Solution to NoOff's VaultCrackme

I will not properly walk through this solution, however I used Binary Ninja to reverse engineer how the user's input is hashed to check against the password. I then made a simple keygen in C.

I have added labels so that it is easier to read what's going on. I used the Psuedo C feature to read the code more clearly. Of course, you can see that the password should be hashed to 'clnooNKnooN;[AE'.

```
00007ff78d7012a0  int64_t main()

00007ff78d7012a0  {
00007ff78d7012ae      void var_88;
00007ff78d7012ae      int64_t rax_1 = (__security_cookie ^ &var_88);
00007ff78d7012c4      int128_t target;
00007ff78d7012c4      __builtin_strcpy(&target, "clnooNKnooN;[AE");
00007ff78d7012d5      int64_t rdx;
00007ff78d7012d5      printf("Please enter a password:\n", printf("!!!SECRET VAULT!!!\n", rdx));
00007ff78d7012fc      void input;
00007ff78d7012fc      printf(&data_7ff78d703324, fgets(&input, 0x40, __acrt_iob_func(0)));
00007ff78d70130b      hashcmp(&input, &target);
00007ff78d70131a      __security_check_cookie((rax_1 ^ &var_88));
00007ff78d701326      return 0;
00007ff78d7012a0  }
```

Following the function 'hashcmp', we get the following:

```
00007ff78d701070  int64_t hashcmp(char* input, char* target)
00007ff78d701070  {
00007ff78d701080      void var_88;
00007ff78d701080      int64_t rax_1 = (__security_cookie ^ &var_88);
00007ff78d70108e      int64_t length = -1;
00007ff78d70108e
00007ff78d70109c      do
00007ff78d701095          length += 1;
00007ff78d70109c      while (input[length] != 0);
00007ff78d70109c
00007ff78d7010a2      int32_t correct;
00007ff78d7010a2
00007ff78d7010a2      if (length == 16)
00007ff78d7010a2      {
00007ff78d7010a8          int64_t i = 0;
00007ff78d7010ab          int32_t j = 2;
00007ff78d7010ab
00007ff78d7011f2          do
00007ff78d7011f2          {
00007ff78d7010c8              int32_t rcx_1 = ((j - 1) % 5);
00007ff78d7010ca              uint32_t changeBy;
00007ff78d7010ca
00007ff78d7010ca              if ((j - 1) == (((j - 1) / 5) * 5))
```

```
00007ff78d7010cc                changeBy = 14;
00007ff78d7010ca            else if (rcx_1 == 4)
00007ff78d7010d5                changeBy = 13;
00007ff78d7010d3            else if (rcx_1 != 3)
00007ff78d7010dc            {
00007ff78d7010e5                changeBy = rcx_1 == 2;
00007ff78d7010e8                changeBy += 0xa;
00007ff78d7010dc            }
00007ff78d7010dc            else
00007ff78d7010de                changeBy = 12;
00007ff78d7010de
00007ff78d7010ea            input[i] += changeBy;
00007ff78d701105            int32_t rcx_3 = (j % 5);
00007ff78d701107            uint32_t changeBy2;
00007ff78d701107
00007ff78d701107            if (j == ((j / 5) * 5))
00007ff78d701109                changeBy2 = 14;
00007ff78d701107            else if (rcx_3 == 4)
00007ff78d701112                changeBy2 = 13;
00007ff78d701110            else if (rcx_3 != 3)
00007ff78d701119            {
00007ff78d701122                changeBy2 = rcx_3 == 2;
00007ff78d701125                changeBy2 += 10;
00007ff78d701119            }
00007ff78d701119            else
00007ff78d70111b                changeBy2 = 0xc;
00007ff78d70111b
00007ff78d701127            input[(i + 1)] += changeBy2;
00007ff78d701143            int32_t rcx_5 = ((j + 1) % 5);
00007ff78d701145            uint32_t changeBy3;
00007ff78d701145
00007ff78d701145            if ((j + 1) == (((j + 1) / 5) * 5))
00007ff78d701147                changeBy3 = 0xe;
00007ff78d701145            else if (rcx_5 == 4)
00007ff78d701150                changeBy3 = 0xd;
00007ff78d70114e            else if (rcx_5 != 3)
00007ff78d701157            {
00007ff78d701160                changeBy3 = rcx_5 == 2;
00007ff78d701163                changeBy3 += 0xa;
00007ff78d701157            }
00007ff78d701157            else
00007ff78d701159                changeBy3 = 0xc;
00007ff78d701159
00007ff78d701165            input[(i + 2)] += changeBy3;
00007ff78d701181            int32_t rcx_7 = ((j + 2) % 5);
00007ff78d701183            uint32_t changeBy4;
00007ff78d701183
00007ff78d701183            if ((j + 2) == (((j + 2) / 5) * 5))
```

```
00007ff78d701185                changeBy4 = 0xe;
00007ff78d701183            else if (rcx_7 == 4)
00007ff78d70118e                changeBy4 = 0xd;
00007ff78d70118c            else if (rcx_7 != 3)
00007ff78d701195            {
00007ff78d70119e                changeBy4 = rcx_7 == 2;
00007ff78d7011a1                changeBy4 += 0xa;
00007ff78d701195            }
00007ff78d701195            else
00007ff78d701197                changeBy4 = 0xc;
00007ff78d701197
00007ff78d7011a3            input[(i + 3)] += changeBy4;
00007ff78d7011bf            int32_t rcx_9 = ((j + 3) % 5);
00007ff78d7011c1            uint32_t changeBy5;
00007ff78d7011c1
00007ff78d7011c1            if ((j + 3) == (((j + 3) / 5) * 5))
00007ff78d7011c3                changeBy5 = 0xe;
00007ff78d7011c1            else if (rcx_9 == 4)
00007ff78d7011cc                changeBy5 = 0xd;
00007ff78d7011ca            else if (rcx_9 != 3)
00007ff78d7011d3            {
00007ff78d7011dc                changeBy5 = rcx_9 == 2;
00007ff78d7011df                changeBy5 += 0xa;
00007ff78d7011d3            }
00007ff78d7011d3            else
00007ff78d7011d5                changeBy5 = 0xc;
00007ff78d7011d5
00007ff78d7011e1            input[(i + 4)] += changeBy5;
00007ff78d7011e6            j += 5;
00007ff78d7011ea            i += 5;
00007ff78d7011f2        } while (i < 15);
00007ff78d7011f2
00007ff78d701204        correct = strncmp(input, target, 15);
00007ff78d7010a2    }
00007ff78d7010a2
00007ff78d70120c    int64_t result;
00007ff78d70120c
00007ff78d70120c    if ((length != 16 || correct != 0))
00007ff78d70127c        result = printf("VERIFICATION FAILED!\n", target);
00007ff78d70120c    else
00007ff78d70120c    {
00007ff78d70122e        int128_t var_68;
00007ff78d70122e        __builtin_strncpy(&var_68, "CONGRATULATIONS!!!\nYOU
UNLOCKED THE SECRET VAULT!\nYOUR TREASURE ", 0x40);
00007ff78d701253        int32_t var_20;
00007ff78d701253        __builtin_strcpy(&var_20, "0.000");
00007ff78d70125e        uint64_t var_28_1 = 0x30302e3124203e2d;
00007ff78d70126e        result = printf(&data_7ff78d7032c0, &var_68);
```

```
00007ff78d70120c     }
00007ff78d70120c
00007ff78d701289     __security_check_cookie((rax_1 ^ &var_88));
00007ff78d701296     return result;
00007ff78d701070 }
```

The first thing I noticed is that the code is basically doing a loop and is handling characters in blocks of five. I decided to rewrite the code in a clearer way, and then I adapted it to make this keygen:

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


void hashcmp(char* input) {
    int64_t length = -1;

    // Calculate the length of the input string
    do {
        length++;
    } while (input[length] != 0);

    int32_t correct = -1; // Initialise correct variable to ensure it has a valid state.

    // Proceed only if the input string's length is exactly 16
    if (length == 15) {
        int64_t i = 0; // Index for the input modification
        int32_t j = 2; // Variable used in modification logic

        // Modify the input string based on the defined rules
        for (i = 0; i < 15; i++) { // Loop through the entire string
            int32_t rcx = (j - 1) % 5; // Modify based on the j value

            uint32_t changeBy;

            // Apply modification logic
            if ((j - 1) == (((j - 1) / 5) * 5)) {
                changeBy = 14;
            }
            else if (rcx == 4) {
                changeBy = 13;
            }
            else if (rcx != 3) {
                changeBy = (rcx == 2) ? 1 : 0; // Note: rcx_1 == 2 translates to 1, else 0
                changeBy += 10; // Add 10 (0xa)
            }
```

```
            else {
                changeBy = 12;
            }

            input[i] -= changeBy; // Modify the character in the input
            j++; // Increment j for next modification
        }

        // Compare the modified input with the target string using the custom function
    }

    printf(input);



    return;
}

int main() {
    // Sample input and target strings
    char input[16] = "clnooNKnooN;[AE"; // The target string to compare

    // Call hashcmp function and capture the result
    hashcmp(input);



    return 0;
}
```

After running this code, I get the key!