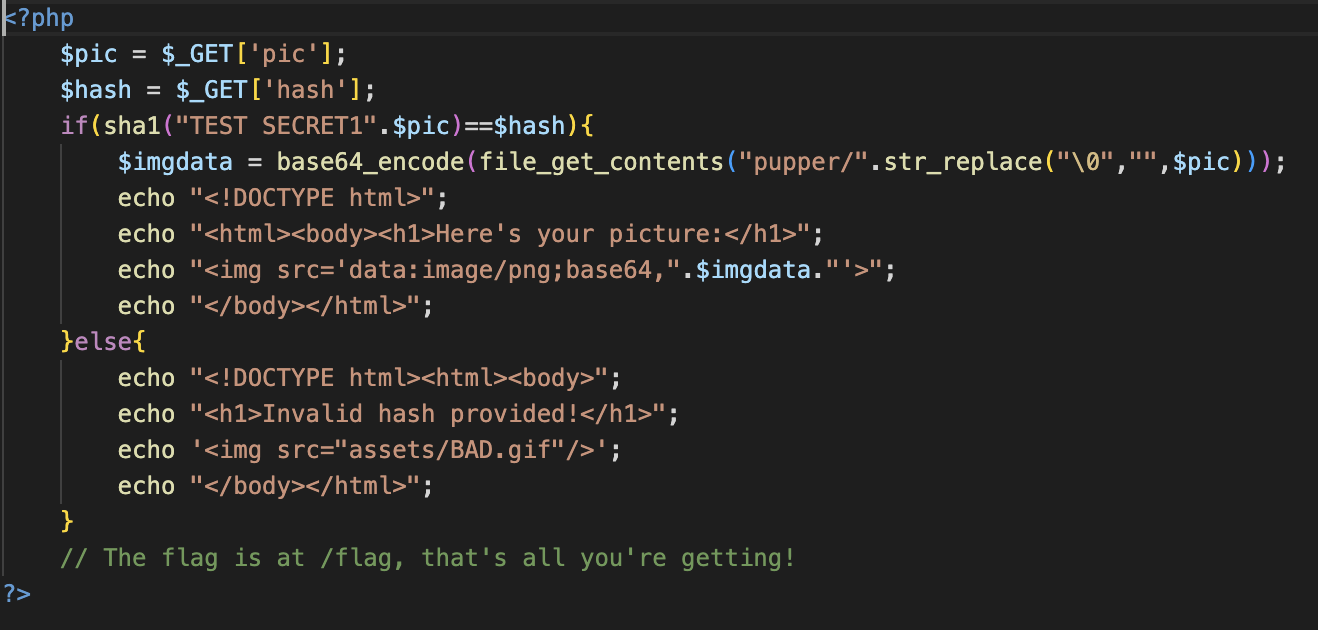
Bài cho source code rất đơn giản:



Flag nằm ở /flag. Ở đây mình sẽ khai thác hàm file\_get\_contents() để đọc flag

Bài cho mình nhập 2 tham số pic và hash. Để đọc được flag, mình phải giải được bài toán sau:

if(sha1("TEST SECRET1".$pic)==$hash){

…

}

secret\_key ở trên là giả

Nhắc đến magic hash sha1 và loose comparison == trong php, mình lại nhớ đến type juggling

Hàm sha1("TEST SECRET1".$pic) trả về chuỗi, $hash cũng là chuỗi, khó khai thác type juggling.

Nếu $hash là json mình có thể truyền số 0 hoặc true(boolean) thì mới khai thác được.

Nhắc đến crypto, mình đã biết đến Known Plaintext Attack. Hôm nay mình sẽ được học thêm 1 kĩ thuật mới là Hash Length Extension Attack

Để khai thác được lỗ hổng trên, bạn phải biết:

* Độ dài của secret key
* Plaintext
* Signature của plaintext
* Thuật toán

Từ đó, bạn có thể nối thêm dữ liệu (payload) vào plaintext và tìm được signature của nó.

Cùng phân tích những điều kiện cần thiết để thực hiện Hash Length Extension Attack nào

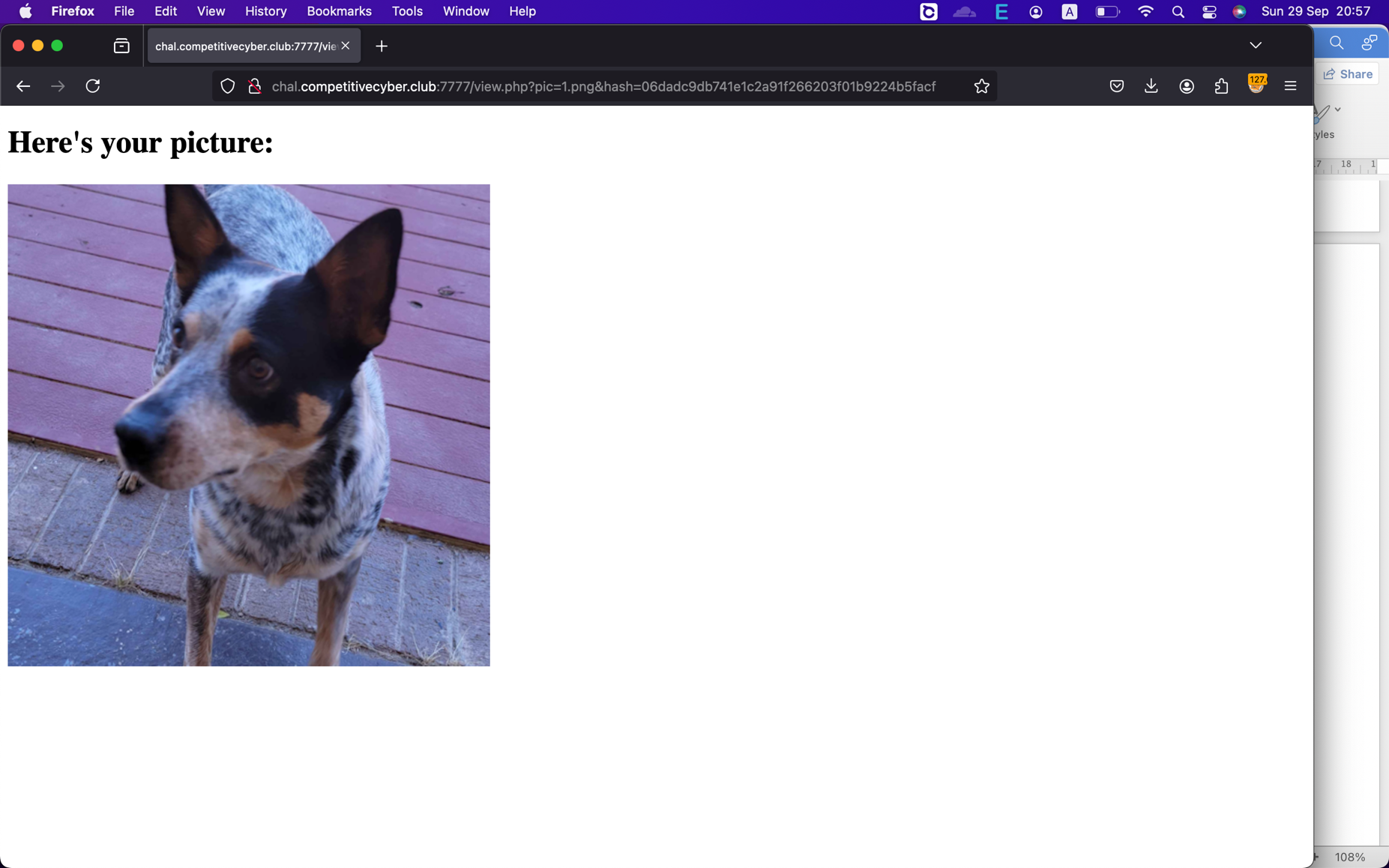
Thuật toán là SHA1

Mình dựa vào secret key mà nó cho để tính độ dài là 12

Trên trang index, bạn thấy nhiều hình ảnh, bật developer tool sẽ thấy mã sha1 tương ứng với plaintext

Ví dụ: 1.png có hash là 06dadc9db741e1c2a91f266203f01b9224b5facf

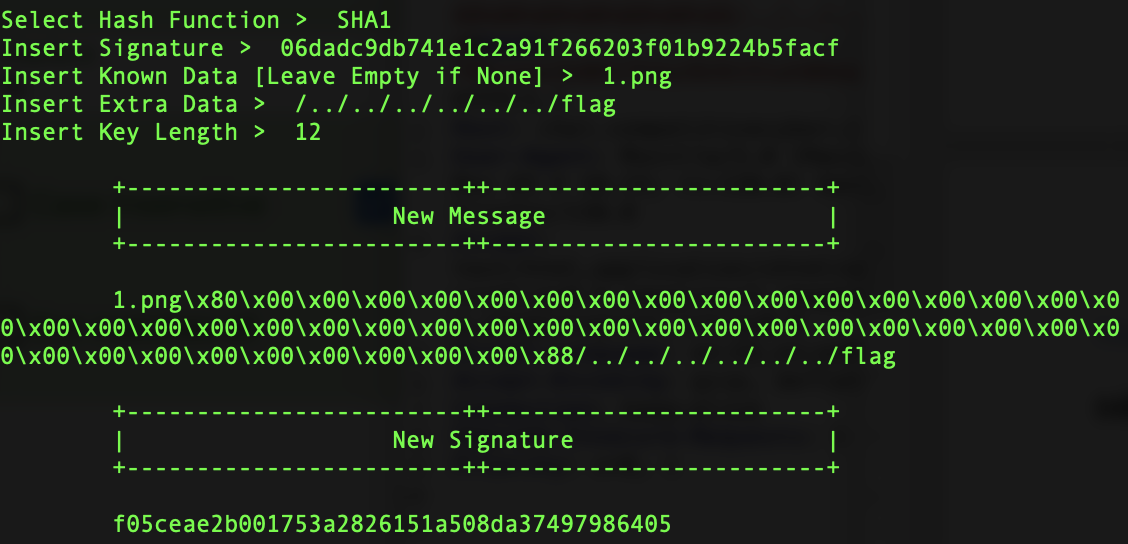
Bạn có thể thử ở trang view.php:



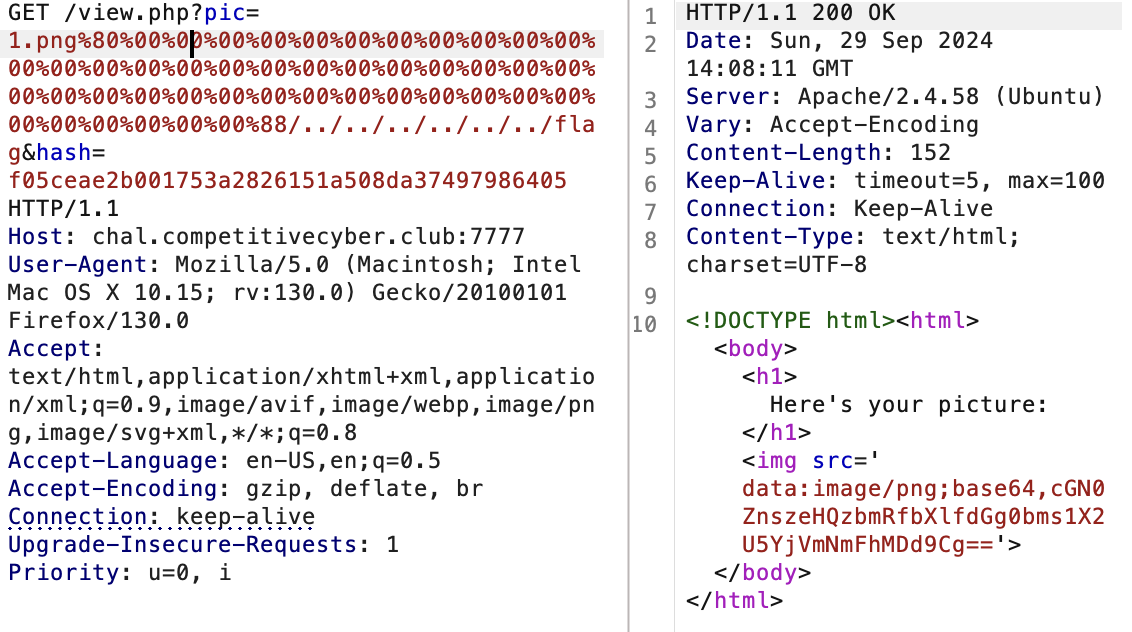
Ta có hàm file\_get\_contents("pupper/1.png"). Để đọc được flag, ta sẽ nối thêm /../../../../../../flag

file\_get\_contents("pupper/1.png/../../../../../../flag")

Để tìm được hash tương ứng, ta dùng đến công cụ Length\_Extender.py:



Vì ta gửi $pic trong Url nên \x trình duyệt sẽ không hiểu, đổi thành %:



**Flag: pctf{3xt3nd\_my\_th4nk5\_e9b5f6aa07}**