Algorithm 1 Compact Secure Login System

```
Initialize: app, database, tracking (login_attempts, blocked_sessions, otp_store)
procedure HASH_PWD(pwd)
    salt ← random_bytes(60)
    return salt + PBKDF2(pwd, salt, iter=100000)
end procedure
procedure VERIFY_PWD(stored, provided)
    salt ← extract_salt(stored)
    return extract_hash(stored) = PBKDF2(provided, salt, iter=100000)
end procedure
procedure CHECK_SQLI(input)
    if empty(input) or input matches "^[a-z0-9_]+$" then return False
    end if
    norm_input ← normalize(input)
    for pattern in DANGEROUS_PATTERNS do
        if pattern in norm_input then return True
        end if
    end for
    return False
end procedure
procedure LOGIN(username, password, captcha, otp)
    if empty(username) or empty(password) then return error
    end if
    if check_sqli(username) or check_sqli(password) then
        log("SQLI"); return error
    end if
    if captcha_enabled and captcha ≠ expected then
        log("CAPTCHA FAIL"); return error
    end if
    if rate_limiting_enabled then
        if session_id in blocked_sessions then
            log("BLOCKED"); return "Session blocked"
        end if
        if too_many_attempts(username) then
            log("RATE LIMIT"); block(session_id); return "Too many attempts"
        end if
    end if
    user ← find_user(username)
    if user = null or not verify_pwd(user.password, password) then
        increment_attempts(username); log("FAIL"); return "Invalid"
    end if
    if two_factor_enabled then
        if otp = null then
            gen_otp ← random_code(6); store_otp(username, gen_otp)
            log("2FA"); return "2FA required"
        else
            stored_otp ← get_stored_otp(username)
            if stored_otp = null or otp ≠ stored_otp then
                log("2FA FAIL"); return "Invalid 2FA"
            end if
            clear_otp(username)
        end if
    end if
    log("SUCCESS"); reset_attempts(username); return success
end procedure
```