ARL-SR-0523 ● SEP 2026

DEVCOM ARMY RESEARCH LABORATORY

Deployment and Configuration of an Android Team Awareness Kit (ATAK) Server Using Docker

by Justin Shumaker, Dan Nikolov, and Derrik Asher

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-SR-0523 ● SEP 2026

Deployment and Configuration of an Android Team Awareness Kit (ATAK) Server Using Docker

Justin Shumaker, Dan Nikolov, and Derrik Asher
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
September 2026	Special Report	6/1/2025	1/7/2026

4. TITLE AND SUBTITLE
Deployment and Configuration of an Android Team Awareness Kit (ATAK) Server Using Docker

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Justin Shumaker, Dan Nikolov, and Derrik Asher

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
DEVCOM Army Research Laboratory ATTN: FCDD-RLA-IH Aberdeen Proving Ground, MD 21005	ARL-SR-0523

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report details the procedures required to successfully deploy a functional Tactical Assault Kit (TAK) server (version 5.6) within a Docker containerized environment. It provides a step-by-step guide derived from a proven automation script covering system prerequisites, Docker network and image creation, software installation, initial configuration, and certificate generation for secure communications. The objective is to provide a clear and repeatable process for ARL personnel and partners to establish their own TAK server instance for tactical communication and situational awareness.

15. SUBJECT TERMS
Military Information Sciences, Tactical Assault Kit, TAK, Docker, server, container, Android Team Awareness Kit, ATAK

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	26

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Justin Shumaker	(520) 691-5669

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Contents

List of Tables

1. Introduction and Purpose

The Android Team Awareness Kit (ATAK), also known as the Tactical Assault Kit (TAK), is a geospatial situational awareness (SA) application designed for mobile platforms. Developed initially for the Android operating system, it provides Warfighters with real-time, map-based information, allowing for collaborative mapping, data sharing, and enhanced communication in a tactical environment. Users can see the precise location of other team members (Blue Force Tracking), share points of interest, exchange chat messages, and collaborate on mission planning through a shared, interactive map.

The TAK server is the critical backend component that transforms individual ATAK clients into a fully connected, collaborative network. It acts as the central hub or "heart" of the TAK ecosystem, receiving data from all connected clients and intelligently routing it to the appropriate recipients. This enables communication and data sharing across disparate networks and beyond-line-of-sight (BLOS) scenarios, which would be impossible with a peer-to-peer connection alone. The server manages user authentication, persists mission data, and provides the authoritative source for SA data across an entire operation.

The client-server relationship is fundamental to the system's architecture. An ATAK client (running on an Android device) or a WinTAK client (on a Windows machine) establishes a secure, certificate-based connection to the TAK server. Once connected, the client sends its Position Location Information (PLI), chat messages, map markers, and other data to the server. The server then distributes this information to all other authorized clients connected to the same mission or channel. This ensures every member of the team shares a common operational picture, greatly enhancing command and control (C2) and team coordination.

This report serves as a standardized guide for deploying the TAK server software using Docker. Docker is an open-source platform that packages an application and all its dependencies into a single isolated unit called a container, ensuring it runs identically on any machine regardless of the underlying operating system. It is used to eliminate the "it works on my machine" problem, streamline deployment pipelines, and optimize server resource utilization. For servers requiring complex multistep configurations, traditional manual setups or distribution via massive virtual machine snapshots are highly inefficient, error-prone, and platform-dependent. Docker solves this by replacing manual steps with a lightweight text file (a Dockerfile) that acts as an automated, reproducible blueprint. Instead of downloading and installing every dependency by hand, community members can pull a pre-configured Docker image from a registry and deploy the exact same

environment instantly with a single command. Additionally, Docker introduces a layered file system that caches each step of the build process, meaning that if a developer updates only the final configuration step, they only need to re-distribute that specific minor layer rather than redistributing a massive multi-gigabyte operating system image.

The use of containerization with Docker dramatically simplifies the complex setup process, isolates the server environment, and ensures a consistent, reproducible deployment across different host systems. The following instructions are based on the provided `README_TAKSERVER.txt` (reproduced in full in Appendix A) and intended to reduce configuration errors, establish a reliable baseline for future deployments, and provide U.S. Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory (ARL) personnel and partners with a clear, repeatable process for establishing their own TAK server instance. Additionally, a quick-reference guide of useful Docker operational and troubleshooting commands is provided in Appendix B.

2. System Prerequisites

Before beginning the installation, ensure the host system meets the requirements shown in Table 1.

Table 1. System requirements.

Component	Requirement
Operating system	A Linux distribution with a current kernel (e.g., Ubuntu 22.04 LTS).
Hardware (minimum)	Sufficient CPU, RAM, and disk space to run Docker and the server containers. Recommend at least 4 CPU cores, 8 GB RAM, and 50 GB disk space.
Software dependencies	Docker Engine, OpenSSL. Optional: `docker-buildx` for alternative build methods.
Networking	Administrative access to configure network settings and firewall rules if necessary.

3. Installation and Setup Procedure

This section translates the setup script into a series of logical steps with explanations.

3.1 Create Docker Network, Image, and Database Volume Image

A dedicated Docker network bridge and named volume are required to manage container communication and persist database-related data.

```bash
# Create a Docker network bridge named 'takserver'
docker network create takserver

# Create a Docker volume to persist the database
docker volume create takserver-db
```

3.2 Initial Configuration: Set Admin Password

Generate a secure random password and insert it into the example configuration file. This password will be used for the server's keystore.

```bash
# Generate a 12-character random password
openssl rand -base64 12 | tr -dc 'a-zA-Z0-9'

# IMPORTANT: Replace the example password in the XML config file with the one generated above.
# The command below uses 'pvtpVEf0zNpyOHCE' as an example. Use your generated password.
sed -i 's/password=""/password="pvtpVEf0zNpyOHCE"/g' tak/CoreConfig.example.xml
```

3.3 Build and Run TAK Server Database Container

Build the Docker image for the PostgreSQL database and then run it as a container on the `takserver` network.

If you have a local instance of PostgreSQL running, it will conflict with the port used by the container (5432). Stop the local service (`sudo systemctl stop postgres`) before proceeding.

```bash
# Build the takserver-db Docker image
docker build takserver-db -f docker/Dockerfile.takserver-db .

# Create and run the takserver-db container
docker run --mount source=takserver-db,destination=/var/lib/postgresql -v ${PWD}/tak:/opt/tak:z -it -p 5432:5432 --network takserver --network-alias tak-database --name takserver-db -d takserver-db
```

3.4 Build and Run TAK Server Container

Next, build the main TAK server image and run it as a container. This command exposes the necessary ports for both encrypted and unencrypted communication.

```bash
# Build the main takserver image
docker build -t takserver -f docker/Dockerfile.takserver .

# Run the takserver container, mapping all required ports
docker run -v $(pwd)/tak:/opt/tak:z -it -p 8080:8080 -p 8087:8087 -p 8088:8088 -p 8089:8089 -p 8443:8443 -p 8446:8446 -p 9001:9001 --network takserver --network-alias takserver --name takserver -d takserver
```

4. Configuration: Certificate Generation

For secure communication over Transport Layer Security, a set of certificates must be generated.

4.1 Create Certificates for Encrypted Communications

First, edit the metadata file with your organization's information. Then, execute the scripts within the running `takserver` container to generate the Root Certificate Authority (CA), Intermediate CA, and server certificates.

```bash
# 1. Edit the metadata for your certificates
nano tak/certs/cert-metadata.sh
# Enter values for: STATE, CITY, ORGANIZATION, ORGANIZATIONAL_UNIT

# 2. Make Root Certificate Authority
docker exec -it takserver bash -c "cd /opt/tak/certs && ./makeRootCa.sh --ca-name TAK-ROOT-CA-01"

# 3. Make Intermediate Certificate
docker exec -it takserver bash -c "cd /opt/tak/certs && echo y | ./makeCert.sh ca TAK-ID-CA-01"

# 4. Make Server Certificate
docker exec -it takserver bash -c "cd /opt/tak/certs && ./makeCert.sh server takserver"
```

4.2 Update Core Configuration

Update the configuration file to use the newly created intermediate certificate as the truststore authority.

```bash
sed -i 's/truststore-root/truststore-TAK-ID-CA-01/g' tak/CoreConfig.example.xml
```

5. Verification

To verify the server is running correctly, restart the container and monitor the log file. A successful start-up will show messages indicating the server and its microservices have started.

```bash
# Restart the container and follow the messaging log
docker restart takserver && tail -f tak/logs/takserver-messaging.log

# Expected output should include lines like:
#   INFO   tak.server.ServerConfiguration   -   Started ServerConfiguration...
#  INFO  tak.server.ServerConfiguration  -  Started  TAK Server messaging Microservice.
```

6. Post-Installation: Client and Admin Setup

6.1 Create WebAdmin User

Create a certificate for the `webadmin` user and set a password for web-based administration.

```bash
# 1. Create the webadmin client certificate
docker exec -it takserver bash -c "cd /opt/tak/certs && ./makeCert.sh client webadmin"

# 2. Add the certificate to the UserManager
docker exec -it takserver bash -c "java -jar /opt/tak/utils/UserManager.jar certmod -A /opt/tak/certs/files/webadmin.pem"

# 3. Set a password for the webadmin user (replace 'ARLR0b0tAdmin!!' with a strong password)
docker exec -it takserver bash -c "java -jar /opt/tak/utils/UserManager.jar usermod -A -p 'ARLR0b0tAdmin!!' webadmin"
```

6.2 Create Client Certificate (Example: WinTAK)

Generate a certificate for a client device.

```bash
# 1. Create the client certificate (e.g., for 'wintak01')
docker exec -it takserver bash -c "cd /opt/tak/certs && ./makeCert.sh client wintak01"

# 2. Add the new client to the UserManager
docker exec -it takserver bash -c "java -jar /opt/tak/utils/UserManager.jar certmod /opt/tak/certs/files/wintak01.pem"
```

6.3 Client Connection

The generated `.p12` certificate files (`caCert.p12`, `wintak01.p12`, `webadmin.p12`) can be copied from the `tak/certs/files/` directory and installed on client devices (ATAK, WinTAK) or in a web browser to connect to the server on the secure ports (e.g., 8443).

7. Summary

This report provides a comprehensive, step-by-step guide for the successful deployment and configuration of a TAK server within a Docker containerized environment. By translating a proven automation script into a clear and repeatable process, this document standardizes the setup of a critical component in the TAK ecosystem. The use of Docker simplifies dependency management, isolates the server environment, and ensures consistent, reproducible deployment across different host systems. The procedures detailed herein, from initial system setup and certificate generation to client configuration, are intended to reduce configuration errors and empower ARL personnel and their partners to rapidly establish a functional TAK server, thereby enhancing tactical communication and shared SA.

8. Future Directions: Server Federation for Robotic Systems

The successful deployment of a single TAK server is the foundational step toward a robust C2 network. The next logical and critical evolution of this capability, especially for Army applications involving teams of robotic systems, is server federation. While a single server is effective for a localized team, the modern battlefield requires a distributed, resilient, and scalable architecture that a federated model provides.

Future work should focus on the following areas:

- Developing a federated architecture: Research should be directed toward creating a "server-of-servers" model. This involves configuring multiple independent TAK servers to communicate with one another, sharing essential data like PLI, chat messages, and sensor data across geographically dispersed units. This is paramount for robotic systems, where a single swarm or team may operate under a local TAK server, which then federates its data to a higher-echelon command server, creating a multi-layered common operational picture.
- Automation and scalability: Building on the containerization work in this report, future efforts should develop automated scripts (e.g., using Docker Compose or Kubernetes) to rapidly deploy and connect multiple TAK servers in a federated topology. The goal is to enable the dynamic creation of networks that can scale to support hundreds or thousands of clients, including autonomous ground and air systems that generate high-frequency data.
- Resilience and redundancy: A federated network is inherently more resilient than a centralized one. If a single server node fails or is disconnected, teams connected to other servers in the federation can maintain communication. This is crucial for robotic teams operating in contested environments where network connectivity may be intermittent. Research should investigate self-healing and auto-discovery mechanisms for servers within a tactical federation.
- Bandwidth optimization for robotic data: Robotic platforms can generate immense volumes of data (e.g., video feeds, light detection and ranging scans). A federated architecture allows for intelligent data routing and filtering. Future work should explore methods to prioritize critical C2 data and manage the flow of high-bandwidth sensor information between servers, ensuring tactical networks are not overwhelmed while still providing necessary information to commanders.

By pursuing server federation, the Army can move from isolated pockets of SA to a truly interconnected, resilient, scalable C2 network capable of supporting complex, multi-domain operations with large teams of robotic and autonomous systems.

Appendix A. Full Original Installation Script "README_TAKSERVER.txt"

Making Docker Image for TAK Server 5.6
Justin Shumaker <justin.l.shumaker.civ@army.mil>
Mar 05, 2025

First create a docker network bridge
The bridge is responsible for allowing network traffic into the running docker container
orin@orin:~/takserver-docker-5.6-RELEASE-24$ docker network create takserver

orin@orin:~/takserver-docker-5.6-RELEASE-24$ docker volume create takserver-db

orin@orin:~/takserver-docker-5.6-RELEASE-24$ openssl rand -base64 12 | tr -dc 'a-zA-Z0-9'

orin@orin:~/takserver-docker-5.6-RELEASE-24$ grep password tak/CoreConfig.example.xml

BE SURE TO UPDATE THE PASSWORD TO THE ONE GENERATED WITH THE ONE FROM THE ABOVE COMMAND
orin@orin:~/takserver-docker-5.6-RELEASE-24$ sed -i 's/password=""/password="pvtpVEf0zNpyOHCE"/g' tak/CoreConfig.example.xml

Make the docker image for the takserver database
orin@orin:~/takserver-docker-5.6-RELEASE-24$ docker build takserver-db -f docker/Dockerfile.takserver-db .
Alternatively if using buildx or required to use buildx then:
orin@orin:~/src/takserver-docker-5.6-RELEASE-24$ docker buildx build -f docker/Dockerfile.takserver-db -t takserver-db .

Downloading for several packages begins (postgresql ...)
Installs docker image in /var/lib/docker

If you need to start the entire installation process all over again:
docker images (to list images)
docker rmi image_name (to delete image(s) and try again)
Remove any docker network, containers, images, volume, and generated certificates (depending on how far you made it through the process)

Create the takserver database container
orin@orin:~/takserver-docker-5.6-RELEASE-24$ docker run --mount source=takserver-db,destination=/var/lib/postgresql -v ${PWD}/tak:/opt/tak:z -it -p 5432:5432 --network takserver --network-alias tak-database --name takserver-db -d takserver-db

List the newly created container
orin@orin:~/takserver-docker-5.6-RELEASE-24$ docker container ps -a
Note that "docker container ls" only lists running containers
Stopped docker containers also persist and consume some disk space.
Containers can be deleted or pruned, but avoid doing so if you plan to use the containers long term.

# If you have postgres database running (ps aux | grep postgres) then you may want to disable postgres since the local
# postgres server port will conflict with the postgres server port running in the takserver-db image.
# Alternatively one could try mapping the ports differently so there isn't a conflict.
sudo systemctl stop postgres # if you just want to stop postgres to free up the port one time only.

Build the image

```
orin@orin:~/takserver-docker-5.6-RELEASE-24$ docker build -t takserver -f docker/Dockerfile.takserver .
# More Downloading may take place while building the image

# Run the docker image (creates and runs the container)
# If planning to also use unencrypted communications add the "-p 8080:8080 -p 8087:8087 -p 8088:8088" ports as well (a total of 6 ports)
orin@orin:~/takserver-docker-5.6-RELEASE-24$ docker run -v $(pwd)/tak:/opt/tak:z -it -p 8080:8080 -p 8087:8087 -p 8088:8088 -p 8089:8089 -p 8443:8443 -p 8446:8446 -p 9001:9001 --network takserver --network-alias takserver --name takserver -d takserver

# The following steps are for creating certificates to communicate on the encrypted ports
orin@orin:~/takserver-docker-5.6-RELEASE-24$ nano tak/certs/cert-metadata.sh
# Enter values for: STATE,CITY,ORGANIZATION,ORGANIZATIONAL_UNIT

# Make Root Certificate Authority and adds to keystore
orin@orin:~/takserver-docker-5.6-RELEASE-24$ docker exec -it takserver bash -c "cd /opt/tak/certs && ./makeRootCa.sh --ca-name TAK-ROOT-CA-01"

# Make Intermediate Certificate and adds to keystore
orin@orin:~/takserver-docker-5.6-RELEASE-24$ docker exec -it takserver bash -c "cd /opt/tak/certs && echo y | ./makeCert.sh ca TAK-ID-CA-01"

# Make Server Certificate and adds to keystore
orin@orin:~/takserver-docker-5.6-RELEASE-24$ docker exec -it takserver bash -c "cd /opt/tak/certs && ./makeCert.sh server takserver"

orin@orin:~/takserver-docker-5.6-RELEASE-24$ grep truststore-root tak/CoreConfig.xml
# Displays the TLS v1.2 trust store
#       <tls keystore="JKS" keystoreFile="/opt/tak/certs/files/takserver.jks" keystorePass="atakatak" truststore="JKS" truststoreFile="/opt/tak/certs/files/truststore-root.jks" truststorePass="atakatak" context="TLSv1.2" keymanager="SunX509"/>
# Replace truststore-root with TAK-ID-CA-01 (making it the new authority for all future cert requests)

orin@orin:~/takserver-docker-5.6-RELEASE-24$ sed -i 's/truststore-root/truststore-TAK-ID-CA-01/g' tak/CoreConfig.example.xml

orin@orin:~/takserver-docker-5.6-RELEASE-24$ grep truststore- tak/CoreConfig.example.xml
#                   truststore="JKS" truststoreFile="/opt/tak/certs/files/truststore-TAK-ID-CA-01.jks" truststorePass="atakatak">

#
# Now it's time to see if it all worked
#
orin@orin:~/takserver-docker-5.6-RELEASE-24$ docker restart takserver && tail -f tak/logs/takserver-messaging.log
takserver
        at tak.server.ServerConfiguration$$SpringCGLIB$$FastClass$$1.invoke(<generated>)
        at org.springframework.cglib.proxy.MethodProxy.invokeSuper(MethodProxy.java:258)
        at org.springframework.context.annotation.ConfigurationClassEnhancer$BeanMethodInterceptor.intercept(ConfigurationClassEnhancer.java:330)
        at tak.server.ServerConfiguration$$SpringCGLIB$$0.jwkSource(<generated>)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77)
at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.base/java.lang.reflect.Method.invoke(Method.java:569)
at org.springframework.beans.factory.support.SimpleInstantiationStrategy.instantiate(SimpleInstantiationStrategy.java:139)
... 98 common frames omitted
2025-03-05-18:21:45.626 [main] INFO tak.server.ServerConfiguration - Starting ServerConfiguration v5.6-RELEASE-24 using Java 17.0.14 with PID 10 (/opt/tak/takserver.war started by root in /opt/tak)
2025-03-05-18:21:45.628 [main] INFO tak.server.ServerConfiguration - The following 2 profiles are active: "plugins", "messaging"
2025-03-05-18:21:53.758 [main] INFO tak.server.util.DataSourceUtils - 8 CPU cores detected. Postgres server maximum allowed connections value is 2100. The computed connection pool size is: 236
2025-03-05-18:21:54.866 [main] WARN org.hibernate.orm.deprecation - HHH90000025: PostgreSQLDialect does not need to be specified explicitly using 'hibernate.dialect' (remove the property setting and it will be selected by default)
2025-03-05-18:22:03.651 [services-deployment-worker-#61] WARN com.bbn.marti.service.SSLConfig - TLS enabled, but no certificate revocation lists, and OSCP is not enabled in Core Config!
2025-03-05-18:22:06.288 [main] ERROR tak.server.util.JAXBUtils - UserAuthenticationFile.xml not found.
2025-03-05-18:22:12.842 [main] INFO c.b.m.sync.JDBCEnterpriseSyncService - file cache explicity disabled.
2025-03-05-18:22:12.848 [main] INFO c.b.marti.service.SubmissionService - input class type: Input
2025-03-05-18:22:12.849 [main] INFO c.b.marti.service.SubmissionService - Configuring network input com.bbn.marti.config.Input stdssl:
2025-03-05-18:22:13.284 [main] WARN com.bbn.marti.service.SSLConfig - TLS enabled, but no certificate revocation lists, and OSCP is not enabled in Core Config!
2025-03-05-18:22:13.809 [Thread-5] INFO c.b.marti.nio.netty.NioNettyBuilder - Successfully Started Netty Server for TlsServerInitializer on Port 8089 with watermark WriteBufferWaterMark(low: 16384, high: 32768)
2025-03-05-18:22:14.760 [main] INFO t.server.federation.FederationServer - Federation disabled in config. Not starting V2 federation server.
2025-03-05-18:22:14.770 [main] INFO com.bbn.marti.nio.server.NioServer - max memory (bytes): 10712252416
2025-03-05-18:22:14.771 [main] INFO com.bbn.marti.nio.server.NioServer - Server started
2025-03-05-18:22:14.875 [main] INFO t.server.cache.DatafeedCacheHelper - Done initializing PluginDatafeedCacheHelper with buffer cache value: 300 seconds
2025-03-05-18:22:14.897 [main] INFO com.bbn.marti.util.VersionBean - TAK Server version 5.6-RELEASE-24-HEAD
2025-03-05-18:22:14.898 [main] INFO c.b.m.sync.EnterpriseSyncCacheHelper - file cache explicity disabled.
2025-03-05-18:22:14.898 [main] INFO c.b.m.sync.EnterpriseSyncCacheHelper - Initialzed EnterpriseSyncCacheHelper with file cache TTL 120 seconds. Implementation: caffeine
2025-03-05-18:22:14.900 [main] INFO tak.server.ServerConfiguration - Started ServerConfiguration in 32.532 seconds (process running for 48.647)
2025-03-05-18:22:14.927 [main] INFO tak.server.ServerConfiguration - Started TAK Server messaging Microservice.

At this point you can create the certificates (Ctrl-C) server will still be running.

#
Create Webadmin Certificate and Password
#

Create the webadmin key
orin@orin:~/takserver-docker-5.6-RELEASE-24$ docker exec -it takserver bash -c "cd /opt/tak/certs && ./makeCert.sh client webadmin"

Place the key in the docker image
orin@orin:~/takserver-docker-5.6-RELEASE-24$ docker exec -it takserver bash -c "java -jar /opt/tak/utils/UserManager.jar certmod -A /opt/tak/certs/files/webadmin.pem"
New User Added:
	Username:	'webadmin'
	Role:	ROLE_ADMIN
	Fingerprint:
0A:F9:52:FA:40:02:8A:CA:30:0A:33:4C:8C:43:A2:F2:AF:FF:15:22:80:DD:99:6F:60:F7:CC:E2:C0:04:9A:26
	Groups (read and write permission):
		__ANON__

orin@orin:~/takserver-docker-5.6-RELEASE-24$ docker exec -it takserver bash
root@44e2eac4c852:/#
root@44e2eac4c852:/# java -jar /opt/tak/utils/UserManager.jar usermod -A -p 'ARLR0b0tAdmin!!' webadmin
User Updated:
	Username:	'webadmin'
	Role:	ROLE_ADMIN
	Fingerprint:
0A:F9:52:FA:40:02:8A:CA:30:0A:33:4C:8C:43:A2:F2:AF:FF:15:22:80:DD:99:6F:60:F7:CC:E2:C0:04:9A:26
	Groups (read and write permission):
		__ANON__
root@44e2eac4c852:/# exit

Move Some PEM files around (The root CA public certificate)
orin@orin:~/takserver-docker-5.6-RELEASE-24$ sudo cp tak/certs/files/truststore-TAK-ID-CA-01.p12 caCert.p12
[sudo] password for orin:

Make a wintak client certificate within the docker image
orin@orin:~/takserver-docker-5.6-RELEASE-24$ docker exec -it takserver bash -c "cd /opt/tak/certs && ./makeCert.sh client wintak01"
Using legacy provider
client
wintak01
Making a client cert for /C=US/ST=MD/L=APG/O=DEVCOM ARL/OU=SISD/CN=wintak01
.+.....+.....+.....+.....+.+.....+.++
++++++++++++++++++++*...+.....+.....+...+.+..+.....+..+....+...+.....+.....+.....+...+.+...+
..+...+...+.....+....+..++

```
+++++++++++++++*...+.+...+.........+...+........+....+..+......+..........+.....+....+...........+...+..........+.....
....+..............+...+...+...+.+...........+....+.....+....+..........+......+.............+..+...+.......+...+.....+....+...+...
+......+.....+.+.............+......+.......+..+.........+.........+.+..+....+...........+...+.......+.....+.......+...+..+.+.....
+.+.....+..........+...+................+...........+....+..+...+...+....+.....+...+.+..............+................+......+.+.....+..
.+...++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.+......+....+..++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++*.++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+*..............+........+.......+.....+...+.......+...+.....+....+...............+.......+.....+....+.....+.........+....+..........
+.......+.....+....+........+...+.......+...+.....+............+.+.....+......+.+.....+....+...............+......+.....+++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
-----
Certificate request self-signature ok
subject=C = US, ST = MD, L = APG, O = DEVCOM ARL, OU = SISD, CN = wintak01
Importing keystore wintak01.p12 to wintak01.jks...
```

Add the wintak key to the user manager within the docker image
orin@orin:~/takserver-docker-5.6-RELEASE-24$ docker exec -it takserver bash -c "java -jar /opt/tak/utils/UserManager.jar certmod /opt/tak/certs/files/wintak01.pem"
New User Added:
Username: 'wintak01'
Role: ROLE_ANONYMOUS
Fingerprint:
16:F3:8C:1F:D5:D7:07:A0:F7:72:4D:C8:56:95:4E:66:BA:E7:87:E0:75:7F:24:3E:24:21:DF:38:58:A5:54:74
Groups (read and write permission):
__ANON__

The wintak public key
orin@orin:~/takserver-docker-5.6-RELEASE-24$ sudo cp tak/certs/files/wintak01.p12 .

The webadmin public key
orin@orin:~/takserver-docker-5.6-RELEASE-24$ sudo cp tak/certs/files/webadmin.p12 .

#
These certificates can be used/installed on clients as needed
#

#
Now go into wintak or atak, remove the old server connection, connect to this docker tak server and install the caCert.p12 (password: atakatak) the server is running on.

WebTAK
In the web browser you will need 2 certificates to make this work
first install the Authority certificate ca.pem as a trusted authority
next install the webadmin.p12 as a certificate (enter "atakatak" for password when prompted)
In the web browser visiting localhost:8443 ought to reveal that takserver is running with encryption

#
Additional Useful Commands
#

#

```
# Viewing existing ports:
#
# List Existing ports for an existing container on a network:
docker ps --format "table {{.ID}}\t{{.Names}}\t{{.Ports}}"
# This will display a list of running containers and their port mappings.
# Another useful command to show the Exposed ports:
docker inspect takserver | grep -A 5 "Ports"
# Another useful command to show the ports in a different format:
docker inspect takserver | jq '.[].NetworkSettings.Ports'

#
# Modifying existing ports (remove container then create new one):
#

# First stop the affected container from running with the "docker container stop <container>"
# Be sure to stop the takserver-db as well
# Next remove the affected container "docker rm <container_id>"

docker network prune # may be needed
docker ps -aq # and remove the existing containers
docker network create takserver # to recreate the network, make sure the existing network named takserver has already been deleted

# Now perform the original "docker run ..." command
# The example below exposes two additional unsecure ports [8080,8087]
# Remember to restart the takserver-db container before creating and starting the takserver container using the following docker run command.
docker run -v $(pwd)/tak:/opt/tak:z -it -p 8080:8080 -p 8087:8087 -p 8089:8089 -p 8443:8443 -p 8446:8446 -p 9001:9001 --network takserver --network-alias takserver --name takserver -d takserver
# The new docker container should reflect the exposed ports now
```

Appendix B. Useful Docker Commands

```bash
# List running containers and their port mappings
docker ps --format "table {{.ID}}\t{{.Names}}\t{{.Ports}}"

# Inspect a container's network settings
docker inspect takserver | grep -A 5 "Ports"

# Stop a running container
docker container stop <container_name_or_id>

# Remove a stopped container
docker rm <container_name_or_id>

# List all Docker images
docker images

# Remove a Docker image
docker rmi <image_name>
```

List of Symbols, Abbreviations, and Acronyms

ARL	Army Research Laboratory
ATAK	Android Team Awareness Kit
BLOS	Beyond-Line-Of-Sight
C2	Command and Control
CA	Certificate Authority
CPU	Central Processing Unit
DEVCOM	U.S. Army Combat Capabilities Development Command
PLI	Position Location Information
RAM	Random Access Memory
SA	Situational Awareness
TAK	Tactical Assault Kit

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLB CB TECH LIB