's Responsible:

- **Requester**: Implements compensating controls
- **Approver**: Accepts the risk on behalf of company
- **CTO**: Tracks all exceptions and ensures reviews happen
- **Auditors**: Will definitely ask about these

What Happens If:

- Controls aren't implemented: Exception is void
- Risk materializes: Incident response + immediate review
- Expiration reached: Must renew or implement full control
- Pattern emerges: Policy might need updating

The exception process ensures we can be flexible when needed while maintaining accountability and not normalizing bad security practices.

SOC2 Mapping: CC9.1, CC9.2 (Risk Mitigation and Acceptance)

14. Keeping Ourselves Honest

14.1 How We Check If This Is Actually Working

We don't need complex compliance monitoring—just simple checks to ensure we're doing what we said we'd do:

Quarterly Quick Check (30 minutes during risk review)

- Did we actually review the risk register? (Check: meeting notes)
- Are exceptions within approved limits? (Check: exception register)
- Did any incidents reveal process failures? (Check: incident log)
- Are people following the security basics? (Check: MFA enrollment, training completion)

Annual Reality Check (Part of policy review)

- Interview 3-4 team members: "Is this policy helping or hindering?"
- Review all incidents: "Would following policy have prevented this?"
- Check audit findings: "What did we miss?"
- Ask customers: "Any security concerns?" (part of annual survey)

What We Track (Simple metrics):

- % of risks reviewed on schedule (Target: 100%)
- Days to patch critical vulnerabilities (Target: <3)
- Security incidents caused by policy gaps (Target: 0)
- Team members trained this year (Target: 100%)

SOC2 Mapping: CC4.1, CC4.2, CC5.1 (Monitoring Activities)

14.2 When Things Go Wrong

Non-compliance happens. Here's how we handle it constructively:

First Response: Understand, don't blame

- What actually happened?
- Why did the person deviate from policy?
- Is the policy unrealistic for our size?
- What's the actual risk impact?

Fix the Root Cause:

- Policy too complex? → Simplify it

- People didn't know? → Better communication
- Too time-consuming? → Find automation
- Repeated issues? → Process problem, not people problem

Example: Developer skips security review for "urgent" fix

- Root cause: Customer pressure + unclear escalation
- Fix: Document emergency change process
- Not: Blame developer for trying to help customer

15. Keeping This Document Alive

15.1 Review Cycle (That Actually Happens)

Annual Review (Every January with planning)

- Owner: CTO
- Duration: 2 hours max
- Participants: CEO, CTO, Senior Developer, Customer Success Lead

What We Review:

- Did any sections cause confusion this year?
- What changed in our business or technology?
- Any new regulations affecting us?
- Lessons learned from incidents?
- Are we still being honest about our size?

Triggered Updates (Don't wait for annual)

- Major incident reveals policy gap
- Audit finding requires change
- New regulation applies to us
- Business model shifts significantly
- We double in size

15.2 Change Process (Keep It Simple)

1. **Identify Need**: Anyone can suggest improvements
2. **Draft Changes**: CTO updates in Git branch
3. **Review**: Leadership team reviews in regular meeting
4. **Approve**: CEO approves (document in commit message)
5. **Communicate**: All-hands announcement + Slack

6. **Train**: Only if changes are significant

Version Control: Git handles all our versioning needs. No separate document management system required.

SOC2 Mapping: CC1.2, CC5.3 (Policy Management)

16. Related Documents

These documents work together as our security framework:

- **Information Security Policy**: Our overarching security approach
- **Incident Response Plan**: What to do when things go wrong
- **Access Control Policy**: Who can access what and how
- **Change Management Policy**: How we safely make changes
- **Business Continuity Policy**: Keeping services running
- **Asset Management Policy**: Protecting what we own
- **Human Resources Security Policy**: People-related security
- **Privacy Addendum**: Our architectural privacy advantages

17. Document Control

NIST Controls: PM-4, SA-5

Version	Date	Author	Changes
1.0	January 1, 2025	CTO	Initial comprehensive version
2.0	Jun 25, 2025	CTO	Added NIST control mappings throughout document and new Appendix A

Review and Approval

- **Prepared By**: _____________ **Date:** ____
- **Approved By**: _____________ **Date:** ____

Next Review Date: January 1, 2026

Distribution:

- All Employees: Via company policy portal

- IT Team: Direct distribution for implementation
- Executive Team
- External Auditors (upon request)

18. Definitions

Accept (Risk): Consciously deciding to take no action to modify a risk, understanding the potential consequences.

Azure Security Center: Microsoft's unified security management system that we use for continuous monitoring.

Compensating Control: Alternative control that provides similar risk reduction when the primary control isn't feasible.

Control: Any action, device, procedure, or technique that reduces risk. Can be preventive, detective, or corrective.

Exception: Formal, time-limited deviation from policy with documented justification and approval.

Impact: What happens to the business if a risk materializes. We measure in dollars, downtime, and customer trust.

Inherent Risk: The risk level before we do anything about it—the worst-case scenario.

Likelihood: Our best guess at whether something will actually happen, based on experience and industry data.

MFA (Multi-Factor Authentication): Requiring two or more verification methods. Non-negotiable for our admin access.

Residual Risk: The risk level after our controls are in place—what we're actually living with.

Risk: Anything that could prevent us from building great software, keeping customers happy, or staying in business.

Risk Appetite: How much risk we're willing to accept. Varies by category—zero for IP theft, higher for market timing.

Risk Owner: The person who loses sleep if this risk materializes. Has authority to implement controls.

Risk Register: Our simple spreadsheet tracking what could go wrong and what we're doing about it.

Risk Treatment: What we do about a risk—eliminate, automate, process, or accept.

SOC2: Service Organization Control 2—the audit standard that proves we handle security responsibly.

Vulnerability: A weakness in our systems or processes that could be exploited. Found by scanning or testing.

19. Appendices

Appendix A: NIST Control Mapping

This policy addresses the following SOC2 Trust Services Criteria:

Control Environment (CC1)

- CC1.1: Demonstrates commitment (Sections 2, 4)
- CC1.2: Exercises oversight (Sections 4, 15)
- CC1.3: Establishes structure (Section 4)
- CC1.4: Demonstrates commitment to competence (Sections 10, 12)

Communication and Information (CC2)

- CC2.1: Obtains/generates information (Section 8)
- CC2.2: Communicates internally (Sections 8, 12)
- CC2.3: Communicates externally (Section 8)

Risk Assessment (CC3)

- CC3.1: Identifies and assesses risk (Sections 3, 5, 10)
- CC3.2: Assesses fraud risk (Sections 5, 7)
- CC3.3: Identifies and assesses changes (Sections 5, 6, 13)
- CC3.4: Evaluates and communicates deficiencies (Sections 6, 13)

Monitoring Activities (CC4)

- CC4.1: Conducts ongoing/separate evaluations (Sections 7, 8, 14)
- CC4.2: Evaluates and communicates deficiencies (Sections 9, 14)

Control Activities (CC5)

- CC5.1: Selects and develops controls (Sections 7, 10)
- CC5.2: Deploys through policies (Sections 7, 8)
- CC5.3: Deploys through technology (Sections 9, 15)

Logical and Physical Access (CC6)

- CC6.1: Implements logical access controls (Section 6)
- CC6.6: Implements security measures (Section 6)

System Operations (CC7)

- CC7.1: Detects and mitigates vulnerabilities (Sections 6, 11)
- CC7.2: Monitors system components (Sections 6, 11)

Risk Mitigation (CC9)

- CC9.1: Identifies and assesses risk (Section 6)
- CC9.2: Implements risk mitigation (Section 13)

Appendix B: Quick Reference Templates

Risk Quick Assessment

```
Date: _______
Identified By: _______
Risk Description: _______
Impact (1-3): _______
Likelihood (1-3): _______
Priority Score: _______
Current Controls: _______
Proposed Action: _______
Owner: _______
Target Date: _______
```

Exception Request Form

```
Request Date: _______
Requester: _______
Policy Section: _______
Business Justification: _______
Risk Without Exception: _______
Proposed Compensating Controls: _______
Exception Duration: _______
Review Frequency: _______
Approver: _______
Approval Date: _______
```

Appendix C: Emergency Contacts

Security Incident: CTO → CEO → Legal Counsel

Azure Issues: Azure Support (Premier tier)

Vulnerability Found: Development Team → CTO

Customer Security Concern: Customer Success → CTO

Media Inquiry: CEO only