ISTEC

NETGUARDIAN

CRDT

CRDT

01

COMO FUNCIONA O CRDT?

CRDT (Conflict-free Replicated Data Type) é uma estrutura de dados desenhada para ser replicada em múltiplos nós de um sistema distribuído, permitindo que todas as réplicas possam ser atualizadas de forma independente e simultânea, sem necessidade de coordenação imediata.

O sistema garante que, eventualmente, todas as réplicas convergem para o mesmo estado, mesmo que tenham recebido as atualizações em ordens diferentes.

O objetivo dos CRDTs é:

- **Evitar conflitos durante a replicação de dados.**
- **Garantir convergência automática.**
- **Permitir disponibilidade máxima mesmo em situações de partição de rede.**

CRDT

02

IMPLEMENTAÇÃO DO CRDT

Um CRDT num servidor com base de dados é uma forma de replicação inteligente, onde cada réplica (servidor ou nó) mantém uma cópia parcial ou total dos dados e consegue aceitar atualizações locais sem coordenação imediata com os outros nós.

- **Vários servidores / réplicas**

Cada servidor mantém uma cópia da base de dados ou de uma parte dela.
Exemplo: 3 nós — Lisboa, Porto, Faro.

- **Camada de sincronização (replicação CRDT)**

É responsável por propagar operações ou estados entre as réplicas.
Pode usar state-based (enviar estado) ou op-based (enviar operações)
.

- **Base de dados local**

Cada réplica armazena localmente o estado do CRDT (por exemplo, conjuntos, contadores, mapas, etc.).
Pode estar em memória, num ficheiro, ou num motor como Redis, RocksDB, PostgreSQL, etc.

- **Camada de rede / sincronização periódica**

Liga os servidores entre si (TCP, HTTP, gRPC, etc.).
Troca estados, merges ou operações de forma assíncrona.

CRDT

03

VANTAGENS DO PROTOCOLO CRDT

O protocolo CRDT permite que múltiplos servidores mantenham réplicas da mesma base de dados, aceitando atualizações locais de forma independente e sincronizando-as depois. Desta forma, mesmo em ambientes distribuídos ou com falhas de rede, todas as réplicas convergem automaticamente para o mesmo estado, sem necessidade de coordenação constante ou bloqueios complexos. Esta abordagem é ideal para sistemas distribuídos, colaborativos ou geograficamente dispersos, oferecendo disponibilidade, escalabilidade e rapidez na resposta aos utilizadores.
Principais vantagens:

- **Alta disponibilidade:** cada nó pode processar atualizações localmente, mesmo durante falhas de rede.
- **Consistência eventual garantida**: todas as réplicas convergem automaticamente para o mesmo estado.
- **Baixa latência para o utilizador:** operações locais permitem respostas rápidas.
- **Escalabilidade:** fácil adicionar novos servidores distribuídos geograficamente.
- **Simplicidade na gestão de conflitos**: merges automáticos evitam bloqueios ou protocolos de consenso complexos.

04

FRAQUEZAS DO PROTOCOLO CRDT

O CRDT não é um mecanismo de segurança, mas sim um método de replicação e consistência de dados.

O seu objetivo é garantir que várias réplicas de um sistema distribuído convergem automaticamente para o mesmo estado, mesmo que as atualizações aconteçam de forma independente ou em momentos diferentes.

Contudo, o CRDT não protege os dados nem controla quem pode aceder ou alterar informação. Ele apenas assegura que, depois das alterações serem propagadas, todos os nós têm a mesma versão dos dados — não que essas alterações sejam seguras, legítimas ou confidenciais.

Pontos principais:

- **Não fornece autenticação nem encriptação**: não impede que utilizadores não autorizados enviem atualizações.
- **Foca-se na consistência, não na proteção**: o objetivo é resolver conflitos, não proteger contra ataques.
- **Não lida com falhas maliciosas:** assume que todas as réplicas são confiáveis.
- **Deve ser combinado com camadas de segurança:** como HTTPS, autenticação, controlo de acesso e encriptação de dados.
- **Garantia funcional, não de segurança**: assegura convergência lógica, não integridade ou confidencialidade.

CRDT

05

EXPLOITS E SUAS MITIGAÇÕES

1) Alterações não autorizadas / falta de autenticação

O que é: se qualquer cliente ou nó puder submeter operações, alguém não autorizado pode inserir dados, corromper registos ou provocar estados inválidos.

Mitigações: autenticação forte (mTLS, tokens com rotação), autorização por recurso/operação, RBAC/ABAC, assinaturas das operações (op-signed), validação de origem.

2) Nós maliciosos / comportamento bizantino

O que é: réplicas comprometidas que enviam operações inválidas ou maliciosas; CRDTs típicos assumem réplicas honestas, logo isto pode corromper o estado.

Mitigações: limitar quem pode ser nó replicante; usar assinaturas/ATTESTATION das operações; auditoria e verificação de operações; modelos híbridos que combinam CRDT com quorum/consensus para operações críticas; monitorização de anomalias.

06

EXPLOITS E SUAS MITIGAÇÕES

3) Replay e adulteração de mensagens / falta de integridade

O que é: mensagens de operações ou estados podem ser interceptadas e re-enviadas ou alteradas por um atacante na rede.

Mitigações: canais cifrados e autenticados (TLS/mTLS), uso de nonces ou contadores, assinatura de operações, e detecção de duplicados idempotente combinada com limites de taxa.

4) Exaustão de recursos (DoS / amplification)

O que é: um atacante gera muitas operações que fazem crescer metadados (tombstones, vectores), enchem a fila de sincronização ou consomem CPU/IO.

Mitigações: rate-limiting por origem, quotas por cliente/ nó, políticas de garbage collection e compactação, validação de carga, limites no tamanho de mensagens/deltas, circuit breakers.

07

EXPLOITS E SUAS MITIGAÇÕES

5) Crescimento de metadados e GC abuse

O que é: operações concebidas para inflar vetores/identificadores/tombstones para degradar performance e armazenamento.

Mitigações: políticas de limitação de histórico por conta, compactação periódica e checkpoints com validação, validação semântica das operações, mecanismos seguros de garbage collection com quorum ou checkpoints assinados.

6) Injeção via API / validação insuficiente

O que é: entradas inválidas ou maliciosas enviadas através das APIs que são incorporadas no estado CRDT e depois propagadas.

Mitigações: validação estrita de input, schemas, saneamento, limites de tamanho, permissões por operação; testes de fuzzing controlado no endpoint.

08

EXPLOITS E SUAS MITIGAÇÕES

7) Vazamento de metadados / privacidade

O que é: metadados CRDT (identificadores, vectores de versão, timestamps) podem revelar topologia, actividade de utilizadores ou padrões sensíveis**.**

Mitigações: minimalizar metadados expostos, encriptação at-rest e in-transit, políticas de retenção, anonymization/aggregation, revisão de dados enviados a terceiros.

8) Ataques de sincronização (manipular ordem/latência)

O que é: manipular sincronização para forçar estados intermediários ou explorar políticas de merge (por exemplo, forçar wins específicos).

Mitigações: políticas de merge bem definidas e auditáveis; assinaturas temporais; monitorização de padrões de sincronização anómalos.

CRDT

09
FUTURAMENTE:

Escolher fraquezas e ver o impacto delas sem as respetivas mitigações (versão teste, própria para explorar vulnerabilidades).