Thala vCISO Security Review

Auditors

Devtooligan, Security Researcher

Report prepared by: Lucas Goiriz

March 15, 2025

Contents

1 About Spearbit

Spearbit is a decentralized network of expert security engineers offering reviews and other security related services to Web3 projects with the goal of creating a stronger ecosystem. Our network has experience on every part of the blockchain technology stack, including but not limited to protocol design, smart contracts and the Solidity compiler. Spearbit brings in untapped security talent by enabling expert freelance auditors seeking flexibility to work on interesting projects together.

Learn more about us at spearbit.com

2 Introduction

Thala is a decentralized protocol that consolidates a suite of trading and yield products, enabling composable leverage and a seamless user experience.

Disclaimer: This security review does not guarantee against a hack. It is a snapshot in time of Thala vCISO according to the specific commit. Any modifications to the code will require a new security review.

3 Risk classification

Severity level	Impact: High	Impact: Medium	Impact: Low
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

3.1 Impact

- High - leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
- Medium - global losses <10% or losses to only a subset of users, but still unacceptable.
- Low - losses will be annoying but bearable--applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.

3.2 Likelihood

- High - almost certain to happen, easy to perform, or not easy but highly incentivized
- Medium - only conditionally possible or incentivized, but still relatively likely
- Low - requires stars to align, or little-to-no incentive

3.3 Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

4 Executive Summary

Over the course of 60 days in total, Thala engaged with Spearbit to review the v2 protocol. In this period of time a total of **29** issues were found.

Summary

Project Name	Thala
Repository	Move & Web2
Type of Project	Infrastructure, Web2
Audit Timeline	Dec 16th to Mar 14th

Issues Found

Severity	**Count**
Critical Risk	0
High Risk	2
Medium Risk	7
Low Risk	12
Informational	8
Total	**29**

4.1 Engagement Scope and Context

This report details the process and outcomes of a 3-month vCISO cybersecurity consulting engagement for Thala, a DeFi protocol on the Aptos blockchain. The engagement focused on reviewing all security-related aspects excluding code review, with emphasis on infrastructure and operational security.

The engagement encompassed operational security (opsec), multisig management, web2 security around DNS, and critical services such as Vercel and AWS and more.

4.2 Approach: Building on a Strong Foundation

It's worth noting that Thala already demonstrated a strong security posture prior to this engagement. The team had implemented many industry best practices and showed a clear commitment to security. This engagement was specifically requested to identify opportunities for further enhancement and to align with evolving security standards in the rapidly changing DeFi space.

4.3 Methodology: Utilizing the SEAL Framework

To manage the extensive scope, the review was based on the SEAL Framework from the Security Alliance, a leading authority in Web3 security. This framework provides comprehensive security guidance for Web3 projects across domains including day-to-day security practices, development operations, vulnerability handling, dependency security, and risk management.

From this framework, a detailed checklist was developed to systematically evaluate Thala's security practices, ensuring thorough coverage despite the broad scope.

4.4 Collaborative and Iterative Process

Throughout the engagement, multiple meetings were held with Thala's team to deeply understand their current security practices. The iterative nature of the review allowed for findings to be shared as they were identified, enabling real-time feedback and discussion. Notably, Thala began implementing remediation for many findings during the engagement period, demonstrating their proactive approach to security enhancement.

4.5 Recommended Next Steps

To further strengthen Thala's security posture:

1. **Address Findings**: Implement the remediation steps outlined for each identified finding below.
2. **Application Security Review**: Consider obtaining a complete Application Security review of Thala's front-end system.
3. **Multisig Management Partner**: Evaluate potential multisig management partners that also provide Incident Response services.
4. **Security Review of Safely**: Arrange for a security review of the Safely codebase.

5 Findings

5.1 High Risk

5.1.1 Missing Incident Response Plan

Severity: High Risk

Description: Thala lacks documented incident response procedures and playbooks. The lack of formal procedures increases risk of delayed or ineffective response during time-critical situations, especially for incidents requiring different handling than past experiences.

Without documented procedures, the team faces increased risks of:

- Delayed detection and response to security incidents.
- Uncoordinated team actions during critical moments.
- Inconsistent external communications.
- Missed opportunities for fund recovery.
- Extended downtime from delayed decisions.
- Impaired judgment under high-stress situations.

Severity Discussion: The impact is high as incident response could mitigate losses during security events. The likelihood is medium given the variety of potential attack vectors across smart contracts, frontend, social media and operations. Based on these factors and the critical nature of timely incident response, this ranks as a high severity issue.

Recommendation:

1. Implement the draft incident response plan outlined in Appendix B which includes:
    - War room procedures and team roles.
    - Communication protocols and templates.
    - Containment and recovery frameworks.
    - Integration with SEAL 911 emergency response.
    - Post-incident review procedures.
2. Create an Incident Response Template for use during incidents. See example in Appendix C.
3. Create detailed playbooks for handling all common incident types:

- Smart contract exploit:
    - Funds lost - see example in Appendix D.
    - Contract disabled.
    - Private key compromise.
- Front-end incident:
    - Malicious code deployed.
    - Phishing attack related to an open redirect.
    - DNS hijacked - see example in Appendix E.
- Discord:
    - Admin compromised.
    - Raid/Spam attack.
- Twitter compromised - see example in Appendix F.

4. Conduct quarterly incident response exercises to:
    - Test and refine response procedures.
    - Help team members understand their roles.
    - Practice coordination under simulated pressure.
    - Identify and address gaps in the plan.

5.1.2 Alert System Coverage and Classification

Severity: High Risk

Description: The system lacks monitoring for several protocol actions:

- Administrative function calls.
- Pool creation and management.
- Balance invariant violations.

Thala's alert system could be improved with a more structured classification and triage system. The November 2024 incident highlighted how TVL drop alerts were mistakenly misclassified as routine changes, delaying investigation by several hours. At the time of the incident, alerts were split between `alerts-activity` and `alerts-protocol` channels without clear prioritization or investigation requirements.

Additionally, alerts in the protocol channel mix critical and non-critical items, contributing to alert fatigue and reducing effectiveness of the notification system.

Note: During the course of the review, the project has implemented `#alerts-protocol` and `#alerts-manager` as high priority alerting channels, both of which require action.

Severity Discussion: The impact is high as missing or misclassified alerts directly affect incident detection and response time. The likelihood is high given evidence from the recent incident where alert misclassification contributed to delayed response. Based on these factors and the critical nature of proper alerting for protocol safety, this ranks as a high severity issue.

Recommendation:

1. Implement enhanced alert coverage for both critical and informational events. See Appendix A for detailed specifications of recommended alerts.
2. Establish clear alert classification and procedures:
    - Move all non-critical alerts out of protocol channel.
    - Require root cause analysis and evidence for critical alert resolution.
    - Document investigation procedures and templates.
    - Define escalation paths and response times.
    - Review alert efficacy quarterly.
3. Define response requirements:
    - Primary on-call (10min) $\rightarrow$ Secondary $\rightarrow$ Management escalation path.
    - Evidence collection requirements before alert closure.
    - Regular review of alert patterns and thresholds.

5.2 Medium Risk

5.2.1 Insufficient Authentication Controls on Critical Services

Severity: Medium Risk

Description: Critical services should use FIDO2 hardware security key authentication, instead of relying on less secure methods.

FIDO2 hardware keys provide stronger security against phishing and account takeover compared to SMS or authenticator app 2FA.

Severity Discussion: The impact is high as compromise of critical service accounts could allow unauthorized access to deployment infrastructure, treasury funds, or public communication channels. The likelihood is low - while the team has been targeted recently and these services present valuable targets, standard 2FA provides baseline protection against most attacks.

Recommendation: Disable password, SMS, and authenticator app 2FA and other auth options and restrict auth to passkey through FIDO2 enabled hardware security keys (such as Yubikey) for all critical services where available including:

- AWS.
- Vercel.
- DNS registrar.
- Slack.
- Pagerduty.
- Twitter.
- Discord.

For services where there is only one login (or for a singular admin account), add a secondary backup hardware security key in case one gets lost or damaged. Store the backup key in a physically separate location.

Additionally, use "cold" email addresses for critical service logins as described in L-01.

5.2.2 Missing Security Vulnerability Reporting Process

Severity: Medium Risk

Description: The project lacks a clear process for security researchers to report vulnerabilities and has no defined bug bounty program to incentivize responsible disclosure.

Not having a published security contact information or bug bounty program, makes it difficult for security researchers to responsibly disclose vulnerabilities. Sometimes, a quick way to contact the development team can make a big difference in defending against hacks.

The lack of a public bounty program may incentivize black hats to conduct a hack instead of reporting the issue.

Recommendation: Establish a public security reporting process and bug bounty program:

- Create `security.txt` at `/.well-known/security.txt` with contact details.
- Add security contact information to website and documentation.
- Publish scope and rewards for bug bounty program.
- Consider using Immunefi platform to manage bounty program and submissions.
- Include encrypted contact method (e.g. PGP key).
- Define SLAs for initial response to reports.

5.2.3 Unconstrained Vercel Spending Limits

Severity: Medium Risk

Description: Spend monitoring and control mechanisms are not configured in Vercel, creating risk of unexpected charges or potential abuse. Currently, spending is monitored manually on a monthly basis without automated

alerts or controls. Automatic controls would provide better oversight and early warning of unusual activity that could indicate security issues or resource abuse.

Recommendation: Enable Vercel's free spend management features:

- Set maximum spend threshold significantly above normal usage.
- Enable automatic pausing when threshold is reached.
- Enable spend alert notifications at:
  - 25% of threshold.
  - 50% of threshold.
  - 75% of threshold.
- Configure notification webhooks for Slack integration.

5.2.4 Single Bot Design in Liquidation System

Severity: Medium Risk

Description: The protocol's liquidation process currently relies on a single bot implementation. When this bot failed due to an interface change, the team successfully fell back to manual liquidations.

Severity Discussion: The impact is medium as manual intervention has proven effective as a backup. The likelihood is low since market conditions requiring liquidations are relatively rare. While the system could be more robust, the existence of working manual procedures makes this a medium severity issue.

Recommendation:

1. Document Manual Process:
   - Create runbook of the successful manual liquidation procedure.
   - Document team member roles and coordination process.
   - Include procedure for flash loan utilization.
2. Improve Bot Resilience:
   - Add monitoring for bot health and execution status.
   - Consider basic redundancy in bot deployment.
   - Maintain thorough test coverage for interface changes.

5.2.5 `node.js` Dependency Management and Supply Chain Security

Severity: Medium Risk

Description: Current dependency management practices expose the protocol to potential supply chain attacks. The front-end repository shows frequent commits of `yarn.lock` with caret (`^`) version ranges, indicating dependencies are being updated automatically when developers install packages locally. This creates risk of malicious code being introduced through compromised dependencies.

Supply chain attacks have become increasingly common as attackers sometimes find it easier to take over some obscure dependency rather than hack protocols directly.

Recommendation: Implement the following dependency management controls:

1. Immediately conduct a comprehensive dependency audit:
   - Run `npm audit` to identify known vulnerabilities.
   - Review every dependency used by the project looking for:
     - Number of GitHub stars.

    - Maintenance activity.
    - Search `README` and issues for any known security issues.
    - Conduct additional searches for any known.
  - Document decisions and rationale for keeping each dependency.
2. Lock dependency versions:
  - Remove caret (`^`) and tilde (`~`) version ranges from `package.json`.
  - Lock to specific commit hashes for critical dependencies.
  - Implement strict version control on yarn.lock file.
  - Require PR review for any dependency changes.
  - Consider using `yarn.lock` checksums in CI/CD.
3. Set up automated monitoring:
  - Enable Dependabot alerts and security updates.
  - Subscribe to:
    - Node.js Security Working Group mailing list.
    - National Vulnerability Database (NVD) RSS feeds.
    - Security announcements from key dependencies.
4. Establish quarterly review process: Repeat steps 1 and 2 every quarter.

5.2.6 Multisig Security Enhancements

Severity: Medium Risk

Description: Current multisig setup uses 3/5 threshold (60%) with all internal signers. While functional, this configuration could be strengthened with a goal of 70% for medium and high risk operations.

For further discussion, see this article, The Right Way To Multisig.

Severity Discussion: The impact is medium as compromise would require multiple signers to be affected. The likelihood is low given existing hardware wallet requirements and signing procedures. Current setup is functional but could be enhanced.

Recommendation:

1. Increase security through external signers:
  - Short-term: Add trusted third party (e.g., security provider partner) as 6th signer.
  - Adjust threshold to 4/6 (66.7%) as an immediate improvement.
  - Long-term: Consider expanding to 5/7 configuration as team and protocol grow.
  - Document verification responsibilities for external signer.
2. Implement formal key management procedures:
  - Annual review of signer set composition.
  - Documented procedures for signer replacement.
  - Regular testing of recovery scenarios.
  - Clear criteria for emergency key revocation.
3. Enhance signing procedures:
  - Require out-of-band transaction hash verification.

- Implement mandatory cool-down period for large transfers (*Note: This is currently not readily available for Aptos multisig wallets but something to keep in mind for the future*).
- Create transaction templates for common operations.
- Document specific verification steps for each transaction type.

5.2.7 Comprehensive Application Security Review of Front-End System

Severity: Medium Risk

Description: Thala's front-end system has not undergone a comprehensive application security (AppSec) review. This system handles user interactions, displays sensitive data, and integrates with backend services, making it a prime target for attacks such as cross-site scripting (XSS), client-side logic manipulation, or phishing exploits. Without a thorough assessment, latent vulnerabilities could persist, increasing the risk of exploitation that could compromise user trust, funds, or system integrity.

The absence of a prior AppSec review is notable given:

- Dependency management risks identified in M-05, which are only one aspect of front-end security.
- Lack of Content Security Policy (CSP) noted in L-06, suggesting broader front-end security gaps may exist.
- SQL injection vulnerabilities in an API route (L-05), highlighting potential weaknesses in systems interacting with the front-end.

Severity Discussion: The impact is high, as front-end vulnerabilities could enable attackers to steal user credentials, inject malicious content, or disrupt protocol operations. The likelihood is medium, as no specific exploits have been identified, but the system's exposure to external users and lack of prior review elevate the risk. This ranks as a medium-severity issue, warranting proactive mitigation.

Recommendation: Conduct a comprehensive application security (AppSec) review of the front-end system, focusing on:

- Code Quality: Assess for secure coding practices, input validation, and output encoding to prevent XSS, CSRF, and injection flaws.
- Client-Side Logic: Verify that business logic cannot be bypassed or manipulated by attackers.
- Security Controls: Evaluate implementation of CSP (see L-06), secure headers, and session management.
- Dependency Security: Expand on M-06 to ensure all libraries are vetted and up-to-date.
- Penetration Testing: Consider pen testing to simulate real-world attacks in order to identify exploitable weaknesses.

5.3 Low Risk

5.3.1 Exposed Email Addresses Used for Critical Services

Severity: Low Risk

Description: Critical service accounts are accessed using email addresses that are publicly known or used for other purposes, increasing security risk.

The organization's critical service accounts include social media (Twitter, Discord), and infrastructure admin accounts which require an email address for login. These should not use email addresses that are publicly disclosed. The email addresses should not be used for any other purpose - including communications, other services, or public contact.

Using shared or known email addresses increases attack surface and makes credential discovery easier for potential attackers.

Severity Discussion: The impact is low since exposed email addresses facilitate but do not directly enable attacks, requiring additional security failures to cause harm. The likelihood is medium as these addresses are easily discoverable and regularly targeted, though existing authentication measures prevent direct compromise.

Recommendation: Implement a dedicated "cold" email address for critical service logins. This email should be:

- Used exclusively for service authentication.
- Not publicly disclosed.
- Not used for any communication.
- Not used for any other purpose.
- Stored securely with limited access.

5.3.2 Vercel Security Controls

Severity: Low Risk

Description: Current Vercel security settings are not fully optimized, leaving potential security gaps in deployment and access control. Basic security features available on the Pro plan are not fully utilized.

Recommendation: Implement the following Vercel security configurations:

- Review and restrict team access permissions to only those who need access.
- Exclusively use passkey login with Yubikey for all team member and owner accounts.
- Use "sensitive" environment variables for Preview and Production environments.
- Enable Git Fork Protection to prevent unauthorized repository forks.
- Disable Vercel Toolbar in production environments. Pre-production environments can be optionally enabled if it is useful.

Consider implementing these optional monitoring controls if budget and team bandwidth permit:

- Enable basic request logging for critical endpoints/routes.
- Configure simple error rate monitoring and alerts.
- Set up Slack notifications for deployment failures and critical errors.
- Document procedures for periodic log review.

5.3.3 Unprotected Vercel Preview URLs

Severity: Low Risk

Description: Preview deployments are currently accessible to anyone who obtains the URL, creating unnecessary security exposure. Without deployment protection, preview URLs (e.g., `my-preview-5678.vercel.app`) are publicly accessible with no access controls or audit capabilities. This presents the following risks:

- Unlimited access to anyone with the preview URL.
- No tracking of preview URL distribution.
- Absence of access audit trails.
- Inability to revoke access selectively.

Severity Discussion: The impact is low since preview deployments contain pre-release code but not production credentials or data. The likelihood is low as preview URLs are not easily discoverable without internal access. Given the limited exposure and existing deployment controls, this ranks as a low severity issue.

Recommendation: Enable Vercel Authentication and implement Shareable Links:

1. Enable Vercel Authentication (Standard Protection) to secure all preview deployments.
2. Use Vercel Shareable Links to:
    - Create per-branch preview access links.

- Control recipient access.
- Revoke access when needed.
- Monitor active link usage.
- Track preview deployment sharing.

This configuration ensures preview deployments remain private by default while maintaining efficient collaboration with external stakeholders through managed access controls.

5.3.4 Supply chain vulnerability in GitHub Actions workflow configuration

Severity: Low Risk

Description: There are multiple instances of unpinned external actions or floating references in the Github workflows. This creates potential supply chain attack vectors which could allow attackers to execute malicious code within the CI environment if any of the referenced action repositories are compromised. Two GitHub Actions workflows were analyzed:

1. CI workflow for testing and code coverage.
2. Sync workflow for maintaining thala-mainnet repository.

The following actions are inadequately pinned:

- `actions/checkout@v3`: Pinned only to major version.
- `pontem-network/get-aptos@main`: Using floating `main` branch reference.
- `actions/github-script@v6`: Pinned only to major version.
- `actions/cache@v3`: Pinned only to major version.

When actions are referenced by branch names or version tags instead of specific commit hashes, there is a risk that the referenced code could be modified maliciously through compromised repositories or social engineering attacks against action maintainers.

If any of the referenced action repositories are compromised, an attacker could inject malicious code that would be automatically executed within Thala's CI environment. This could lead to exposure of sensitive repository data, credentials stored in secrets, or enable supply chain attacks against the built artifacts.

The likelihood of exploitation is low. Compromising well-maintained actions like those from GitHub themselves is relatively difficult.

Recommendation: All external actions should be pinned to specific commit hashes rather than using version tags or branch references. This ensures the exact code being executed is known and won't change unexpectedly.

Example of properly pinned actions:

```
- uses: actions/checkout@8f4b7f84864484a7bf31766abe9204da3cbe65b3  # v3.5.0
- uses: actions/cache@88522ab9f39a2ea568f7027eddc7d8d8bc9d59c8  # v3.3.1
- uses: actions/github-script@d556feaca394842dc55e4734bf3bb9f685482fa0  # v6.3.3
```

For third-party actions like `pontem-network/get-aptos`, review the source code at a specific commit and pin to that commit hash:

```
- uses: pontem-network/get-aptos@7602c9d447e5b92d447737c2f905c1f42c78cd73  # specific commit
```

Additionally, consider implementing these supplementary security measures:

1. Regular audit of pinned commit hashes to ensure they're still the latest verified versions.
2. Documentation of the verification process for third-party actions.
3. Implementation of a policy requiring security review before updating action versions.

5.3.5 SQL Injection In Points API Route

Severity: Low Risk

Description: The `GET()` handler in `echelon/app/api/cron/points/route.txs` contains two SQL injection vulnerabilities. While the likelihood of exploitation is low since both injection points rely on trusted 3rd party data sources, following security best practices would suggest using parameterized queries in both cases.

1. Error Message SQL Injection: In the catch block of the `GET()` handler, error messages are directly interpolated into an SQL query string:

```
catch (e) {
    console.error(e);
    const message: string = e instanceof Error ? e.message : String(e);
    await client.query(`
  INSERT INTO points\_snapshot\_status (success, error\_message)
  VALUES (FALSE, '${message}');`);
}
```

2. Points Data SQL Injection: When inserting points data from the `fetchAllPoints()` response:

```
const values = data
  .map(
    (item) => `(
'${item.account}',
${item.borrowPoints},
${item.lendPoints},
0
)`
  )
  .join(",");
```

Both cases involve direct string concatenation into SQL queries without parameterization. The `VALUES` clause structure makes it possible to break out of both the string context and parentheses using a payload like `') <malicious SQL> --`.

An attacker who could control either error messages or points data could potentially execute arbitrary SQL commands. However, since both data sources are trusted internal systems, the actual risk of malicious exploitation is low.

The likelihood of exploitation is low since both vulnerabilities require control of trusted data sources - either internal error messages or the points API response. However, the technical complexity of exploitation is also low due to the direct string concatenation and simple injection pattern.

Recommendation:

1. For error handling:

```
catch (e) {
    console.error(e);
    const message: string = e instanceof Error ? e.message : String(e);
    await client.query(
        'INSERT INTO points\_snapshot\_status (success, error\_message) VALUES ($1, $2)',
        [false, message]
    );
}
```

2. For points data:

```
const valueParams = data.map((\_, i) => `($${i*3+1}, $${i*3+2}, $${i*3+3}, 0)`).join(',');
const flatValues = data.flatMap(item => [
  item.account,
  item.borrowPoints,
  item.lendPoints
]);

await client.query(`
INSERT INTO points\_snapshot (account, borrow\_points, lend\_points, referral\_points)
VALUES ${valueParams}
`, flatValues);
```

5.3.6 Missing Content Security Policy Header

Severity: Low Risk

Description: The application does not implement a Content Security Policy (CSP). CSP provides critical protection against various web security vulnerabilities, particularly cross-site scripting (XSS) attacks, by controlling which resources a page is allowed to load and execute. While not a vulnerability in itself, the absence of CSP removes an important layer of defense that could help mitigate the impact of other security issues.

As defined in the Mozilla documentation:

> Content Security Policy (CSP) is a feature that helps to prevent or minimize the risk of certain types of security threats. It consists of a series of instructions from a website to a browser, which instruct the browser to place restrictions on the things that the code comprising the site is allowed to do.

CSP can help protect against:

- Cross-site scripting (XSS) by controlling which scripts can execute.
- Clickjacking attacks through frame-ancestors directives.
- Data injection attacks by restricting valid content sources.
- Malicious resource loading by enforcing HTTPS.

Recommendation: Implement a Content Security Policy (CSP) by adding the Content-Security-Policy header to all responses.

Note: Prefer defining the CSP at the application level, such as in next.config.js or directly in response headers, rather than relying on Vercel's CSP management feature. Doing so aligns with the 'configuration as code' paradigm, which encourages treating configuration as code to allow it to benefit from the existing code review process, to receive more scrutiny, and to avoid manual checks by logging into the Vercel dashboard.

To develop a CSP for a new application, start with a strict policy and adjust based on application needs:

```
Content-Security-Policy:
    default-src 'self';
    script-src 'self';
    style-src 'self';
    img-src 'self';
    frame-ancestors 'none';
    form-action 'self';
```

Consider starting with report-only mode using `Content-Security-Policy-Report-Only` to identify and address any legitimate resources that would be blocked before enforcing the policy:

```
Content-Security-Policy-Report-Only: default-src 'self'; report-uri /csp-violation-report-endpoint/
```

Note: When creating a new app, it is recommended to start with this very strict policy. It can then be gradually tightened while monitoring for violations.

Note for project team: When adding a CSP to an existing app that previously hadn't been using one (as is the case here), it is recommended to do take the opposite approach. Start with a very loose policy and then gradually tighten it ensuring the application functionality does not break.

Remember that CSP should be used as part of a defense-in-depth strategy alongside other security controls like input validation, output encoding, and secure coding practices.

5.3.7 Asset Inventory System

Severity: Low Risk

Description: Thala lacks a formal system for tracking and managing critical assets including personal devices, hardware security modules, and service account access. Without a comprehensive asset inventory, the team faces increased risk during security incidents and may miss critical areas during access reviews or employee offboarding.

The current state presents several risks:

- No central record of which personal devices have access to critical systems.
- Incomplete tracking of hardware wallets and signing devices.
- Service account access and privilege levels not systematically documented.
- Difficulty ensuring complete access removal during offboarding.
- Limited ability to perform security audits across all assets.
- Challenges in disaster recovery planning without full asset visibility.

Recommendation: Implement a comprehensive asset inventory system with the following components:

- Personal Computing Devices. Track for each team member:
  - Device type/model.
  - Operating system and version.
  - Status of security tools (antivirus, disk encryption).
  - Access level to critical systems.
  - Primary user and backup contact.
  - Date of last security review.
- Hardware Security. Document for each device:
  - Device type (e.g., Yubikey 5C, Ledger Nano X).
  - Serial number.
  - Assigned user.
  - Associated accounts/services.
- Service Accounts. Track for each service:
  - Service name and purpose.
  - Account holders and access levels.
  - Authentication method.

  Current services requiring documentation:
  - Vercel: Development team access (Member role).
  - AWS: Infrastructure management.
  - GCP: Validator and bot operations.
  - GitHub: Repository access.

- Coingecko: API access.
- Namecheap: Domain management.
- Google Workspace: Team collaboration.
- Twitter: Social media management (Adam, San).

- Implementation Plan.

1. Create Initial Inventory:
   - Use a secure, shared spreadsheet or documentation system.
   - Assign an owner responsible for maintenance.
   - Include all current team members and assets.
2. Regular Maintenance:
   - Schedule quarterly reviews of all assets.
   - Update during any team member changes.
   - Review during security incidents.
   - Verify during compliance audits.
3. Offboarding Integration:
   - Add asset inventory review to offboarding checklist.
   - Include verification of access removal.
   - Document transfer of critical assets/accounts.

5.3.8 Fix Invalid Nameserver Configuration

Severity: Low Risk

Description: `WHOIS` data for Thala.fi revealed two valid Vercel nameservers and one invalid gitbook.io entry:

```
Nameservers

nserver.............: ns1.vercel-dns.com [OK]
nserver.............: ns2.vercel-dns.com [OK]
nserver.............: 94b0511f40-hosting.gitbook.io [Technical Error]
```

Verification via authoritative .fi TLD name servers confirms that the misconfigured gitbook.io nameserver is still present:

```
$ dig NS thala.fi @a.fi

; <<>> DiG 9.10.6 <<>> NS thala.fi @a.fi
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34818
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 3, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;thala.fi.          IN  NS

;; AUTHORITY SECTION:
thala.fi.       21600   IN  NS  ns1.vercel-dns.com.
thala.fi.       21600   IN  NS  ns2.vercel-dns.com.
thala.fi.       21600   IN  NS  94b0511f40-hosting.gitbook.io.

;; Query time: 170 msec
;; SERVER: 2001:708:10:53::53#53(2001:708:10:53::53)
;; WHEN: Sat Feb 08 15:04:00 PST 2025
;; MSG SIZE  rcvd: 130
```

The misconfigured gitbook.io nameserver does not pose a direct security risk since DNS resolution is functioning correctly through the valid Vercel nameservers. However, it represents poor hygiene and could complicate future DNS troubleshooting.

Recommendation: Remove the invalid gitbook.io nameserver entry through the domain management interface and verify that only the required Vercel nameservers remain.

5.3.9 Strengthen Domain Transfer Protection for Critical Domains

Severity: Low Risk

Description: Domain hijacking remains a critical risk, even with strong account security.

Standard transfer locks (`clientTransferProhibited`) prevent unauthorized transfers but do not protect against unauthorized DNS modifications or registry-level attacks. A **Registry Lock**, where available, adds an extra layer of protection by requiring out-of-band manual approval before any changes can be made at the registry level.

For `.fi` domains, **Traficom is responsible for enabling Registry Lock**, not AWS or any other provider. The team must contact Traficom directly to determine eligibility and request activation.

A WHOIS lookup confirms that **Registry Lock is not enabled** for `thala.fi`:

```
RegistryLock.......: no
```

Recommendation:

1. Contact Traficom to enable Registry Lock (if available).
    - Email: `firootadmin@traficom.fi`.
    - Phone: `+358 295 345 000`.
    - Ask if Registry Lock is supported for `.fi` domains and how to enable it.
2. Restrict AWS Route 53 domain modification permissions:
    - Ensure only authorized users can change DNS records.
    - Require MFA (Multi-Factor Authentication) for accounts with domain access.

3. Set up a simple notification system for unexpected DNS changes:
    - Using **AWS EventBridge (or CloudWatch)**, configure alerts for Route 53 DNS modifications.
    - Route notifications to an **SNS topic** to send alerts via email or webhook.

5.3.10 Strengthen GCP Infrastructure Security

Severity: Low Risk

Description: The current GCP configuration has several areas that could be strengthened:

- A single service account with “Editor” permissions is used for the collateral auction bot, granting it full control over most resources and violating the principle of least privilege.
- Default-deny firewall rules are not yet enabled, allowing all traffic unless explicitly blocked, which increases the risk of unauthorized access or lateral movement within the network.
- Basic VPC flow logging is not configured, reducing visibility into network traffic and making it harder to detect anomalies, troubleshoot connectivity issues, or audit security events.

Recommendation:

1. Implement service account separation:
    - Create a dedicated service account for the validator with minimum required permissions.
    - Create a dedicated service account for the liquidator bot with minimum required permissions.
    - Remove "*Editor*" permission from the existing shared service account
    - Document ownership and purpose of each service account (L-07 Asset Inventory System)
2. Additional security controls.

Here are some practical security measures that minimize risk without adding unnecessary complexity or cost:

- **Enable default-deny firewall rules** for both ingress and egress (gradually).

  Start by logging traffic first, then apply rules incrementally, ensuring all necessary services have explicit allow rules to prevent disruptions.
- **Enable Basic VPC Flow Logs** to detect unusual traffic patterns.

  Basic flow logs are free and help monitor network activity. Avoid Detailed logs unless you need deeper insights, as they can become expensive.
- **Enable Cloud Audit Logs** for admin activity monitoring.
- **Admin Activity logs** are free and track security-critical actions like IAM changes and API calls. Skip Data Access logs for now.
- Consider enabling **Security Command Center** (free tier). Provides automated security insights but can generate noisy findings. Enable only if someone will review alerts and act on them.

5.3.11 Discord Security Controls

Severity: Low Risk

Description: During our assessment, we identified the following implemented controls:

- CAPTCHA based user verification.
- 2 external moderators + core team with 'core' role.
- Using several bots including MEE6, AFK Bot, and Wick.
- Large member base (33,823 members).
- Any member can create unlimited-duration invite links.

Our analysis revealed several potential security exposures:

- Limited protection against coordinated malicious activity.
- No documented security procedures or incident response.
- Unclear bot permission boundaries and usage.

Recommendations:

1. Access Controls:
    - Enable Discord's verification level "High" requiring verified email + 10-minute wait.
    - Use a team password manager for sharing sensitive credentials.
    - Regularly reviews of active invite links.
2. Bot Security:
    - Audit each bot's permissions and remove unnecessary access.
    - Document active bots' purposes and configurations.
    - Remove unused integrations.
    - Enable Discord's AutoMod for baseline protection:
        - Set spam filters.
        - Configure mention limits.
        - Block suspicious links.
        - Set message rate limits.
3. Moderation Controls:
    - Create clear moderator guidelines including:
        - Allowed/disallowed actions.
        - When to escalate issues.
        - Emergency contact info.
    - Set up a private mod coordination channel.
    - Define incident response steps for common scenarios.
4. Documentation:
    - Simple security policies (<2 pages).
    - Current role/permission matrix.
    - Bot configurations and owners.
    - Incident response procedures (H-01).

5.3.12 Enhance Team Security Awareness

Severity: Low Risk

Description: Thala would benefit from implementing structured security awareness training. While technical security measures are in place, the human element requires strengthening through targeted education, particularly given the increasing sophistication of attacks targeting DeFi protocols.

Recommendation: Consider implementing KnowBe4's security training platform (knowbe4.com) to establish baseline security awareness and conduct regular phishing simulations. Their small-team pricing options and crypto-specific scenarios make this a practical choice for Thala's size and needs.

The Red Guild's Phishing Dojo (phishing.rektgames.xyz) offers free, interactive training specifically designed for crypto teams. This can complement KnowBe4 by providing hands-on practice with signature verification, transaction analysis, and scam site identification. Additionally, schedule monthly team security sessions to discuss "The Art of Social Engineering" concepts and rotate security topic presentations among team members.

- Implementation Plan: Given the small team size:
  - Assign security champion role.
  - Use simple spreadsheet for tracking.
  - Integrate training into existing team meetings.
  - Focus on high-impact, low-friction activities.

5.4 Informational

5.4.1 Operational Security for Company Workstations

Severity: Informational

Description: The laptops and workstations used for company business will benefit from standardized security configurations to protect against common attack vectors and maintain operational security. These are especially critical on developer workstations which have privileged access to critical systems and repositories, making them high-value targets for attackers.

Developer workstations serve as primary access points to critical infrastructure, smart contracts, and deployment systems. Tight operational security policies can help mitigate various attacks including:

- Malware and ransomware infections.
- Silent installation of malicious startup items.
- Browser-based attacks through malicious ads.
- Multi-account session confusion.

Recommendation: Implement the following baseline security controls for all developer workstations:

- System Security:
  - Enable full-disk encryption (FileVault for MacOS, LUKS for Linux).
  - Install and maintain active malware protection software.
  - Enable host-based firewall with default-deny rules.
  - Install BlockBlock (MacOS) or equivalent startup monitoring.
  - Configure automatic security updates.
  - Implement fail2ban for SSH protection.
  - Restrict remote access to authorized IPs only.
- Browser Security:
  - Standardize on Chrome as the primary browser.
  - Install security extensions:
    - Password Alert for phishing protection.
    - uBlock Origin or Ghostery for ad/tracker blocking.
  - Configure separate Chrome profiles for different security contexts.
  - Prevent simultaneous logins across different security boundaries.
- Access Control:

  - Enforce individual user accounts with unique SSH keys.
  - Implement RBAC for administrative access.
  - Enable MFA for all service accounts.
  - Follow NIST 800-123 guidelines for system hardening.
  - Document and maintain standard configuration baselines.
- Monitoring:
  - Consider implementing HIDS/HIPS for enhanced threat detection.
  - Monitor for unauthorized configuration changes.
  - Regularly audit system logs and access patterns.

5.4.2 Heightened Operational Security for Signer Workstations

Severity: Informational

Description: Critical blockchain operations like multisig transactions require heightened security controls beyond standard workstation hardening. Dedicated signing workstations should be configured with minimal attack surface and maximum isolation.

The protocol's multisig signers use dedicated laptops for transaction signing operations. These workstations have privileged access to critical protocol functions and handle high-value transactions, making them prime targets for attackers. A compromise of these systems could lead to unauthorized protocol changes or loss of funds.

While finding I-01 provides a baseline, signing workstations require additional isolation and hardening measures.

Recommendation: In addition to all controls from I-01, implement these specific measures for signing workstations:

- Network Isolation:
  - Disable all wireless communications:
    - WiFi.
    - Bluetooth.
    - NFC.
    - Cellular/mobile data if present.
  - Remove or physically disable network hardware where possible.
  - If network connectivity is required for specific operations:
    - Use wired connections only.
    - Implement strict firewall rules allowing only essential traffic.
    - Consider air-gapping when not actively signing.
- Software Controls:
  - Remove or disable web browsers.
  - Install transaction payload data decoding tool.
  - Disable unnecessary system services and daemons.
  - Remove all non-essential applications.
  - Disable USB ports not required for hardware wallets.
- Physical Security:
  - Store workstations in secure location when not in use.

  - Consider tamper-evident seals.
  - Maintain chain of custody documentation.
  - Label devices clearly as "Transaction Signing Only".
- Operational Controls:
  - Use only for transaction signing - no other activities.
  - Do not install or use GitHub applications.
  - Disable package managers like npm and `pip`/`pypi`.
  - Remove email clients and disable email access.
  - Disable file synchronization services and cloud storage.
  - Document clear procedures for:
    - Transaction verification steps.
    - Payload decoding and validation.
    - Emergency response if compromise suspected.
  - Regular integrity checks of system state.
- Backup & Recovery:
  - Document base system configuration and setup steps.
  - Keep spare workstation(s) available for redundancy.
  - Test hardware wallet signing on backup machine periodically.

5.4.3 Web Domain Best Practices and General Recommendations

Severity: Informational

Description: As a `.fi` domain, `thala.fi` falls under Finnish jurisdiction, meaning legal or administrative actions must comply with Finnish regulations. The **Finnish Transport and Communications Agency (Traficom)** enforces strict domain policies, including verification requirements and registry oversight, which contribute to the overall security and stability of `.fi` domains.

Secure domain name management

A guide for domain name registrars and holders

FI Domain Team

The `thala.fi` domain infrastructure consists of:

- **Registration:** Managed through Traficom (expires Oct 2025).
- **DNS Hosting:** Vercel nameservers.
- **Security Controls:**
  - **CAA records restrict certificate issuance** to Let's Encrypt.
  - No unnecessary services exposed.

1. Results of `WHOIS thala.fi`:

```
$ whois thala.fi

% IANA WHOIS server
% for more information on IANA, visit http://www.iana.org
% This query returned 1 object

refer:        whois.fi

domain:       FI

organisation: Finnish Transport and Communications Agency Traficom
address:      PO Box 320
address:      TRAFICOM
address:      Helsinki 00059
address:      Finland

contact:      administrative
name:         Domain Names
organisation: Finnish Transport and Communications Agency Traficom
address:      PO Box 320
address:      TRAFICOM
address:      Helsinki 00059
address:      Finland
phone:        +358 295 345 000
e-mail:       firootadmin@traficom.fi

contact:      technical
name:         Sami Salmensuo
organisation: Finnish Transport and Communications Agency Traficom
address:      PO Box 320
address:      TRAFICOM
address:      Helsinki 00059
address:      Finland
phone:        +358 295 345 000
e-mail:       firootadmin@traficom.fi

nserver:      A.FI 193.166.4.1 2001:708:10:53:0:0:0:53
nserver:      B.FI 194.146.106.26 2001:67c:1010:6:0:0:0:53
nserver:      C.FI 194.0.11.104 2001:678:e:104:0:0:0:53
nserver:      D.FI 2a01:3f0:0:302:0:0:0:53 77.72.229.253
nserver:      E.FI 194.0.1.14 2001:678:4:0:0:0:0:e
nserver:      G.FI 2001:500:14:6098:ad:0:0:1 204.61.216.98
nserver:      H.FI 2001:678:a0:0:0:0:0:aaaa 87.239.120.11
nserver:      I.FI 194.0.25.30 2001:678:20:0:0:0:0:30
nserver:      J.FI 185.159.199.190 2620:10a:80ac:0:0:0:0:190
nserver:      K.FI 2001:14b8:188d:0:0:0:0:53 213.186.229.226
ds-rdata:     35221 8 2 d4a489017f68a6c02836d4f126a4b9f6af34c371993bee97ed759d83db9cd38e

whois:        whois.fi
```

```
status:       ACTIVE
remarks:      Registration information: https://domain.fi

created:      1986-12-17
changed:      2023-08-18
source:       IANA

# whois.fi

domain.....: thala.fi
status.....: Registered
created.....: 9.10.2022 07:18:39
expires.....: 9.10.2025 07:18:39
available.....: 9.11.2025 07:18:39
modified.....: 5.9.2024 04:08:37
RegistryLock.....: no

Nameservers

nserver.....: ns1.vercel-dns.com [OK]
nserver.....: ns2.vercel-dns.com [OK]
nserver.....: 94b0511f40-hosting.gitbook.io [Technical Error]

DNSSEC

dnssec.....: no

Holder

holder.....: Private person

Registrar

registrar.....: Gandi SAS
www.....: www.gandi.net

>>> Last update of WHOIS database: 8.2.2025 22:47:24 (EET) <<<
```

2. DNS Configuration:

```
# A records point to Vercel IPs
dig +short thala.fi A
216.198.79.194
66.33.60.66

# CAA record properly configured
dig +short thala.fi CAA
0 issue "letsencrypt.org"

# No unnecessary MX/TXT records
dig +short thala.fi MX
<no response>
```

3. Certificate Analysis: Reviewed certificate transparency logs to analyze historical certificate issuance. Learn More → here

- crt.sh analysis revealed:
  - All certificates properly issued by authorized CAs (Let's Encrypt, Google Trust Services).

  - Expected subdomains: app, docs, classic.app, foundry, careers, www.
  - Regular 90-day renewal pattern for Let's Encrypt certs.

Learn More → here

- Censys.io search showed:
  - Active certificates match log records.
  - No unauthorized or suspicious certificates.
  - CAA record enforcement active.

Recommendation: To learn more, read this excellent Cantina blogpost written by some of the industry's top security researchers: Prevent DNS Hijacking: Web3 Security Best Practices.

To prevent **DNS hijacking, unauthorized modifications, and misconfigurations**, implement the following:

1. DNS Monitoring & Alerting:
   - Use **AWS EventBridge** to detect Route 53 DNS modifications.
   - Route notifications to an **SNS topic** to send alerts via email or webhook.
   - (Optional) Use **CloudWatch Logs** to store DNS change history for audit purposes.
2. Registrar & Access Security:
   - Confirm with Traficom whether transfer lock is available and enable it if supported.
   - Use a dedicated email account for domain management, separate from public-facing accounts (L-01).
   - Require FIDO2 security keys (YubiKey, Ledger) for domain-related logins. (M-01)
   - Maintain an asset inventory list including domain management credentials (L-07).
   - If possible, disable SMS-based 2FA and enforce SSO-based logins.
3. DNS Configuration Hardening:
   - Remove invalid GitBook nameserver (L-08).
   - Continue restricting certificate issuance using **CAA records**.
   - Consider using **Cloudflare Custom Domain Protection** for additional DNS change verification.
4. Incident Response & Recovery:
   - Document **domain recovery procedures** (H-01):
     - **Traficom registry contacts** for `.fi` domains.
     - **Vercel DNS restoration steps** in case of misconfiguration or compromise.
     - Assign team responsibilities for handling domain-related incidents.
5. Follow the recommendations provided in related DNS findings:
   - **Fix Invalid Nameserver Configuration** (L-08).
   - **Enable Domain Registry Lock for Critical Domains** (L-09).

5.4.4 Configure DNS Security Extensions (DNSSEC)

Severity: Informational

Description: DNSSEC (Domain Name System Security Extensions) cryptographically signs DNS records using public-key cryptography, allowing DNS resolvers to verify record authenticity and integrity.

The following analysis confirms DNSSEC is not currently enabled for thala.fi domain, leaving DNS responses without cryptographic verification.

1. `WHOIS` shows DNSSEC disabled. Verification:

```
dnssec.............: no
```

2. No DNSSEC records present:

```
$ dig +short thala.fi DNSKEY
<no response>

$ dig +short thala.fi DS
<no response>
```

3. SOA record (Start of Authority - the root DNS record for the domain) shows no RRSIG (Resource Record Signature) which would be present if DNSSEC was enabled:

```
$ dig +cd +dnssec thala.fi SOA

; <<>> DiG 9.10.6 <<>> +cd +dnssec thala.fi SOA
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10631
;; flags: qr rd ra cd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 512
;; QUESTION SECTION:
;thala.fi. IN SOA

;; ANSWER SECTION:
thala.fi. 3600 IN SOA ns1.vercel-dns.com. hostmaster.nsone.net. 1666032549 43200 7200 1209600 600

;; Query time: 252 msec
;; SERVER: 2001:1998:f00:1::1#53(2001:1998:f00:1::1)
;; WHEN: Sat Feb 08 13:14:23 PST 2025
;; MSG SIZE rcvd: 111
```

4. DNSSEC validation using `delv` (Domain Entity Lookup & Validation - a tool for testing DNSSEC validation) confirms no DNSSEC implementation:

```
$ delv thala.fi
;; none:29: no crypto support
delv: No trusted keys were loaded
```

DNSSEC provides an additional layer of trust by cryptographically validating DNS responses. Its absence could enable DNS spoofing. However, it is not a critical vulnerability as other controls like HTTPS and certificate pinning significantly mitigate this risk.

Recommendation: *Note: During the review, the team experimented with enabling DNSSEC and experienced site outages in certain regions. We believe those outages happened in regions that use DNS relays which were caching and causing DNS sec to fail to resolve.*

DNSSEC provides an extra layer of security but is not mandatory due to other existing security measures. Given the team's experience, caution is advised when implementing it. If the team chooses to pursue this options, use a gradual rollout:

- If possible test in a non-production domain first.
- Enable DNSSEC only for subdomains initially before applying to the root domain.
- Verify DNS propagation across multiple geographic regions.
- Monitor resolver behavior before finalizing changes.

Once DNSSEC is enabled and stable, the main operational risk is key management. Ensure key rotation follows AWS Route 53 best practices.

5.4.5 X(Twitter) Account Security

Severity: Informational

Description: Our review identified the following security measures in place:

- Account access limited to 2 founders. One other "delegated" access to team member.
- Using shared password manager.
- No phone linked to the account, eliminating the possibility of sim-swap attacks.
- No policies or formal incident response plan.

Recommendations:

1. Implement hardware security keys with FIDO2 (YubiKeys) for 2FA (M-01).
2. Store backup codes securely in the team password manager.
3. Conduct periodic audits of connected apps and sessions.
4. Add authorized logins to the asset inventory list.
5. Set up periodic password rotation schedule.
6. Document policies and procedures related to twitter and create an incident response plan (H-01 Incident Response Plans).

5.4.6 Design Review Considerations for Safely

Severity: Informational

Description: As part of this engagement, we conducted a design review of the Safely product, a client-side SDK in development to enhance the user experience of Aptos's built-in multisig functionality. The current design provides a robust foundation for multisig operations, with no critical security flaws identified at the design level. The following sections outline considerations to strengthen the SDK as development progresses, ensuring its reliability, security, and sustainability within the Aptos ecosystem.

1. Template Security: The transaction annotation system is central to Safely's usability, requiring templates to accurately reflect transaction intent to prevent misinterpretation. Suggested considerations include:

- **Template Rules:** Template rules, such as mandating "transfer" in titles for transfer transactions, could ensure clarity and consistency.
- **Validation:** Implement checks to confirm template accuracy and alignment with transaction purpose.
- **Question:** How might template design prevent misrepresentation of transaction actions?

2. Community Contribution Security: The community-driven development model introduces risks of malicious or erroneous contributions, potentially from compromised accounts or bad actors. To mitigate these, consider:

- **Contribution Guidelines:** Define security-focused submission requirements for contributions.
- **Review Process:** Establish a multi-level review process, potentially involving trusted contributors.
- **CI/CD Security:** Integrate security checks, such as template validation, into the CI/CD pipeline.
- **Questions:**
  - How could contributions be verified to block misleading templates?
  - Could a trusted contributor system balance openness with security?

3. Version Stability and Production Readiness: A clear strategy for stability and feature maturity is essential as Safely targets production use, maintaining user trust and operational reliability. Suggested considerations include:

- **Feature Control:** Implement access controls for experimental functionality, such as feature flags, to isolate untested components.
- **Versioning Strategy:** Define a versioning approach (e.g., semantic versioning) with explicit stability guarantees.
- **Documentation:** Differentiate production-ready features from experimental ones in documentation, alongside a detailed changelog noting security implications.
- **Migration Support:** Provide migration guides for breaking changes to ensure smooth transitions.
- **Questions:**
  - How will users be informed of a feature's stability status?
  - What criteria will determine when Safely is production-ready?

4. Security Validation Process: Given Safely's critical role in multisig management, a structured security validation process is necessary to uphold its integrity. Recommendations include:

- **External Review:** Conduct an external security review of the initial Safely codebase to establish a baseline.
- **Incremental Validation:** Plan for incremental security reviews of significant community-driven updates to address evolving risks.
- **Security Documentation:** Document the SDK's security properties and guarantees for user awareness.
- **Bug Bounty:** Explore a bug bounty program, potentially in collaboration with the Aptos Foundation, to incentivize vulnerability discovery.
- **Incident Response:** Develop a security incident response plan to handle disclosed issues efficiently.
- **Question:** How might incremental reviews be prioritized as community contributions grow?

5. Long-term Maintenance: Sustaining Safely's security and functionality beyond initial development requires a proactive maintenance strategy. Key points to address:

- **Ownership Plan:** Establish a long-term security maintenance plan, identifying who will oversee the project post-MVP—will key community members transition into maintainer roles?
- **Critical Components:** Document security-critical components to guide future maintainers.
- **Dependency Management:** Minimize dependencies and monitor them for security issues to reduce external risk exposure.
- **Question:** What steps could ensure a smooth handoff to community maintainers if the core team steps back?

5.4.7 Slack Best Practices

Severity: Informational

Description: Slack workspaces containing sensitive protocol discussions, operational coordination, and deployment planning require robust security controls. Without proper security measures, Slack workspaces can become vectors for unauthorized access, data exposure, or communication disruption. A comprehensive set of controls helps protect against these risks while maintaining operational efficiency.

Recommendation:

1. **Workspace Security Configuration:**

- Enable mandatory two-factor authentication under Settings > Security > Two-Factor Authentication.
- Set specific message retention timeframes:

  - Configure 30-day retention for general channels.
  - Set 90-day minimum retention for deployment and security channels.
  - Navigate to Settings > Message Retention to configure.
- Under Settings > Email Domain, claim and verify all company email domains.
- In Settings > File Sharing, disable public link creation unless specifically required.

2. **Access Management:**

- Review admin/owner list monthly, limit to 2-3 key team members.
- Under Settings > Permissions, configure specific role capabilities:
  - Channel creation permissions.
  - App installation rights.
  - Member invitation controls.
- For guest accounts:
  - Set 30-day maximum access duration.
  - Restrict to specific channels only.
  - Review all guest access monthly.

3. **Integration Security:** Under Settings > Apps:

- Document purpose of each authorized integration.
- Remove all unused applications.
- Restrict installation permissions to admin role only.
- Configure app-specific channel access restrictions.

4. **File Controls:** Under Settings > File Storage:

- Set appropriate retention periods for different file types.
- Disable external file sharing by default.
- Require admin approval for public link creation.
- Limit file upload permissions in sensitive channels.

5.4.8 Streamlined PagerDuty Configuration for Small Teams

Severity: Informational

Description: PagerDuty serves as Thala's primary alert escalation system. The current setup uses a single service with SMS and call notifications, rotating on-call duty across the engineering team, with escalation to leadership after ~10 minutes of no acknowledgment. While functional, this configuration could be refined to enhance effectiveness without adding significant operational overhead.

Recommendation:

1. **Create two distinct alert services:**
   - Critical service: True emergencies requiring immediate response (major TVL drops, invariant violations).
   - Standard service: Important but non-urgent notificationsThis simple distinction helps combat alert fatigue by ensuring critical notifications stand out.
2. **Reduce admin access:**
   - Limit admin privileges to 2 team members instead of the current 4.

- This improves security posture with zero ongoing maintenance cost.

3. **Implement a brief monthly review:**
   - Schedule a 15-minute monthly check of alert patterns.
   - Identify which alerts were valuable vs. which were noise.
   - Adjust thresholds based on findings.

```
# IR: Thala

**Date:** YYYY.MM.DD

**Severity:** Sev[0-5]

**Status:** Triage | Containment | Mitigation | Recovery | Retro

## Roles

* **Commander**
* **Scribe**
* **Comms**
* **Responders**
* **Other Participants**

## Summary

**What happened? Scope? Impact?**

## Action Items



| Action | Owner | Status |
| --- | --- | --- |
|  |  |  |
|  |  |  |

## Timeline

* **2025-MM-DD XX:XX** - `<event/action>`
* **2025-MM-DD XX:XX** - `<event/action>`
* **2025-MM-DD XX:XX** - `<event/action>`

## Artifacts

## Comms
```

6 Appendix

6.1 Appendix A: Alert Specifications

6.1.1 Critical Alerts (Protocol Channel + PagerDuty)

Rate Limiting

1. Coin Usage Alert
    - **Trigger**: Usage exceeding 75% of limit (curr_qty vs. window_max_qty)
    - **Data Points**: Current usage, limit, address
    - **Frequency**: Real-time
2. Parameter Changes
    - **Trigger**: Changes to window_duration or window_max_qty
    - **Data Points**: Old value, new value, modifier
    - **Frequency**: Real-time
3. New Address Activity
    - **Trigger**: New address using >50% of rate limit in first interaction
    - **Data Points**: Address, usage amount, timestamp
    - **Frequency**: Real-time

Administrative Actions

1. Admin Function Monitoring
    - **Trigger**: All admin-only function calls
    - **Data Points**: Function name, caller, parameters
    - **Frequency**: Real-time
2. Pool Changes
    - **Trigger**: Changes to stake_paused or unstake_paused flags
    - **Data Points**: Flag name, new state, modifier
    - **Frequency**: Real-time
3. Parameter Updates
    - **Trigger**: Modifications to max_boost_multiplier_bps
    - **Data Points**: Old value, new value, modifier
    - **Frequency**: Real-time

Pool Operations

1. Creation Events
    - **Trigger**: New pool creation
    - **Data Points**: Pool ID, parameters
    - **Frequency**: Real-time
2. Reward Token Changes
    - **Trigger**: New reward token addition

- **Data Points**: Pool ID, token info
- **Frequency**: Real-time

3. Stake Monitoring
    - **Trigger**: retroactive_user_stake > remaining_retroactive_stake
    - **Data Points**: User stake, remaining stake
    - **Frequency**: Real-time

6.1.2 Informational Alerts (Activity Channel)

Volume/Activity

1. Trading Activity
    - **Trigger**: High borrowing/lending activity
    - **Data Points**: Volume, participant count
    - **Frequency**: Per minute
2. Balance Changes
    - **Trigger**: Nominal balance changes by tier
    - **Data Points**: Amount, token, type
    - **Frequency**: Real-time
3. Pool Updates
    - **Trigger**: Pool allocation changes under threshold
    - **Data Points**: Old allocation, new allocation
    - **Frequency**: Real-time
4. Integration Activity
    - **Trigger**: New integrator activity
    - **Data Points**: Integrator ID, action type
    - **Frequency**: Real-time

6.2 Appendix B - Incident Response Plan Guidelines

The following outlines practical high-level steps and procedures for security incident handling. Appendix C contains an Incident Response Template that may be used in an actual incident.

6.2.1 Security Incident Definition

A **security incident** is an event that may compromise **confidentiality, integrity, or availability** of the Project's infrastructure, smart contracts, cryptocurrency assets, and other critical components.

A security incident may lead to financial losses, data breaches, reputational damage, legal and regulatory implications, and other harm to the Project ecosystem or its employees.

Security incidents are distinct and have a higher impact than **regular engineering incidents**, which primarily cause disruptions limited to system downtime, service degradation, or inconvenience to users.

6.2.2 Incident Triage and Analysis

Incident triage is performed on a **rotating assignment** on a **weekly basis**. The triager's responsibilities include:

- Analyze security-related signals (e.g., smart contract alerts, social media reports, log data, etc.).
- Determine if there is a **security incident** by consulting triage sections in respective playbooks.
- Determine incident scope and initial severity.
- Declare the incident and initiate the incident handling process.

6.2.3 Incident Severity Guidelines

Severity	Description
Sev 0	- Project or customers lost or at risk of losing assets (e.g., treasury compromise). - Project or customer service disru
Sev 1	- Temporary function degradation that does not affect funds access or availability for the entire protocol. - Ecosystem
Sev 2	- Web2 infrastructure compromise that does not affect core operations or funds availability (e.g., API hosting).
Sev 3	- Non-security and temporary API outage. - Non-security and temporary Web2 infrastructure outage.

6.2.4 Incident Declaration

Once a security incident is declared, the following procedure must be followed:

- **Incident Document** – Create a new incident document and fill out the title, date, and severity fields.
- **Incident Roles** – Assign incident participant roles (e.g., incident commander, scribe, comms, etc.).
- **Incident Channel** – Create a new Slack channel and invite all incident participants.

6.2.5 Incident Participant Roles

Some roles may be shared by the same person depending on the incident size and availability.

Role	Responsibilities
Incident Commander	Manages incident threads, assigns tasks, follows up on the status, sets incident severity, and provides
Scribe	Records incident timeline, communications, findings, and other incident artifacts in the incident docum
Comms	Communicates with external parties, provides updates at appropriate cadence, and manages custom
Incident Responders	Perform investigations, mitigations, and other roles depending on the nature of the incident based on

6.2.6 Incident Lifecycle

The incident handling lifecycle consists of the following stages:

Stage	Description
Containment	Stop the bleeding as quickly as possible. Steps are outlined in incident-specific playbooks. Examples: - Pausi
Mitigation	Fully eliminate and/or mitigate the root cause of the incident as described in incident-specific playbooks. Exam
Recovery	Restore normal operations once all containment and mitigation steps have been taken.
Retro	Host a retrospective to determine what went well, what could be improved, and action items to better prepare f

6.2.7 Incident Contact Book

Team/Org/Company	Name	Contact Info
SEAL 911	Whitehat security researchers	**SEAL 911 Telegram Bot**

6.3 Appendix C - Incident Response Template

The following template may be used during an incident. It should be saved in a shared document such as a Google Doc.

```
# IR: Thala

**Date:** YYYY.MM.DD

**Severity:** Sev[0-5]

**Status:** Triage | Containment | Mitigation | Recovery | Retro

## Roles

* **Commander**
* **Scribe**
* **Comms**
* **Responders**
* **Other Participants**

## Summary

**What happened? Scope? Impact?**

## Action Items



| Action | Owner | Status |
| --- | --- | --- |
|  |  |  |
|  |  |  |

## Timeline

* **2025-MM-DD XX:XX** - `<event/action>`
* **2025-MM-DD XX:XX** - `<event/action>`
* **2025-MM-DD XX:XX** - `<event/action>`

## Artifacts

## Comms
```

6.4 Appendix D - Incident Response Playbook Example - On-Chain Loss of Funds Playbook

6.4.1 Type

Unauthorized loss of funds from smart contracts or protocol-controlled wallets, including exploits, compromised access, or oracle manipulation.

6.4.2 Detection

- Automated alerts from Sentios showing balance drops
- User/team reports of missing funds

6.4.3 Triage

1. Quick Assessment:
    - Check if balance drop is expected from normal operations
    - Calculate total value at risk
    - Determine if loss is ongoing
    - Identify affected contracts
2. Severity:
    - Sev 0: Loss over $100k or active exploit
    - Sev 1: Loss under $100k, single user affected
    - Sev 2: Minor discrepancies, no direct loss

6.4.4 Containment

1. Stop the Bleeding:
    - Execute emergency pause
    - Document pause transaction hash
    - Alert team and SEAL 911
    - Notify users through official channels

6.4.5 Mitigation

1. Investigation & Fix:
    - Review transaction traces to identify exploit
    - Develop and test fix (prioritize speed over perfection)
    - Get quick security review if possible
    - Document all changes

6.4.6 Recovery

1. Restore Operations:
    - Deploy fixed contracts
    - Verify deployment
    - Gradually unpause
    - Monitor for any issues

6.4.7 Retro

1. Quick Review:
    - Document attack timeline
    - Identify how we could have detected sooner
    - List concrete security improvements
    - Update monitoring based on learnings

6.5 Appendix E - Incident Response Playbook Example - DNS Hijacking Incident Response Playbook

6.5.1 Type

Unauthorized modification of DNS records leading to traffic redirection or service disruption.

6.5.2 Detection

- AWS EventBridge alerts for Route 53 record changes
- Traffic drops or geographic anomalies
- Community reports

6.5.3 Triage

1. Quick Check:
    - Verify DNS records across multiple providers
    - Look for unauthorized A record or nameserver changes
    - Check if changes affect smart contracts/fund access
2. Severity:
    - Sev 0: Changes affect smart contracts or funds
    - Sev 1: Changes affect non-critical infrastructure
    - Sev 2: Suspicious activity but no confirmed changes

6.5.4 Containment

1. Lock Down:
    - Revoke current DNS management access
    - Screenshot current DNS config
    - Export DNS logs for evidence
    - Restore known-good records from backup

6.5.5 Mitigation

1. Secure Access:
    - Rotate all DNS credentials
    - Enable mandatory 2FA
    - Document all changes made

- Decrease TTL values

6.5.6 Recovery

1. Verify & Restore:
    - Check DNS propagation
    - Test critical functions
    - Notify users if needed
    - Monitor for residual issues

6.5.7 Retro

1. Review & Improve:
    - Document timeline and entry point
    - Update DNS change process
    - Enhance monitoring
    - Schedule follow-up audit

6.6 Appendix F - Incident Response Playbook Example - X (Twitter) Account Compromise Playbook

6.6.1 Type

Social media account compromise affecting official Twitter presence

6.6.2 Detection

Primary Detection

- Monitor for login attempts from new locations/devices
- Track unusual tweet patterns or high tweet volume
- Watch for changes to account settings, especially email/2FA changes

Secondary Signals

- Community reports of suspicious tweets
- Automated monitoring of tweet content for known scam patterns
- Monitor follower/following count changes

6.6.3 Triage

1. Confirm unauthorized access by checking:
    - Recent login history in Twitter settings
    - Tweet history for unauthorized posts
    - Account setting changes in last 24 hours
2. Assess Severity:
    - Sev 1: Active unauthorized tweets, especially involving funds
    - Sev 2: Suspicious access without visible account abuse

- Sev 3: Failed unauthorized access attempts

6.6.4 Containment

1. External containment (when locked out):
    - Contact Twitter support emergency line immediately to report compromise
    - Submit account recovery form through Twitter
    - Alert community through backup communications channels about compromise
    - Document unauthorized tweets/actions for Twitter support

6.6.5 Mitigation

1. Secure access:
    - Reset password manager credentials
    - Review and revoke third-party app access
    - Update email account security linked to Twitter
    - Document all changes made

6.6.6 Recovery

1. Restore account control:
    - Delete unauthorized tweets
    - Issue brief statement acknowledging incident
    - Restore correct profile information
    - Verify account settings are properly configured

6.6.7 Retro

1. Document and analyze:
    - Timeline of events
    - Entry point of compromise
    - Response effectiveness
    - Update security measures based on findings