Secure Coding Review Report

1. Selected Language and Application

Language: Python
Application: Streamlit-based Login/Signup System

2. Insecure Code Example

```
import streamlit as st

st.title("■ Insecure Login/Signup System")

if "users" not in st.session_state:
st.session_state["users"] = {} # {username: password}

menu = st.sidebar.radio("Menu", ["Signup", "Login"])

if menu == "Signup":
username = st.text_input("Username")
password = st.text_input("Password", type="password")
if st.button("Signup"):
st.session_state["users"][username] = password
st.success("Signup successful!")

elif menu == "Login":
username = st.text_input("Username")
password = st.text_input("Password", type="password")
if st.button("Login"):
if username in st.session_state["users"] and st.session_state["users"][username] ==
password:
st.success(f"Welcome {username}! ■")
else:
st.error("Invalid credentials ■")
```

3. Vulnerabilities Found

- Passwords stored in plain text.
- No password policy enforced.
- No input validation for usernames or passwords.
- Unlimited login attempts allowed (brute force risk).
- Session state storing sensitive data without protection.

4. Recommendations

- Hash and salt passwords using bcrypt.
- Enforce strong password policy (minimum length, complexity).
- Validate inputs before storing or checking credentials.
- Implement brute-force protection (lockout after multiple failed attempts).
- Use secure databases instead of in-memory session storage.

5. Secure Code Example

```
import streamlit as st
import bcrypt
import time

st.set_page_config(page_title="Secure Auth System", page_icon="■")
st.markdown("■ Secure Login / Signup System", unsafe_allow_html=True)
```

```
if "users" not in st.session_state:
st.session_state["users"] = {}
if "failed_attempts" not in st.session_state:
st.session_state["failed_attempts"] = {}

menu = st.sidebar.radio("Menu", ["Signup", "Login"])

def signup():
st.subheader("■ Create an Account")
username = st.text_input("Enter Username")
password = st.text_input("Enter Password", type="password")

if st.button("Signup"):
if username in st.session_state["users"]:
st.error("■■ Username already exists!")
elif len(password) < 8:
st.warning("■■ Password must be at least 8 characters.")
else:
hashed = bcrypt.hashpw(password.encode("utf-8"), bcrypt.gensalt())
st.session_state["users"][username] = hashed
st.success(f"■ Account created for {username}")

def login():
st.subheader("■ Login to Your Account")
username = st.text_input("Username")
password = st.text_input("Password", type="password")

if st.button("Login"):
if st.session_state["failed_attempts"].get(username, 0) >= 3:
st.error("■ Account locked due to multiple failed attempts. Try again later.")
time.sleep(3)
return

if username in st.session_state["users"] and
bcrypt.checkpw(password.encode("utf-8"), st.session_state["users"][username]):
st.success(f"■ Welcome {username}!")
st.session_state["failed_attempts"][username] = 0
else:
st.error("■ Invalid username or password")
st.session_state["failed_attempts"][username] =
st.session_state["failed_attempts"].get(username, 0) + 1

if menu == "Signup":
signup()
elif menu == "Login":
login()
```

6. Documented Findings

Vulnerability	Risk	Fix Applied
Plain-text passwords	Credential theft	Passwords hashed with bcrypt
Weak passwords	Easily guessed	Minimum 8 characters enforced
No input validation	Empty/malformed input	Validation added
Unlimited attempts	Brute force risk	Lockout after 3 failures
Insecure storage	Sensitive info in session state	Recommendation: use DB