11장. 보안

> 데이터를 더욱 더 안전하게 관리하기 위해 레디스는 어떤 기능을 제공할까요?

커넥션 제어

bind

레디스 서버는 여러 개 ip를 가질 수 있다.

bind 설정을 통해 레디스가 여러 ip 중 어떤 ip를 통해 들어오는 연결을 받을 것인지 지정한다.

- **기본값: 127.0.0.1**
  - 기본값 지정 시 레디스는 오직 동일한 서버 내 연결만 허용한다.
  - `0.0.0.0` or `*`로 설정 시 모든 연결을 허용한다.

외부 인터넷에 노출 && 운영 목적으로 사용 시 bind를 특정 IP로 설정해서 의도하지 않은 연결을 방지하자.

패스워드

1. 노드에 직접 패스워드 지정
2. ACL 기능 사용 (버전 6.0 ~)

Protected mode

레디스를 운영 용도로 사용한다면 설정하는 것을 권장

- **기본값: yes**
  - 패스워스 미설정 시 로컬에서 들어오는 연결만 허용 (bind 설정 무시)

커맨드 제어

CONFIG SET

- 중요한 설정값 제어
- 설정값 읽기, 파라미터 재설정 등

주의! 레디스에 접근할 수 있는 모든 클라이언트가 레디스 인스턴스를 제어할 수 있다 ?!

커맨드 이름 변경

redis.conf 파일에서 변경, 실행 중 동적 변경 불가능

```
rename-command CONFIG CONFIG_NEW
```

- 커맨드를 다른 이름으로 변경
- 커맨드 비활성화
  - `rename-command CONFIG “”` –> CONFIG 비활성화
- 커맨드 커스터마이징, 보안 강화

주의! 센티널이 사용하는 커맨드의 경우 sentinel.conf에서도 변경해줘야 한다.

커맨드 실행 환경 제어

레디스가 악성 공격에 노출될 가능성을 줄이기 위해 도입

```
enable-protected-configs no
enable-debug-command no
enable-module-command no
```

- **no**: 모든 연결에 대해서 명령어의 수행 차단
- **yes**: 모든 연결에 대해서 명령어의 수행 허용
- **local**: 로컬 연결에 대해서만 명령어의 수행 허용

결론

- 보안 강화를 위해 `protected-mode`를 `yes`로 설정하자.
- 패스워드를 설정하자.

- 사용하지 않을거면, `enable-protected-configs` 옵션을 `local` or `no`로 설정하자.
- 외부에서 레디스의 중요한 설정 파일을 변경하는 것을 막을 수 있다.

ACL

유저라는 개념을 도입해 유저별로 실행 가능한 커맨드와 접근 가능한 키를 제한하는 기능

```
> ACL SETUSER meun on >password ~cached:* &* +@all -@dangerou
OK
```

- **ACL SETUSER *meun***: *meun*이라는 이름을 가진 유저 생성
- **on**: 활성 상태 유저
- ***>password:*** *password*라는 패스워드 설정
- ***~cached*:***: *cached:* 프리픽스를 가진 모든 키에 대해 접근 가능
- **+@all**: 모든 커맨드 사용 가능
- **-@dangerous**: 위험한 커맨드 사용 불가

유저 생성/삭제

- **ACL SETUSER**: 유저 생성
- **ACL DELUSER**: 유저 삭제
- **ACL GETUSER {user}**: 특정 유저 확인

유저 상태 제어

- 유저 비활성화 시(off) 이미 접속해 있는 유저의 연결은 여전히 유지된다.

패스워드

- 패스워드는 1개 이상 지정할 수 있다.
- 패스워드를 지정하지 않으면 유저에 접근할 수 없다.
- `nopass` 권한을 부여하면 패스워드 없이 접근할 수 있다.

- 기존 유저에 `nopass` 권한을 부여하면 설정된 모든 패스워드가 삭제된다.
- `resetpass` 권한을 부여하면 패스워드를 지정하거나 `nopass` 권한 부여 전까지 접근할 수 없다.
- `ACL`을 이용해 패스워드를 저장하면 내부적으로 SHA256 방식으로 암호화된다.
  - `requirepass`는 평문 저장🤣, 그리고 `CONFIG SET requirepass`로 누구나 확인 가능
  - `ACL GENPASS`로 난수를 생성하는 등 예측할 수 없는 복잡한 패스워드를 사용하자

커맨드 권한 제어

1. 개별 커맨드 제어
2. 그룹화된 카테고리로 제어
   - 카테고리에 포함된 상세 커맨드 확인하기: `ACL CAT {카테고리명}`
3. 서브 커맨드 제어

주요 카테고리

- **dangerous**: 아무나 사용하면 위험할 수 있는 커맨드
- **admin**: dangerous - 장애 유발 커맨드
- **fast**: O(1)로 수행되는 커맨드
- **slow**: fast에 속하지 않은 커맨드
- **keyspace**: 키와 관련된 커맨드
- **read**: 데이터를 읽어오는 커맨드
- **write**: 데이터를 쓰는 커맨드

키 접근 제어

- `~*` or `allkeys`: 모든 키에 대한 접근이 가능
- `~<pattern>`: 접근 가능한 키 정의

키에 대한 읽기/쓰기 권한 나누기 (버전 7 ~)

- `% R ~<pattern>`: 읽기 권한
- `% W ~<pattern>`: 쓰기 권한

- `% RW ~<pattern>`: 읽기/쓰기 권한 (~<pattern> 과 동일)

```
# loguser에게 log: 프리픽스 모든 권한, mail:, sms: 프리픽스 읽기 권한
ACL SETUSER loguser ~log:* %R~mail:* %R~sms:*
```

셀렉터

다른 읽기 커맨드가 아닌 오직 GET 커맨드만 사용하도록 강제하고 싶다면 ?

```
ACT SETUSER loguser resetkeys ~log:* (+GET ~mail:*)
```

괄호 안에 셀렉터를 정의할 수 있다.

유저 초기화

`reset` 커맨드로 모든 권한을 회수하고 기본 상태로 변경할 수 있다.

`ACL SETUSER`를 한 직후와 동일해진다.

ACL 규칙 파일로 관리하기

```
# redis.conf
aclfile /etc/redis/users.acl

# 유저 데이터 레디스로 로드
ACL LOAD
# 유저 데이터 저장
ACL SAVE
```

ACL 데이터는 어디에 저장되든 형태는 동일하되 위치가 달라질 뿐이다.

- 따로 사용하면 `CONFIG REWRITE`를 통해 ACL 정보를 저장할 수 없다.
- 대신 `ACL LOAD`, `ACL SAVE`로 로드/저장이 가능하기 때문에 운영 시 유용하다.

SSL/TLS

버전 6부터 SSL/TLS를 이용한 보안 연결을 지원한다.

역할

- 데이터 전송 과정에서 정보를 암호화
- 중간에 데이터가 노출되거나 조작되는 것을 방지
- 핸드셰이크 과정으로 상호 인증을 진행해 두 통신 당사자가 모두 신뢰할 수 있는지 확인
- 암호화 기술과 인증서로 통신의 무결성과 기밀성을 확보
- 데이터 전송 과정을 암호화하여 보안 위험을 감소시킨다.

레디스에서 사용하기

처음 빌드할 때부터 정의해야 한다.

```
make BUILD_TLS=yes
```

- 레디스 인스턴스와 클라이언트 간 동일한 인증서를 사용한다.
- key, cert, ca-cert파일은 레디스를 실행할 클라이언트에 동일하게 복사해둬야 한다.

```
# redis.conf
tls-port <포트 번호> # SSL/TLS 연결 사용
tls-cert-file /path/to/redis.crt
tls-key-file /path/to/redis.key
tls-ca-cert-file /path/to/ca.crt
```

`port 0`을 명시에 기본 포트를 비활성화하여 SSL/TLS 사용 없이 레디스에 접근할 수 없도록 할 수 있다.

HA 구성

복제 구성

복제본도 마스터와 동일하게 설정을 추가한다.

```
tls-port <포트 번호>

tls-replication yes # 기본값 no
```

```
tls-cert-file /path/to/redis.crt
tls-key-file /path/to/redis.key
tls-ca-cert-file /path/to/ca.crt
```

센티널 구성

센티널도 동일하게 설정을 추가한다.

```
# sentinel.conf
tls-port <포트 번호>

tls-replication yes # 기본값 no

tls-cert-file /path/to/redis.crt
tls-key-file /path/to/redis.key
tls-ca-cert-file /path/to/ca.crt
```

클러스터 구성

```
tls-port <포트 번호>

tls-replication yes # 기본값 no

tls-cluster yes

tls-cert-file /path/to/redis.crt
tls-key-file /path/to/redis.key
tls-ca-cert-file /path/to/ca.crt
```