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.

No caso de mysig, a attack surface é

if (exptime <= now) { printf("ignoring SIG: expiration %s is in the past", p\_secstodate (exptime)); return ((cp - rrp) + dlen); }

Problema: Se um atacante manipular os campos de tempo (como o tempo de expiração e o tempo de assinatura), pode tentar fazer com que a aplicação aceite assinaturas expiradas ou inválidas. Isso pode permitir ataques de falsificação de registos DNS.

--2

Buffer overflow –

No programa *map\_path*, há cinco buffer overflows. Acontecem nas linhas 30, 68, 69 e 86. Na linha 30 e 86, ocorre pois está a ser utilizada a função strcpy (copia uma string). O problema de utilizar esta função é que não é feita nenhuma verificação de tamanho da string que vai ser copiada em relação ao local de destino.

Na linha 30, strcpy copia orig\_path para path. A variável orig\_path é uma string contida dentro do input de main. Esta variável pode dar overflow no buffer criado para *path*.

strcpy(path, *orig\_path*);

Caso mapped\_path seja maior ou igual a path (\0 será acrescentado no final da string, causando buffer overflow nestes casos também), ocorrerá overflow. Na linha 35, pathspace cria um buffer com tamanho [MAXPATHLEN], (definido em include-map.h como tendo 20 bytes de tamanho). Este buffer, como será usado depois para (fazer cenas idk), também pode receber input maior do que o seu tamanho.

char pathspace[MAXPATHLEN]

A função strcat concatena strings sem verificar se o tamanho do resultado ultrapassa o espaço reservado para a operação. É este o caso na linha 68 e 69, com as seguintes linhas:

strcat ( mapped\_path, “/” );

strcat ( mapped\_path, dir );

A variável mapped\_path é concatenada com “/” e depois com dir. O resultado pode eventualmente originar um buffer overflow, dependendo do input inicial.

6 –

Na função RRextract, o valor de dlen (comprimento dos dados do RR) é extraído diretamente da mensagem DNS recebida: GETSHORT(dlen, cp); Aqui, dlen é extraído de uma posição na mensagem DNS (o ponteiro cp), que está fora do controle da aplicação. Isso significa que um atacante pode manipular esse valor no pacote DNS para ser maior do que o esperado, causando assim um integer overflow. Após ler dlen, o código tenta garantir que a área referida por cp tem espaço suficiente para ler os dados, usando a macro BOUNDS\_CHECK: BOUNDS\_CHECK(cp, dlen); A macro BOUNDS\_CHECK verifica se o ponteiro cp mais o valor dlen não ultrapassa o fim da mensagem (eom), mas isso não verifica se o buffer onde os dados serão armazenados tem espaço suficiente para conter esses dados. Esta verificação é limitada à estrutura da mensagem DNS, mas não protege contra um buffer overflow quando os dados são copiados ou processados. A cópia dos dados do RR para o buffer é feita logo após a verificação de limites, sem verificar se o espaço disponível no buffer é suficiente para os dados que estão a ser copiados: rdatap = cp; Neste ponto, rdatap (um ponteiro para a posição atual no buffer) é configurado para apontar para os dados que serão lidos de cp. Se o valor de dlen for maior do que o espaço disponível no buffer, a função poderá copiar mais dados do que o buffer suporta, causando um buffer overflow.

1º Linha 30 - strcpy( path, *orig\_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* );

35º Linha – char pathspace [maxpathlen]