1 shellcode là một đoạn mã máy nhỏ, thường đc viết bằng ngôn ngữ Asembly, được sử dụng trong các cuộc tấn công khai thác lỗ hổng bảo mật để thực thi lệnh hoặc mở 1 phiên shell trên hệ thống mục tiêu
2 Cách hoạt động
Có thể chia thành nhiều giai đoạn, từ việc tạo ra shellcode đến khi nó được thực thi thành công trên hệ thống mục tiêu
2.1 Phát hiện lỗ hổng bảo mật: xác định 1 lỗ hổng bảo mật trong phần mềm hoặc hệ điều hành. các lỗ hổng phổ biến thường đc chèn shellcode
 +Buffer Overflow: Tràn bộ đệm khi ứng dụng không kiểm tra giới hạn bộ nhớ
 +Format String vulnerability lỗi định dạng chuỗi khi xử lí đầu vào không an toàn
 +Integer overflow lỗi tràn số nguyên dẫn đến ghi đè bộ nhớ
 +Use-After-free: sử dụng vùng nhớ sau giải phóng
2.2 Hoạt động Shellcode
Shellcode đc viết bằng ngôn ngữ Assembly hoặc được tạo tự động bằng công cụ Metasploit, MSFvenom, pwntools
+ Lựa chọn mục tiêu: shellcode phải phù hợp với hệ điều hành, kiến trúc CPU(x64, x86) và môi trường thực thi
+ viết shellcode
	- system calls: cách shellcode tương tác với hệ điều hành. Shellcode không thể goi trực tiếp các hàm trong kernel
	->Đặt số syscall vào một thanh ghi cụ thể: Mỗi syscall được gán một số duy nhất(ví dụ Linux x86, syscall execve là 11, Shellcode cần đặt số này vào thanh ghi eax
	->Đặt các đối số syscall vào các thanh ghi khác: Các syscall thường yêu cầu các đối số. Shellcode phải đặt các đối số này vào các thanh ghi đã được quy định trước. Ví dụ, trên Linux x86, execve yêu cầu:
*ebx: Đường dẫn đến chương trình cần thực thi (/bin/sh).
*ecx: Một con trỏ đến một mảng các chuỗi (argv) chứa các tham số cho chương trình. Thường là NULL nếu không có tham số nào.
*edx: Một con trỏ đến một mảng các chuỗi (envp) chứa các biến môi trường. Thường là NULL.
	- Sử dụng thanh ghi (registers): Shellcode thường sử dụng các thanh ghi để truyền tham số cho các syscall
	-Tránh các ký tự null (null bytes): Shellcode thường được chèn vào các chuỗi (strings) và các ký tự null có thể làm gián đoạn quá trình thực thi. Cố gắng tránh các byte \x00 trong shellcode của bạn. Có nhiều kỹ thuật để loại bỏ ký tự null, ví dụ: sử dụng xor eax, eax thay vì mov eax, 0
	- Viết mã vị trí độc lập (Position Independent Code - PIC): Shellcode nên có khả năng chạy ở bất kỳ vị trí nào trong bộ nhớ. Điều này có nghĩa là bạn không nên sử dụng địa chỉ tuyệt đối trong mã của mình. Thay vào đó, hãy sử dụng các kỹ thuật như CALL/POP để lấy địa chỉ hiện tại của shellcode
+tối ưu và mã hóa: shellcode có thể được mã hóa học ẩn giấu để tránh bị phát hiện bởi các hệ thống bảo mật
2.3 Chèn shellcode vào bộ nhớ: Shellcode được chèn vaoif vùng nhớ của chương trình đích để có thể thực thi thông qua
+ Đầu vào không an toàn: kẻ tấn công lợi dụng dữ liệu đầu vào của người dùng để ghi shellcode vào vùng nhớ của chương trinh
+ tấn công vào bộ đệm(buffer overflow): ghi đè vùng nhớ stack hoặc heap bằng shellcode , thay địa chỉ trả về bằng địa chỉ trỏ đến shleecode
+ sử dụng kỹ thuật ROP(return oriented programming) sắp xếp các đoạn mã (gadgets) hiện có trong chương trình để chuyển hướng thực thi đến shellcoder
2.4 Kích hoạt shellcode: để thực thi shellcode được , luồng thực thi của chương trình cần được chuyển hướng đến địa chỉ chứa shellcode.
Thực hiện bằng cách:

+ Ghi đè địa chỉ trả về (Return address overwrite): ghi đè địa chỉ trả về trên stack để trỏ đến shellcode
+ Sử dụng hàm hệ thống: Thay đổi con tỏ hàm(function pointer) hoặc bảng nhập bằng(IAT-import Address Table) để trỏ đến shellcode
2.5 Duy trì quyền truy nhập: sau khi thực thi thành công, kẻ tấn công có thể thực thi các nước dể duy trì quyền kiểm soát hệ thống
+ Cài đặt backdoor: Tạo 1 cửa hậu để truy cập lại hệ thống sau này
+ Vô hiệu hóa tường lửa, phần mềm diệt vius hoặc các hệ thống phát hiện xâm nhập
+ Thu thập thông tin: đánh cắp thông tin nhạy cảm, thông tin đăng nhập hoặc tài liệu quan trọng