ECB Oracle

שוב היה לי סקריפט שפותר את התרגיל מהעבר, אז הסתכלתי קצת ועכשיו אני אפילו יכול להסביר אותו.

הרעיון נובע מכך שאין IV רנדומלי בECB והpadding עצמו קבוע, כלומר הבלוק ABCD... תמיד יתורגם לאותו ciphertext.

בנוסף, יש לנו אופציה להוסיף plaintext בתחילת הpayload שיוצפן, אנחנו גם יודעים שAES מיושר לבלוקים של 16. בתור התחלה נוסיף תווים עד שנראה שהגענו לבלוק חדש, ברגע זה נעצור ונבין מה קורה, למעשה מכיוון שמתבצע:

```
padded = pad(plaintext + FLAG.encode(), 16)
```

הבלוק החדש ייראה כך:

$$FLAG[n-1], known\ padding\ \ldots$$

אז יש לנו את הצורה המוצפנת של התו האחרון מהדגל עם פדינג. איך ננצל את זה?

זה פשוט – נוסיף בלוק שלם משלנו שהוא עצמו גם באותו פורמט, תו כלשהו עם פדינג, ונשלח להצפנה.

נשווה את הצורה המוצפנת עם הבלוק שבידינו, במידה והיא זהה – מצאנו תו אחרון של הדגל!

לאחר מכן נמשיך בדיוק כך – הבלוק הבא יהיה *FLAG[n-1], FLAG[n-2]* אבל את *FLAG[n-1]* אנחנו כבר יודעים והברוט פורס נשאר רק בגודל של בית אחד בלבד.

```
$ python3 ecb_oracle.py
c
cr
cry
cryp
crypt
crypto
crypto{
crypto{p
crypto{p3
crypto{p3n
crypto{p3n6
crypto{p3n6u
crypto{p3n6u1
crypto{p3n6u1n
crypto{p3n6u1n5
crypto{p3n6u1n5_
crypto{p3n6u1n5_h
crypto{p3n6u1n5_h4
crypto{p3n6u1n5_h47
crypto{p3n6u1n5_h473
crypto{p3n6u1n5_h473_
crypto{p3n6u1n5_h473_3
crypto{p3n6u1n5_h473_3c
crypto{p3n6u1n5_h473_3cb
crypto{p3n6u1n5_h473_3cb}
crypto{p3n6u1n5_h473_3cb}
```

(מה שאני עושה בפועל בסקריפט זה לדחוף את התו הראשון למקום האחרון בבלוק, כנראה שחשבתי על זה קצת שונה אז, אבל ההשמשה מתבסס על רעיון זהה).

אגב, אני לא חושב שצריך את אורך הדגל אבל לכאורה ניתן לגלות בקלות מה הוא ע"י הוספת בתים עד שגלשנו בלוק כלשהו. למשל אם הוספנו 7 בתים ונוצר הבלוק הרביעי, זה אומר ש6 בתים השלימו לכפולה של 16 (במקרה הזה 64) ולכן אורך הדגל הוא 58.