# Lỗ hổng phần mềm

## Khái niệm: là một điểm yếu trong cách thức cài đặt hoặc là một lỗi trong phần mềm có thể bị khai thác bởi một kẻ tấn công để làm thay đổi hoạt động bình thường của phần mềm.

## Nguyên nhân của lỗ hổng: là lỗi ở một công đoạn nào đó của quy trình phát triển phần mềm. Có lỗ hổng thì nhất định là vì có lỗi.

## Phân loại:

Tiêu chí phân loại

* Theo nguyên nhân xuất hiện
  + Lỗ hổng do kiểm tra dữ liệu: Tràn bộ đệm, Chuỗi định dạng, XSS, SQL Injection...
  + Lỗ hổng khác: Race condition, Sử dụng các thành tố mật mã không an toàn...
* Theo thời điểm xuất hiện (trong quy trình phát triển phần mềm)
  + Giai đoạn Phân tích (đặc tả) yêu cầu
    - Lỗ hổng xuất hiện do không có yêu cầu về tính năng an toàn
    - Ví dụ: không yêu cầu cơ chế chống spam ở trang "Liên hệ"
  + Giai đoạn Thiết kế
    - Lỗ hổng xuất hiện do:
      * Thiết kế luồng thực thi không an toàn
      * Lựa chọn hoặc cho phép lựa chọn sử dụng các thành tố không an toàn
    - Ví dụ: tấn công Padding Oracle lên chế độ CBC của mã khối
  + Giai đoạn Xây dựng (lập trình)
    - Lỗ hổng xuất hiện do sử dụng các hàm, các cấu trúc không an toàn, do không kiểm tra thỏa đáng dữ liệu đầu vào
    - Ví dụ:
      * buffer overflow
      * format string
      * race condition
      * integer overflow...
* Theo mức độ nguy hiểm
  + Định tính
    - Lỗ hổng loại C (Mức thấp): cho phép tấn công từ chối dịch vụ (DoS)
    - Lỗ hổng loại B (Mức trung bình): cho phép người dùng cục bộ leo thang đặc quyền hoặc truy cập trái phép.
    - Lỗ hổng loại A (Mức cao): cho phép người dùng từ xa có thể truy nhập trái phép vào hệ thống
  + Định lượng
    - Có 3 nhóm đại lượng đặc trưng cho mỗi lỗ hổng
      * Base Metric Group
      * Temporal Metric Group
      * Environmental Metric Group
    - Mỗi đại lượng đều có thể đo được và nhận một giá trị nhất định
    - Có công thức để tính điểm chung cho lỗ hổng từ giá trị của các đại lượng
    - Thang điểm: 0.0 đến 10.0; điểm càng cao càng nguy hiểm

## Một số lỗ hổng phần mềm cụ thể:

* Buffer Overflow: Lỗ hổng khiến dữ liệu có kích thước lớn có thể tràn ra khỏi vùng đệm để chứa nó
* Format String: Lỗ hổng khiến dữ liệu chuỗi bị diễn giải như một chuỗi định dạng
* Interger Overflow: Lỗ hổng khiến kết quả phép toán trên số nguyên bị diễn giải sai khi vượt quá phạm vi giá trị
* Off-by-one: Lỗ hổng khiến ký tự kết thúc chuỗi bị ghi đè
* Race Condition: Lỗ hổng trong vấn đề đồng bộ dữ liệu khiến một tiến trình vẫn xử lý dữ liệu cũ, trong khi dữ liệu đã được cập nhật bởi một tiến trình khác

# Stack

Ngăn xếp (stack) là một vùng nhớ được hệ điều hành cấp phát cho chương trình khi nạp

Kích thước stack được xác định khi biên dịch chương trình

* Có thể chỉ định kích thước stack qua tham số cho trình biên dịch
* Mặc định khoảng 1 MB

Chức năng của stack:

* Chứa các biến cục bộ
* Lưu địa chỉ trả về khi gọi hàm
* Truyền tham số khi gọi hàm
* Lưu giữ con trỏ "this" trong lập trình hướng đối tượng

Thao tác trên ngăn xếp

* Trong x86 mỗi phần tử stack là 4 byte
* Stack được quản lý qua ESP
* Hai thao tác cơ bản: PUSH và POP
* PUSH
  + Giảm giá trị của ESP: ESP = ESP – 4
  + Ghi đối số (4 byte) vào [ESP]
* POP
  + Đọc 4 byte tại [ESP] vào đối số
  + Tăng giá trị của ESP: ESP = ESP + 4

# Procedure

Hàm (Procedure) là một đoạn chương trình con mà có thể được gọi bởi một chương trình khác để thực thi một nhiệm vụ nhất định

Thông thường, nếu hàm có trả về một kết quả thì kết quả đó được đặt trong EAX trước khi hàm kết thúc.

Việc gọi một hàm bao gồm:

* Nạp các tham số cần thiết
  + Tham số có thể được nạp vào thanh ghi
  + Ưu: nhanh
  + Nhược: có thể không đủ thanh ghi
  + Cần kết hợp nạp tham số vào stack
  + Người xây dựng hàm có toàn quyền lựa chọn cách thức nạp tham số. Nhưng cần có quy ước chung:
    - Mọi người hiểu mã của nhau
    - Mọi người có thể sử dụng hàm của nhau
* Thực hiện lệnh CALL
  + Phổ biến: stdcall (Windows API) và cdecl (standard C library)
    - Giống
      * Truyền tham số qua stack
      * caller phải bảo quản EAX, ECX và EDX nếu cần (callee phải bảo quản các thanh ghi khác)
    - Khác
      * cdecl: caller phải cân bằng stack
      * stdcall: callee phải cân bằng stack
  + Có thể gặp: fastcall

# Buffer Overflow

## Khái niệm:

* Lỗ hổng tràn bộ đệm (Buffer Overflow) là lỗ hổng trong lập trình, cho phép dữ liệu được ghi vào một buffer có thể tràn ra ngoài buffer đó, ghi đè lên dữ liệu khác và dẫn tới hoạt động bất thường của chương trình
* Dễ tránh nhưng phổ biến và nguy hiểm nhất hiện nay
* Đứng thứ 3/25 trong bảng xếp hạng lỗi lập trình nguy hiểm nhất
* Có nhiều cơ chế bảo vệ và cũng có nhiều kỹ thuật khai thác

## Phân loại:

* Tràn bộ đệm trên Stack: biến cục bộ
  + Stack overflow: sẽ xuất hiện khi buffer tràn trong stack space và là hình thức tấn công phổ biến nhất của lỗi tràn bộ đệm.
  + Mục đích:
    - Ghi đè một biến địa phương nằm gần bộ nhớ đệm trong stack để thay đổi hành vi của chương trình nhằm phục vụ ý đồ của hacker.
    - Ghi đè địa chỉ trả về trong khung stack. Khi hàm trả về thực thi sẽ được tiếp tục tại địa chỉ mà hacker đã chỉ rõ, thường là tại một bộ đệm chứa dữ liệu vào của người dùng.
  + Hệ quả:
    - Ghi đè lên biến cục bộ khác trong stack
    - Ghi đè lên địa chỉ trả về
  + Hướng khai thác lỗ hổng tràn bộ đệm
    - Làm tràn ngẫu nhiên làm chương trình (server!!!) bị crash
    - Ghi đè lên biến khác để chương trình rẽ nhánh theo ý muốn
    - Ghi đè địa chỉ trả về để trỏ tới shellcode (được nạp vào qua dữ liệu)
  + Nguyên tắc khai thác
    - Dữ liệu quan trọng phải nằm phía sau (ở địa chỉ cao hơn) so với bộ đệm
    - Phần dữ liệu tràn phải đủ lớn để đè lên được dữ liệu quan trọng
    - Những dữ liệu khác nằm giữa vùng đệm và dữ liệu mục tiêu cũng bị ghi đè. Việc ghi đè đó có thể ảnh hưởng đến logic làm việc của chương trình, đến khả năng thành công của việc khai thác.
* Tràn bộ đệm trên Heap: cấp phát động