Buffer Overflow

ERNESTO ROBLES

Dispotivi coinvolti

MACCHINA KALI: 192.168.1.60

Buffer Overflow

BUFFER OVERFLOW SI VERIFICA QUANDO UN PROGRAMMA RICEVE PIÙ DATI DI QUELLI CHE PUÒ GESTIRE. GLI INPUT UTENTE O I DATI PROVENIENTI DA UNA FONTE ESTERNA VENGONO SCRITTI IN UN'AREA DI MEMORIA "BUFFER" SE L'INPUT SUPERA LA DIMENSIONE MASSIMA PREVISTA PER IL BUFFER, IL DATO IN ECCESSO PUÒ SOVRASCRIVERE ALTRE PARTI DELLA MEMORIA.

SFRUTTANDO IL BUFFER OVERFLOW UN AGGRESSORE PUÒ SOVRASCRIVERE DATI IMPORTANTI NELLA MEMORIA, COME VARIABILI, PUNTATORI O INDIRIZZI DI RITORNO DELLA FUNZIONE E CON QUEST'ULTIMO PUÒ SOVRASCRIVERE L'INDIRIZZO DI RITORNO DELLA FUNZIONE CON UN INDIRIZZO DI UN CODICE MALEVOLO OTTENENDO L'ESECUZIONE DI CODICE ARBITRARIO NEL PROGRAMMA COMPROMESSO.

UN SIMILE ACCADIMENTO IN UN'AZIENDA POTREBBE PORTARE A FURTI DI DATI SENSIBILI, DANNI ALL'INTEGRITÀ DEI DATI O AL SISTEMA, O ADDIRITTURA AL CONTROLLO COMPLETO DEL SISTEMA DA PARTE DELL'ATTACCANTE. PORTANDO I CLIENTI A PERDERE FIDUCIA E PREFERIRE PRODOTTI O SERVIZI DI COMPETITORS CHE DIMOSTRANO MAGGIORE IMPEGNO PER LA SICUREZZA.

DIMOSTRAZIONE

```c
#include <stdio.h>

int main (){

char buffer [10];

printf("Prego inserire il nome utente:");
scanf("%s", buffer);

printf("Nome utente inserito: %s\n", buffer);

return 0;

}
```

```
┌──(kali3㉿kali3)-[~/Desktop]
└─$ ./BOF
Prego inserire il nome utente:ernestojavier
Nome utente inserito: ernestojavier

┌──(kali3㉿kali3)-[~/Desktop]
└─$ ./BOF
Prego inserire il nome utente:ernestojavierroblesperezcioane
Nome utente inserito: ernestojavierroblesperezcioane
zsh: segmentation fault  ./BOF
```

Nelle slide qui presenti è possibile notare come il codice raffigurato non gestisca la possibilità di un Buffer Overflow, l'esecuzione del programma ci dimostra come ci sia comunque una certa tolleranza e pur inserendo più caratteri nel primo esempio non abbiamo alcun errore, nella seconda esecuzione un inserimento molto più grande rispetto all'area riservata nel Buffer ci da un errore di segmentazione in quanto il programma ha continuato a eseguire la copia nonostante il buffer fosse troppo piccolo.

DIMOSTRAZIONE

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

int main() {
    char buffer[10]; // Dimensione massima della stringa

    printf("Prego inserire il nome utente:");
    fgets(buffer, sizeof(buffer), stdin); // Legge al massimo 9 caratteri

    if (strlen(buffer) == 9 && buffer[8] != '\n') {
        // Buffer completamente riempito senza carattere di nuova riga
        printf("Hai inserito troppi caratteri. Massimo consentito: 9.\n");
    } else {
        printf("Nome utente inserito: %s", buffer);
    }

    return 0;
}
```

```
└─$ gcc -g BOFris.c -o BOFris

┌──(kali3㉿kali3)-[~/Desktop]
└─$ ./BOFris
Prego inserire il nome utente:ernestojavierroblesperezcioane
Hai inserito troppi caratteri. Massimo consentito: 9.

┌──(kali3㉿kali3)-[~/Desktop]
```

Nelle slide qui presenti è possibile notare come il codice prima visto sia stato modificato per gestire un eventuale Buffer Overflow, dopo la lettura l'input viene controllato che il buffer contenga non più di 9 caratteri e che l'ultimo carattere non è un carattere di nuova riga se queste condizioni non sono verificate viene visualizzato l'input inserito altrimenti viene stampato un messaggio di errore.
Un'altra possibile soluzione potrebbe essere l'allocazione di memoria dinamicamente in base alla dimensione dell'input.