Relatorio - Laboratorio IoT Security

Alunos: Wagner Junior e Pedro Oliveira

Link do repositório com o código: Github

Execução das Etapas

Passamos por todas as etapas propostas no laboratorio:

1. MQTT funcionando com autenticacao:

Utilizamos Mosquitto Broker local, com usuario 'aluno' e senha 'senha123'. A BitDogLab se conecta com sucesso e publica no topico 'escola/sala1/temperatura'.

2. Publicação de mensagens JSON com timestamp: A BitDogLab gera uma string JSON contendo valor e ts (timestamp em epoch), por exemplo: {"valor":26.5,"ts":1716589200}

```
int len = snprintf(plain, sizeof plain,
                   "{\"valor\":26.5,\"ts\":%lu}", time(NULL));

// ✅ Publica JSON puro
mqtt_comm_publish(topic, (uint8_t*)plain, len);
```

3. Subscriber (PC) recebendo e validando: O programa subscriber.c foi implementado em C com a lib Paho. Ele: - Decifra (caso as mensagens estejam cifradas com XOR) - Faz o parse do JSON - Verifica se o timestamp e mais recente que o anterior (anti-replay)

```
static int msg_cb(void *ctx, char *topic, int tlen, MQTTClient_message *msg)
{
    char plain[64];

    /* 1. Copia payload puro */
    memcpy(plain, msg->payload, msg->payloadlen);
    plain[msg->payloadlen] = '\0';

    printf("Recebido: %s\n", plain);

    /* 2. Extrai valor e timestamp */
    float    valor;
    uint32_t ts;
    if (sscanf(plain, "{\"valor\":%f,\"ts\":%u}", &valor, &ts) != 2) {
        puts("Parse ERROR");
        goto done;
    }

    /* 3. Anti-replay */
    if (ts > ultima_ts) {
        ultima_ts = ts;
        printf("Nova leitura: %.1f  (ts=%u)\n", valor, ts);
    } else {
        printf("Replay detectado! ts=%u <= %u\n", ts, ultima_ts);
    }

done:
    MQTTClient_freeMessage(&msg);
    return 1;                   /* mensagem tratada */
}
```

4. Deteccao de Replay:

Mensagens antigas (com ts menor ou igual ao ultimo recebido) sao detectadas e ignoradas.

5. Testes com cifragem XOR: Foi implementado e testado o uso de cifra XOR com chave 42. Isso simula uma criptografia simples para evitar leitura direta do payload.

```
uint8_t enc[64];
const uint8_t XOR_KEY = 42;
xor_encrypt((uint8_t*)plain, enc, len, XOR_KEY);
mqtt_comm_publish(topic, enc, len);
```

Analise Critica: Tecnicas escalaveis

1. Publicacao em MQTT com autenticacao:

Altamente escalavel.

Pode ser usado em uma escola com varias BitDogLab publicando para topicos especificos.

2. JSON com timestamp:

Boa pratica padrao.

Facil de interpretar, leve, compativel com varias linguagens.

3. Verificacao de replay:

Escalavel em cenarios locais.

Requer controle do ultimo timestamp por dispositivo.

4. Cifra XOR:

Nao escalavel para seguranca real.

Deve ser substituida por TLS/SSL ou criptografia forte (como AES) em aplicacoes reais.

Como aplicar em uma rede escolar com varias BitDogLab

1. Topologia com um broker central (Mosquitto):

Todas as BitDogLab se conectam ao broker com IP fixo.

2. Divisao de topicos por ambiente:

- escola/sala1/temp
- escola/sala2/umidade
- escola/lab1/presenca

3. Autenticacao por dispositivo ou grupo:

Cada BitDogLab pode ter um usuario no broker com permissoes diferentes.

4. Central de coleta em um servidor/PC:

Subscriber coleta dados de varios topicos e armazena ou processa.

5. (Extra) Dashboard com Node-RED ou Grafana:

Visualizacao dos dados em tempo real.