Đây thực chất là một “stream cipher” tự chế dựa trên SHA-1 làm PRG kết hợp với PKCS#7-like padding, và dịch vụ chỉ cho bạn oracle kiểm tra padding (padding oracle) chứ không cho encrypt trực tiếp nữa. Mình chia việc phân tích và tấn công ra hai phần:

**1. Phân tích hệ thống mã hóa**

1. **Gen OTP (keystream)**

def gen\_otp(key: bytes, message: bytes) -> bytes:

iv = key

otp = b''

for \_ in range(len(message)//20):

iv = hashlib.sha1(iv).digest()

otp += iv

return otp

* + Lấy iv = key ban đầu.
  + Lần lượt SHA1 lên iv để sinh ra các block 20-byte, ghép lại thành keystream có độ dài bằng độ dài *đã pad* của message.
  + **Chú ý**: keystream chỉ phụ thuộc vào key và số block (=> phụ thuộc vào độ dài pad của message), không phụ thuộc vào nội dung message.

1. **Padding và XOR**
   * pad(message) thêm PKCS#7-like để độ dài thành bội số 20.
   * Encrypt: lấy C = pad(M) ⊕ OTP(key, pad(M)).
   * Decrypt (dịch vụ): nhận hex-string C, decode thành bytes, tính OTP(key, pad(C)), rồi M′ = C ⊕ OTP. Sau đó gọi unpad(M′) để kiểm tra và tách padding.
2. **Oracle chỉ trả về**
   * Nếu sau khi XOR và unpad mà nội dung chứa substring FLAG hoặc byuctf: in "I didn't make it that easy" và không lộ plaintext.
   * Nếu unpad thất bại (padding không hợp lệ): decrypt trả về "Error decrypting" (và dịch vụ sẽ in b'Error decrypting').
   * Ngược lại (unpad thành công, không chứa từ cấm): in luôn plaintext.encode().

=> Ta có **padding oracle**: chỉ cần gửi một ciphertext bất kỳ, dịch vụ sẽ cho biết padding hợp lệ hay không.

**2. Khai thác padding oracle trên stream cipher**

Dù đây không phải CBC, về mặt toán học padding oracle attack trên CBC cũng hoàn toàn áp dụng được cho stream cipher có block-sized padding:

* Gọi block cuối của ciphertext gốc (đã pad) là CnC\_nCn​ (20 byte), keystream tương ứng là KnK\_nKn​, và plaintext block cuối là Pn=Cn⊕KnP\_n = C\_n \oplus K\_nPn​=Cn​⊕Kn​.
* Chúng ta muốn tìm từng byte của KnK\_nKn​, hay tương đương tìm PnP\_nPn​ bởi vì Kn=Cn⊕PnK\_n = C\_n \oplus P\_nKn​=Cn​⊕Pn​.
* Cách làm (giả sử block size = 20):
  1. **Byte cuối**:
     + Muốn tạo padding hợp lệ 0x01 ở vị trí cuối, ta cần Pn′[19]=0x01P\_n'[19] = 0x01Pn′​[19]=0x01.
     + Chọn một giá trị giả định g∈[0..255]g\in[0..255]g∈[0..255] cho Pn[19]P\_n[19]Pn​[19]. Khi đó Kn[19]=Cn[19]⊕gK\_n[19] = C\_n[19]\oplus gKn​[19]=Cn​[19]⊕g.
     + Ta biết CnC\_nCn​, chưa biết KnK\_nKn​. Để thử ggg, ta tạo một ciphertext giả:

Cn′[19]=Cn[19]⊕Pn[19]⊕0x01 C\_n'[19] = C\_n[19] \oplus P\_n[19] \oplus 0x01Cn′​[19]=Cn​[19]⊕Pn​[19]⊕0x01

(các byte khác giữ nguyên). Gửi toàn bộ ciphertext với block cuối là Cn′C\_n'Cn′​ đến oracle.

* + - Nếu oracle trả padding hợp lệ (trả về plaintext chứ không phải "Error decrypting"), tức là sau XOR ra Pn′P\_n'Pn′​ có 0x01 hợp lệ => dự đoán thành công. Từ đó suy ra được byte cuối của PnP\_nPn​.
  1. **Các byte tiếp**:
     + Để dò byte thứ iii (từ phải về trái), mình giả lập padding length là b=20−ib = 20 - ib=20−i. Đầu tiên đã tìm được byte Pn[19]P\_n[19]Pn​[19], Pn[18]P\_n[18]Pn​[18], ..., đến byte i+1i+1i+1.
     + Muốn padding hợp lệ là b+1 lặp lại b+1b+1b+1 lần ở cuối block, ta điều chỉnh các byte đã “dò” như:

Cn′[j]=Cn[j]⊕Pn[j]⊕(b+1)∀j>i C\_n'[j] = C\_n[j] \oplus P\_n[j] \oplus (b+1) \quad \forall j>iCn′​[j]=Cn​[j]⊕Pn​[j]⊕(b+1)∀j>i

Còn byte ở vị trí iii ta thử các giá trị để ép Pn′[i]=b+1P\_n'[i] = b+1Pn′​[i]=b+1.

* + - Gửi oracle, nếu padding hợp lệ, dò được byte thứ iii.
  1. Lặp cho đến khi giải hết 20 byte cuối. Rồi tiếp tục chuyên sang block kế cuối nếu flag dài hơn 20 byte.
* **Kết quả**: thu được toàn bộ P=pad(FLAG)P = \texttt{pad(FLAG)}P=pad(FLAG). Xử lý unpad thủ công sẽ cho đúng nội dung FLAG.

**Số lượng truy vấn**

* Mỗi byte cần tối đa 256 lần thử (trung bình 128), block 20 byte => ~2560 truy vấn cho block.
* Nếu flag dài LLL byte, pad lên bội 20 thành NNN block, tổng truy vấn ~N×20×128N \times 20 \times 128N×20×128. Với NNN nhỏ (thường < 5), hoàn toàn khả thi.

**3. Hướng triển khai**

1. **Lấy độ dài flag padded**
   * Chuẩn bị một ciphertext giả có độ dài bytes bằng độ dài hex của encrypted flag chia 2, ví dụ toàn bộ C\_flag. Từ đó biết có bao nhiêu block (đổi hex thành bytes).
2. **Viết script tự động padding-oracle attack**
   * Mỗi lần gửi nc hash.chal.cyberjousting.com 1351, đọc prompt, gửi ciphertext sửa đổi, đọc phản hồi xem có "Error decrypting" hay không, từ đó xác định valid padding.
   * Đóng gói vòng dò byte theo byte, block theo block.
3. **Thu được padded-plaintext**
   * Sau khi dò xong, bạn có pad(FLAG). Bỏ padding bằng cách xem byte cuối (giá trị vvv) và cắt v bytes đi.