niolabs

nio security compliance

white paper

INTRODUCTION

Operating with a **security-first** mindset

As part of our security best practices, niolabs closely follows and conforms to multiple industry standard security guidelines. These include both the **OWASP Top IoT Vulnerabilities** and the **Google IoT Security Requirements** documents. niolabs also publishes a **Security Response Policy** so that all users are informed of how security incidents are reported and handled.

table of contents

OWASP: **Top 10** IoT **Vulnerabilities**

OWASP, an international community driving the safety and security of the world's software, has published a Top 10 list of common IoT Vulnerabilities. The niolabs engineering team pays close attention to these and other reported vulnerabilities when designing software. Below you will find how niolabs addresses and handles each of the OWASP vulnerabilities.

1. Insecure Web Interface

The nio management app is the main web interface used for connecting directly with nio instances. It consists of the nio System Designer, the nio Deployment Manager, and the nio System Monitoring Tool. Access to this app is handled through a valid nio license and authentication and authorization are handled through our identity provider Auth0.

The nio management app is always available via HTTPS. Use of the HTTPS app also requires that your nio instances be running with TLS support. This is required in the nio deployment best practices.

Tokens for the management web interface use JSON Web Tokens (JWTs) for interaction with the nio hosted APIs and use the nio instance's security mechanism for communication. The default options for interacting with a nio instance are the use of basic authentication (username and password) and JWTs.

When using JSON Web Tokens, the tokens are signed at the identity provider, Auth0, and then validated against their public key and the token claims are confirmed.

The use of basic authentication in a nio instance requires creating a user and password when setting up the nio instance and running the instance behind TLS. This can be performed using the nio command line interface. Passwords are hashed using bcrypt before being stored in the nio instance.

Internal security audits are performed to protect this interface against common attacks like cross site scripting, SQL injection, or cross site request forgery. Updates are deployed to this hosted interface daily.

2. Insufficient Authentication/Authorization

All running instances of nio implement granular access control through built in security modules. We provide modules that authenticate users of a nio instance using basic credentials (username & password) or JSON Web Tokens (JWTs). Instance authorization settings for these modules can be configured in the project configuration settings. Authorization can be configured on each nio instance for each authenticated user. Through user level permissions, granular instance authorization can be configured for read, write, and execution access.

Instances running with the basic security module can have their users configured through the nio command line interface. Each user is created with permissions assigned to them. User credentials are stored on the device after being hashed and salted using bcrypt.

The nio authentication and authorization security module can also be extended to work with other identity providers. This makes it possible for enterprises to integrate their own identity solution into nio instances and then manage user access control from preexisting tools.

Pubkeeper Server topic permissions can be configured using the Pubkeeper Token Management tool. This allows for the generation of tokens that can be distributed to Pubkeeper clients with topic specific publish/subscribe rules attached. Through this tool, tokens can be revoked as necessary. Access to the Pubkeeper Token Management tool should be limited to administrators of a nio organization. This access is controlled in the Pubkeeper server's user database and is documented in the deployment best practices

3. Insecure Network Services

By default, nio instances open one port to traffic within a network. This port is attached to the nio API which provides authorization and authentication for all traffic within the network. This port can be configured through a setting in the project configuration. niolabs strongly recommends configuring TLS encryption for an instance before opening this port to external/public network traffic. This port can be disabled if needed through the project configuration settings on an instance.

Depending on the communication protocols that are used, running a nio instance may open additional ports. For example, running the ZeroMQ Pubkeeper brew means opening ports for network communication to occur on. The ports used by Pubkeeper brews are always configurable via the nio.conf file.

Internal security audits are run to identify and protect nio instances against common attacks like fuzzing, buffer overflow, or DoS. The built in instance security module acts as a gateway layer preventing malicious, malformed, or incorrect requests from affecting the device or instance logic.

4. Lack of Transport Encryption

nio instances can be configured with SSL/TLS certificates, either self-signed, or obtained from a trusted certificate authority. This allows for all nio API communication to be encrypted with TLS encryption. This behavior comes with zero configuration for all nio managed cloud instances and requires installing a certificate for local instances.

Data transfer through the Pubkeeper communication broker is end-to-end encrypted using AES encryption, by default. A shared key is exchanged between sender and receiver when the Pubkeeper match is established. All data packets sent and received by Pubkeeper clients are verified against an expected frame pattern.

5. Privacy Concerns

Because niolabs does not offer a cloud environment that collects data on the user's behalf, privacy of data in nio is fully in control of the one building nio services. This means getting through privacy and compliance teams happens faster and users have more control over where data lives.

niolabs optionally collects diagnostic data about nio instances for internal use. This data is not of any sensitive nature and user identity is not tied to the diagnostic data collected. The two types of diagnostic data collected are service status changes and signal volume information. This data is all gathered and sent through a diagnostic component built into the nio binary. This component can be disabled through settings in the project configuration of an instance.

Within a nio organization, user and team level permissions are implemented for access control of systems and instances. These permissions are configured by an organization administrator. For more information about organization level privacy settings the docs on organization management: https://docs.n.io/organizations/management.html.

Additional privacy information can be found within the niolabs privacy policy: https://niolabs.com/legal/privacy-policy.

6. Insecure Cloud Interface

All niolabs hosted cloud interfaces are secured using a third-party identity provider, Auth0, for user and authorization management. This trusted service handles account credential setup, lockout, and recovery.

When signing up for an account on the nio portal, no default password is created for a new user. The user enters their password following a secure password policy to create their account. Account recovery is handled through an Auth0 password reset. This process does not provide any unneeded information to identify active accounts.

Account lockout happens after repeated invalid attempts to login to a niolabs hosted cloud interface. Through this secure authorization service, we provide appropriate access to nio accounts in the nio account portal and management app.

When using any cloud hosted interface within the nio Platform all requests are run through TLS encryption by default. The niolabs portal which handles all user data creation is required to use an HTTPS connection for account security. This prevents any credentials from being exposed through the network.

Internal security audits are performed to protect these interfaces against common attacks like Cross Site Scripting, SQL Injection, or Cross Site Request Forgery. Updates are deployed to this hosted interface daily.

7. Insecure Mobile Interface

niolabs does not provide a downloadable mobile interface to interact with running nio instances. All cloud interfaces can be used with a mobile browser. Behavior through niolabs mobile browser interfaces are secured in the same way as cloud interfaces. There is no difference in how account recovery and lockout is handled with Auth0. TLS encryption is still used to protect account credentials and provide data security.

8. Insufficient Security Configurability

The nio Platform allows for many different security configurations. Different security modules can be packaged into a nio binary to allow different types of authentication and authorization for a running instance. The nio command line interface provides tools for managing users of an instance. Permissions can be configured for each user specified allowing for separation of regular users and administrative users. Permissions can grant access to read, write, or execute logic on an instance.

User credentials of an instance are hashed using bcrypt when stored on an instance for reference. This ensures that critical data is secured both in transit using TLS encryption and when stored on a device.

9. Insecure Software/Firmware

niolabs does not provide a means to remotely update binaries running on devices. As updates are made and released, the latest nio binaries can be obtained by a user through the nio portal binary download page. Documentation is provided for updating a nio binary using the latest file obtained from the nio portal. A full changelog of nio binary changes, including any potential breaking changes with new nio binaries can be found within the niolabs documentation at: https://docs.n.io/binaries/changelog.html.

niolabs recommends using third party tools in deployments to manage binaries running on devices in a production environment. For convenience, niolabs has reference Ansible playbooks and Chef recipes that can be used with configuration management tools for deployment. The nio binary can also be packaged in a Docker container with this reference repository for use in container orchestration deployments.

Updates to the logic running on nio instances can be managed using the nio Deployment Manager. This tool allows for service and block configurations to be changed and sent directly to running nio instances. For instances that are behind a protected network and cannot be reached from the web interface, asynchronous component deployments can be configured. Component deployments utilize a component that runs on a nio instance and periodically pulls the latest version of block and/or service configurations from the nio cloud or another configuration source.

10. Poor Physical Security

As a software company, the physical security of deployed nio instances cannot be controlled by niolabs. nio can run on many different types of devices. It is up to nio users to determine the physical security of the devices they run in the field. We provide deployment best practices that includes physical security as a consideration.

Google IoT Partner **Security Requirements**

Google publishes up-to-date security requirements for its IoT partners to follow. While many of the requirements are geared towards IoT devices, many of the principles apply to IoT software products as well. niolabs keeps current with these requirements and ensures that relevant items are accounted for in the nio software.

1. Best Secure Coding Practices Should be Followed

Best secure coding practices are always followed in the development of niolabs software. The OWASP top 10 IoT vulnerabilities are identified in the niolabs product security compliance with appropriate security work devoted to preventing them. Through the niolabs internal software review process, secure coding practices are prioritized before additions are released as part of the product offering.

2. Use of TLS for all Network Communications

All nio instance communications can be secured using TLS encryption. Instances primarily communicate in two ways. A REST interface is used to control the configuration of a nio instance. This interface is created by default in nio instances. Steps can be taken to secure this endpoint with TLS encryption. This requires a trusted SSL/TLS certificate used by the nio instance for a secure connection between it and the nio management application.

Instances can also use publish-subscribe communication to interact with other nodes within a system. This communication uses the niolabs secure communication broker, Pubkeeper. The protocols used by Pubkeeper are specific to each deployment but data is encrypted in transit using AES encryption. For deployments that use the Websocket Brew, TLS encryption can be configured to add an additional layer of security to network communications. This behavior is used by default in all nio managed cloud instances. Other Pubkeeper brews have similar encryption and security features that can be used.

3. Verified Firmware Updates

Updates to nio services and blocks that are running on existing instances can be performed using many different industry standard deployment tools. This allows configuration updates to occur over secure channels and also be signed if needed. Best practices for deploying service and block configuration updates can be found in the deployment best practices documentation.

4. Scalable Process for Firmware Updates

The nio Deployment Manager provides a scalable way to deploy service/block updates to multiple devices. When deploying a new version of an instance configuration, multiple instances within a system can be selected to receive updates. Indirect deployments can be configured for instances that cannot be accessed remotely by the nio Deployment Manager. The deployment API component allows for nio instances to poll for assigned instance configurations at a configurable interval.

5. Strong Authentication Mechanisms

nio instances use a combination of TLS encryption and password hashing to secure access credentials both in transit and when stored on a device. Instructions for setting up a local nio instance using a trusted SSL certificate can be found in the niolabs documentation: https://docs.n.io/cli/new.html#new-project-creation.

When creating new users for an instance using the nio command line interface, the password used to create a new user is salted and hashed using bcrypt before it is stored on the device. This process prevents insecure plain text password storage and passwords cannot be looked up using attack vectors like a rainbow table.

6. Unique MAC Addresses

The use of a device with a unique MAC address is considered a deployment specific requirement. Best practices for this are described in the deployment best practices. The nio Platform does not rely on MAC addresses as a unique identifier, spoofing them has no impact on the software itself.

7. No Communication With Third-Party Servers

By default, nio instances do not communicate with any third party servers. Blocks are provided that would allow instances to communicate with outside servers. The use of these blocks is fully in control of the one building nio services. Enforcing the secure use of such blocks is a deployment specific requirement. Best practices for this are described in the deployment best practices.

8. No Hardcoded Credentials

nio instances are created with default access credentials to enable quick access to development instances. These credentials are not coded in to the software at all, they are fully visible and configurable. The nio command line interface offers commands to change these default access credentials on a new nio instance. Additionally, multiple users can be added with different credentials and permissions set. Instructions on configuring credentials on a nio instance can be found in the niolabs documentation: https://docs.n.io/cli/users.html.

9. Unique & Replaceable Certificates

TLS encrypted communication for nio instances can be configured by specifying a private key and certificate file in the nio configuration settings. These files can be self-signed or signed by a trusted certificate authority. Ensuring that the private keys used on each instance are unique is a deployment specific requirement.

10. Commitment to Security Updates

niolabs is committed to providing security updates to all parts of our product offering in a timely manner. For more information on niolabs security updates, see our security response policy.

11. Minimum Service Exposure

By default nio opens one port for communication on a network interface. This port is secured by the nio security module for the authentication and authorization of all network requests. No undocumented functionality is exposed through this port. This port can be changed or disabled as needed through the nio configuration settings.

Depending on the communication protocols that are used, running a nio instance may open additional ports. For example, running the ZeroMQ Pubkeeper brew means opening ports for network communication to occur on. Any additional ports used by Pubkeeper or other components are always configurable via the nio configuration file.

12. WiFi Must Use WPA2

Enforcing the use of the WPA2 security protocol on devices is a deployment specific requirement. The nio Platform does not make any assumptions or contain any requirements for the type of network connectivity or the security used.

13. Bluetooth Security

Enforcing secure Bluetooth support on devices is a deployment specific requirement. The nio Platform does not make any assumptions or contain any requirements for the type of network connectivity or the security used.

14. Sync Clocks with NTP

The use of a Network Time Protocol server to sync device clocks is a deployment specific requirement. The nio Platform does not rely on clocks to be synchronized by default. Many third party tools do require this though so the use of NTP is recommended in the deployment best practices.

15. No External Network Connectivity

By default, nio does not provide any external network connectivity which would bypass network firewalls. Enforcing this security requirement on running instances of nio is a deployment specific requirement.

16. Use of Non-WiFi Wireless Interfaces

In some deployments of nio, devices are used for instance to instance communication over non-WiFi protocols such as Bluetooth and Xbee. This communication runs through an application layer and prevents maliciously crafted packets from entering a system. Additionally, Pubkeeper bridge mode could allow for traffic to be routed between non-WiFi and WiFi enabled interfaces. The use of this mode can be configured or disabled through Pubkeeper configuration settings. More information on Pubkeeper bridge mode can be found in the Pubkeeper documentation: https://docs.pubkeeper.com/protocol/protocol.html.

17. Identification and Delivery of Open Source Components

Open source components used by the nio Platform are identified on the niolabs legal page: https://niolabs.com/legal/third-party-software. Additionally, open source components used by nio binaries are delivered within the binary package that users download.

18. Graceful Degradation

nio was made to provide logic to devices where a network connection is not guaranteed. In the event of a network outage, nio does not revert to any insecure or unencrypted state. When attempting to perform a function requiring a network connection, retry settings can be configured to allow nio to retry until a network connection is available.

19. Test Resilience

nio regularly goes through independent security audits, both internally and through third party vendors to ensure resilience of the software. For more information on these audits please contact nio engineering: engineering@niolabs.com.