Pad Thai

אין פה הרבה קוד, מתחילים עם מפתח קבוע מגונרט רנדומלית, הודעה רנדומלית שתופסת בלוק אחד בדיוק ו-IV רנדומלי שאינו קבוע 😊 כלומר החולשה היא לא בקינפוג הAES

יש לנו יכולת לקרוא לget_ct שמביאה לנו את ההודעה עם IV שונה בכל פעם, לcheck_message שתבדוק אם ההודעה שהכנסנו זהה להודעה הסודית ואם כן תביא לנו דגל ולcheck_padding.

המעניינת היא כמובן check_padding, ספציפית מתבצע unpad(plaintext, 16) ואם הpadding נכון חוזר True, אחרת False.

Crypto.Util.Padding.unpad(*padded_data, block_size, style='pkcs7'*)

Remove standard padding.

Parameters:
- **padded_data** (*byte string*) – A piece of data with padding that needs to be stripped.
- **block_size** (*integer*) – The block boundary to use for padding. The input length must be a multiple of `block_size`.
- **style** (*string*) – Padding algorithm. It can be *'pkcs7'* (default), *'iso7816'* or *'x923'*.

Returns: data without padding.

Return type: byte string

Raises: **ValueError** – if the padding is incorrect.

אז פונקציית הunpad מקבלת 3 פרמטרים, המידע, גודל הבלוק וסוג הפדינג. נראה שהכל נכון אז נצטרך לקרוא על pkcs7 padding.

One is to pad the last block with n bytes all with value n, which is what Alex Wien suggested. This has issues (including restricting you to block sizes that are less than 256 bytes long). This padding mode is known as PKCS#7 padding (for 16 byte blocks) or PKCS#5 padding (for 8 byte blocks).

אז מסתבר שpkcs#7 זה פשוט למלא את שאר הבלוק עם בתים שהערך שלהם זה הגודל שמשלים אותו ל16.

As for how to know whether a message is padded, the answer is a message will *always* be padded: even if the last chunk of the message fits perfectly inside a block (i.e. the size of the message is a multiple of the block size), you will have to add a dummy last block.

זה מגניב – תמיד יהיה padding למרות שבמקרה הזה ההודעה שלנו בגודל בלוק בלבד.

לאחר קצת מחשבה עלה לי כיוון מעניין, יש לי שליטה על הIV שמשומש בפענוח ואני יודע (ובדקתי עם אתגר קודם) שAES מבצע Decrypt ואז XOR עם הIV, משהו מגניב שאני יכול לעשות זה brute-force מוזר לפי סכמת הפדינג.

משהו בסגנון של:

- נוריד את בלוק הפדינג ונשאר עם 2 בלוקים בלבד, IV וCT
- נשנה את הבית האחרון בIV כל פעם עד שunpad מחזירה true, במקרה הזה אני יודע שמתקיים $IV[15] \oplus PT[15] = 0x1$ ומכיוון שXOR אסוציאטיבי אני יכול לקבוע ש$PT[15] = IV[15] \oplus 0x1$
- לאחר שיש לי את הערך הזה אני יכול לחשב $0x1 \oplus 0x2$ ולקבל $0x3$, ולבצע $IV[15] = PrevIv[15] \oplus 0x3$ והפעם יתקיים $IV[15] \oplus PT[15] = PrevIv[15] \oplus PT[15] \oplus 0x3$ $0x1 \oplus 0x3 = 0x2$ (הנקודה היא שאחרי שעשינו ברוט-פורס כדי לגלות את הערך שיביא ל1, אפשר בזמן קבוע לחשב מה צריך לעשות כדי שהבית הPT יהפוך ל2)
- ואז נעשה ברוט-פורס על הבית ה14 בIV כדי לגלות מה יביא את הבית ה14 בPT להיות 2...

וככה נבצע שוב ושוב, הסיבוכיות תהיה $256 \cdot 16$ 😊

מכאן כדי לגלות את ההודעה נצטרך רק לפתור 16 משוואות בסגנון של $pt[i] = 16 \oplus x$

ולאחר התפסטנות כהוגן על מימוש יעיל בפייתון וכל מני הסרות חלודה, הבנתי שקרתה פדיחה. הרצתי את השרת לוקאלית ואני קולט שהצפנו 32 בתים (כי message.encode('ascii')), מה שאומר שלא מספיק לי להדליף בלוק אחד. יתרה מכך – שיחקתי עם זה קצת והצלחתי להדליף בלוק אחד, עכשיו צריך להבין איך מכלילים את המתקפה לn בלוקים.

ואז נזכרתי שאני לא מהנדס ביטים אלא עושה קריפטו, בסוף – אין הבדל בין אם הIV המשומש הוא הבלוק ה0 או ה1, לבלוק השני הIV הוא הבלוק ה1, אני יכול לעשות בדיוק אותו דבר עליו כמו שעשיתי לIV המקורי (שהוא הIV לבלוק הראשון)

עכשיו רק נותר לחכות להצלחה

```
[12:35 PM]-[liam@liampc]-[~/ctfs/cryptohack]- |main ✓|
$ python3 pad_thai.py
Let's practice padding oracle attacks! Recover my message and I'll send you a flag.

{"ct": "a53370bcf0408c50e9dd5b1a81ec6e16a16078173cd001d6d7926462763174ccffa3f74ec4f9826063935ecf9b26bf
5b"}

 19%|                                                    | 3/16 [00:21<01:40,  7.70s/it]
```

בזמן הזה אני אבהיר את המשוואה שיש בסוף.

נניח שהIV באינדקס 15 הוא 0x4, כדי שהpadding יהיה נכון צריך להתקיים (בבלוק עם פדינג מגדול 1)

$$IV[15] \oplus Pt[15] = 0x1$$

אנחנו יודעים את IV ואת ה0x1 כמובן, אז מה נשאר לעשות כדי להדליף את הבלוק?

ברוט-פורס על ערך x כלשהו שעבורו נבצע $IV[15] \oplus Pt[15] \oplus x = 0x1$ כדי שנהפוך את הפדינג של הבלוק לפדינג בגודל 1 (הדרך שהאלגוריתם של הפדינג מגלה כמה גדול הפדינג זה לקרוא את הערך שבבית הזה ולכן הוא יחשוב שהפדינג הוא 1, בלי תלות בכמה הפדינג באמת)

ברגע שנמצא את זה נוכל לשחזר את $Pt[15]$ בקלות ע"י $IV[15] \oplus x \oplus 0x1 = Pt[15]$

כמובן שאת זה נכליל לn כללי כדי להדליף את כל הבלוק, כדי להדליף כמה בלוקים פשוט חוזרים על זה ככמות הבלוקים ומשנים את IV בהתאם.

```
[12:35 PM]-[liam@liampc]-[~/ctfs/cryptohack]- |main ✓|
$ python3 pad_thai.py
Let's practice padding oracle attacks! Recover my message and I'll send you a flag.

{"ct": "a53370bcf0408c50e9dd5b1a81ec6e16a16078173cd001d6d7926462763174ccffa3f74ec4f9826063935ecf9b26bf
5b"}

100%|                                                                    | 16/16 [02:50<00:00, 10.64s/it]
100%|                                                                    | 16/16 [02:47<00:00, 10.50s/it]
{'option': 'check', 'message': 'f6d661bee26e45a3290cb5f9fb3717b2'}
{"flag": "crypto{if_you_ask_enough_times_you_usually_get_what_you_want}"}
```