Overview sobre Code Review

----------cpy.c----------

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

int main(int argc, char *argv[]){

    char nome[16];

    if (argc < 2)
    {
        printf("Modo de uso: check.exe username\n");
        return 0;
    }
    strcpy(nome, argv[1]);
    printf("Seja bem vindo: %s \nSeu usuario foi adicionado\n", nome);
    return 0;

}
```

----------check_gets.cpp----------

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

int main(int argc, char *argv[]){

    char nome[16];

    printf("Digite seu nome de usuario:\n");
    gets(nome);

    printf("Seja bem vindo: %s \nSeu usuario foi adicionado!\n", nome);
    return 0;

}
```

----------check_scanf.cpp----------

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

int main(int argc, char *argv[]){

    char nome[16];

    printf("Digite seu nome de usuario:\n");
    scanf("%s", &nome);

    printf("Seja bem vindo: %s \nSeu usuario foi adicionado!\n", nome);
    return 0;

}
```

→ Nenhum dos três códigos faz a validação da qtd de bytes fornecidos pelo usuário: Perdemos esse poder e passamos ao usuário

→ Ao passarmos esses programas para o Immunity Debugger, devemos, ao selecionar a função que recebe os dados como breakpoint e apertarmos o Fn+F7 duas vezes até chegar nela, apertar o CTRL+Fn+F9 pra que possamos passar os argumentos que queremos no executável.
→ A partir daí, vemos que a análise segue da mesma maneira que a anterior
→ Isso mostra que essas funções também são vulneráveis

+ Para resolver esse tipo de problema:

⇒ troca a gets pela fgets

⇒ troca a strcpy pela strncpy

→ Mesmo passando argumento maior do que o reservado, ele só vai considerar o argumento até o limite do que foi reservado para o buffer