1. Giới thiệu và nguyên nhân

Lỗi tràn bộ đệm (Buffer overflow) là một trong các lỗi thường gặp trong các hệ điều hành và đặc biệt nhiều ở các phần mềm ứng dụng [6]. Lỗi tràn bộ đệm xảy ra khi một ứng dụng cố gắng ghi dữ liệu vượt khỏi phạm vi của bộ nhớ đệm, là giới hạn cuối hoặc cả giới hạn đầu của bộ đệm. Lỗi tràn bộ đệm có thể khiến ứng dụng ngừng hoạt động, gây mất dữ liệu hoặc thậm chí giúp kẻ tấn công chèn, thực hiện mã độc để kiểm soát hệ thống. Lỗi tràn bộ đệm chiếm một tỷ lệ lớn trong số các lỗi gây lỗ hổng bảo mật [6]. Tuy nhiên, trên thực tế không phải tất cả các lỗi tràn bộ đệm đều có thể bị khai thác bởi kẻ tấn công.

Lỗi tràn bộ đệm xuất hiện trong khâu lập trình phần mềm (coding) trong quy trình phát triển phần mềm. Nguyên nhân của lỗi tràn bộ đệm là người lập trình không kiểm tra, hoặc kiểm tra không đầy đủ các dữ liệu đầu vào nạp vào bộ nhớ đệm. Khi dữ liệu có kích thước quá lớn hoặc có định dạng sai được ghi vào bộ nhớ đệm, nó sẽ gây tràn và có thể ghi đè lên các tham số thực hiện chương trình, có thể khiến chương trình bị lỗi và ngừng hoạt động. Một nguyên nhân bổ sung khác là việc sử dụng các ngôn ngữ với các thư viện không an toàn, như hợp ngữ, C và C++.

Ví dụ 1:

Một ví dụ khác về tấn công khai thác lỗi tràn bộ đệm là khi một ứng dụng đọc một file có kích thước không xác định hoặc kích thước quá lớn và lưu trữ nó vào bộ nhớ đệm mà không kiểm tra kích thước của file này. Khi đó, kẻ tấn công có thể khai thác lỗi tràn bộ đệm bằng cách thực hiện các thao tác ghi đè vào địa chỉ bộ nhớ khác nhau trong hệ thống, thực thi các mã độc hoặc lấy được các thông tin quan trọng từ bộ nhớ đó.

Một cách ngăn chặn tấn công khai thác lỗi tràn bộ đệm trong trường hợp này là đảm bảo rằng ứng dụng chỉ đọc file với kích thước cụ thể, kiểm tra kích thước của file trước khi đọc và lưu trữ nó vào bộ nhớ đệm. Điều này sẽ giúp giới hạn kích thước của file và đảm bảo rằng không có dữ liệu quá lớn sẽ được lưu trữ vào bộ nhớ đệm. Ngoài ra, các bản vá bảo mật và phần mềm chống virus cũng rất quan trọng để bảo vệ hệ thống khỏi các cuộc tấn công khai thác lỗi tràn bộ đệm.

Ví dụ về sử dụng ngôn ngữ bậc thấp:

#include <iostream>

#include <cstring>

using namespace std;

int main() {

char buffer[10];

int value = 0;

cout << "Enter a value: ";

cin >> buffer;

value = atoi(buffer);

cout << "Value is: " << value << endl;

return 0;

}

Ứng dụng này cho phép người dùng nhập vào một chuỗi ký tự và chuyển đổi chuỗi đó thành một số nguyên bằng cách sử dụng hàm atoi của thư viện chuẩn C.

Tuy nhiên, nếu người dùng nhập một chuỗi có độ dài lớn hơn 10 ký tự, thì chương trình sẽ ghi đè lên các vùng nhớ khác trong bộ nhớ, có thể gây ra lỗi hoặc thực thi mã độc. Điều này có thể được khai thác bởi kẻ tấn công để tấn công ứng dụng và chiếm quyền điều khiển của hệ thống.

Để ngăn chặn tấn công này, chúng ta có thể sử dụng các thư viện an toàn hơn như thư viện chuẩn C++ hoặc sử dụng các công cụ để kiểm tra mã nguồn tự động để phát hiện và sửa lỗi tràn bộ đệm. Ngoài ra, chúng ta cũng nên kiểm tra độ dài của đầu vào từ người dùng trước khi lưu trữ nó trong bộ nhớ và đảm bảo rằng các vùng nhớ được cấp phát đầy đủ để tránh lỗi tràn bộ đệm.

1. Cơ chế gây tràn và khai thác
2. Cơ chế gây tràn:

Cơ chế hoạt động của Buffer Overflow Attacking:

Tấn công tràn bộ nhớ đệm là một loại tấn công vào các phần mềm ghi dữ liệu vào trong bộ nhớ buffer (nằm trong vùng lưu trữ tạm thời của bộ nhớ). Do vùng nhớ quá nhỏ dẫn đến tràn dữ liệu sang vùng nhớ khác có thể gây ra hiện tượng crash phần mềm hoặc cũng có thể cho phép kẻ tấn công thực thi những mã lệnh độc hại vào máy tính.

Cách tấn công Buffer Overflow:

1. Kẻ tấn công xác định phần mềm dễ bị tấn công tràn bộ nhớ điển hình như một phần mềm đọc dữ liệu từ một người dung hoặc kết nối mạng và lưu trữ nó ở trong bộ nhớ đệm

2. Sau đó kẻ tấn công gửi nhiều dữ liệu hơn bộ nhớ đệm có thể lưu trữ khiến tràn bộ nhớ sang những vùng nhớ bên cạnh

3. Dữ liệu tràn có thể có khả năng ghi đè lên những dữ liệu quan trọng của bộ nhớ ví dụ như trả về địa chỉ của phần mềm hoặc là 1 chức năng trỏ. Điều này gây ra crash phần mềm hoặc thực thi những lệnh không xác định

4. Nếu kẻ tấn công có khả năng ghi đè địa chỉ tra về, chúng có thể điều hướng chương trình để thực thi những mã lệnh độc hại thay vì trả về mã lệnh của chương trình gốc. Điều này khiến kẻ tấn công có thể điều khiển toàn bộ hệ thống và cho phép chúng thực thi bất kì mã lệnh nào hoặc làm những hành động độc hại khác

1. Khai thác lỗi tràn bộ đệm:

Kẻ tấn công khái thác lỗi tràn bộ đệm để thay đổi đương dẫn thc thi của ứng dụng bằng cách ghi đè lên các phần dữ liệu của bộ nhwos. Dữ liệu bổ dung độc hại có thể chứa mã độc được thiết kế để kích hoạc các hành động cụ thể - thực tế là gứi các hướng dẫn mới đến ứng dụng bị tấn công có thể dẫn đến truy cập trái phép và hệ thống. Các kỹ thuật của kẻ tấn công khai thác lỗ hổng tràn bộ đệm tùy theo kiến trúc và hệ điều hành.

Có rất nhiều cách khai thác lỗ hổng này tùy theo kiến trúc máy tính, hệ điều hanhf và vùng ô nhớ. Một số kiểu khai thác là lỗi trà bộ đệm trên stack hoặc lỗi tràn bộ đệm trên heap hay là khai thác dựa vào các lỗ hong phần mần thông qua ngôn ngữ lập trinh, khai thác qua các trang web có tương tasv lớn nhưng ràng buộc dữ liệu không chặt chẽ.

Ví dụ về khai thác lỗi tràn bộ đệm bằng cách ghi đè biến cục bộ

Ta có đoạn mã nguồn như sau:

#include <stdio.h>

int main()

{

int cookie=0;

char buf[16];

printf("Your name: ");

gets(buf);

if(cookie == 0x41424344)

puts("You win!");

else

puts("Try again!");

return 0;

}

Đoạn mã trên cho phép người dùng nhập vào một chuỗi kí tự, sau đó kiểm tra biến cookie với 0x41424344, nếu bằng thì in ra "You win!", nếu không thì in ra "Try again!". Nhưng biến cookie ở đây được gán giá trị ban đầu là 0 và chương trình không có đoạn code nào thây đổi giá trị của nó, vì vậy chương trình sẽ luôn chạy vào nhánh "Try again!"

Đoạn mã trên đã bị lỗi buffer overflow do sử dụng hàm gets, hàm này sẽ lưu toàn bộ những gì chúng ta nhập vào biến buf, nếu người dùng nhập số kí tự lớn hơn số ô nhớ được cấp phát cho biến buff thì sẽ xảy ra lỗi buffer overflow.

Và mục đích của chúng ta là phải khai thác lỗi này, làm cho chương trình chạy vào nhánh "You win!".

Ví dụ ta có một file thực thi của mã nguồn trên, sau khi dịch ngược bằng IDA Pro ta thu được đoạn mã như sau:

int \_\_cdecl main(int argc, const char \*\*argv, const char \*\*envp)

{

char Buffer; // [esp+1Ch] [ebp-14h]

int v5; // [esp+2Ch] [ebp-4h]

v5 = 0;

printf("Your name: ");

gets(&Buffer);

if ( v5 == 0x41424344 )

puts("You win!");

else

puts("Try again!");

return 0;

}

Từ đoạn mã này ta có thể thấy:

• biến Buffer(buf) nằm ở stack có địa chỉ [ebp-14h], v5(cookie) nằm ở stack có địa chỉ [ebp-4h].

• Buffer nằm thấp hơn v5, vậy ta có thể ghi đè giá trị của biến Buffer lên biến v5.

• Khoảng cách từ Buffer đến v5 (-4h) – (-14h) = 10h = 16.

• Cần 16 bytes bất kỳ để tiếp cận, tiếp đó là dữ liệu muốn ghi đè lên v5 (cookie).

Khi chúng ta chạy chương trình và thử nhập một chuỗi "123456789abc" thì sẽ nhận được kết quả: Kết quả này đúng như luống chạy của chương trình

Các kí tự được nhập vào sẽ được chuyến sang mã ASCII và lưu vào biến buff, các kí tự này vẫn nằm trong vùng nhớ được cấp cho biến buf, do vậy vẫn chưa có gì xảy ra, giờ ta sẽ tạo ra một đoạn mã khai thác chương trình theo công thức:

"c"x16 + "DCBA"

- c là kí tự bất kì

- x16 là lặp lại 16 lần

Đoạn mã này sử dụng 16 kí tự bất kì để lấp đầy ô nhớ được cấp cho biến buff(16 ô) rồi sử dụng 4 kí tự cuối để ghi đè lên biến cookie.

Ta chạy chương trình và nhập vào chuỗi "0123456789abcdefDCBA" thu được kết quả:

Nhìn vào hình trên ta có thể thấy lúc này biến cookie sẽ có giá trị là 0x41424344 (biến trong stack được lưu theo kiểu little endian). Và khi chương trình kiểm tra cookie == 0x41424344 là đúng và sẽ chạy vào nhánh "You win!".

1. Phòng chống

Để phòng chống lỗi tràn bộ đệm một cách hiệu quả, cần kết hợp nhiều biện pháp. Các biện pháp có thể thực hiện bao gồm:

• Kiểm tra thủ công mã nguồn hay sử dụng các công cụ phân tích mã tự động để tìm và khắc phục các điểm có khả năng xảy ra lỗi tràn bộ đệm, đặc biệt lưu ý đến các hàm xử lý xâu ký tự.

• Sử dụng cơ chế không cho phép thực hiện mã trong dữ liệu DEP (Data Excution Prevention). Cơ chế DEP được hỗ trợ bởi hầu hết các hệ điều hành (từ Windows XP và các hệ điều hành họ Linux, Unix,…) không cho phép thực hiện mã chương trình chứa trong vùng nhớ dành cho dữ liệu. Như vậy, nếu kẻ tấn công khai thác lỗi tràn bộ đệm, chèn được mã độc vào bộ đệm trong ngăn xếp, mã độc cũng không thể thực hiện

• Ngẫu nhiên hóa sơ đồ địa chỉ cấp phát các ô nhớ trong ngăn xếp khi thực hiện chương trình, nhằm gây khó khăn cho việc gỡ rối và phát hiện vị trí các ô nhớ quan trọng như ô nhớ chứa địa chỉ trở về.

• Sử dụng các cơ chế bảo vệ ngăn xếp, theo đó thêm một số ngẫu nhiên (canary) phía trước địa chỉ trở về và kiểm tra số ngẫu nhiên này trước khi trở về chương trình gọi để xác định khả năng bị thay đổi địa chỉ trở về.

• Sử dụng các ngôn ngữ, thư viện và công cụ lập trình an toàn. Trong các trường hợp có thể, sử dụng các ngôn ngữ không gây tràn, như Java, các ngôn ngữ lập trình trên Microsoft .Net. Với các ngôn ngữ có thể gây tràn như C, C++, nên sử dụng các thư viện an toàn (Safe C/C++ Libraries) để thay thế các thư viện chuẩn có thể gây tràn.