# Redis Enumeration - Complete Theory & Practice Notes

## Introduction to Databases & Redis

### Database Fundamentals

**Databases** are collections of organized information that can be easily accessed, managed and updated. In most environments, database systems are critical because they communicate information related to:

- Sales transactions
- Product inventory
- Customer profiles
- Marketing activities

### Types of Databases

There are different types of databases, and **Redis** is an **'in-memory' database**.

**In-memory databases**:

- Rely essentially on primary memory (RAM) for data storage
- Database is managed in the RAM of the system
- Contrast to databases that store data on disk or SSDs
- Primary memory is significantly faster than secondary memory
- Data retrieval time is very small, offering efficient & minimal response times

### Redis Usage Scenarios

**Common Implementation Pattern:**

1. Website needs to display prices on front page
2. Website first checks if prices are in Redis cache
3. If not found, checks traditional database (MySQL/MongoDB)
4. When value is loaded from database, it's stored in Redis for shorter period (seconds/minutes/hours)
5. Handles similar requests during that timeframe from cache

**Benefits for High-Traffic Sites:**

- Much faster retrieval for majority of requests
- Stable long-term storage in main database
- Reduced load on primary database

## What is Redis?

**Redis (REmote DIctionary Server)** is an open-source advanced NoSQL key-value data store used as:

- **Database**: Primary data storage
- **Cache**: Temporary fast storage
- **Message broker**: Communication between services

### Core Architecture

#### Server Component

- Redis runs as server-side software
- Core functionality is in server component
- Server listens for connections from clients
- Accepts connections programmatically or through CLI

#### Database Storage

- Data stored in dictionary format with key-value pairs
- Database stored in server's RAM for fast data access
- Redis writes database contents to disk at varying intervals
- Provides persistence as backup in case of failure
- Typically used for short-term storage of data needing fast retrieval

#### The CLI (Command-Line Interface)

- Powerful tool giving complete access to Redis data and functionalities
- Essential for developing software/tools that interact with Redis
- Provides direct database manipulation capabilities

## Enumeration Methodology

### Phase 1: Target Verification

**Connectivity Check**: Verify target availability and connection quality

```bash
ping <target_IP>
# Run until satisfied with connection (2-3 replies sufficient)
# Sometimes getting snippet/overview is more time-efficient than detailed reports
```

### Phase 2: Network Reconnaissance

**Port Scanning**: Identify open services and versions

```bash
nmap -p- -sV <target_IP>
# -sV: Probe open ports to determine service/version info
```

#### Expected Redis Indicators:

- Port 6379/tcp open
- Service: Redis key-value store
- Version information displayed

### Phase 3: Redis-Specific Enumeration

#### Tool Installation

```bash
# Install redis-cli utility
apt install redis-tools
# Alternative: netcat can connect but redis-cli is more convenient
```

#### Connection Establishment

```bash
redis-cli -h <target_IP>
# Should display prompt: target_IP:6379>
```

## Redis CLI Reference

### Essential Connection Options

- `-h <hostname>`: Server hostname (default: 127.0.0.1)
- `-p <port>`: Server port (default: 6379)
- `-t <timeout>`: Server connection timeout in seconds
- `-a <password>`: Password for authentication
- `--user <username>`: ACL style authentication with username
- `-u <uri>`: Server URI format: redis://user:password@host:port/dbnum
- `-n <db>`: Database number to select
- `-c`: Enable cluster mode

- `--tls`: Establish secure TLS connection

### Advanced Options

- `-r <repeat>`: Execute specified command N times
- `-i <interval>`: Wait interval between repeated commands
- `--raw`: Use raw formatting for replies
- `--csv`: Output in CSV format
- `--json`: Output in JSON format
- `--scan`: List all keys using SCAN command
- `--pattern <pat>`: Key pattern for scanning operations
- `--bigkeys`: Sample keys looking for large elements
- `--hotkeys`: Sample keys looking for frequently accessed keys

## Core Redis Commands & Theory

### Information Gathering

```bash
info
```

**Purpose**: Returns comprehensive information and statistics about Redis server **Output Sections**:

- **Server**: Version, OS, configuration files, process info
- **Clients**: Connected clients, buffer information
- **Memory**: Usage statistics, fragmentation, allocator info
- **Persistence**: RDB and AOF status, backup information
- **Stats**: Connection counts, command processing, network I/O
- **Replication**: Master/slave configuration
- **CPU**: System and user CPU usage
- **Keyspace**: Database statistics including key counts

### Database Selection & Navigation

```bash
select <database_number>
```

**Theory**: Redis supports multiple logical databases (default 16, numbered 0-15) **Usage**: Switch between databases for organization/isolation

### Key Discovery

```bash
keys *
keys <pattern>
```

**Theory**:

- Lists keys matching pattern (* = all keys)
- Can be slow on large databases (use SCAN for production)
- Pattern matching supports glob-style patterns

### Data Retrieval

```bash
get <key_name>
type <key_name>
dbsize
```

**Functions**:

- `get`: Retrieve string value associated with key
- `type`: Determine data type (string, list, set, hash, etc.)
- `dbsize`: Count total keys in current database

## Practical Enumeration Example

### Target Analysis

**Network Scan Results:**

```
$ nmap -p- -sV 10.129.136.187
PORT     STATE SERVICE VERSION
6379/tcp open  redis   Redis key-value store 5.0.7
```

**Interpretation**:

- Only port 6379 open (typical Redis configuration)
- Redis version 5.0.7 identified
- No additional services running

## Connection & Initial Reconnaissance

bash

```
redis-cli -h 10.129.136.187
10.129.136.187:6379>
```

## Server Information Analysis

bash

```
10.129.136.187:6379> info
# Server
redis_version:5.0.7
os:Linux 5.4.0-77-generic x86_64
tcp_port:6379
config_file:/etc/redis/redis.conf

# Memory
used_memory_human:839.48K
total_system_memory_human:1.94G

# Keyspace
db0:keys=4,expires=0,avg_ttl=0
```

**Key Findings**:

- Redis 5.0.7 on Linux system
- Low memory usage (839KB) indicates small dataset
- Database 0 contains 4 keys with no expiration set
- No authentication required (security issue)

## Database Exploration

bash

```
10.129.136.187:6379> select 0  # Select database 0
10.129.136.187:6379> keys *     # List all keys
1) "flag"
2) "stor"
3) "numb"
4) "temp"

10.129.136.187:6379> get flag   # Retrieve flag value
"03e1d2b376c37ab3f5319922053953eb"
```

## Security Implications & Vulnerabilities

### Common Redis Security Issues

1. **No Authentication**: Default installations often lack password protection
2. **Network Exposure**: Redis accessible from internet without firewall
3. **Privilege Escalation**: Redis running with elevated privileges
4. **Data Exposure**: Sensitive information stored in plaintext
5. **Command Injection**: Unsafe handling of user input in applications

### Enumeration Indicators

- Direct connection without credentials
- `info` command execution successful
- Access to key listing and data retrieval
- Server configuration details exposed

### Real-World Attack Scenarios

- **Data Exfiltration**: Extracting cached user sessions, API keys
- **Configuration Theft**: Application settings, database credentials
- **Cache Poisoning**: Injecting malicious data into cache
- **Lateral Movement**: Using Redis as pivot point in network

## Enumeration Best Practices

### Systematic Approach

1. **Start with `info`**: Comprehensive server overview
2. **Analyze keyspace**: Identify available databases and key counts
3. **Database enumeration**: Check each database systematically
4. **Pattern-based searching**: Use specific patterns for targeted discovery
5. **Data type analysis**: Understand data structures before extraction

### Performance Considerations

- **Large databases**: Use `SCAN` instead of `KEYS *`
- **Pattern matching**: Specific patterns reduce response time
- **Batch operations**: Group related commands for efficiency

### Documentation

- Record Redis version and configuration
- Document database structure and key naming conventions
- Note security configurations and access controls
- Catalog sensitive data discovered

## Advanced Enumeration Techniques

### Database Mapping

```bash
info keyspace           # Quick database overview
scan 0 match *pattern*  # Efficient key discovery
```

### Data Structure Analysis

```bash
type <key>              # Identify data type
strlen <key>            # String length
llen <key>              # List length
hlen <key>              # Hash field count
```

### Configuration Analysis

```bash
config get "*"          # Retrieve configuration parameters
client list             # Show connected clients
```

This comprehensive approach ensures thorough Redis enumeration while understanding the underlying technology and security implications.