sekurity Write Up

Decompiling

We can use something like ghidra or binaryninja to decompile the file.
For binaryninja, we get this.

```
void _init()
{
    if (__gmon_start__)
        __gmon_start__();
}

int64_t sub_1020()
{
    int64_t var_8 = data_3ff0;
    /* jump -> data_3ff8 */
}

int32_t puts(char const* str)
{
    /* tailcall */
    return puts(str);
}

int64_t sub_1036()
{
    int64_t var_8 = 0;
    /* tailcall */
    return sub_1020();
}

uint64_t strlen(char const* arg1)
{
    /* tailcall */
    return strlen(arg1);
}

int64_t sub_1046()
{
    int64_t var_8 = 1;
    /* tailcall */
    return sub_1020();
}
```

```
int32_t printf(char const* format, ...)
{
    /* tailcall */
    return printf();
}

int64_t sub_1056()
{
    int64_t var_8 = 2;
    /* tailcall */
    return sub_1020();
}

int32_t strcmp(char const* arg1, char const* arg2)
{
    /* tailcall */
    return strcmp(arg1, arg2);
}

int64_t sub_1066()
{
    int64_t var_8 = 3;
    /* tailcall */
    return sub_1020();
}

int32_t __isoc99_scanf(char const* format, ...)
{
    /* tailcall */
    return __isoc99_scanf();
}

int64_t sub_1076()
{
    int64_t var_8 = 4;
    /* tailcall */
    return sub_1020();
}

void exit(int32_t status) __noreturn
{
    /* tailcall */
    return exit(status);
}

int64_t sub_1086()
{
    int64_t var_8 = 5;
```

```
    /* tailcall */
    return sub_1020();
}

void __cxa_finalize(void* d)
{
    /* tailcall */
    return __cxa_finalize(d);
}

void _start(int64_t arg1, int64_t arg2, void (* arg3)()) __noreturn
{
    int64_t stack_end_1;
    int64_t stack_end = stack_end_1;
    __libc_start_main(main, __return_addr, &ubp_av, nullptr, nullptr, arg3, &stack_end);
    /* no return */
}

void deregister_tm_clones()
{
    return;
}

void register_tm_clones()
{
    return;
}

void __do_global_dtors_aux()
{
    if (__TMC_END__)
        return;

    if (__cxa_finalize)
        __cxa_finalize(__dso_handle);

    deregister_tm_clones();
    __TMC_END__ = 1;
}

void frame_dummy()
{
    /* tailcall */
    return register_tm_clones();
}

int64_t swap(void* arg1, int64_t arg2, void* arg3)
{
```

```
    char result = *(arg1 + arg2);
    *(arg1 + arg2) = *(arg1 + arg3);
    *(arg3 + arg1) = result;
    return result;
}

int32_t main(int32_t argc, char** argv, char** envp)
{
    int64_t var_68;
    __builtin_strcpy(&var_68, "Y1ta4c4rkN2uxs3Bhr1{n_tu_0uyg_mr_dnh4n1}0_1");
    int64_t var_70;
    __builtin_strcpy(&var_70, "}nb!{y?");
    puts("Really High Security Vault 100,0…");
    printf("Please enter the passcode: ");
    void var_d8;
    __isoc99_scanf("%99s", &var_d8);
    uint64_t rax_2 = strlen(&var_d8);
    uint64_t rax_3 = strlen(&var_68);

    if (rax_2 == rax_3)
    {
        for (int64_t i = 0; i < rax_3; i += 1)
        {
            uint8_t temp1_2 = COMBINE(0, *(&var_70 + COMBINE(0, i) % strlen(&var_70))) %
strlen(&var_68);

            if (i & 3)
                swap(&var_d8, i, temp1_2);
        }

        if (!strcmp(&var_68, &var_d8))
        {
            puts("Correct passcode! Welcome!");
            exit(0);
            /* no return */
        }
    }

    puts("Wrong passcode! Exiting!");
    return 0;
}

int64_t _fini() __pure
{
    return;
}
```

Solving

We see something that looks a lot like a flag.

```
    int64_t var_68;
    __builtin_strcpy(&var_68, "Y1ta4c4rkN2uxs3Bhr1{n_tu_0uyg_mr_dnh4n1}0_1");
    int64_t var_70;
    __builtin_strcpy(&var_70, "}nb!{y?");
```

Looking at the code here,

```
    uint64_t rax_2 = strlen(&var_d8);
    uint64_t rax_3 = strlen(&var_68);

    if (rax_2 == rax_3)
    {
        for (int64_t i = 0; i < rax_3; i += 1)
        {
            uint8_t temp1_2 = COMBINE(0, *(&var_70 + COMBINE(0, i) % strlen(&var_70))) %
strlen(&var_68);

            if (i & 3)
                swap(&var_d8, i, temp1_2);
        }

        if (!strcmp(&var_68, &var_d8))
        {
            puts("Correct passcode! Welcome!");
            exit(0);
            /* no return */
        }
    }
```

We can reverse engineer this to get our flag.

Heres a sample code in c to do it.

```
#include <stdio.h>
#include <string.h>
#include <stdint.h>
#include <stdlib.h>

void swap(char* str, int i, int j) {
    char temp = str[i];
    str[i] = str[j];
    str[j] = temp;
}

int main() {
```

```
    char flag[] = "Y1ta4c4rkN2uxs3Bhr1{n_tu_0uyg_mr_dnh4n1}0_1";
    char key[] = "}nb!{y?";
    size_t flag_len = strlen(flag);
    size_t key_len = strlen(key);
    char reversed_flag[100];

    strcpy(reversed_flag, flag);

    for (int64_t i = flag_len - 1; i >= 0; i--) {
        uint8_t temp_index = key[i % key_len] % flag_len;
        if (i & 3) {
            swap(reversed_flag, i, temp_index);
        }
    }

    printf("Recovered flag: %s\n", reversed_flag);
    return 0;
}
```