Map\_path –

1

A attack surface do Map\_path é o conteúdo do ficheiro input. Uma vez que nada dentro do programa verifica se o utilizador tem acesso às diretorias indicadas pelo ficheiro input no terminal, correr o programa permite a entrada em qualquer diretoria desejada. A partir daí, é possível entrar onde o atacante quiser e, eventualmente, aceder aos ficheiros aí presentes.

strcat(mapped\_path, "/");

strcat(mapped\_path, dir);

Aqui, a função vai concatenando o caminho dos diretórios à variável *mapped\_path*, sem fazer uma verificação de segurança adequada no conteúdo de *dir*. Isto permite que um atacante insira partes do caminho como ".." para manipular o caminho e aceder a diretórios superiores.

if ((ret = chdir(mapped\_path)) < 0){

printf("couldn't chdir to %s !\n", mapped\_path);

strcpy(mapped\_path, old\_mapped\_path);

}

Aqui, a função tenta mudar o diretório para o valor acumulado em *mapped\_path*, o que pode ser controlado por um atacante se *orig\_path* tiver valores manipulados. Isto abre a possibilidade de mudar entre diretórios, permitindo acesso a áreas não autorizadas do sistema de ficheiros.

f = fopen(argv[1], "r");

fgets(orig\_path, MAXPATHLEN + 20, f);

fclose(f);

map\_dir\_chdir(orig\_path);

O código lê o caminho original de um ficheiro (através de *fopen* e *fgets*) e depois tenta mudar de diretoria para esse caminho com *map\_dir\_chdir*. Se um atacante puder modificar o conteúdo do ficheiro entre a abertura e a leitura, ou se manipular o sistema de ficheiros entre as chamadas a *fopen* e *chdir*, ele pode alterar o comportamento do programa, levando a um aumento de privilégios ou à execução de operações indesejadas.

--2

Buffer overflow –

1º Linha 30 - strcpy( path, mapped\_path ); /\* copies mapped\_path to path without doing a size check \*/

2º Linha 86 - strcpy( path, *orig\_path* );

3º Linha 68 - strcat( mapped\_path, "/" );

4º Linha 69 - strcat( mapped\_path, *dir* );