Reverse Hill Write up:

You have been given an executable when running it you see that some string gets printed. But this string changes every time you run it. If you reverse the code and analyze it you will see that there are some operations happening on a matrix. This is actually hill cypher decryption code. After reversing the code you get the key and the encrypted flag. In Hill cypher 3 letters gets encrypted/decrypted at a time if the key is 3*3 matrix i.e if it's a 6 letter string. So in the executable given at each decryption step the key is changed to random values so at each step its get decrypted to a wrong string. All we have to do is get the values of the key and the encrypted string values stored in a matrix. We then have to write our own code for hill cypher decryption.

This is the key:

```
a[3][3] =
{{7.000000,8.000000,11.000000},{11.000000,2.000000,24.000000},{5.000000,4.000000,
17.000000}};
```

This is the encrypted string in a matrix.

```
in[6][3] = { {293.000000,588.000000,383.000000},
                  {436.000000,755.000000,508.000000},
                  {233.000000,319.000000,191.000000},
                  {418.000000,818.000000,518.000000},
                  {290.000000,370.000000,306.000000},
                  {277.000000,318.000000,203.000000} };
```

The code for decryption is:

```
#include<stdio.h>
#include<math.h>
#include <unistd.h>
#include<stdlib.h>
#include<time.h>
float decrypt[3][1],  b[3][3],  c[3][3];
char d[6][3];

float a[3][3] =
{{7.000000,8.000000,11.000000},{11.000000,2.000000,24.000000},{5.000000,4.000000,
17.000000}};


float in[6][3] = { {293.000000,588.000000,383.000000},
                  {436.000000,755.000000,508.000000},
                  {233.000000,319.000000,191.000000},
                  {418.000000,818.000000,518.000000},
                   {290.000000,370.000000,306.000000},
                   {277.000000,318.000000,203.000000} };

void function1();
void function2();

void main()
{
        for(int i=0;i<6;i++)
        {
            function1(i);
        }
        printf("\nFLAG IS:\n");
        for(int j=0;j<6;j++)
        {
            for(int k=0;k<3;k++)
            {
                printf("%c ",d[j][k]);
            }
        }

}

void function1(int n)
{
    int i, j, k,p,q;
    function2();
    float encrypt[3][1];
    for(p=0;p<3;p++)
    {
        encrypt[p][0] = in[n][p];
    }
    printf("\n");
    for(i=0;i<3;i++)
    {
        decrypt[i][0] = 0;
    }
    for(i = 0; i < 3; i++)
    {
        for(j = 0; j < 1; j++)
        {
            for(k = 0; k < 3; k++)
            {
                decrypt[i][j] = decrypt[i][j] + b[i][k] * encrypt[k][j];
            }
```

```c
        }
    }
    for(q = 0; q < 3; q++)
    {
        d[n][q] = (char)( (round(fmod(decrypt[q][0], 26))) + 97);
    }
    printf("\n");
}

void function2()
{
    int  i, j;
    float determinant = 0;
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            c[i][j] = a[i][j];
        }
    }
    for(i = 0; i < 3; i++)
    {
        determinant = determinant + (c[0][i] * (c[1][(i+1)%3] * c[2][(i+2)%3] - 
c[1][(i+2)%3] * c[2][(i+1)%3]));
    }
    for(i = 0; i < 3; i++)
    {
        for(j = 0; j < 3; j++)
        {
            b[i][j] = ((c[(j+1)%3][(i+1)%3] * c[(j+2)%3][(i+2)%3]) - 
(c[(j+1)%3][(i+2)%3] * c[(j+2)%3][(i+1)%3]))/ determinant;
        }
    printf("\n");
    }
}
```

The code on the given executable was:

```c
#include<stdio.h>
#include<math.h>
#include <unistd.h>
#include<stdlib.h>
#include<time.h>
float decrypt[3][1],  b[3][3],  c[3][3];
char d[6][3];
float a[3][3] =
{{7.000000,8.000000,11.000000},{11.000000,2.000000,24.000000},{5.000000,4.000000,
17.000000}};
float in[6][3] = { {293.000000,588.000000,383.000000},
                   {436.000000,755.000000,508.000000},
                   {233.000000,319.000000,191.000000},
                   {418.000000,818.000000,518.000000},
                   {290.000000,370.000000,306.000000},
                   {277.000000,318.000000,203.000000} };

void function1();
void function2();
void function3();

void main()
{
    // while(1)
    // {
        for(int i=0;i<6;i++)
        {
            function1(i);
        }
        printf("\nFLAG IS:\n");
        for(int j=0;j<6;j++)
        {
            for(int k=0;k<3;k++)
            {
                printf("%c ",d[j][k]);
            }
        }
    //      sleep(20);
    // }
}

void function1(int n)
{
    int i, j, k,p,q;
    function2();
    float encrypt[3][1];
    for(p=0;p<3;p++)
    {
        encrypt[p][0] = in[n][p];
    }
    printf("\n");
    for(i=0;i<3;i++)
    {
        decrypt[i][0] = 0;
    }
    for(i = 0; i < 3; i++)
    {
        for(j = 0; j < 1; j++)
        {
            for(k = 0; k < 3; k++)
            {
                decrypt[i][j] = decrypt[i][j] + b[i][k] * encrypt[k][j];
            }
        }
    }
    for(q = 0; q < 3; q++)
    {
        d[n][q] = (char)( (round(fmod(decrypt[q][0], 26))) + 97);
    }
    printf("\n");
}

void function2()
{
    int  i, j;
    float determinant = 0;
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            c[i][j] = a[i][j];
        }
    }
    function3();
    for(i = 0; i < 3; i++)
    {
        determinant = determinant + (c[0][i] * (c[1][(i+1)%3] * c[2][(i+2)%3] -
c[1][(i+2)%3] * c[2][(i+1)%3]));
```

```c
    }
    for(i = 0; i < 3; i++)
    {
        for(j = 0; j < 3; j++)
        {
            b[i][j] = ((c[(j+1)%3][(i+1)%3] * c[(j+2)%3][(i+2)%3]) -
(c[(j+1)%3][(i+2)%3] * c[(j+2)%3][(i+1)%3]))/ determinant;
        }
    printf("\n");
    }
}

void function3()
{
    srand(time(NULL));
    int rand_i;
    int rand_j;
    for (int i=0;i<3;i++)
    {
        for (int j=0;j<3;j++)
        {
            rand_i = rand() % 3;
            rand_j = rand() % 3;
        }
    }

    c[rand_i][rand_j] = ((double)rand()) / 10000000000;
}
```

Flag: VishwaCTF{matrix_the_way_around}