BaroPAM 가이드(Mattermost)

목차

1. Mattermost

1.1 개 요

Mattermost는 파일 공유, 검색, 통합 기능을 제공하는 오픈 소스로 셀프 호스팅이 가능한 온라인 채팅 서비스이다. 단체와 기업을 위한 내부 채팅으로 설계되어 있으며 대부분 그 자체를 슬랙과 마이크로소프트 팀즈의 오픈 소스 대안이다.

고정되어 있는 정적인 비밀번호를 동적인 BaroPAM 솔루션의 일회용 인증키로 대체했을 때 이점은 다음과 같다.
-비밀번호 단방향 암호화 불필요.
-비밀번호 관리지침 적용 불필요.
-비밀번호를 기억할 필요 없음.
-사용자 정보 유출되어도 로그인 불가능.
-비밀번호 도용 및 불법접속 불가능.
-브라우저 자동 로그인 불가능
-중간자 공격에도 안전.
-비밀번호 분실 및 도용 등에 따른 초기화 불필요.
-일회성 또는 휘발성 같은 동적보안 지원.

1.2 Matermost 구성 환경

1) Mattermost 서버
IP: 192.168.56.1
OS: Ubuntu 22.04.3 LTS x86_64
Mattermost: Version 9.5.2

2) DB 서버
IP: 192.168.56.2
OS: Ubuntu 22.04.3 LTS x86_64
DB: Postgresql 14.11 (Ubuntu 14.11-0ubuntu0.22.04.1)
Java: openjdk version 11.0.22
WAS: apache-tomcat-9.0.85
BaroPAM: Version 1.0

2. Postgresql 설정

2.1 환경설정 파일 변경

1) 데이터베이스와 Mattermost 앱 서버에 다른 서버를 사용하는 경우 PostgreSQL이 할당된 모든 IP 주소를 수신하도록 허용

```
root@mattermostdb:~# vi /etc/postgresql/14/main/postgresql.conf
.....
# - Connection Settings -

#listen_addresses = '192.168.56.2'                  # what IP address(es) to listen on;
listen_addresses = '*'                          # what IP address(es) to listen on;
                                                # comma-separated list of addresses;
                                                # defaults to 'localhost'; use '*' for all
                                                # (change requires restart)
port = 5432                                     # (change requires restart)
.....
```

2) Mattermost 서버가 데이터베이스와 통신할 수 있도록 pg_hba.conf 파일을 수정

```
root@mattermostdb:~# vi /etc/postgresql/14/main/pg_hba.conf
.....
# Database administrative login by Unix domain socket
local   all             postgres                                peer

# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
#local   all             all                                     peer
local   all             all                                     trust
# IPv4 local connections:
host    all             all             127.0.0.1/32            scram-sha-256
# IPv6 local connections:
#host    all             all             ::1/128                 scram-sha-256
host    all             all             ::1/128                 trust
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     peer
host    replication     all             127.0.0.1/32            scram-sha-256
host    replication     all             ::1/128                 scram-sha-256

host all all 192.168.56.1/32 md5
.....
```

2.2 사용자 설정

1) postgres Linux 사용자 계정으로 전환

```
root@mattermostdb:~# sudo -iu postgres
```

2) PostgreSQL 대화형 터미널을 시작

```
root@mattermostdb:~# sudo -u postgres psql
could not change directory to "/root": Permission denied
psql (14.11 (Ubuntu 14.11-0ubuntu0.22.04.1))
Type "help" for help.
```

3) 현재 생성된 데이터베이스 학인

```
postgres=# SELECT datname FROM pg_database;
  datname
-----------
 postgres
 template1
 template0
 KMatterDB
(4 rows)
```

4) KMatterDB 데이터베이스 접속

```
postgres=# \c KMatterDB
You are now connected to database "KMatterDB" as user "postgres".
```

5) 사용자 정보 확인

```
KMatterDB=# SELECT Username FROM Users;
   username
---------------
 mc529
  ......
(88 rows)

KMatterDB=# \q
```

6) baropamdb 데이터베이스를 생성

```
root@mattermostdb:~# sudo -u postgres psql
could not change directory to "/root": Permission denied
psql (14.11 (Ubuntu 14.11-0ubuntu0.22.04.1))
Type "help" for help.

postgres=# CREATE DATABASE baropamdb WITH ENCODING 'UTF8' LC_COLLATE='en_US.UTF-8'
LC_CTYPE='en_US.UTF-8' TEMPLATE=template0;
CREATE DATABASE
```

7) baropamdb 데이터베이스 사용자 'nurit'를 생성

```
postgres=# CREATE USER nurit WITH PASSWORD 'baropam';
CREATE ROLE
```

8) baropamdb 데이터베이스에 대한 사용자 액세스 권한을 부여

```
postgres=# GRANT ALL PRIVILEGES ON DATABASE baropamdb to nurit;
GRANT
```

9) baropamdb 데이터베이스 접속

```
postgres=# \c baropamdb
You are now connected to database "baropamdb" as user "postgres".
```

```
```

10) nurit 사용자에 대한 스키마 액세스 권한을 부여

```
postgres=# GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO nurit;
GRANT
```

권한을 부여하지 않으면 다음과 같은 오루가 발생함.

```
org.postgresql.util.PSQLException: ERROR: permission denied for table tb_secure_key
```

11) Secure key 테이블 생성

```
baropamdb=# CREATE TABLE IF NOT EXISTS TB_SECURE_KEY (
    USERNAME   varchar(64) NOT NULL,
    SECURE_KEY varchar(32) NOT NULL,
    CYCLE_TIME varchar(2) DEFAULT '30',
    LOGIN_TIME varchar(10) DEFAULT '0',
    PRIMARY KEY (USERNAME)
);
CREATE TABLE
```

참고)
USERNAME: 로그인-ID
SECURE_KEY: 개인별로 부여된 보안 키
CYCLE_TIME: 일회용 인증키 생성 주기(3~60초)
LOGIN_TIME: 로그인 최종시간으로 일회용 인증키 생성 주기 내에 사용자 한 명만 로그인 가능하게 제한하여
재사용 및 중간자 공격(Man-in-the-middle attack)에 대비하기 위하여 사용함

11) Secure key 테이블에 테스트할 정보 등록

```
baropamdb=#  INSERT INTO TB_SECURE_KEY (USERNAME, SECURE_KEY ) VALUES
('mc529','jlqlcHbVqdpj7b4PzBpM2DileBvmHFV/');
(1 row)

baropamdb=#  SELECT * FROM TB_SECURE_KEY WHERE USERNAME = 'mc529';
 username |           secure_key            | cycle_time | login_time
----------+---------------------------------+------------+------------
 mc529    | jlqlcHbVqdpj7b4PzBpM2DileBvmHFV/ | 30         | 0
(1 row)
```

3. Tomcat(WAS) 설정

3.1 Java 설치

Tomcat 9를 사용하려면 Java SE 8 이상이 시스템에 설치되어 있어야 한다. 자바 플랫폼의 오픈 소스 구현체인 OpenJDK 11을 설치한다.

1) openjdk-11-jdk 설치

```
root@mattermostdb:~#  sudo apt install openjdk-11-jdk
```

2) 설치된 Java 버전 확인

```
root@mattermostdb:~# java -version
openjdk version "11.0.22" 2024-01-16
OpenJDK Runtime Environment (build 11.0.22+7-post-Ubuntu-0ubuntu222.04.1)
OpenJDK 64-Bit Server VM (build 11.0.22+7-post-Ubuntu-0ubuntu222.04.1, mixed mode, sharing)
```

3.2 Tomcat 설치

Apache Tomcat은 오픈 소스 웹 서버 및 Java 서블릿 컨테이너이다. Java 기반 웹 사이트 및 응용 프로그램을 구축하는 데 가장 많이 사용되는 선택 중 하나이다. Tomcat은 가볍고 사용하기 쉬우며 강력한 애드온 생태계를 갖추고 있다.

1) Tomcat 사용자 생성
Tomcat 서비스를 실행할 홈 디렉토리 /opt/tomcat을 사용하여 새 시스템 사용자와 그룹을 생성한다.

```
root@mattermostdb:~#  sudo useradd -m -U -d /opt/tomcat -s /bin/false tomcat
```

2) 설치할 Tomcat 모듈 다운로드

```
root@mattermostdb:~#  wget https://downloads.apache.org/tomcat/tomcat-9/v9.0.85/bin/apache-tomcat-9.0.85.tar.gz
```

3) 다운로드 받은 Tomcat 파일 압축 해제

```
root@mattermostdb:~#  sudo tar -xf apache-tomcat-9.0.85.tar.gz -C /opt/tomcat/

root@mattermostdb:~# ls /opt/tomcat
apache-tomcat-9.0.85
```

4) Tomcat 설치 디렉토리 소유권을 사용자 및 그룹을 Tomcat으로 변경

```
root@mattermostdb:~# sudo chown -R tomcat: /opt/tomcat
```

5) Tomcat의 bin 디렉토리 내에 있는 셸 스크립트에 실행 권한 부여

```
root@mattermostdb:~# sudo sh -c 'chmod +x /opt/tomcat/apache-tomcat-9.0.85/bin/*.sh'
```

6) SystemD 단위 파일 생성
셸 스크립트를 사용하여 Tomcat 서버를 시작하고 중지하는 대신 서비스로 실행되도록 설정한다.

```
root@mattermostdb:~# vi /etc/systemd/system/tomcat.service
# /etc/systemd/system/tomcat.service

[Unit]
Description=Tomcat 9 servlet container
After=network.target

[Service]
Type=forking

User=tomcat
Group=tomcat

Environment="JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64"
Environment="JAVA_OPTS=-Djava.security.egd=file:///dev/urandom -Djava.awt.headless=true"

Environment="CATALINA_BASE=/opt/tomcat/apache-tomcat-9.0.85"
Environment="CATALINA_HOME=/opt/tomcat/apache-tomcat-9.0.85"
Environment="CATALINA_PID=/opt/tomcat/apache-tomcat-9.0.85/temp/tomcat.pid"
Environment="CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC"

ExecStart=/opt/tomcat/apache-tomcat-9.0.85/bin/startup.sh
ExecStop=/opt/tomcat/apache-tomcat-9.0.85/bin/shutdown.sh

[Install]
WantedBy=multi-user.target
```

7) 서비스 설정을 데몬에 즉시 반영

```
root@mattermostdb:~# sudo systemctl daemon-reload
```

8) Tomcat 서비스를 사용하도록 설정하고 시작

```
root@mattermostdb:~# sudo systemctl enable --now tomcat
Created symlink /etc/systemd/system/multi-user.target.wants/tomcat.service →
/etc/systemd/system/tomcat.service.
```

9) Tomcat 서비스 상태를 확인

```
root@mattermostdb:~# sudo systemctl status tomcat
● tomcat.service - Tomcat 9 servlet container
     Loaded: loaded (/etc/systemd/system/tomcat.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2024-04-21 00:57:54 UTC; 17s ago
    Process: 308391 ExecStart=/opt/tomcat/apache-tomcat-9.0.85/bin/startup.sh (code=exite>
   Main PID: 308398 (java)
      Tasks: 29 (limit: 4477)
     Memory: 119.6M
        CPU: 2.169s
```

```
     CGroup: /system.slice/tomcat.service
             └─308398 /usr/lib/jvm/java-11-openjdk-amd64/bin/java -Djava.util.logging.con>

Apr 21 00:57:54 mattermostdb systemd[1]: Starting Tomcat 9 servlet container...
Apr 21 00:57:54 mattermostdb startup.sh[308391]: Tomcat started.
Apr 21 00:57:54 mattermostdb systemd[1]: Started Tomcat 9 servlet container.
```

참고)
root@mattermostdb:~# sudo systemctl start tomcat ➔서비스 시작
root@mattermostdb:~# sudo systemctl stop tomcat ➔서비스 종료
root@mattermostdb:~# sudo systemctl restart tomcat ➔ 서비스 재시작
root@mattermostdb:~# sudo systemctl status tomcat ➔ 서비스 상태

10) 방화벽을 구성
서버가 방화벽으로 보호되고 로컬 네트워크 외부에서 Tomcat에 액세스하려면 포트 8080을 열어야 한다.

```
root@mattermostdb:~# sudo ufw allow 8080/tcp
Rules updated
Rules updated (v6)
```

3.3 BaroPAM 모듈 반영

Mattermost 로그인 시 비밀번호를 BaroPAM의 일회용 인증키로 대체한 일회용 인증키를 검증하는 모듈은 Tomcat 기반 하에 운영되도록 구성한다.

1) 공통 라이브러리

```
root@mattermostdb:/opt/tomcat/apache-tomcat-9.0.85/lib#
barokey.jar
json.jar
json-lib-2.4.jar
json-simple-1.1.1.jar
log4j-1.2.17.jar
postgresql-42.7.3.jar
```

2) Log4j 속성 설정

```
root@mattermostdb:/opt/tomcat/apache-tomcat-9.0.85/webapps/ROOT/WEB-INF/classes# vi
log4j.properties
# A sample log4j configuration file
# Create two appenders, one called stdout and the other called rolling
log4j.rootLogger=INFO, stdout, rolling
log4j.logger.JspLogger=DEBUG

# Configure the stdout appender to go to the console
log4j.appender.stdout=org.apache.log4j.ConsoleAppender

# Configure the stdout appender to use the PatternLayout
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
```

```
# Pattern to output the caller's filename and line number
log4j.appender.stdout.layout.ConversionPattern=%5p [%t] (%F:%L)- %m%n

# Configure the rolling appender to be a RollingFileAppender
log4j.appender.rolling=org.apache.log4j.RollingFileAppender

# Configure the name of the logout for the rolling appender
log4j.appender.rolling.File=/opt/tomcat/apache-tomcat-9.0.85/logs/output.log

# Set up the maximum size of the rolling log file
log4j.appender.rolling.MaxFileSize=10MB
#log4j.appender.rolling.DatePattern='.'yyyy-MM-dd

# Keep one backup file of the rolling appender
log4j.appender.rolling.MaxBackupIndex=1

# Configure the layout pattern and conversion pattern for the rolling appender
log4j.appender.rolling.layout=org.apache.log4j.PatternLayout
log4j.appender.rolling.layout.ConversionPattern=%d{ABSOLUTE} - %p %c - %m%n
```

3) 일회용 인증키 검증 모듈

```
root@mattermostdb:/opt/tomcat/apache-tomcat-9.0.85/webapps/ROOT/baropam# vi result_baropam.jsp
result_baropam.jsp
<%@ page contentType="text/html; charset=UTF-8" language="java" pageEncoding="UTF-8" %>
<%@ page import="org.apache.log4j.*"%>
<%@ page import="java.text.*"%>
<%@ page import="java.util.*"%>

<%@ page import="java.sql.Connection"%>
<%@ page import="java.sql.DriverManager"%>
<%@ page import="java.sql.ResultSet"%>
<%@ page import="java.sql.PreparedStatement"%>

<%@ page import="org.json.JSONObject"%>
<%@ page import="org.json.simple.*"%>
<%@ page import="com.barokey.*"%>
<% request.setCharacterEncoding("utf-8"); %>
<% response.setContentType("text/html; charset=utf-8"); %>
<%!
private Logger logger = Logger.getLogger("JspLogger");
%>
<%
/*-------------------------------------------------------------------------*/
/* 변수선언 및 초기화.                                                     */
/*-------------------------------------------------------------------------*/
int     ii = 0, jj = 0, kk = 0, ll = 0;                 // Index

String  jdbc_driver = "org.postgresql.Driver";          // JDBC Driver명
String  jdbc_url    = "jdbc:postgresql://localhost:5432/baropamdb";
String  select_stmt = "SELECT USERNAME, SECURE_KEY, CYCLE_TIME, LOGIN_TIME FROM TB_SECURE_KEY
WHERE USERNAME = ?";
```

```
String  update_stmt = "UPDATE TB_SECURE_KEY SET LOGIN_TIME = ? WHERE USERNAME = ?";

String  secure_key  = "";                               // Secure key
String  cycle_time  = "30";                             // 생성주기
long    login_time  = 0;                                // 최종 로그인 시간
long    curr_time   = 0;                                // 현재 시간

String  result      = "Fail";                           // Result
boolean bota_key    = false;                            // 인증키 검증
/*-------------------------------------------------------------------------*/
/* Request에서 데이터를 얻어옴(로그인 정보).                               */
/*-------------------------------------------------------------------------*/
String  username = request.getParameter("username");
String  password = request.getParameter("password");

logger.info("(result_baropam.jsp)Starting.....................");
String  param = request.getServerName() + request.getRequestURI()
              + "?remote_addr="         + request.getRemoteAddr()
              + "&username="            + username
              + "&password="            + password
              ;
logger.info(param);
/*-------------------------------------------------------------------------*/
/* Begin.                                                                  */
/*-------------------------------------------------------------------------*/
try {
    /*---------------------------------------------------------------------*/
    /* JDBC 설정 및 연결.                                                  */
    /*---------------------------------------------------------------------*/
    Class.forName(jdbc_driver).newInstance();
    Connection conn = DriverManager.getConnection(jdbc_url, "nurit", "baropam");

    PreparedStatement pstmt = conn.prepareStatement(select_stmt);
    pstmt.setString(++ii, username);
    /*---------------------------------------------------------------------*/
    /* 사용자 정보 조회.                                                   */
    /*---------------------------------------------------------------------*/
    ResultSet rs = pstmt.executeQuery();
    while(rs.next()){
        username   = rs.getString("USERNAME"  );
        secure_key = rs.getString("SECURE_KEY");
        cycle_time = rs.getString("CYCLE_TIME");
        login_time = rs.getLong  ("LOGIN_TIME");

        logger.info("username   = [" + username   + "]");
        logger.info("secure_key = [" + secure_key + "]");
        logger.info("cycle_time = [" + cycle_time + "]");
        logger.info("login_time = [" + login_time + "]");
    }
    /*---------------------------------------------------------------------*/
    /* 사용자 정보가 존재하는 경우.                                        */
    /*---------------------------------------------------------------------*/
    if (!"".equals(username) && !"".equals(secure_key) && !"".equals(cycle_time)) {
```

```
        /*------------------------------------------------------------------*/
        /* 현재 시간 Edit.                                                  */
        /*------------------------------------------------------------------*/
        curr_time = barokey.get_logintime(cycle_time);

        logger.info("curr_time = [" + curr_time + "]");
        /*------------------------------------------------------------------*/
        /* 로그인 최종시간이 생성주기 보다 큰 경우.                         */
        /*------------------------------------------------------------------*/
        if (curr_time > login_time) {
            /*--------------------------------------------------------------*/
            /* 인증키 검증.                                                 */
            /*--------------------------------------------------------------*/
            bota_key = barokey.verifyKEY(username, secure_key, cycle_time, password);
            /*--------------------------------------------------------------*/
            /* 인증키 검증(성공).                                           */
            /*--------------------------------------------------------------*/
            if (bota_key == true) {
                /*----------------------------------------------------------*/
                /* 최종 로그인 시간 Update.                                 */
                /*----------------------------------------------------------*/
                try {
                    conn.setAutoCommit(false);
                    pstmt = conn.prepareStatement(update_stmt);
                    pstmt.setString(1, Long.toString(curr_time));
                    pstmt.setString(2, username );
                    if (pstmt.executeUpdate() > 0) {
                        conn.commit();
                    } else {
                        conn.rollback();
                    }
                } catch (java.sql.SQLException e) {
                    logger.info("SQL = [" + update_stmt + "]");
                    logger.info("SQLException = [" + e + "]");
                    e.printStackTrace();
                } catch(Exception e) {
                    logger.info("Exception = [" + e + "]");
                    e.printStackTrace();
                }
                result = "OK";
            }
        }
    /*------------------------------------------------------------------*/
    /* 사용자 정보가 존재하지 않는 경우.                                */
    /*------------------------------------------------------------------*/
    } else {
        logger.info("no data found. username = [" + username + "]");
    }
/*----------------------------------------------------------------------*/
/* 예외사항 처리(SQLException).                                         */
/*----------------------------------------------------------------------*/
} catch (java.sql.SQLException e) {
    logger.info("SQL = [" + select_stmt + "]");
```

```
    logger.info("SQLException = [" + e + "]");
    e.printStackTrace();
    result = "Fail";
/*-------------------------------------------------------------------------*/
/* 예외사항 처리(Exception).                                               */
/*-------------------------------------------------------------------------*/
} catch(Exception e) {
    logger.info("Exception = [" + e + "]");
    e.printStackTrace();
    result = "Fail";
/*-------------------------------------------------------------------------*/
/* Finally.                                                                */
/*-------------------------------------------------------------------------*/
} finally {
    logger.info("Result = [" + result + "]");
    JSONObject jsons = new JSONObject();
    jsons.put("Body", result);
    out.println(jsons);
    logger.info("json = [" + jsons + "]");
    logger.info("(result_baropam.jsp)Ending.....................");
}
%>
```

4. Mattermost 설정

4.1 Mattermost 반영

1) Mattermost 실행 파일 반영

Mattermost 서버용 바이너리 파일을 빌드한 후 생성된 "**mattermost**" 바이너리 파일을 sftp 툴을 이용하여 AP 서버 "**/opt/mattermost/bin**" 디렉토리로 전송한다.

Mattermost 서버용 바이너리 파일이 존재하는 디렉토리(/opt/mattermost/bin)에 다음과 같은 파일이 존재한다.

```
mattermost     => 기존 바이너리 파일
mattermost.org => 기존 바이너리 파일
mattermost.ota => BaroPAM이 적용된 바이너리 파일
```

2) 비밀번호 최소 자릿수(8 → 6) 변경

```
root@mattermost:/# vi /opt/mattermost/config/config.json
.............
"PasswordSettings": {
        "MinimumLength": 6,
        "Lowercase": false,
        "Number": false,
        "Uppercase": false,
        "Symbol": false,
        "EnableForgotLink": true
    },
.............
```

2) Mattermost에 BaroPAM 관련 환경변수 설정

환경변수	설명	비고
ACL_TYPE	기존 비밀번호를 사용할 건지, BaroPAM의 일회용 인증키로 대체할 것인 지정 (**allow/deny**)	
ACL_NAME	**allow**(허용)을 선택한 경우 ACL_NAME 에 지정된 파일 내에 존재하는 Username는 BaroPAM의 일회용 인증키를 사용하고 존재하지 않는 Usernam는 기존 비밀번호를 사용해야 함. **deny**(제외)를 선택한 경우 ACL_NAME 에 지정된 파일 내에 존재하는 Username는 기존 비밀번호를 사용하고 존재하지 않은 Username는 BaroPAM의 일회용 인증키를 사용해야 함.	
BAROPAMADDR	비밀번호를 대체한 BaroPAM의 일회용 인증키를 검증하기 위하여 호출하는 Tomcat URL 정보를 설정함.	

```
root@mattermost:/# vi /lib/systemd/system/mattermost.service

[Unit]
Description=Mattermost
```

```
After=network.target

[Service]
Type=notify
ExecStart=/opt/mattermost/bin/mattermost
TimeoutStartSec=3600
KillMod=mixed
Restart=always
RestartSec=10
WorkingDirectory=/opt/mattermost
User=mattermost
Group=mattermost
LimitNOFILE=49152

Environment="ACL_TYPE=deny"
Environment="ACL_NAME=/opt/mattermost/bin/.baro_acl"
Environment="BAROPAMADDR=http://192.168.56.2/:8080"

[Install]
WantedBy=multi-user.target
```

3) 서비스 설정을 데몬에 즉시 반영

```
root@mattermost:/# sudo systemctl daemon-reload
```

4) Mattermost 서비스를 사용하도록 설정하고 시작

```
root@mattermost:/# sudo systemctl enable —now mattermost
```

5) Mattermost 서비스 상태를 확인

```
root@mattermost:/# sudo systemctl status mattermost
● mattermost.service - Mattermost
     Loaded: loaded (/lib/systemd/system/mattermost.service; enabled; vendor pr>
     Active: active (running) since Tue 2024-04-09 00:47:36 UTC; 1 week 5 days >
   Main PID: 137847 (mattermost)
      Tasks: 65 (limit: 4477)
     Memory: 372.2M
        CPU: 1h 2min 17.784s
     CGroup: /system.slice/mattermost.service
             ├─137847 /opt/mattermost/bin/mattermost
             ├─137861 plugins/jira/server/dist/plugin-linux-amd64
             ├─137888 plugins/com.mattermost.nps/server/dist/plugin-linux-amd64
             ├─137896 plugins/playbooks/server/dist/plugin-linux-amd64
             ├─137904 plugins/com.mattermost.calls/server/dist/plugin-linux-amd>
             └─339221 plugins/com.mattermost.badges/server/dist/plugin-linux-am>

Apr 21 01:58:01 mattermost mattermost[137847]: {"timestamp":"2024-04-21 01:58:0>
Apr 21 01:58:31 mattermost mattermost[137847]: {"timestamp":"2024-04-21 01:58:3>
Apr 21 02:02:30 mattermost mattermost[137847]: {"timestamp":"2024-04-21 02:02:3>
Apr 21 02:13:30 mattermost mattermost[137847]: {"timestamp":"2024-04-21 02:13:3>
Apr 21 02:24:30 mattermost mattermost[137847]: {"timestamp":"2024-04-21 02:24:3>
Apr 21 02:30:31 mattermost mattermost[137847]: {"timestamp":"2024-04-21 02:30:3>
```

```
Apr 21 02:35:31 mattermost mattermost[137847]: {"timestamp":"2024-04-21 02:35:3>
Apr 21 02:46:31 mattermost mattermost[137847]: {"timestamp":"2024-04-21 02:46:3>
```

참고)
root@mattermost:/# sudo systemctl start mattermost ➔서비스 시작
root@mattermost:/# sudo systemctl stop mattermost ➔서비스 종료
root@mattermost:/# sudo systemctl restart mattermost ➔ 서비스 재시작
root@mattermost:/# sudo systemctl status mattermost ➔ 서비스 상태

4.2 Mattermost 로그인

1) 로그인 화면

Mattermost 로그인 화면에서 본인의 Username를 입력한 후 BaroPAM 앱에서 생성한 일회용 인증키(613045)를 Password 입력 항목에 입력한 후 "**Log in**" 버튼을 클릭한다.

현재 BaroPAM 앱과 Secure key 테이블에 사용되는 Secure key를 동일하게 **Secure key**는 "**jlqlcHbVqdpj7b4PzBpM2DileBvmHFV/**", **일회용 인증키 생성주기**는 "**30**"초로 반영되어 있음.

2) Mattermost 서버 로그 확인

```
root@mattermost:/opt/mattermost/logs# tail -f mattermost.log

{"timestamp":"2024-04-21 04:12:22.408
Z","level":"error","msg":"ComparePasswords==========>","caller":"users/baropam.go:27"}

{"timestamp":"2024-04-21 04:12:22.409 Z","level":"error","msg":"Get
W"http://192.168.56.2/:8080/baropam/result_baropam.jsp?password=613046&username=mc529W": dial tcp
192.168.56.2:80: connect: connection refused","caller":"users/baropam.go:50"}
```

3) Tomcat 서버 로그 확인

```
root@mattermostdb:/opt/tomcat/apache-tomcat-9.0.85/logs# tail -f catalina.out
 INFO [http-nio-8080-exec-5] (result_005fbaropam_jsp.java:185)-
(result_baropam.jsp)Starting.....................
 INFO [http-nio-8080-exec-5] (result_005fbaropam_jsp.java:191)-
192.168.56.2/baropam/result_baropam.jsp?remote_addr=192.168.56.1&username=mc529&password=613046
 INFO [http-nio-8080-exec-5] (result_005fbaropam_jsp.java:214)- username   = [mc529]
 INFO [http-nio-8080-exec-5] (result_005fbaropam_jsp.java:215)- secure_key =
[jlqIcHbVqdpj7b4PzBpM2DileBvmHFV/]
 INFO [http-nio-8080-exec-5] (result_005fbaropam_jsp.java:216)- cycle_time = [30]
 INFO [http-nio-8080-exec-5] (result_005fbaropam_jsp.java:217)- login_time = [0]
 INFO [http-nio-8080-exec-5] (result_005fbaropam_jsp.java:228)- curr_time = [57122437]
 INFO [http-nio-8080-exec-5] (result_005fbaropam_jsp.java:290)- Result = [OK]
 INFO [http-nio-8080-exec-5] (result_005fbaropam_jsp.java:294)- json = [{"Body":"OK"}]
 INFO [http-nio-8080-exec-5] (result_005fbaropam_jsp.java:295)-
(result_baropam.jsp)Ending.....................
```

만약, Postgresql 데이터베이스의 Secure key 테이블에 사용자가 등록되어 있지 않으면 "**no data found. username = [*username*]**" 메시지가 출력된다.

5. About BaroPAM

Version 1.0 - Official Release - 2016.12.1

http://www.nurit.co.kr

제 조 사 : 주식회사 누리아이티
등록번호 : 258-87-00901
대표이사 : 이종일
대표전화 : 02-2665-0119(영업문의/기술지원)
이 메 일 : mc529@nurit.co.kr
주 소 : 서울시 강서구 마곡중앙2로 15, 913호(마곡동, 마곡테크노타워2)