Arbeidskrav 1

**1. Decompilation and analysis of an x64 binaryEt bilde som inneholder tekst, skjermbilde, datamaskin

Automatisk generert beskrivelse**

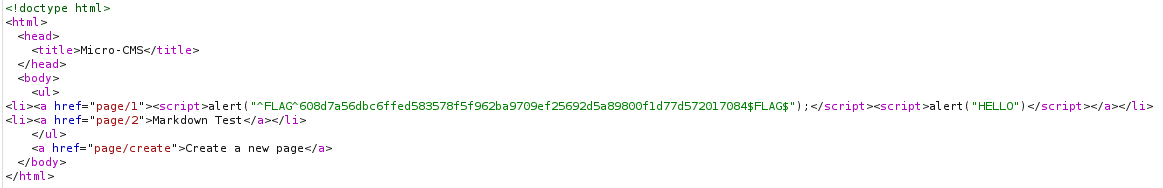
The software vulnerability can be found in the main function, where we can see that the printf-function is used with user-input stored in the local\_28 variable. The printf-function can be used to pass malicious code, for example using %s or %x to print data from the call stack or other locations in memory. This can be fixed by using the fputs() or puts()-function instead of printf() to print the user-input.

**2. Finding a vulnerabilites in a small web application with Burpsuite**

Et bilde som inneholder tekst

Automatisk generert beskrivelse**Flag 1:**

The first Flag was found when typing a script-tag in the title-field of the “Edit this page” in the “Test” page. We first tried typing the script in the text-field, but nothing happened. After the script-tag with an ‘alert’ was added to the title-field, we navigated back to the homepage.

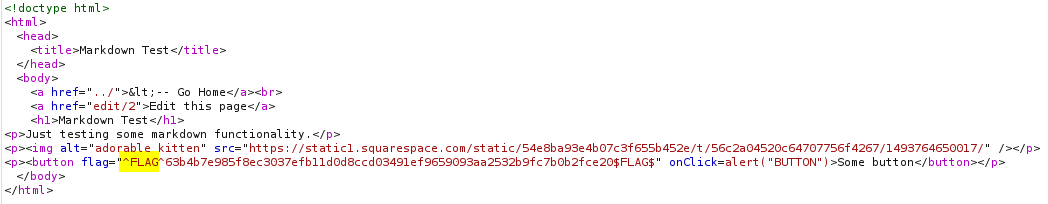
When forwarding the Intercept in BurpSuite, we could see the Flag planted in the link to the homepage. Whenever we clicked the link to the homepage, the alert-box with the Flag popped up instead of being redirected.

**Flag 2:**

The second flag was found while editing the Markdown-page, where there was a button-tag in the text-field. We added an onClick-event to the button with an alert and saved the changes.

Et bilde som inneholder tekst

Automatisk generert beskrivelse

When we forwarded the intercept in BurpSuite, we could see that the second Flag was planted in the button-tag. If we pressed the button after having saved the changes, the alert box popped up on the page with the message that I wrote.

**Flag 3:**

The third flag was found by messing around with the url of the website. We could see that the url changed numbers after the “page/edit/…” part of the url, where the “Test” page had the url “page/1” and when you clicked edit on it, the url was “page/edit/1”. The url behaved the same for the Markdown-page, but had the number 2 in the url instead. When we created a new page, the number that was assigned at the end of the url was 12, which made us think that there might be some other hidden pages that had not been completely removed or was hidden. We changed the number in the url from 3-11, and found a private page in the url “page/edit/7” with the third flag in the text field:

Et bilde som inneholder tekst

Automatisk generert beskrivelse

**Flag 4:**

The fourth flag was also found by messing around with the url. We first thought that there might be a way to perform SQL-injection on the page somewhere, but we couldn’t find any text-fields where it made sense to make a call to the database. To our surprise we found that you could check if a site was likely to send SQL queries via the url by looking at how it was structured. Since the edit-pages on the site had a number that represented the id of the page, these url’s were most likely sending SQL queries. We tried adding a “ **‘** “ at the end of the url to check if this interrupted the query for identifying the page we were on, which sent us directly to a page where the last flag was hidden.

Et bilde som inneholder tekst

Automatisk generert beskrivelse