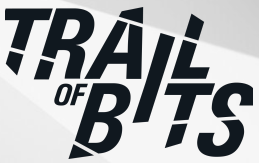


Linkerd

Threat Model

May 2, 2022

Prepared for:
Oliver Gould
Buoyant, Inc

Prepared by:
Alex Useche and David Pokora

About Trail of Bits

Founded in 2012 and headquartered in New York, Trail of Bits provides technical security assessment and advisory services to some of the world's most targeted organizations. We combine high-end security research with a real-world attacker mentality to reduce risk and fortify code. With 80+ employees around the globe, we've helped secure critical software elements that support billions of end users, including Kubernetes and the Linux kernel.

We maintain an exhaustive list of publications at https://github.com/trailofbits/publications, with links to papers, presentations, public audit reports, and podcast appearances.

In recent years, Trail of Bits consultants have showcased cutting-edge research through presentations at CanSecWest, HCSS, Devcon, Empire Hacking, GrrCon, LangSec, NorthSec, the O'Reilly Security Conference, PyCon, REcon, Security BSides, and SummerCon.

We specialize in software testing and code review projects, supporting client organizations in the technology, defense, and finance industries, as well as government entities. Notable clients include HashiCorp, Google, Microsoft, Western Digital, and Zoom.

Trail of Bits also operates a center of excellence with regard to blockchain security. Notable projects include audits of Algorand, Bitcoin SV, Chainlink, Compound, Ethereum 2.0, MakerDAO, Matic, Uniswap, Web3, and Zcash.

To keep up to date with our latest news and announcements, please follow @trailofbits on Twitter and explore our public repositories at https://github.com/trailofbits. To engage us directly, visit our "Contact" page at https://www.trailofbits.com/contact, or email us at info@trailofbits.com.

Trail of Bits, Inc.
228 Park Ave S #80688
New York, NY 10003
https://www.trailofbits.com
info@trailofbits.com

Notices and Remarks

Copyright and Distribution

Test Coverage Disclaimer

All activities undertaken by Trail of Bits in association with this project were performed in accordance with a statement of work and mutually agreed upon project plan.

Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase.

Table of Contents

Executive Summary

Engagement Overview

The Linux Foundation engaged Trail of Bits to create a component-focused threat model of its Linkerd project. From February 7 to February 11, 2022, a team of two consultants scheduled four meetings over two person-weeks of effort to evaluate relevant components and the architectural design of the project. Details of the project's timeline, test targets, and coverage are provided in subsequent sections of this report.

Project Scope

Our assessment focused on the identification of security control flaws that could result in a compromise of confidentiality, integrity, or availability of the target system, especially with respect to the controls noted in the category breakdown table below. An exhaustive list of security control types and their definitions can be found in appendix B.

Summary of Findings

While preparing the threat model, we uncovered some flaws that could impact system confidentiality, integrity, availability. A summary of findings is provided below.

FINDINGS BY SEVERITY

Severity	*Count*
Medium	3
Low	8
Informational	3

FINDINGS BY CONTROL TYPE

Category	*Count*
Access Control	4
Audit and Accountability	2
Awareness and Training	4
Denial of Service	2
System and Communications Protection	2

Project Summary

Contact Information

The following managers were associated with this project:

Dan Guido, Account Manager
dan@trailofbits.com

Cara Pearson, Project Manager
cara.pearson@trailofbits.com

The following engineers were associated with this project:

Alex Useche, Consultant
alex.useche@trailofbits.com

David Pokora, Consultant
david.pokora@trailofbits.com

Project Timeline

The significant events and milestones of the project are listed below.

Date	Event
January 27, 2022	Pre-project kickoff call
February 8, 2022	Discovery meeting #1
February 9, 2022	Discovery meeting #2
February 10, 2022	Discovery meeting #3
February 11, 2022	Discovery meeting #4
February 18, 2022	Delivery of report draft and report readout meeting
May 2, 2022	Delivery of revised report

Project Coverage

During a threat modeling assessment, engineers generally aim to cover the entire target system as a coherent whole. In some cases, however, certain components may be either unnecessary to examine or impossible to review thoroughly.

Exclusions

The following components were explicitly excluded from the assessment scope:

- **Multi-cluster configurations.** Multi-cluster configurations require cluster operators to perform a series of tasks to enable a service mesh across multiple Kubernetes clusters. This setup adds a number of variables that significantly increase the complexity of the threat model. As a result, and due to time constraints, multi-cluster configurations were considered out of scope for this assessment.

- **Other extensions.** Other extensions outside of the `linkerd-viz` namespace, such as Jaeger and those used for tracing, were not considered as part of the threat model.

Limitations

Because of the time-boxed nature of testing work, it is common to encounter coverage limitations. During this project, we were unable to perform comprehensive review of the following components, which may warrant further review:

- **`linkerd-viz` extensions.** Operators can enable a number of extensions used for metrics and diagnostics, including Prometheus metrics, TAP diagnostics, Grafana dashboards, and a web UI. The extensions are deployed as individual pods in the `linkerd-viz` namespace. Due to time constraints, we could not model the extensions individually; instead, we grouped them under a single abstract component called "`linkerd-viz` extensions," for which we achieved only partial coverage.

Security Control Maturity Evaluation

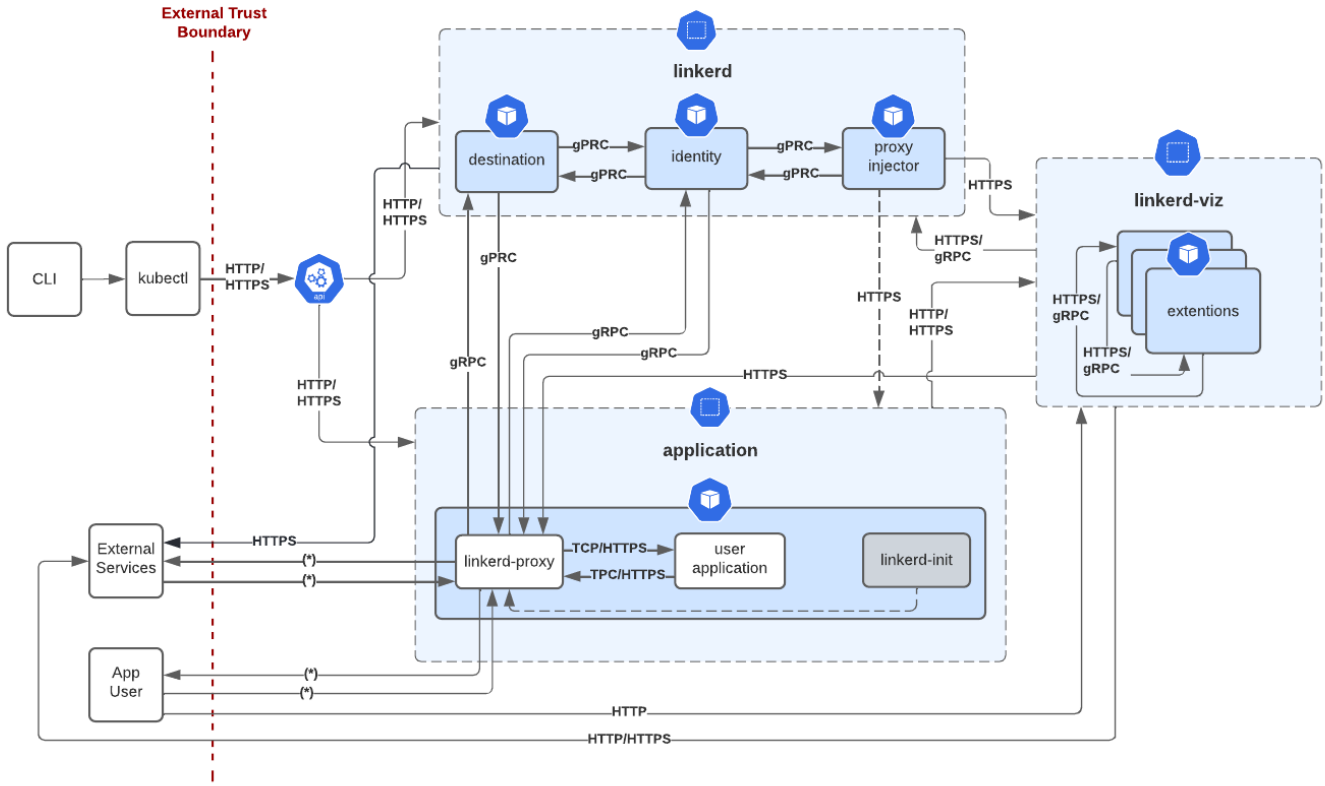
Trail of Bits uses a traffic-light protocol to provide each client with a clear understanding of the areas in which its security controls are mature, immature, or underdeveloped. See appendix B for a detailed description of each category.

Category	Summary	Result
Access Controls	Linkerd relies on Mutual Transport Layer Security (mTLS) provided by sidecar proxies for most authorization controls between components. Outside of mTLS, cluster operators rely on fine-grained Kubernetes role-based access controls to restrict access to some components, such as `linkerd-viz` dashboards and Prometheus metrics. Although most access controls are appropriate for the current architectural design, we believe that isolating components more granularly and implementing access controls for some APIs, such as proxy admin and metrics endpoints, may result in a stronger security posture.	**Moderate**
Audit and Accountability	Every component enables at least `INFO` level logging to `STDOUT`. Operators can enable other extensions to obtain metrics and diagnostics data for Linkerd components (e.g., `linkerd-viz` and Jaeger extensions).	**Satisfactory**
Awareness and Training	Throughout the Linkerd documentation, Linkerd provides general guidance on securing a Kubernetes cluster and technical guidance on performing various tasks. Buoyant also provides general guidance on Linkerd security. However, there is no centralized and comprehensive security documentation that enumerates common misconfigurations and system caveats.	**Moderate**
Denial of Service	A number of basic controls that aid in mitigating denial-of-service conditions, such as timeouts and retries, are in place. However, there is no support for rate-limiting controls.	**Moderate**
Identification and Authorization	Components that are part of the Linkerd infrastructure must obtain a certificate from the identity controller in	**Satisfactory**

	order to communicate and authenticate via mTLS through their sidecar proxies.	
System and Communications Protection	Due to the use of mTLS and Kubernetes role-based access controls, Linkerd's system and communication protection is satisfactory. However, as indicated under the "Access Controls" category, we believe that isolating components more granularly and implementing access controls and encryption for some APIs, such as proxy admin and metrics endpoints, may result in a stronger security posture. Regarding encryption, TLS communications are enforced for pod-to-pod communications. However, plain HTTP communications from external endpoints to the cluster could occur if an operator uses port forwarding to forward endpoints such as the dashboard to the network.	**Moderate**

System Diagram

The following diagram depicts the relationships between the target system's various components and trust zones, as well as the paths that threat actors could take within them.



Linkerd system diagram

Components

Linkerd is a lightweight service mesh for Kubernetes. Linkerd provides features to route messages between complex infrastructural components with ease and to improve system observability. Proxies in the form of containers are injected into user-application pods, while init containers are used to route application traffic to the proxies. To communicate securely between each other, these proxies look up relevant certificate information from "identity" containers and routing information from "destination" containers. Linkerd also offers load balancing functionality for outgoing traffic.

Component	Description
Command-Line Interface (CLI) Tool	The CLI is used to configure an existing Kubernetes deployment with Linkerd. With the CLI, users can set up relevant service accounts, run health checks on deployments, and annotate pods housing user applications that should have Linkerd proxies installed. The CLI does not deploy applications directly, but rather modifies an application's YAML file, which will be deployed with `kubectl`.
`kubectl`	The `kubectl` command is used to deploy CLI-modified YAML definitions to integrate Linkerd with an application.
`proxy-injector`	The `proxy-injector` is a single Kubernetes controller per cluster used to monitor pods for annotations indicating that Linkerd proxies should be installed. When it discovers such annotations, the `proxy-injector` configures the `proxy-init` and `linkerd-proxy` containers into the relevant pod to begin routing traffic appropriately.
`proxy-init`	The `proxy-init` is a Kubernetes init container used to configure IP tables for a given application in a pod. This container starts and exits before the application container runs to ensure traffic is routed to the `linkerd-proxy`. Each component in the Linkerd control and data planes also has its own proxy, each of which is deployed as a container in its respective pod.
`linkerd-proxy`	The `linkerd-proxy` is a Kubernetes container injected into an application pod by the `proxy-injector`, accepts traffic routed from the application, and queries the identity and destination containers to route traffic to the intended destination, leveraging mTLS for communication when possible. The `linkerd-proxy` is included in the initial Linkerd deployment for components such as the `proxy-injector` and the destination and identity services. Proxies also offer load balancing for each application

	pod.
Identity Service	The identity service is a single Kubernetes pod per cluster; it maintains certificate information for components managed by Linkerd proxies. It acts as a TLS certificate authority that accepts certificate signing requests from proxies and returns signed certificates. The identity service is issued at proxy initialization time and is used for proxy-to-proxy connections to implement mTLS.
Destination Service	The destination service is a single Kubernetes pod per cluster that maintains connection routing information for proxied traffic.
User Applications	This component refers to user-deployed Kubernetes applications that can be integrated with Linkerd.
`linkerd-viz` Extensions	The `linkerd-viz` Kubernetes namespace includes a number of diagnostics and metrics extensions that can be added to a Linkerd-integrated Kubernetes deployment, including the TAP injector, the web UI, and Prometheus metrics.
External Services	External services are those that do not live in the Kubernetes environment but are reachable by application pods. For instance, an external service could be a public server or a service hosted on a local machine/network.

Trust Zones

Systems include logical “trust boundaries” or “zones” in which components may have different criticality or sensitivity. Therefore, to further analyze a system, we decompose components into zones based on shared criticality rather than physical placement in the system. Trust zones capture logical boundaries in which controls should or could be enforced by the system and that allow designers to implement interstitial controls and policies between zones of components as needed.

Zone	Description	Included Components
External	This zone comprises external components that do not live in the Kubernetes environment, such as external websites and locally run tools.	• CLI Tool • `kubectl` • External Services

User Application Namespaces	This zone comprises the underlying components of user applications, including components that Linkerd injects into the application namespaces to facilitate proxying.	• User Applications
		  • `proxy-init`
		  • `linkerd-proxy`

Linkerd Namespace	This zone describes components in the Linkerd namespace that are shared among pods within a cluster. These components help install Linkerd components into user applications and record routing information for underlying proxies.	• `proxy-injector` (controller)
		  • `linkerd-proxy`
		• Identity Service (controller)
		  • `linkerd-proxy`
		• Destination Service (controller)
		  • `linkerd-proxy`

`linkerd-viz` Namespace	This zone comprises extensions that collect metrics and provide diagnostics information for Linkerd-integrated Kubernetes deployments.	• `linkerd-viz` Extensions

Trust Zone Connections

At a design level, trust zones are delineated by the security controls that enforce the differing levels of trust within each zone. Therefore, it is necessary to ensure that data cannot move between trust zones without first satisfying the intended trust requirements of its destination. We enumerate such connections between trust zones below.

Originating Zone	Destination Zone	Data Description	Connection Type	Authentication Type
External	External	The “check” command returns the latest version of Linkerd from linkerd.io.	HTTPS	TLS
		The “inject” command pulls a YAML file from a given URL.	URI-specified protocol (HTTP, HTTPS, etc.)	URI-specified protocol (None, TLS, etc.)
External	User Application Namespaces	User application containers can interact with arbitrary external services (e.g., GitHub).	*	*
		The CLI tool modifies infrastructural YAML definitions for `kubectl` to deploy. When `kubectl` deploys these definitions, Linkerd is integrated into relevant infrastructural components over the Kubernetes API.	HTTP, HTTPS	None, TLS

External	Linkerd Namespace	The CLI tool modifies infrastructural YAML definitions for `kubectl` to deploy. When `kubectl` deploys these definitions, Linkerd is integrated into relevant infrastructural components over the Kubernetes API. This includes control of trusted root and webhook certificates.	HTTP, HTTPS	None, TLS
External	`linkerd-viz` Namespace	Users can reach the web UI if the dashboard is enabled and exposed outside of the localhost.	HTTP (external to `linkerd-viz` proxies)	None
Linkerd Namespace	External	The Heartbeat cron job contacts linkerd.io with runtime telemetry; telemetry is enabled by default.	HTTPS	TLS (terminated by CloudFlare)
Linkerd Namespace	User Application Namespaces	The `proxy-injector` injects proxy components into annotated pods in user application namespaces through patches provided to the Kubernetes API.	HTTPS	TLS
		The identity service acts as a certificate	gRPC	One-way TLS, then mTLS

		authority and signs certificate signing requests for proxies.		after the certificate is obtained from the identity service
		The destination service provides routing information for `linkerd-proxy` sidecars. Proxies query this information (read-only). A race condition could cause some traffic to be one-way/partially TLS-encrypted until certificates are exchanged.	gRPC	One-way TLS, then mTLS after the certificate is obtained from the identity service
Linkerd Namespace	Linkerd Namespace	All components within this trust zone reach out to the identity provider to be able to communicate.	gRPC	mTLS
		Outbound connections go through the Kubernetes API server and may hit the destination controller to resolve routing information.	gRPC	mTLS
Linkerd Namespace	`linkerd-viz` Namespace	Proxies in the Linkerd namespace	HTTPS	mTLS

		issue responses returned from queries made by `linkerd-viz` components.		
User Application Namespaces	External	User application containers can interact with arbitrary external services with or without the use of Linkerd proxies. Proxies record TCP metrics for connections (number of times connected, etc.).	*	*
User Application Namespaces	User Application Namespaces	A user application maintains two-way communication with another application through its sidecar proxy (`linkerd-proxy`), as defined by IP tables configured earlier by the `proxy-init` container. Only TCP traffic can be proxied.	TCP	*
		The `linkerd-proxy` exposes admin endpoints on the localhost to perform actions such as shutting down the machine or checking the logging settings.	HTTPS	mTLS

User Application Namespaces	Linkerd Namespace	When a user application's sidecar proxy (`linkerd-proxy`) is first launched, it reaches out to the identity provider for a new certificate. It then reaches out to the identity provider periodically for certificate rotation. The lifetime of a certificate is 24 hours by default.	gRPC	One-way TLS, then mTLS after the certificate is obtained from the identity service
		A user application's sidecar proxy (`linkerd-proxy`) reaches out to the destination provider for routing information. The proxy also reaches out to the destination provider for the inbound authorization policy at startup.	gRPC	One-way TLS, then mTLS after the certificate is obtained from the identity service
User Application Namespaces	`linkerd-viz` Namespace	`linkerd-viz` extensions inject a TAP server component into the user application namespace. The user application reads config maps from the `linkerd-viz` namespace to provide diagnostics	HTTPS	TLS

		data.		
		`linkerd-viz` endpoints are reachable from the user application namespace.	HTTP	None for Prometheus endpoints Namespace validation for web endpoints
`linkerd-viz` Namespace	External	The web UI serves a front end to clients, and JavaScript makes calls to external content services.	HTTP	None
		Prometheus can export metrics to external services. Grafana can send data to external Grafana APIs.	HTTPS	TLS Authentication controls imposed by external services
`linkerd-viz` Namespace	User Application Namespaces	`linkerd-viz` extensions reach out to user application namespace pods to collect metrics and diagnostics data via their respective admin server endpoints. When TAP is enabled, a TAP injector injects TAP controllers in user application pods.	HTTPS	mTLS
`linkerd-viz` Namespace	Linkerd Namespace	`linkerd-viz` extensions query Linkerd namespace proxies for metrics	HTTPS	TLS

		data. The web container reaches out to the Linkerd namespace to read config maps.		
		The TAP controller reaches out to proxies in the Linkerd namespace for diagnostics data.	gRPC	mTLS
		`linkerd-viz` proxies obtain routing data from the destination controller and identities from the identity controller.	gRPC	mTLS
`linkerd-viz` Namespace	`linkerd-viz` Namespace	`linkerd-viz` extensions reach out to their own proxies when querying metrics data.	HTTPS	TLS
		The TAP controller reaches out to proxies in `linkerd-viz` for diagnostics data.	gRPC	TLS
		The web container reaches out to the Prometheus container to obtain and display metrics data in the dashboard.	HTTPS	TLS Linkerd authorization policies (inbound)

Threat Actors

Similarly to establishing trust zones, defining malicious actors before conducting a threat model is useful in determining which protections, if any, are necessary to mitigate or remediate a vulnerability. Additionally, we define other "users" of the system who may be impacted by, or induced to undertake, an attack. For example, in a confused deputy attack such as cross-site request forgery, a normal user could be both the victim and the direct attacker, even if the user were induced to undertake the action by a secondary attacker. We will refer to these actors in the descriptions of the findings uncovered by creating the threat modeling process.

Actor	Description
Internal Attacker	An attacker who has transited one or more trust boundaries, such as an attacker with cluster access
External Attacker	An attacker who is external to the cluster and is unauthenticated, such as an attacker with control over external services
Infrastructure Operator	An administrator tasked with operating and maintaining infrastructure within one or many of the namespaces of a Linkerd-integrated cluster
Application User	An end user who accesses meshed applications

Threat Actor Paths

Additionally, defining attackers' paths through the various zones is useful when analyzing potential controls, remediations, and mitigations that exist in the current architecture.

Originating Zone	Destination Zone	Actor	Description
Any	Any	Internal Attacker	Internal attackers may have existing privileges or access to a wide range of resources, such as the Kubernetes cluster that hosts Linkerd and user application namespace infrastructure. They may also be able to obtain signed mTLS certificates from the identity provider to communicate with other components within the system. Therefore, for strong non-repudiation of actions, controls must be in place to log all actions within the system and to ensure that actors are authorized to undertake certain actions.
		External Attacker	External attackers may target any exposed paths within the Linkerd or user application namespaces. Proxied connections and endpoints are managed by the infrastructure operator and are exposed as needed. Documentation should describe best practices when configuring forwarding rules and Kubernetes access controls separating trust boundaries. This will deter external attackers from transiting trust boundaries.
		Infrastructure Operator	Operators may misconfigure Linkerd or user application services such that they do not allow actors to access them as expected. To mitigate such concerns, errors in

			the system should be logged accordingly to provide an audit trail alongside extensive administrator documentation.
		Application User	An application user may interact with an exposed user application in any number of ways. Therefore, the availability or integrity of any user application may be at risk. It is important for Linkerd to employ access controls for isolation and to log events so that an audit trail is available in the event of an attack or system deficiency.
External	External	Internal Attacker	An internal attacker with access to external services such as linkerd.io servers can misreport version checks made by the CLI tool, install backdoors in distributions, or otherwise affect users in a number of ways without immediate access to a cluster.
		External Attacker	An external attacker may attempt to intercept communications between the CLI tool and external services if connections are insecure. The CLI tool may load YAML files from an external source during configuration. If insecure protocols such as HTTP are specified in the URI path, an external attacker may manipulate traffic to inject malicious payloads into the underlying Kubernetes infrastructure. Best practices regarding the use of external configuration files within Linkerd should be described in the documentation. This will decrease the risk of the use of insecure protocols.
External	User	Internal	User applications may decide to route

	Application Namespaces	Attacker	proxied or unproxied connections to arbitrary external services. User applications are responsible for effectively and appropriately leveraging authentication, cryptography, data validation, and Kubernetes access controls to prevent lateral movement across namespaces. Care must be taken not to expose critical endpoints to the user application's sidecar proxy admin endpoints to the outside world, as they can be used to shut down the proxy, leak metrics, or perform denial-of-service attacks.
		Infrastructure Operator	User applications share a pod with their sidecar proxies and respective init containers. Therefore, operators of user application infrastructure should be aware that if a user application is compromised, lateral components such as the sidecar proxy could also be compromised. This may expose routing information and certificates within the namespace. Effective logging is crucial for maintaining an audit trail for such scenarios.
External	Linkerd Namespace	Internal Attacker	An internal attacker with access to an external service that hosts an infrastructure operator's YAML files may be able to manipulate the underlying infrastructure. The documentation should note that accepting external configurations from arbitrary sources is always risky.
		External Attacker	If the specified connection protocol is insecure, an external attacker without access to any external service may

			perform a man-in-the-middle attack against the CLI tool as it fetches infrastructural YAML definitions. The documentation should note the insecure protocols that are accepted by the CLI tool, which should be used with caution.
External	`linkerd-viz` Namespace	External Attacker	An external attacker can conduct supply chain attacks to compromise the confidentiality of data provided by `linkerd-viz` APIs and extensions. For instance, an attacker could modify JavaScript libraries used by `linkerd-viz` components for UI dashboards. An attacker may be able to reach `linkerd-viz` APIs externally via a number of common web application attacks, such as cross-site scripting, cross-site request forgery, and clickjacking.
		Infrastructure Operator	Infrastructure operators must ensure that they do not unintentionally expose `linkerd-viz` applications externally, and they must understand the risks inherent in exposing `linkerd-viz` related data. This is especially important given that `linkerd-viz` components such as the web UI do not require authorization once exposed.
User Application Namespaces	External	Internal Attacker	An internal attacker could exploit external services by leveraging connections to them from user applications. For instance, if a Linkerd-integrated user application connects to a remote web server, an internal attacker could send malicious payloads to the server

			through such a connection.
User Application Namespaces	User Application Namespaces	Internal Attacker	An internal attacker can disable an application's sidecar proxy, effectively bringing the user application offline. This can be done by exposing admin endpoints on the sidecar proxy. Exposing admin endpoints can also expose proxy metrics and other potentially sensitive information.
		Infrastructure Operator	Infrastructure operators must ensure that appropriate logging procedures are in place; logs can serve as an audit trail in the event of an attack. Logs can help determine whether two components have tried to communicate inappropriately, indicating that the pod may have been compromised.
User Application Namespaces	Linkerd Namespace	Internal Attacker	An internal attacker could send requests to the destination and identity controllers to try to perform a denial-of-service attack against them. For instance, if an attacker changes the availability of an application, causing updates to be made in the destination controller, and continuously requests a newly signed certificate (as if he were a new proxy initializing for the first time), the identity service would be forced to perform signing operations for the attacker. This highlights the need for appropriate logging within the Linkerd namespace to prevent such attacks and for access controls to ensure that routing information for unrelated pods or containers cannot be obtained.

			Additionally, the documentation should describe best practices regarding the use of Kubernetes access controls to ensure that lateral movement across namespaces is not possible.
		Infrastructure Operator	Infrastructure operators must ensure that they maintain restrictive access controls, limiting access to admin endpoints and other services that could increase the attack surface area or interrupt the availability of the system. Additionally, they must be conscious of access controls and namespace isolations so as not to allow lateral movement across namespaces.
User Application Namespaces	`linkerd-viz` Namespace	Internal Attacker	Internal attackers with access that is restricted to the application namespace could reach Prometheus endpoints to obtain metrics data that could give them insight into other cluster components that they would not otherwise have visibility into.
		External Attacker	If an external attacker finds a vulnerability in an application in the user application namespace that gives her access to the application server, she can call the Prometheus API to obtain information about other resources in the cluster.
Linkerd Namespace	External	Internal Attacker	An internal attacker with access to the Linkerd namespace may be able to change certificate and routing information stored within the identity and destination services, respectively. This could prevent traffic from reaching the intended external services.

		Infrastructure Operator	An operator may unintentionally expose access to the sidecar proxies for the identity, destination, and `proxy-injector` controllers, allowing external traffic to be routed into the proxy and exposing the admin panel to the outside world. Kubernetes access controls must be managed appropriately to prevent lateral movement within the cluster.
Linkerd Namespace	User Application Namespaces	Internal Attacker	An internal attacker with access to the Linkerd namespace may be able to change certificate and routing information stored within the identity and destination services, respectively. This could expose user application traffic to unintended parties and could leak sensitive secrets. Furthermore, an internal attacker with access to the Linkerd namespace may be able to inject malicious components into the user application namespaces, as the `proxy-injector` does. This could compromise the user application namespaces.
Linkerd Namespace	Linkerd Namespace	Internal Attacker	An internal attacker may obtain access to certificates or destination routing information for Linkerd proxy–integrated components within the cluster. An attacker may also be able to change routing information to leak sensitive application data or to break Linkerd controllers across the cluster. An internal attacker may also access sidecar proxy admin shutdown endpoints, which may affect service availability.

Linkerd Namespace	`linkerd-viz` Namespace	Internal Attacker	An internal attacker who is able to forward ports for services running in the Linkerd namespace may be able to shut down control plane proxies, preventing the `linkerd-viz` extensions from querying Linkerd control plane components for metrics data.
`linkerd-viz` Namespace	External	Internal Attacker	An internal attacker with the ability to configure external connections for the Prometheus and Grafana extensions could modify their configurations so that they communicate with attacker-controlled endpoints, allowing the attacker to collect cluster metrics that could be useful when formulating attacks.
`linkerd-viz` Namespace	User Application Namespaces	Internal Attacker	An internal attacker could compromise the TAP injector, allowing her to leak user application data and compromise pods in the user application namespaces.
`linkerd-viz` Namespace	Linkerd Namespace	Internal Attacker	An internal attacker could compromise the TAP injector to modify annotations made to proxies in the Linkerd namespace, allowing him to compromise the proxies and potentially disable communication with control plane components.
`linkerd-viz` Namespace	`linkerd-viz` Namespace	Internal Attacker	An internal attacker who is able to forward ports for the `linkerd-viz` endpoint may be able to disable `linkerd-viz` proxies by leveraging the shutdown admin endpoint exposed on the sidecar proxies, effectively removing the ability to collect metrics and diagnostics data typically provided by `linkerd-viz` extensions.

		Infrastructure Operator	An infrastructure operator must ensure that `linkerd-viz` extensions are not exposed on the internet, as the exposure of a `linkerd-viz` extension could affect the availability of other components within the same `linkerd-viz` namespace. Operators should be careful to minimize the exposed attack surface area.

Summary of Findings

The table below summarizes the findings of the review, including type and severity details.

ID	Title	Type	Severity
1	Lack of rate-limiting mechanisms in the identity service	Denial of Service	**Low**
2	Lack of rate-limiting mechanisms in the destination service	Denial of Service	**Low**
3	CLI tool allows the use of insecure protocols when externally sourcing infrastructure definitions	Awareness and Training	**Medium**
4	Exposure of admin endpoint may affect application availability	Awareness and Training	**Medium**
5	Go's pprof endpoints enabled by default in all admin servers	Audit and Accountability	**Informational**
6	Lack of access controls on the linkerd-viz dashboard	Access Controls	**Low**
7	Prometheus endpoints reachable from the user application namespace	Access Controls	**Low**
8	Lack of egress access controls	Access Controls	**Low**
9	Prometheus endpoints are unencrypted and unauthenticated by default	Access Controls	**Low**
10	Shared identity and destination services in a cluster poses risks to multi-application clusters	System and Communications Protection	**Medium**
11	Lack of isolation between components and their sidecar proxies	System and Communications Protection	**Low**

12	Lack of centralized security best practices documentation	Awareness and Training	**Informational**
13	Unclear distinction between Linkerd and Linkerd2 in official Linkerd blog post guidance	Awareness and Training	**Informational**
14	Insufficient logging of outbound HTTPS calls	Audit and Accountability	**Low**

Detailed Findings

1. Lack of rate-limiting mechanisms in the identity service	
Severity: **Low**	Difficulty: **High**
Type: Denial of Service	Finding ID: TOB-LKDTM-1
Target: Identity service	

Description
The identity service issues signed certificates to sidecar proxies within Linkerd-integrated infrastructure. When proxies initialize for the first time, they request a certificate from the identity service. However, the identity service lacks sufficient rate-limiting mechanisms, which may make it prone to denial-of-service attacks.

Because identity controllers are shared among pods in a cluster, a denial of service of an identity controller may affect the availability of applications across the cluster.

Threat Scenario
An attacker obtains access to the sidecar proxy in one of the user application namespaces. Due to the lack of rate-limiting mechanisms within the identity service, the proxy can now repeatedly request a newly signed certificate as if it were a proxy sidecar initializing for the first time.

Recommendations
Short term, add rate-limiting mechanisms to the identity service to prevent a single pod from requesting too many certificates or performing other computationally intensive actions.

Long term, ensure that appropriate rate-limiting mechanisms exist throughout the infrastructure to prevent denial-of-service attacks. Where possible, implement stricter access controls to ensure that components cannot interact with APIs more than necessary. Additionally, ensure that the system sufficiently logs events so that an audit trail is available in the event of an attack.

2. Lack of rate-limiting mechanisms in the destination service	
Severity: **Low**	Difficulty: **High**
Type: Denial of Service	Finding ID: TOB-LKDTM-2
Target: Destination service	

Description
The destination service contains traffic-routing information for sidecar proxies within Linkerd-integrated infrastructure. However, the destination service lacks sufficient rate-limiting mechanisms, which may make it prone to denial-of-service attacks if a pod repeatedly changes its availability status.

Because destination controllers are shared among pods in a cluster, a denial of service of a destination controller may affect the availability of applications across the cluster.

Threat Scenario
An attacker obtains access to the sidecar proxy in one of the user application namespaces. Due to the lack of rate-limiting mechanisms within the destination service, the proxy can now repeatedly request routing information or change its availability status to force updates in the controller.

Recommendations
Short term, add rate-limiting mechanisms to the destination service to prevent a single pod from requesting too much routing information or performing state updates too quickly.

Long term, ensure that appropriate rate-limiting mechanisms exist throughout the infrastructure to prevent denial-of-service attacks. Where possible, implement stricter access controls to ensure that components cannot interact with APIs more than necessary. Additionally, ensure that the system sufficiently logs events so that an audit trail is available in the event of an attack.

3. CLI tool allows the use of insecure protocols when externally sourcing infrastructure definitions

Severity: **Medium**	Difficulty: **Low**
Type: Awareness and Training	Finding ID: TOB-LKDTM-3
Target: CLI tool	

Description
When using the command-line interface (CLI) tool, an operator may source infrastructural YAML definitions from a URI path specifying any protocol, such as `http://` or `https://`.

Therefore, a user could expose sensitive information when using an insecure protocol such as HTTP. Furthermore, the Linkerd documentation does not warn users about the system's use of insecure protocols.

Threat Scenario
An infrastructure operator integrates Linkerd into her infrastructure. When doing so, she uses the CLI tool to fetch YAML definitions over HTTP. Unbeknownst to her, the use of HTTP has made her data visible to attackers on the local network. Her data is also prone to man-in-the-middle attacks.

Recommendations
Short term, disallow the use of insecure protocols within the CLI tool when sourcing external data. Alternatively, provide documentation and best practices regarding the use of insecure protocols when externally sourcing data within the CLI tool.

4. Exposure of admin endpoint may affect application availability	
Severity: **Medium**	Difficulty: **Medium**
Type: Awareness and Training	Finding ID: TOB-LKDTM-4
Target: `linkerd-proxy`	

Description
User application sidecar proxies expose an admin endpoint that can be used for tasks such as shutting down the proxy server and collecting metrics. This endpoint is exposed to other components within the same pod. Therefore, an internal attacker could shut down the proxy, affecting the user application's availability.

Furthermore, the admin endpoint lacks access controls, and the documentation does not warn of the risks of exposing the admin endpoint over the internet.

Threat Scenario
An infrastructure operator integrates Linkerd into his Kubernetes cluster. After a new user application is deployed, an underlying component within the same pod is compromised. An attacker with access to the compromised component can now laterally send a request to the admin endpoint used to shut down the proxy server, resulting in a denial of service of the user application.

Recommendations
Short term, employ authentication and authorization mechanisms behind the admin endpoint for proxy servers.

Long term, document the risks of exposing critical components throughout Linkerd. For instance, it is important to note that exposing the admin endpoint on a user application proxy server may result in the exposure of a shutdown method, which could be leveraged in a denial-of-service attack.

5. Go's pprof endpoints enabled by default in all admin servers	
Severity: **Informational**	Difficulty: **High**
Type: Audit and Accountability	Finding ID: TOB-LKDTM-5
Target: HTTP admin servers	

Description
All core components of the Linkerd infrastructure, in both the data and control planes, have an admin server with Go's server runtime profiler (`pprof`) endpoints on `/debug/pprof` enabled by default. These servers are not exposed to the rest of the cluster or to the local network by default.

Threat Scenario
An attacker scans the network in which a Linkerd cluster is configured and discovers that an operator forwarded the admin server port to the local network, exposing the `pprof` endpoints to the local network. He connects a profiler to it and gains access to debug information, which assists him in mounting further attacks.

Recommendations
Short term, add a check to `http.go` that enables `pprof` endpoints only when Linkerd runs in debug or test mode.

Long term, audit all debug-related functionality to ensure it is not exposed when Linkerd is running in production mode.

References
- Your pprof is showing: IPv4 scans reveal exposed net/http/pprof endpoints

6. Lack of access controls on the linkerd-viz dashboard	
Severity: **Low**	Difficulty: **High**
Type: Access Controls	Finding ID: TOB-LKDTM-6
Target: `linkerd-viz` extensions	

Description
Linkerd operators can enable a set of metrics-focused features by adding the `linkerd-viz` extension. Doing so enables a web UI dashboard that lists detailed information about the namespaces, services, pods, containers, and other resources in a Kubernetes cluster in which Linkerd is configured. Operators can enable Kubernetes role-based access controls to the dashboard; however, no access control options are provided by Linkerd.

Threat Scenario
An attacker scans the network in which a Linkerd cluster is configured and discovers an exposed UI dashboard. By accessing the dashboard, she gains valuable insight into the cluster. She uses the knowledge gained from exploring the dashboard to formulate attacks that would expand her access to the network.

Recommendations
Short term, document recommendations for restructuring access to the `linkerd-viz` dashboard.

Long term, add authentication and authorization controls for accessing the dashboard. This could be done by implementing tokens created via the CLI or client-side authorization logic.

7. Prometheus endpoints reachable from the user application namespace	
Severity: **Low**	Difficulty: **High**
Type: Access Controls	Finding ID: TOB-LKDTM-7
Target: `linkerd-viz` extensions	

Description
The `linkerd-viz` extension provides a Prometheus API that collects metrics data from the various proxies and controllers used by the control and data planes. Metrics can include various labels with IP addresses, pod IDs, and port numbers.

Threat Scenario
An attacker gains access to a user application pod and calls the API directly to read Prometheus metrics. He uses the API to gain information about the cluster that aids him in expanding his access across the Kubernetes infrastructure.

Recommendations
Short term, disallow access to the Prometheus extension from the user application namespace. This could be done in the same manner in which access to the web dashboard is restricted from within the cluster (e.g., by allowing access only for specific hosts).

8. Lack of egress access controls	
Severity: **Low**	Difficulty: **High**
Type: Access Controls	Finding ID: TOB-LKDTM-8
Target: Destination service, `linkerd-proxy`	

Description
Linkerd provides access control mechanisms for ingress traffic but not for egress traffic. Egress controls would allow an operator to impose important restrictions, such as which external services and endpoints that a meshed application running in the application namespace can communicate with.

Threat Scenario
A user application becomes compromised. As a result, the application code begins making outbound requests to malicious endpoints. The lack of access controls on egress traffic prevents infrastructure operators from mitigating the situation (e.g., by allowing the application to communicate with only a set of allowlisted external services).

Recommendations
Short term, add support for enforcing egress network policies. A GitHub issue to implement this recommendation already exists in the Linkerd repository.

9. Prometheus endpoints are unencrypted and unauthenticated by default	
Severity: **Low**	Difficulty: **High**
Type: Access Controls	Finding ID: TOB-LKDTM-9
Target: `linkerd-viz` extensions	

Description
The `linkerd-viz` extension provides a Prometheus API that collects metrics data from the various proxies and controllers used by the control and data planes. However, this endpoint is unencrypted and unauthenticated, lacking access and confidentiality controls entirely.

Threat Scenario
An attacker gains access to a sibling component within the same namespace in which the Prometheus endpoint exists. Due to the lack of access controls, the attacker can now laterally obtain Prometheus metrics with ease. Additionally, due to the lack of confidentiality controls, such as those implemented through the use of cryptography, connections are exposed to other parties.

Recommendations
Short term, consider implementing access controls within Prometheus and Kubernetes to disallow access to the Prometheus metrics endpoint from any machine within the cluster that is irrelevant to Prometheus logging. Additionally, implement secure encryption of connections with the use of TLS within Prometheus or leverage existing Linkerd mTLS schemes.

10. Shared identity and destination services in a cluster poses risks to multi-application clusters

Severity: **Medium**	Difficulty: **High**
Type: System and Communications Protection	Finding ID: TOB-LKDTM-10
Target: Destination and identity services	

Description
The identity and destination controllers are meant to convey certificate and routing information for proxies, respectively. However, only one identity controller and one destination controller are deployed in a cluster, so they are shared among all application pods within a cluster. As a result, a single application pod could pollute records, causing denial-of-service attacks or otherwise compromising these cluster-wide components.

Additionally, a compromise of these cluster-wide components may result in the exposure of routing information for each application pod.

Although the Kubernetes API server is exposed with the same architecture, it may be beneficial to minimize the attack surface area and the data that can be exfiltrated from compromised Linkerd components.

Threat Scenario
An attacker gains access to a single user application pod and begins to launch attacks against the identity and destination services. As a result, these services cannot serve other user application pods. The attacker later finds a way to compromise one of these two services, allowing her to leak sensitive application traffic from other user application pods.

Recommendations
Short term, implement per-pod identity and destination services that are isolated from other pods. If this is not viable, consider documenting this caveat so that users are aware of the risks of hosting multiple applications within a single cluster.

11. Lack of isolation between components and their sidecar proxies	
Severity: **Low**	Difficulty: **High**
Type: System and Communications Protection	Finding ID: TOB-LKDTM-11
Target: `linkerd-proxy`	

Description
Within the Linkerd, `linkerd-viz`, and user application namespaces, each core component lives alongside a `linkerd-proxy` container, which proxies the component's traffic and provides mTLS for internal connections. However, because the sidecar proxies are not isolated from their corresponding components, the compromise of a component would mean the compromise of its proxy, and vice versa.

This is particularly interesting when considering the lack of access controls for some components, as detailed in TOB-LKDTM-4: proxy admin endpoints are exposed to the applications they are proxying, allowing metrics collection and shutdown requests to be made.

Threat Scenario
An attacker exploits a vulnerability to gain access to a `linkerd-proxy` instance. As a result, the attacker is able to compromise the confidentiality, integrity, and availability of lateral components, such as user applications, identity and destination services within the Linkerd namespace, and extensions within the `linkerd-proxy` namespace.

Recommendations
Short term, document system caveats and sensitivities so that operators are aware of them and can better defend themselves against attacks. Consider employing health checks that verify the integrity of proxies and other components to ensure that they have not been compromised.

Long term, investigate ways to isolate sidecar proxies from the components they are proxying (e.g., by setting stricter access controls or leveraging isolated namespaces between proxied components and their sidecars).

12. Lack of centralized security best practices documentation	
Severity: **Informational**	Difficulty: **Informational**
Type: Awareness and Training	Finding ID: TOB-LKDTM-12
Target: Linkerd	

Description
While security recommendations are included throughout Linkerd's technical guidance documents, there is no centralized guidance on security best practices. Furthermore, the documentation on securing clusters lacks guidance on security best practices such as configuring timeouts and retries, authorization policy recommendations for defense in depth, and locking down access to `linkerd-viz` components.

Threat Scenario
A user is unaware of security best practices and configures Linkerd in an insecure manner. As a result, her Linkerd infrastructure is prone to attacks that could compromise the confidentiality, integrity, and availability of data handled by the cluster.

Recommendations
Short term, develop centralized documentation on security recommendations with a focus on security-in-depth practices for users to follow. This guidance should be easy to locate should any user wish to follow security best practices when using Linkerd.

13. Unclear distinction between Linkerd and Linkerd2 in official Linkerd blog post guidance

Severity: **Informational**	Difficulty: **Informational**
Type: Awareness and Training	Finding ID: TOB-LKDTM-13
Target: Linkerd	

Description
The official Linkerd documentation clearly indicates the version of Linkerd that each document pertains to. For instance, documentation specific to Linkerd 1.x displays a message stating, “This is not the latest version of Linkerd!” However, guidance documented in blog post form on the same site does not provide such information. For instance, the first result of a Google search for “Linkerd RBAC” is a Linkerd blog post with guidance that is applicable only to linkerd 1.x, but there is no indication of this fact on the page. As a result, users who rely on these blog posts may misunderstand functionality in Linkerd versions 2.x and above.

Threat Scenario
A user searches for guidance on implementing various Linkerd features and finds documentation in blog posts that applies only to Linkerd version 1.x. As a result, he misunderstands Linkerd and its threat model, and he makes configuration mistakes that lead to security issues.

Recommendations
Short term, on Linkerd blog post pages, add indicators similar to the UI elements used in the Linkerd documentation to clearly indicate which version each guidance page applies to.

14. Insufficient logging of outbound HTTPS calls	
Severity: **Low**	Difficulty: **High**
Type: Audit and Accountability	Finding ID: TOB-LKDTM-14
Target: `linkerd-viz`	

Description
Linkerd operators can use the `linkerd-viz` extensions such as Prometheus and Grafana to collect metrics for the various proxies in a Linkerd infrastructure. However, these extensions do not collect metrics on outbound calls made by meshed applications. This limits the data that operators could use to conduct incident response procedures if compromised applications reach out to malicious external services and servers.

Threat Scenario
A meshed application running in the data plane is compromised as a result of a supply chain attack. Because outbound HTTPS calls are not logged, Linkerd operators are unable to collect sufficient data to determine the impact of the vulnerability.

Recommendations
Short term, implement logging for outbound HTTPS connections. A GitHub issue to implement this recommendation already exists in the Linkerd repository but is still unresolved as of this writing.

A. Methodology

A Trail of Bits threat modeling assessment is intended to provide a detailed analysis of the risks that an application faces at the structural and operational level; the goal is to assess the security of the application's design rather than its implementation details. During these assessments, engineers rely heavily on frequent meetings with the client's developers and on extensive reading of all documentation provided by the client. Code review and dynamic testing are not part of the threat modeling process, although engineers may occasionally consult the codebase or a live instance of the project to verify assumptions about the system's design.

Engineers begin a threat modeling assessment by identifying the safeguards and guarantees that are critical to maintaining the target system's confidentiality, integrity, and availability. These *security controls* dictate the assessment's overarching scope and are determined by the requirements of the target system, which may relate to technical and reputational concerns, legal liability, and regulatory compliance.

With these security controls in mind, engineers then divide the system into logical *components*—discrete elements that perform specific tasks—and establish *trust zones* around groups of components that lie within a common trust boundary. They identify the types of data handled by the system, enumerating the points at which data is sent, received, or stored by each component, as well as within and across trust boundaries.

After establishing a detailed map of the target system's structure and data flows, engineers then identify *threat actors*—anyone who might threaten the target's security, including both malicious external actors and naive internal actors. Based on each threat actor's initial privileges and knowledge, engineers then trace *threat actor paths* through the system, determining the controls and data that a threat actor might be able to improperly access, as well as the safeguards that prevent such access. Any viable attack path discovered during this process constitutes a *finding*, which is paired with design recommendations to remediate gaps in the system's defenses.

Finally, engineers rate the strength of each security control, indicating the general robustness of that type of defense against the full spectrum of possible attacks. These ratings are provided in the Security Control Maturity Evaluation table.

B. Security Controls and Rating Criteria

The following tables describe the security controls and rating criteria used in this report.

Security Controls	
Category	**Description**
Access Controls	Controls related to authorization, session management, separation of duties, etc.
Audit and Accountability	Controls related to logging, non-repudiation, monitoring, analysis, reporting, etc.
Awareness and Training	Controls related to policies, procedures, and related capabilities
Denial of Service	Controls to defend against denial-of-service attacks impacting availability
Identification and Authentication	User and system identification and authentication controls
System and Communications Protection	Network-level controls to protect data

Rating Criteria	
Rating	**Description**
Strong	No concerns related to the security control were found.
Satisfactory	Only minor issues related to the security control were found; though the control may lack certain non-critical operational procedures or security measures, their absence does not expose users to a significant degree of risk. Remediation in this area is suggested but not urgent.
Moderate	Several issues that may expose users to some degree of risk were found. Remediation in this area is recommended.
Weak	Significant issues that are likely to expose users to a substantial degree of risk were found. Remediation in this area should be prioritized.
Missing	The security control was found to be nonexistent or totally ineffective for its intended purpose, despite being necessary for the system's security. The implementation of this control should be prioritized.

Not Applicable	The security control is not applicable to this review.
Not Considered	The security control was not considered in this review.
Further Investigation Required	Further investigation is required to reach a meaningful conclusion.

Severity Levels	
Severity	**Description**
Informational	The issue does not pose an immediate risk but is relevant to security best practices.
Undetermined	The extent of the risk was not determined during this engagement.
Low	The risk is small or is not one the client has indicated is important.
Medium	User information is at risk; exploitation could pose reputational, legal, or moderate financial risks.
High	The flaw could affect numerous users and have serious reputational, legal, or financial implications.

Difficulty Levels	
Difficulty	**Description**
Undetermined	The difficulty of exploitation was not determined during this engagement.
Low	The threat is well known or common; an attacker can exploit it without significant effort or specialized knowledge.
Medium	An attacker must acquire in-depth knowledge of the system or expend a non-trivial amount of effort in order to exploit this issue.
High	An attacker must acquire complex insider knowledge or privileged access to the system in order to exploit this issue.