きざまれし証
解　説

配布されたファイルを解凍しデータベースファイルを取り出します。

```
┌──(kali㉿kali)-[~/challange]
└─$ unzip secret.zip
Archive:  secret.zip
  inflating: secret.db

┌──(kali㉿kali)-[~/challange]
└─$ ls -lh
total 2.3M
-rw-r--r-- 1 kali kali 1.6M Nov  3 23:26 secret.db
-rw-rw-r-- 1 kali kali 722K Nov  4 01:34 secret.zip
```

次にfileコマンドでファイルの種類を特定します。

```
┌──(kali㉿kali)-[~/challange]
└─$ file secret.db
secret.db: SQLite 3.x database, last written using SQLite version 3046001, file counter 8, database pages 390, 1st free page 143, free pages 3, cookie 0×5, schem
a 4, UTF-8, version-valid-for 8
```

fileコマンドによると、SQLite3データベースファイルであることがわかりました。

sqlite3 コマンドでデータベースが開けるか確認します。具体的には、sqlite3 でデータベースを開いた後、テーブル一覧を表示します。

```
┌──(kali㉿kali)-[~/challange]
└─$ sqlite3 secret.db
SQLite version 3.46.1 2024-08-13 09:16:08
Enter ".help" for usage hints.
sqlite> .tables
files     logs      messages  sessions  users
sqlite>
```

表示されました。files, logs, messages, sessions, usersテーブルが確認できます。それぞれSELECT * FROM table_name LIMIT 5;にて5件取得してみます。table_nameをそれぞれのテーブル名に変えることで、最初の5件を取得できます。

```
sqlite> SELECT * FROM files LIMIT 5;
1|SYC7I3IwU1Bzl63.zip|/uploads/Engineering/SYC7I3IwU1Bzl63.zip|5771144|image/png|55|2024-02-14 04:13:00|2024-02-14 04:13:00
2|MujtzdzFUojynK2.docx|/uploads/Operations/MujtzdzFUojynK2.docx|5318815|application/vnd.openxmlformats-officedocument.wordprocessingml.document|351|2024-01-14 13
:46:00|2024-01-14 13:46:00
3|XGTcOjS8QgQfRTD.docx|/uploads/Security/XGTcOjS8QgQfRTD.docx|8408787|application/zip|15|2024-02-14 00:08:00|2024-02-14 00:08:00
4|whPOjcGmPlcYu9B.xlsx|/uploads/Operations/whPOjcGmPlcYu9B.xlsx|3496984|text/plain|220|2024-01-22 07:44:00|2024-01-22 07:44:00
5|aiNiYOSrp75BM2m.pdf|/uploads/Marketing/aiNiYOSrp75BM2m.pdf|4444659|application/vnd.openxmlformats-officedocument.wordprocessingml.document|422|2024-01-06 16:50
:00|2024-01-06 16:50:00
sqlite> SELECT * FROM logs LIMIT 5;
1|412|message_send|105.192.189.118|2024-02-06 04:04:00|Action details: ka3peMR8iwOUh3EXT8uw5KunSJRDyjvmwbxCLg1vRrCJF1e4Jc
2|426|file_download|231.12.152.166|2024-01-12 05:39:00|Action details: yV4VI9eFnj6EQgiWcAqZwZ28PQ7GtlUzBZfPv7zJOVliXtcn4D
3|163|file_upload|202.244.146.25|2024-01-16 13:01:00|Action details: Y9XHL4eIDtqpmDlNwpIhXNu2MlX1geDq9u5n0svKaFpRWW5A2X
4|208|logout|204.23.160.199|2024-02-11 14:41:00|Action details: UbXPxMZm8rNj7DUDeSoZSnS3lAm6aSWwFBDqdbWAwSqgF95zeJ
5|300|profile_update|195.1.21.203|2024-01-02 12:38:00|Action details: EQCFvUfyWUHgxSKUfrqeL4PDnPcwD7w0sE8n2W0Hn8khEvL1L4
sqlite> SELECT * FROM messages LIMIT 5;
1|hfxenobk@test.org|cmvz53n6@demo.net|Schedule Change - 8sMuq1QtSI|This is a normal message with content: EqZtLoD8mLYRhCwFednfe52N2j0zHyYhN8X6o1hqf4JKcErODs0NX3I
55S2yruVOQvWWpG1XXuvMv9PywXljYKhQyKjztnEQYRFd|2024-02-22 19:08:00|3|0
2|itka5t1s@test.org|8tjkrcn7@test.org|Project Update - 4r1jbVmmPJ|This is a normal message with content: 5pwxgSpxHF1kCft532NZjEpPY9W9je0rCM5Oy4OOV7Rk9xrqZYbLseEm
ggNvFhdBGQNjiysIS2UP0upxSJtPaIJG9w96PBFxMxvV|2024-02-27 06:01:00|2|0
3|jpdel7mv@company.com|gefnndzl@example.com|Weekly Report - qDx10ge3xc|This is a normal message with content: GjaN1DtZjLv0SI3tVxoMWuHSuxxGXAHz0pDDNtBgZc8Lw3pd4bS
qU8s7j3qXAU5GVmkpPPG9j9cVWoeRT5SFcURewQxyVpQTYNeJ|2024-01-19 06:31:00|1|1
4|gej9rycs@company.com|uqlw22li@company.com|Question about ... - 3NJu97Xjcf|This is a normal message with content: QhtFMuq45tNrIviG5DiB5rAOyI7YKhF7UQ5xsS1sjeZLTbG
WK5CARkZRu93PnlJz2Z0RApRF1DUWl8VoM991yqV339pF37C40dZK|2024-02-12 22:48:00|2|1
5|obf6drnq@example.com|s2htk8oq@example.com|Budget Review - qqRGN5xgd3|This is a normal message with content: bnqcq9M1HyrWPdplMR5wJCv77vP8cpKX657vFeqbjTDsg8Keqve
m6lANYI5jpF2BRGvOSJg0KuQY0MyEZDAwsaCMQBrMnhBvBFr9|2024-02-21 04:17:00|0|1
sqlite> SELECT * FROM sessions LIMIT 5;
1|200|2vXxrt23RtfdZipALqFFbSjzPfs7C5ocGxebnYVvsxdfAQ1IhMZ8i3tMUXm3sFe4|236.17.235.135|Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36|2024-01-25 20:19:00|2024
-01-26 13:19:00
2|63|EECYEPONjCjb79hcIPDVgMqH4QIR87aWtY4sSpKvWnJhAAL2K04zVu45oJ7mWp0O|173.91.112.103|Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36|2024-01-0
7 09:57:00|2024-01-08 00:57:00
3|261|m7dElEBJafV6VKAAE4sZ3C0UBzxQ1ufHrkAbvd4JcWpLVc55MK2nqZfVPotaNVYj|149.216.186.21|Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36|2024-03-
08 06:09:00|2024-03-08 10:09:00
4|93|ZKrBjeNE9VD91lzwGmKUbWsyOzfoVAP6xTf9bnzbOsx8FlTCiRq9tB1NwFCMsqkm|250.160.217.105|Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36|2024-01-05 00:
18:00|2024-01-05 14:18:00
5|357|o5BPSe2lbu1HlHUrwrQT4DRUQwB8VYgDRWpIdJowVNOwAn3lImQzHsn4SeTwHVwk|197.192.210.123|Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36|2024-03
-04 06:04:00|2024-03-04 07:04:00
sqlite> SELECT * FROM users LIMIT 5;
1|x8y5s0wn|x8y5s0wn@demo.net|NKSdJw KmtHnulO|Engineering|2023-09-11 07:51:00|2023-09-11 07:51:00
2|ex13xx0l|ex13xx0l@demo.net|QpwuFa Lc3Z4eCc|HR|2023-12-26 23:40:00|2023-12-26 23:40:00
3|wfh4zxjb|wfh4zxjb@example.com|CBIdwi z10l5Bb4|HR|2023-09-16 08:54:00|2023-09-16 08:54:00
4|zwbeh2tg|zwbeh2tg@mail.com|RX0BGu 8rnY4ohY|Sales|2023-02-03 01:24:00|2023-02-03 01:24:00
5|tnd2wq4h|tnd2wq4h@test.org|Eloe12 vARglDkD|Sales|2023-09-14 16:34:00|2023-09-14 16:34:00
sqlite>
```

flagに繋がる文字列は見つかりませんでした。テーブルのスキーマ(カラム名)を確認します。それぞれのテーブルのスキーマを確認する場合は、.schema table_name コマンドで確認ができます。
それぞれ確認しますが、例として、messagesのテーブルを指定した際の表示を確認します。

```
sqlite> .schema messages
CREATE TABLE messages (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                sender TEXT NOT NULL,
                recipient TEXT NOT NULL,
                subject TEXT NOT NULL,
                message TEXT NOT NULL,
                timestamp TEXT NOT NULL,
                priority INTEGER DEFAULT 0,
                read_status BOOLEAN DEFAULT 0
        );
```

先程取得した5件を確認すると、
messagesテーブルの中でも
messageカラムの文字列が一番多い
ため、messageカラムに対して、
flag文字列を検索してみます。

```
sqlite> SELECT * FROM messages WHERE message LIKE '%flag%';
sqlite>
```

現存するレコードには平文のフラグが
含まれていないことが確認できます。
同じようにuserテーブルのfull_name,
logsテーブルのdetails, filesテーブル
のfilenameにも試してみます。

```
sqlite> SELECT * FROM users WHERE full_name LIKE '%flag%';
sqlite> SELECT * FROM logs WHERE details LIKE '%flag%';
sqlite> SELECT * FROM files WHERE filename LIKE '%flag%';
sqlite>
```

同様に平文のフラグは含まれていないことが確認できました。SQLクエリでフラグが発見出来なかったため、データベースファイルをバイナリとして直接解析します。stringsコマンドを使用してファイル内のテキストデータを抽出します。

まず、平文のフラグを検索します。

```
┌──(kali㉿kali)-[~/challange]
└─$ strings secret.db | grep "flag"

┌──(kali㉿kali)-[~/challange]
└─$
```

平文のフラグはバイナリ内にも存在しないことが確認できます。

配布されたファイル名がsecret.dbのため、secretという文字列を検索してみます。

```
┌──(kali㉿kali)-[~/challange]
└─$ strings secret.db | grep -i "secret"
        commandh@secret.govagent1007@secret.govMission Update #8CLASSIFIED COMMUNICATION - LEVEL 8
        commando@secret.govagent1014@secret.govMission Update #15CLASSIFIED COMMUNICATION - LEVEL 15
        director@nso.secret.govagent007@verified.govCRITICAL - Mission CodeTOP SECRET COMMUNICATION - CLASSIFIED
k3              commandl@secret.govagent1011@secret.govMission Update #12CLASSIFIED COMMUNICATION - LEVEL 12
k3              commandk@secret.govagent1010@secret.govMission Update #11CLASSIFIED COMMUNICATION - LEVEL 11
k3              commandj@secret.govagent1009@secret.govMission Update #10CLASSIFIED COMMUNICATION - LEVEL 10
        commandi@secret.govagent1008@secret.govMission Update #9CLASSIFIED COMMUNICATION - LEVEL 9
        commandh@secret.govagent1007@secret.govMission Update #8CLASSIFIED COMMUNICATION - LEVEL 8
        commando@secret.govagent1014@secret.govMission Update #15CLASSIFIED COMMUNICATION - LEVEL 15
        commandn@secret.govagent1013@secret.govMission Update #14CLASSIFIED COMMUNICATION - LEVEL 13
k3              commandm@secret.govagent1012@secret.govMission Update #13CLASSIFIED COMMUNICATION - LEVEL 13
        director@nso.secret.govagent007@verified.govCRITICAL - Mission CodeTOP SECRET COMMUNICATION - CLASSIFIED
```

マッチしました。前後10行を確認してみます。

```
┌──(kali㉿kali)-[~/challange]
└─$ strings secret.db | grep -10 -i "secret"
oqaf701y@demo.netoaqwpay9@test.orgBudget Revi
QQmZTVfpNzMOfYHjxFktnY16cfHA6zbG7bnFKw2024-01
#3                    rq56aqfk@company.comwu20jshy@
I8Ura0vkEyunkBVj3rYaspuGDCMZ0wPNZACFSIszA3OUg
yw29a3zc@mail.com9auvtckz@example.comQuestion
```

文字列を確認してみると、
base64(最後の文字が=になっている文字列)らしき文字列が確認できます。

```
Your encoded access code is: c2FtcGxle3IzZF9oM3JyMW5nX2Q0dDR9
This is a test communication. Please disregard.
Mission parameters are subject to change.
Decode using standard Base64 protocol.
MuHJdTd01sLixiZJ74mkzq0LgOFr3rXJUsu8Dow1YLzLsgTCp9E7tboU3mgRqG
nfSTQv3Q3td8YHFYoItqlJwwEh1ftOIZ05qg5BcmCpbYCDVNentnkDbjnFp9mp
[TRANSMISSION ENDS]
2024-01-18 00:00:00
        commando@secret.govagent1014@secret.govMission Update
From: Command Center O
To: Agent 1014
Subject: Mission Update #15
Agent,
Your encoded access code is: bnVsbHsxbnY0bDFkXzRjYzNzc30=
```

「Your encoded access code is」から始まる行にbase64でエンコードされたらしき文字列が確認できます。

その行のみを抜き出してみます。

```
┌──(kali㉿kali)-[~/challange]
└─$ strings secret.db | grep "Your encoded access code is"
Your encoded access code is: c2FtcGxle3IzZF9oM3JyMW5nX2Q0dDR9
Your encoded access code is: bnVsbHsxbnY0bDFkXzRjYzNzc30=
Your encoded access code is: b2xkezN4cDFyM2RfZDR0NF94MDF9
Your encoded access code is: dGVtcHt0M21wMHI0cnlfYzBkM30=
Your encoded access code is: YXJjaGl2ZXtjNG5jM2xsM2RfMHB9
Your encoded access code is: YmFja3VwezBiczBsM3QzX2MwZDNfMDFkfQ==
Your encoded access code is: c2FtcGxle3IzZF9oM3JyMW5nX2Q0dDR9
Your encoded access code is: bnVsbHsxbnY0bDFkXzRjYzNzc30=
Your encoded access code is: dm9pZHtuMHRfNGN0aXYzX2MwZDN9
Your encoded access code is: bGVnYWN5ezB1dGQ0dDNkXzFuZjB9
```

正規表現にて文字列のみを抜き出してみます。

```
┌──(kali㉿kali)-[~/challange]
└─$ strings secret.db | grep "Your encoded access code is" | grep -Eo ': [^$]+' | grep -Eo '[^ :$]+'
c2FtcGxle3IzZF9oM3JyMW5nX2Q0dDR9
bnVsbHsxbnY0bDFkXzRjYzNzc30=
b2xkezN4cDFyM2RfZDR0NF94MDF9
dGVtcHt0M21wMHI0cnlfYzBkM30=
YXJjaGl2ZXtjNG5jM2xsM2RfMHB9
YmFja3VwezBiczBsM3QzX2MwZDNfMDFkfQ==
c2FtcGxle3IzZF9oM3JyMW5nX2Q0dDR9
bnVsbHsxbnY0bDFkXzRjYzNzc30=
dm9pZHtuMHRfNGN0aXYzX2MwZDN9
bGVnYWN5ezB1dGQ0dDNkXzFuZjB9
```

うまく表示されました。パイプライン
で、base64デコードを進めてみます。

```
┌──(kali㉿kali)-[~/challange]
└─$ strings secret.db | grep "Your encoded access code is" | grep -Eo ': [^$]+' | grep -Eo '[^ :$]+' | base64 -d
sample{r3d_h3rr1ng_d4t4}null{1nv4l1d_4cc3ss}old{3xp1r3d_d4t4_x01}temp{t3mp0r4ry_c0d3}archive{c4nc3ll3d_0p}backup{0bs0l3t3_c0d3_01d}sample{r3d_h3rr1ng_d4t4}null{1
nv4l1d_4cc3ss}void{n0t_4ctiv3_c0d3}legacy{0utd4t3d_1nf0}
```

フラグ形式は、flag{}です。これらは
全てダミーフラグです。

ここでわかるのは、flagはbase64で
エンコードされている可能性が
考えられます。

flag{という文字列自体をbase64に
エンコードして、エンコードした
文字列で検索することで見つかる
可能性があります。base64で
エンコードした**flag{**を確認します。

```
┌──(kali㉿kali)-[~/challange]
└─$ echo -n "flag{" | base64
ZmxhZ3s=

┌──(kali㉿kali)-[~/challange]
└─$ echo -n "flag{AA" | base64
ZmxhZ3tBQQ==

┌──(kali㉿kali)-[~/challange]
└─$ echo -n "flag{CC" | base64
ZmxhZ3tDQw==
```

エンコードの結果、ZmxhZ3もしくは、ZmxhZから始まる文字列はflag文字列である可能性が高い事がわかりました。この結果、からstringsで出力された文字列を正規表現にて抽出してみます。

```
┌──(kali㉿kali)-[~/challange]
└─$ strings secret.db | grep "ZmxhZ"
ENCODED MISSION CODE (Base64): ZmxhZ3tkM2wzdDNkX3MzY3IzdF9tM3NzNGczfQ==
ENCODED MISSION CODE (Base64): ZmxhZ3tkM2wzdDNkX3MzY3IzdF9tM3NzNGczfQ==
```

見つかりました。base64でデコードしてみます。

```
┌──(kali㉿kali)-[~/challange]
└─$ echo -n "ZmxhZ3tkM2wzdDNkX3MzY3IzdF9tM3NzNGczfQ==" | base64 -d
flag{d3l3t3d_s3cr3t_m3ss4g3}
```

フラグが確認できました。

flag{d3l3t3d_s3cr3t_m3ss4g3}