ERP/1: Комунікатор

Техноробочий проект

на створення та впровадження
локальної системи безпечного зв'язку,
видачі сертифікатів та обміну повідомленнями

Згідно з Постановою КМУ № 205 від 21.02.2025

Формалізація вимог за стандартом ISO/IEC/IEEE 29148:2018

Зміст

Зміст

1. Загальні відомості про засіб інформатизації

1.1. Найменування та підстави для розробки

Найменування засобу інформатизації: Модуль «Комунікатор» (інфраструктура безпечного зв'язку CHAT v2 / MAIL v3) у складі інформаційної системи управління підприємством ERP/1.

Підстави для виконання робіт:

- Закон України «Про захист персональних даних»;
- Закон України «Про захист інформації в інформаційно-комунікаційних системах»;
- Порядок використання засобів інформатизації (Постанова КМУ № 205 від 21.02.2025);
- Регламент взаємодії систем безпеки та шифрування X.422.

1.2. Призначення та цілі створення засобу

Модуль призначений для розгортання периферійних систем захищеного обміну повідомленнями та документами, локального управління сертифікатами відкритих ключів (CA), VPN-тунелювання, служби каталогів (LDAP) та Identity Access Management (IAS) без використання сторонніх хмарних рішень та мови Rust на периферії.

1.3. План-графік виконання етапів робіт

1. **Етап 1: Технічне проектування** — опис ASN.1 DER структур, схем каталогів LDAP та специфікацій WireGuard тунелів. (1.5 місяці).
2. **Етап 2: Реалізація CA та EST/CMP** — запуск локального засвідчувального центру для видачі та перевірки статусів X.509. (2 місяці).
3. **Етап 3: Розробка CHAT v2 та MAIL v3 серверів** — впровадження асинхронного брокера повідомлень на базі Erlang/OTP та поштової черги доставки документів. (2.5 місяці).
4. **Етап 4: Інтеграція клієнтських додатків** — реалізація клієнтських Swift/SwiftUI SDK для роботи за протоколом Buddha X.422. (1.5 місяці).
5. **Етап 5: Державна експертиза КСЗІ** — отримання атестату відповідності КСЗІ України. (2 місяці).

2. Відомості про робочий процес та умови експлуатації

2.1. Опис бізнес-процесів (BPMN / FSM)

2.1.1. Процес автентифікації клієнта за сертифікатом (chat_auth.erl)

Керує процесом перевірки клієнтського сертифіката X.509 при встановленні зв'язку:

1. **ClientConnect**: Клієнт ініціює TCP/TLS з'єднання.
2. **RequestCertificate**: Сервер вимагає клієнтський сертифікат (Mutual TLS).
3. **VerifyCertificateChain**: Перевірка підпису та ланцюжка сертифікатів через локальний CA.
4. **CheckOCSPStatus**: Запит до локального OCSP-сервера для перевірки наявності сертифіката в списку відкликаних.
5. **LoadLDAPAttributes**: Читання профілю користувача з LDAP для перевірки ABAC-атрибутів доступу.
6. **GrantAccess**: Створення сесії зв'язку, генерація тимчасового токена (AccessToken).

2.1.2. Процес обміну повідомленнями за протоколом Buddha (chat_message.erl)

Керує життєвим циклом передачі повідомлень (CHAT v2):

1. **ReceiveEnvelope**: Прийом DER-кодованого CMS-пакета від відправника.
2. **RouteByProfile**: Читання незашифрованого заголовка отримувача та пошук його сесії.
3. **StoreTransientQueue**: Збереження зашифрованого конверта в оперативній таблиці Mnesia.
4. **DeliverToClient**: Відправка пакета отримувачу при встановленні з'єднання.
5. **WaitDeliveryReceipt**: Очікування підтвердження отримання (Ack) від клієнта.
6. **DestroyMessage**: Повне видалення повідомлення з оперативної пам'яті сервера.

2.2. Опис ролей та прав доступу (ABAC)

Автентифікація користувачів базується на атрибутах LDAP. Приклад Erlang-функції авторизації доступу до каналів зв'язку:

```
allow(User, Action, Channel) ->
    UserSecurityLevel = maps:get(clearance_level, User),
    ChannelSecurityLevel = maps:get(required_level, Channel),
    UserDept = maps:get(department, User),
    ChannelDept = maps:get(department, Channel),

    HasClearance = (UserSecurityLevel >= ChannelSecurityLevel),
    MatchesDept = (UserDept == ChannelDept),

    case {HasClearance, Action} of
        {true, write_message} -> MatchesDept;
        {true, read_message} -> true;
        _ -> false
    end.
```

2.3. Специфікація апаратного забезпечення та мережеві порти

Підсистема «Комунікатор» використовує такі мережеві порти на периферійному сервері:

Порт	Протокол	Сервіс	Призначення
8000	TCP (WebSocket)	CHAT v2	Асинхронний обмін DER-повідомленнями
4221	UDP (Multicast)	CHAT v1 / Discovery	Локальне виявлення пристроїв у LAN
8443	TCP (HTTPS)	IAS / EST	Реєстрація клієнтів та перевипуск сертифікатів
8089	TCP (HTTP)	CA / OCSP	Онлайн-перевірка статусу сертифікатів
8088	TCP (HTTP)	CA / CMP	Управління життєвим циклом сертифікатів
636	TCP (LDAPS)	LDAP	Служба каталогів користувачів
51820	UDP	VPN / Тунелі	Захищений WireGuard-канал
465	TCP (SMTPS)	MAIL v3	Відправка документів (SMTP over TLS)
993	TCP (IMAPS)	MAIL v3	Отримання документів (IMAP over TLS)

2.4. Архітектурна діаграма взаємодії та зв'язків додатків

Взаємодія між сервером та клієнтом, а також внутрішня інтеграція Erlang/OTP додатків у межах вузла «Комунікатор» представлена на схемі:

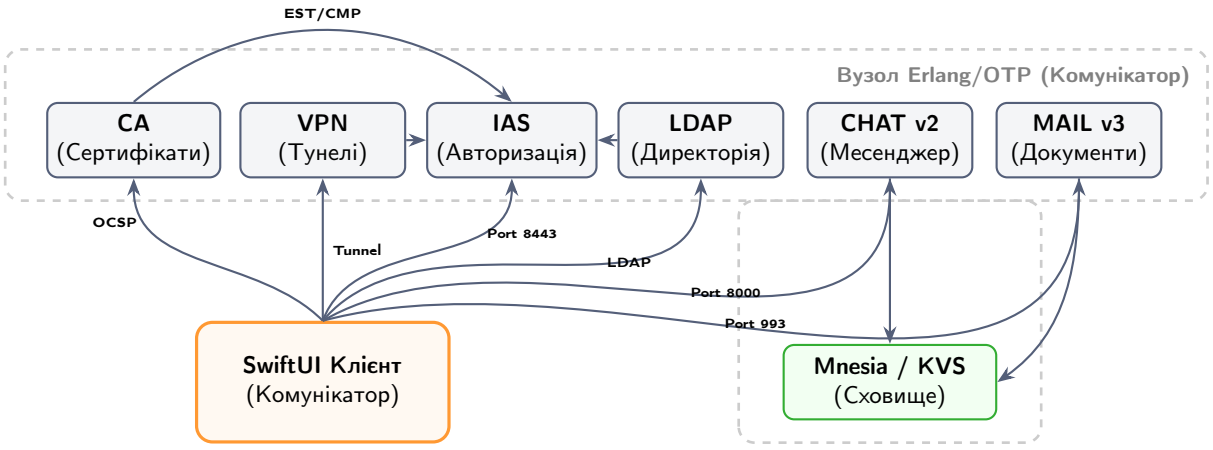

Рис. 1: Архітектурна діаграма взаємодії OTP-додатків та мережевих портів

2.5. Детальна архітектура Комунікатора та інтеграція репозиторіїв

Для побудови підсистеми використовуються репозиторії екосистеми ERP/1:

1. **Репозиторії екосистеми Erlang/OTP** (шляхи пошуку):
    - `synrc/chat/` — Сервер CHAT v2 та клієнтські N2O/Nitro залежності.
    - `synrc/ca/` — Локальний сертифікаційний центр.
    - `synrc/ldap/` — Реалізація вбудованого LDAP сервера.
    - `erpuno/mail/` — Сервер документів MAIL v3.
    - `zencrypted/ias/` — Сервер авторизації та ідентифікації.
    - `zencrypted/vpn/` — Модулі VPN тунелів.
    - `zencrypted/protocol/` — Опис та специфікації протоколу Buddha.

3. Інформаційне та програмне забезпечення

3.1. Інформаційне забезпечення (Схеми та Дані)

3.1.1. Визначення структур даних (Erlang Records)

```
-record(chat_session, {
    session_id,                 % UUID сесії (PK)
    client_id,                  % Серійний номер сертифіката X.509
    device_id,                  % Однозначний ідентифікатор пристрою
    ip_address,                 % Поточна IP-адреса
    access_token,               % Токен доступу сесії
    status = active,            % active | expired | revoked
    created_at = 0
}).

-record(chat_roster, {
    roster_id,                  % UUID ростера (PK)
    owner_id,                   % UUID власника
    contacts = []               % Список контактів
}).

-record(ca_certificate, {
    serial_number,              % Серійний номер X.509 (PK)
    subject_dn,                 % Відмітне ім'я суб'єкта
    public_key_der,             % Публічний ключ у DER кодуванні
    not_before = 0,             % Timestamp початку дії
    not_after = 0,              % Timestamp завершення дії
    status = valid              % valid | revoked | expired
}).

-record(vpn_tunnel_state, {
    tunnel_id,                  % UUID тунелю (PK)
    client_ip,                  % Виділена IP-адреса клієнта
    public_key,                 % Публічний ключ WireGuard
    bytes_sent = 0,             % Надіслано байт
    bytes_recv = 0,             % Отримано байт
    last_handshake = 0          % Timestamp останнього рукостискання
}).
```

3.1.2. Специфікації ASN.1 для обміну даними

Нижче наведено специфікацію першого шару для UDP-мультікаст та локальних мереж з файлу `MESSAGE-v1.asn1`:

```
MESSAGE-v1 DEFINITIONS ::= BEGIN

    IMPORTS ContentInfo FROM CryptographicMessageSyntax-2009
            Certificate FROM PKIX1Explicit-2009 ;

    OS  ::= ENUMERATED { apple (1), microsoft (2), google (3),
                         ericsson (4), nokia (5), cisco (6) }
    P2P ::= SEQUENCE { from OCTET STRING, to OCTET STRING }
    MUC ::= SEQUENCE { room OCTET STRING }

    CHATString ::= OCTET STRING

    Feature ::= SEQUENCE { id OCTET STRING, key OCTET STRING,
                           value OCTET STRING, group OCTET STRING }
    AuthorityMethod ::= ENUMERATED { announce (0), register (1),
```

```
                                    authenticate (2), dismiss (3),
                                    renew (4) }
    Authority ::= SEQUENCE {
       session OCTET STRING,
       type AuthorityMethod,
       cert OCTET STRING,
       settings SEQUENCE OF feature Feature }

    File ::= SEQUENCE {
       id OCTET STRING,
       mime OCTET STRING,
       payload ANY,
       parentid OCTET STRING }

    Channel ::= SEQUENCE { channel OCTET STRING }
    MessageType ::= ENUMERATED { sys (1), reply (2), forward (3),
                                 read (4), edited (5) }
    MessageStatus ::= ENUMERATED { async (1), delete (2), clear (3),
                                   update (4), edit (5) }
    MessageFeed ::= CHOICE { p2p [0] P2P, muc [1] MUC, chan [2] Channel }
    Message ::= SEQUENCE {
       id CHATString,
       feed MessageFeed,
       from CHATString,
       to CHATString,
       files SEQUENCE OF file File,
       type MessageType,
       link CHOICE { empty [1] NULL, message [2] Message },
       seenby CHATString,
       repliedby CHATString,
       mentioned SEQUENCE OF name CHATString,
       status MessageStatus DEFAULT clear }

    Ack ::= SEQUENCE { table OCTET STRING, id OCTET STRING }

END
```

3.2. Програмне забезпечення (Реалізація)

3.2.1. chat_message.erl (DSL процесу обміну повідомленнями)

```
-module(chat_message).
-include("bpe.hrl").
-compile(export_all).

def() ->
    #process{
        name = 'Buddha Protocol Message Delivery',
        beginEvent = 'ReceiveEnvelope',
        endEvent = 'DestroyMessage',
        tasks = [
            #beginEvent{name='ReceiveEnvelope'},
            #serviceTask{name='RouteByProfile', module=?MODULE},
            #serviceTask{name='StoreTransientQueue', module=?MODULE},
            #serviceTask{name='DeliverToClient', module=?MODULE},
            #serviceTask{name='WaitDeliveryReceipt', module=?MODULE},
            #endEvent{name='DestroyMessage'}
        ],
        flows = [
            #sequenceFlow{name='1', source='ReceiveEnvelope', target='RouteByProfile'},
            #sequenceFlow{name='2', source='RouteByProfile', target='StoreTransientQueue'},
            #sequenceFlow{name='3', source='StoreTransientQueue', target='DeliverToClient'},
            #sequenceFlow{name='4', source='DeliverToClient', target='WaitDeliveryReceipt'},
            #sequenceFlow{name='5', source='WaitDeliveryReceipt', target='DestroyMessage'}
        ],
```

```
        roles = [operator, system]
    }.

action({serviceTask, 'RouteByProfile'}, Proc) ->
    % Визначення одержувача та активної сесії
    {reply, Proc};
action({serviceTask, 'StoreTransientQueue'}, Proc) ->
    % Запис у транзієнтну Mnesia таблицю
    {reply, Proc};
action({serviceTask, 'DeliverToClient'}, Proc) ->
    % Відправка через TCP сокет
    {reply, Proc};
action(_, Proc) -> {reply, Proc}.
```

3.2.2. chat_nif.c (C99 NIF обгортка для декодування ASN.1 DER)

Код Erlang NIF модуля мовою C99 для швидкого синтаксичного розбору ASN.1 DER пакетів без Rust:

```
#include "erl_nif.h"
#include <string.h>

// Функція розбору ASN.1 TLV (Type-Length-Value) блоку
static ERL_NIF_TERM parse_tlv_nif(ErlNifEnv* env, int argc, const ERL_NIF_TERM argv[]) {
    ErlNifBinary bin;
    if (!enif_inspect_binary(env, argv[0], &bin)) {
        return enif_make_badarg(env);
    }

    if (bin.size < 2) {
        return enif_make_tuple2(env, enif_make_atom(env, "error"),
                                enif_make_string(env, "too_short", ERL_NIF_LATIN1));
    }

    unsigned char tag = bin.data[0];
    unsigned int length = bin.data[1];
    unsigned int header_len = 2;

    // Спрощена підтримка довгих довжин DER
    if (length & 0x80) {
        unsigned int num_bytes = length & 0x7F;
        if (bin.size < 2 + num_bytes) {
            return enif_make_tuple2(env, enif_make_atom(env, "error"),
                                    enif_make_string(env, "invalid_length", ERL_NIF_LATIN1));
        }
        length = 0;
        for (unsigned int i = 0; i < num_bytes; i++) {
            length = (length << 8) | bin.data[2 + i];
        }
        header_len = 2 + num_bytes;
    }

    if (bin.size < header_len + length) {
        return enif_make_tuple2(env, enif_make_atom(env, "error"),
                                enif_make_string(env, "payload_mismatch", ERL_NIF_LATIN1));
    }

    ERL_NIF_TERM type_term = enif_make_uint(env, tag);

    unsigned char* payload_data;
    ERL_NIF_TERM payload_term;
    payload_data = enif_make_new_binary(env, length, &payload_term);
    memcpy(payload_data, bin.data + header_len, length);

    return enif_make_tuple3(env, enif_make_atom(env, "ok"), type_term, payload_term);
}

static ErlNifFunc nif_funcs[] = {
```

```
    {"parse_tlv", 1, parse_tlv_nif}
};

ERL_NIF_INIT(chat_nif, nif_funcs, NULL, NULL, NULL, NULL)
```

4. Регламент взаємодії X.422 та криптографічні стандарти

Регламент взаємодії X.422 визначає дворівневий стандарт обміну:

4.1. X.422.1: Транспортний рівень

- Взаємодія між клієнтами та брокерами зв'язку CHAT v2 виконується через асинхронні WebSocket з'єднання, захищені двостороннім mTLS 1.3.
- Поштовий обмін MAIL v3 реалізує доставку підписаних S/MIME конвертів за стандартними захищеними SMTP/IMAP протоколами.
- Для випуску, оновлення та перевірки статусів сертифікатів використовуються стандарти EST (RFC 7030) та OCSP (RFC 6960).

4.2. X.422.2: Криптографічний конверт CMS

Всі текстові та медіа-повідомлення, а також поштові вкладення перед відправленням проходять обов'язкову криптографічну обробку на боці клієнта:

1. **Формування підпису (Signature)**: створюється CMS-структура SignedData. Обчислення хешу виконується за алгоритмом ДСТУ 7564 (Купина-384) або SHA-384. Електронний підпис обчислюється на еліптичних кривих за стандартом ДСТУ 4145-2002 або ECDSA.
2. **Шифрування вмісту (EnvelopedData)**: дані шифруються симетричним шифром ДСТУ 7624 (Калина-256) або AES-GCM-256 на сесійному ключі.
3. **Обмін сесійними ключами**: симетричний ключ шифрується на публічному ключі отримувача за алгоритмом еліптичних кривих ECDH (SECP-384 або SECP-571).

5. Вимоги до засобу за ISO/IEC/IEEE 29148

5.1. Вимоги до надійності та продуктивності

- Середній показник доступності (SLA) периферійного сервера «Комунікатор» не менше 99.9%.
- Швидкість обробки та маршрутизації повідомлень брокером CHAT v2 — не менше 10 000 повідомлень на секунду при затримці доставки не більше 15 мс.
- Пропускна здатність поштової підсистеми MAIL v3 — не менше 500 повідомлень на секунду.

5.2. Вимоги до безпеки та захисту інформації

- Незмінність журналів аудиту, автентифікації користувачів та відкликання сертифікатів.
- Заборона збереження та розшифрування закритого тексту (plaintext) повідомлень та поштових вкладень на серверних вузлах.