# The Symbolic Execution Debugger Experiment

The purpose of this experiment is to evaluate the symbolic execution debugger in C language code security auditing.

**How does the experiment work?**

The experiment consists of three sections:

1. **Background Survey**: A short study registration survey about participants' background to determine whether participants are suitable for the experiment. We will send an invitation link by email to qualified participants.
2. **Learning and Identifying the vulnerabilities:** At this stage, you need to complete the two tasks to identify the bugs and inputs that would trigger the vulnerability using your preferred tools or using our tool.
3. **Post-study questionnaire or interview:** You will complete a post-study questionnaire following the completion of each task.

Participants may skip a question or quit at any time. Our website will record the remote desktop session (not your desktop), but we will not record your voice or face. All responses will remain anonymous. The aggregated and anonymous results will show in reports, presentations, or publications.

**Eligibility:** Participants should be 1) at least 18 years old, and 2) have programming and debugging experience 3) pass the pre-screen questions.

**Compensation:** Participants will receive a $10 - $50 Amazon Gift Card based on the quality of their answers. We may ask for your email address to send you a redemption code for the gift card.

(The survey began with an informed consent statement and experiment explanation, omitted here for brevity.)

What's your age?

○ Under 18+
○ 18 - 24
○ 25 - 34
○ 35 - 44
○ 45 - 54
○ 55 - 64
○ 65 - 74
○ 75 - 84
○ 85 or older

What is your gender?
○ Male
○ Female
○ Non-binary / third gender
○ I prefer not to say

I have experience with: (choice all options apply)
☐ Programming in C/C++
☐ Programming in Assembly
☐ Managing memory by malloc/free
☐ Participation in bug-hunting programs
☐ Participation in Capture the Flag (CTF)
☐ Working in a software security group in my company/university
☐ Exploitation stack/heap overflow in the real world
☐ None of the above

Do you have experience with the following symbolic execution tool?
☐ KLEE http://klee.github.io/
☐ angr https://angr.io/
☐ triton https://triton-library.github.io/
☐ KEY https://www.key-project.org/
☐ None of the above

Which IDE / tools / software are you familiar with when you review / bebug source codes?
We may pre-install the common choices in experiment environment.

For the C code snippet below, it may contain a memory-related vulnerability.

```
#define MAX_INPUT_LEN 32
//copy user-control input string to output string, replace & with '&amp;'
char * escaped_copy_input(char *s){
```

```
if (strlen(s) > MAX_INPUT_LEN)
        return NULL; //input too long
char *dst = (char*)malloc(MAX_INPUT_LEN * 4 + 1);
size_t dst_index = 0;
for (size_t i = 0; i < strlen(s); i++ )
        if( '&' == s[i] ){
                dst[dst_index++] = '&';
                dst[dst_index++] = 'a';
                dst[dst_index++] = 'm';
                dst[dst_index++] = 'p';
                dst[dst_index++] = ';';
        }
        else
                dst[dst_index++] = s[i];
        dst[dst_index] = '\0';
        return dst;
}
```

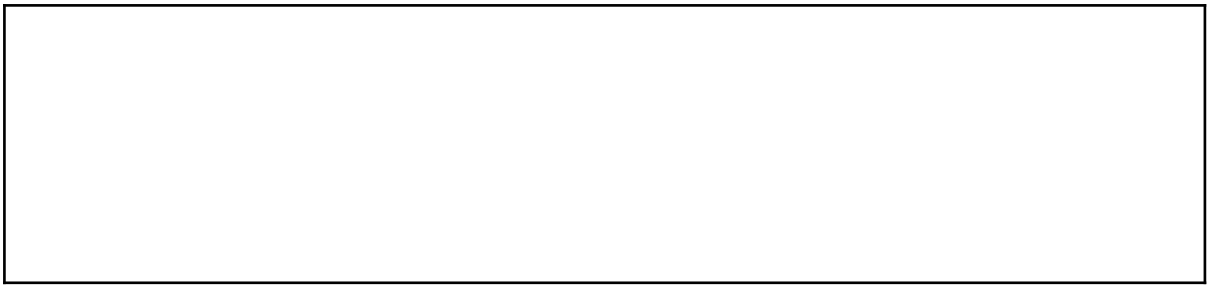
Do you find any bugs in the source code?

○ Yes

○ No

If possible, please briefly describe how the bug occurs by code line number and vulnerability type?

If you have successfully identified the bug, what is value of the parameter (char*s) that triggers it?



(The survey ended with optional contact questions, omitted here for brevity.)