missing-user-entrypoint

Open

Analyzed | Ignore... | Fix

BackendApi/Dockerfile:27

Assistant thinks this is a true positive

Container entrypoint runs as root user with no USER directive, allowing any compromised process full container privileges.

Agree or Disagree

RULE DETAILS

- High severity
- Medium confidence
- Monitor
- CWE-269
- missing-user-entrypoint

FINDING DETAILS

- 3 days ago
- is241307/simsfh_ws25_SAST
  managed-scan
- main
- 5cef085
  by puywei

Finding description | Rule description

The `ENTRYPOINT` instruction runs without a preceding `USER` directive, meaning the dotnet process will execute as the root user inside the container.

How this could be exploited:

1. An attacker gains code execution inside the container (e.g., via a vulnerability in the ASP.NET application or a dependency)
2. Since the process runs as root, the attacker has root privileges within the container
3. The attacker can then access sensitive files, modify application behavior, escape to the host system, or pivot to other containers on the same host

For example, if a web endpoint in `BackendApi.dll` has a path traversal vulnerability, an attacker could read `/etc/shadow` or other sensitive files that would normally be restricted to root. With root access already granted by the Dockerfile configuration, this becomes trivial to exploit.
The `ENTRYPOINT ["dotnet", "BackendApi.dll"]` instruction doesn't specify which user should run the application, so Docker defaults to root—giving any code executed within the container full system privileges.

Your code | Example code

BackendApi/Dockerfile:27 build process

```
8  RUN dotnet restore
9
10 # Kopiere den gesamten Quellcode
11 COPY . ./
12
13 # Build
14 RUN dotnet build -c $BUILD_CONFIGURATION -o /app/build
15
16 # Stage 3: Publish
17 FROM build AS publish
18 ARG BUILD_CONFIGURATION=Release
19 RUN dotnet publish -c $BUILD_CONFIGURATION -o /app/publish /p:UseAppHost=false
20
21 # Stage 4: Final image
22 FROM mcr.microsoft.com/dotnet/aspnet:9.0 AS final
23 WORKDIR /app
24 EXPOSE 8080
25 EXPOSE 8081
26 COPY --from=publish /app/publish ./
27 ENTRYPOINT ["dotnet", "BackendApi.dll"]
28
```

Assistant suggested fix | Rule fix

```
# Stage 2: Build image
FROM mcr.microsoft.com/dotnet/sdk:9.0 AS build
ARG BUILD_CONFIGURATION=Release
WORKDIR /src

# Kopiere nur die Projektdatei für restore
COPY *.csproj ./
RUN dotnet restore

# Kopiere den gesamten Quellcode
COPY . ./

# Build
RUN dotnet build -c $BUILD_CONFIGURATION -o /app/build

# Stage 3: Publish
FROM build AS publish
ARG BUILD_CONFIGURATION=Release
RUN dotnet publish -c $BUILD_CONFIGURATION -o /app/
publish /p:UseAppHost=false

# Stage 4: Final image
FROM mcr.microsoft.com/dotnet/aspnet:9.0 AS final
WORKDIR /app
EXPOSE 8080
EXPOSE 8081
COPY --from=publish /app/publish ./

# Add a non-root user and switch to it
RUN useradd -m appuser
RUN chown -R appuser /app
```

1. Add a non-root user creation step before the ENTRYPOINT instruction in your Dockerfile, for example: `RUN useradd -m appuser`.
2. Change the current user to the newly created non-root user by adding `USER appuser` after copying files and before the ENTRYPOINT instruction.
3. Ensure any directory that your application writes to is owned or writable by `appuser` (you can run `RUN chown -R appuser /app` if needed).

For example:

```
RUN useradd -m appuser
...
COPY --from=publish /app/publish ./
RUN chown -R appuser /app
USER appuser
ENTRYPOINT ["dotnet", "BackendApi.dll"]
```

Running your application as a non-root user reduces the impact of a potential compromise.