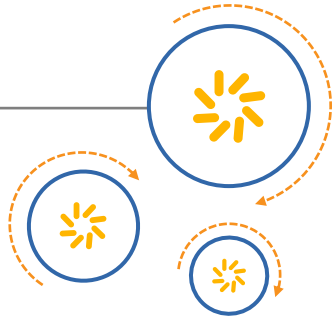
Qualcomm Technologies, Inc.



DIRBS System Security Guidelines

DIRBS-System-Security-Guidelines-2.0.0

January 07, 2021

Revision history

Revision	Date	Description
A	November 2018	Initial release
B	January 2021	Added section on Apache KAFKA security

Contents

1. Introduction **4**

1.1 Purpose & Scope 4

1.2 Definitions, Acronyms & Abbreviations 4

1.3 References 4

2. Security Guidelines **5**

Tables

Table 1 - Definitions, Acronyms & Abbreviations 4

1. Introduction

1.1 Purpose & Scope

This document provides security recommendations and guidelines for installing and deploying the DIRBS System.

As with any software system, the security of the DIRBS system depends on properly administering and maintaining the physical hardware, as well as the network and operating system upon which the DIRBS software is deployed on. Firewalls and network access will also need to be configured to minimize system access.

General guidelines that should be considered when installing and deploying the DIRBS System are defined in Section 2.

1.2 Definitions, Acronyms & Abbreviations

Table 1 - Definitions, Acronyms & Abbreviations

Term	Explanation
MNO	Mobile Network Operator
SFTP	Secure File Transfer Protocol
SSH	Secure Shell
SSL	Secure Sockets Layer
TLS	Transport Layer Security
Nginx	A web server which can be used as a reverse proxy, load balancer, mail proxy and HTTP cache.
XSS	Cross Site Scripting
DOS	Denial of Services

1.3 References

N.A

2. Security Guidelines

The following items are general guidelines that should be considered when installing and deploying the DIRBS System.

- Only machines providing external services should be reachable from the Internet and only required ports should be accessible
- Externally accessible DIRBS components should have minimal network access back into the internal network (should not be able to SSH into an internal host or access any internal host/port not required for essential operation)
    - MNOs SFTP Server:
        - All MNOs account created on SFTP server should have no shell associated (i.e., /bin/false as shell) to restrict SSH login
        - MNOs can only run (i.e., ls, put, get, cd etc) SFTP commands to upload data
        - MNOs account should be jailed to own home directory
        - MNOs should be restricted to only upload specific file format i.e., CSV, ZIP, TAR, GZ etc
        - MNOs can only upload file and no rights to remove, rename or change owner / rights of the uploaded file
        - Uploaded data should be moved to secure locations on an internal network to prevent data being downloaded if the operator upload server is compromised
        - Web Servers:
            - SSL / TLS certificate should be implemented on NGINX
            - Directory access should be disabled in NGINX
            - XSS / DOS prevention should be done in NGINX and web application
            - Only SSL port 443 should be exposed on web servers
            - Domain name should be used instead of Public IPs for web servers
- Operating system installation is hardened and secure

- All software installed on machines is kept up to date and patches are installed in the event of security vulnerabilities
- UNIX file permissions are set correctly to prevent unauthorized access of data
- Minimize root access on all hosts
- Minimize access to the docker group on all hosts
- Host-based access setup appropriately for PostgreSQL. This should be done at database granularity – even though subsystems connect directly to PostgreSQL, no externally accessible host should be allowed to connect to the DIRBS Core database and should instead proxy requests through the API Gateway, which in turn queries the DIRBS Core API host
    - Separate database user accounts used for each subsystem vs. the DIRBS Core
    - Default PostgreSQL user account should be disabled
    - PostgreSQL default encryption MD5 should be changed Scram-SHA-256
    - All subsystem connected to PostgreSQL database should have separate users
    - All subsystem database users should only have access to concerned database and restricted to login / view other database on server
    - All database user info table should be encrypted / hashed
- Strong encryption ciphers used for all external network data
- Host intrusion detection software installed on externally-accessible machines to detect unauthorized access
- DIRBS Core API should only be accessible inside LAN and specific host can have access CORE API service
- VPN access should be used to access backend operations

In addition to the guidelines listed above, the DIRBS system does utilize several application-level security features including:

- **Host-based access** (PostgreSQL Feature): Database access is restricted to a set of machines that are supposed to have access
    - No externally-accessible machine should have access to the DIRBS Core DB
    - All DIRBS Core software can be deployed inside a firewall and all external requests for data from DIRBS Core go through a single machine (API Gateway), which then speaks to the API machines (deployed

internally only)

- **Application isolation** (Docker Feature): All DIRBS SW runs inside of Docker
    - If the DIRBS Core software is compromised via the network, the attacker would be sandboxed to a container and not have automatic access to the entire physical host
- **DB credential prompting:** Support for password prompting when running commands that require a high level of access so that those credentials never need to be stored in a file at all and therefore potentially read by an attacker
- **Key-based authentication for MNOs in the upload/SFTP host image:** Private/public keypairs should be used to allow MNOs to connect to the upload servers, meaning that passwords are not used at all. Using keys rather than password means that security is not vulnerable to weak or guessable passwords
- **Privilege separation:** We have defined security roles for the application, where each role only has access to the data needed for that role. Tasks in DIRBS Core require a specific role to run. Write access to the DB is only granted where strictly necessary and on a per-table basis to further reduce the risk of malicious tampering
    - *Provided by PostgreSQL*: All passwords for user accounts required to connect to PostgreSQL are securely stored (hashed) in the database
    - Whilst stored securely inside PostgreSQL, clients must store credentials in plain text to allow for automation. That is why we don't want externally accessible hosts to connect to the PostgreSQL DB – so we don't need to put any credentials on them
    - Due to host-based access, they would need to do this from the internal network and from one of the allowed internal IP addresses
    - Due to privilege separation, stolen credentials would only have access to a subset of the data required for the compromised job / web service to function
    - Due to password-prompting feature, high-powered credentials should never be stored on disk to reduce this risk
    - Subsystems do connect directly to the PostgreSQL from externally accessible machines, so care must be taken to ensure that credentials used for the subsystems are not shared with DIRBS Core
    - *Provided by PostgreSQL*: All network traffic transmitted between the database and DIRBS Core software is encrypted. We have recommended a configuration where any attempt to connect over a non-encrypted connection will be rejected
    - *Provided by nginx/Web server*: All web traffic transmitted outside the

firewall will also be encrypted to prevent snooping of traffic.

- **Securing Apache KAFKA deployment:** Here are some general security points which must be considered while deploying KAFKA clusters in a production environment. For details, please refer to Apache KAFKA security guideline

    - Connections from clients (producers & consumers) to brokers should be properly authenticated using either SSL (Secure Sockets Layer) or SASL (Simple Authentication and Security Layer) protocols.

    - Connections from brokers to Zookeeper server should be properly authenticated.

    - The data transfer between brokers and clients, between brokers or between brokers or tools should be encrypted using SSL.

    - Only authorized operations (read or write) by clients should be allowed on brokers.