IAT / EAT / Inline /SSDT Hooking

Import Address Table (IAT)

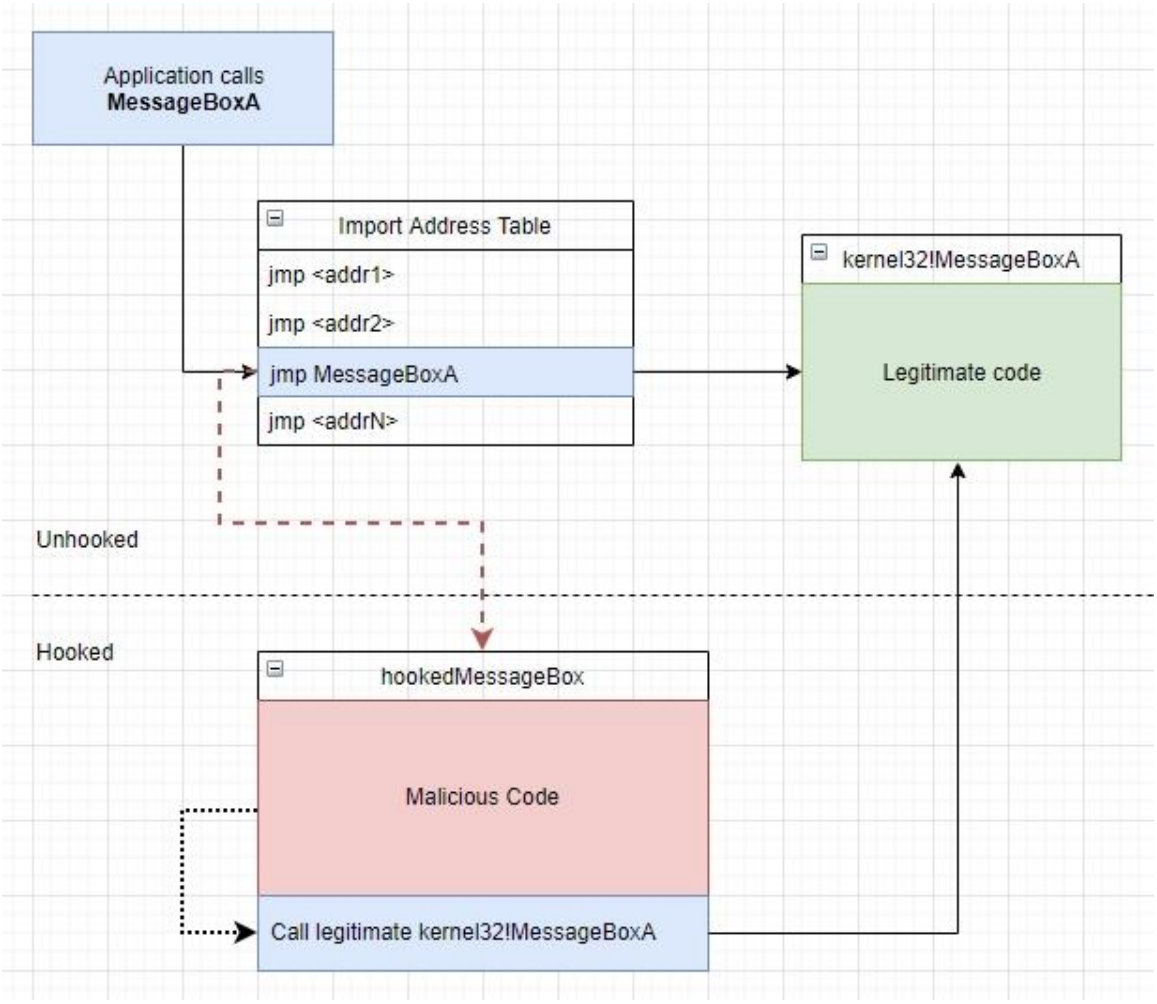
Windows portable executable chứa một cấu trúc gọi là Import Address Table (IAT).
IAT chứa các con trỏ đến thông tin cần thiết cho một executable hoạt động:

- danh sách các DLL mà nó phụ thuộc để cung cấp chức năng,
- danh sách tên hàm và địa chỉ của những hàm đó từ các DLL, những hàm mà binary có thể gọi tại một thời điểm nào đó.

Có thể hook các function pointer được chỉ định trong IAT bằng cách ghi đè địa chỉ hàm mục tiêu bằng địa chỉ của một hàm giả mạo (rogue) và (tùy chọn) vẫn gọi hàm gốc sau đó.

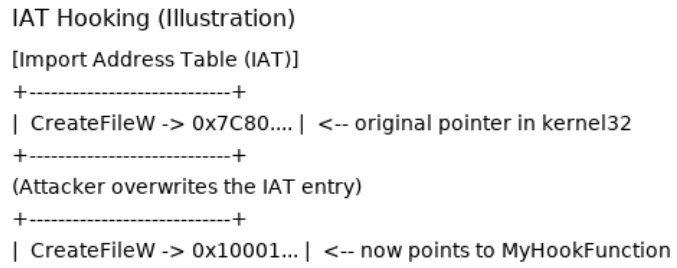
IAT Hooking

IAT Hooking là một kỹ thuật dùng để can thiệp vào các API mà một process Windows gọi tới khiến nó không còn gọi đến địa chỉ gốc, mà gọi đến hàm do attacker chỉ định.



IAT Hooking chỉ ảnh hưởng đến một module, tức là nếu hook MessageBoxA trong IAT của main.exe, chỉ main.exe bị ảnh hưởng, các DLL khác không bị ảnh hưởng.

IAT Hooking chỉ ảnh hưởng đến process hiện tại, không ảnh hưởng đến toàn bộ hệ thống hoặc các process khác.



Export Address Table (EAT)

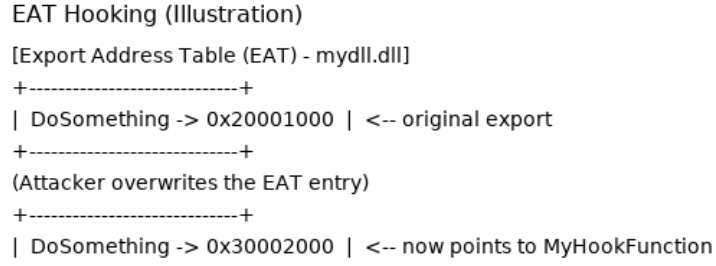
EAT (Export Address Table) được dùng để lưu trữ địa chỉ của các API mà DLL export ra. Các DLL cung cấp thông tin về RVA (Relative Virtual Address) cho các API mà chúng export để các chương trình khác có thể sử dụng.

Nếu DLL không cung cấp thông tin export chính xác, các chương trình sẽ không biết được địa chỉ cụ thể của những API và sẽ không thể sử dụng chúng.

EAT Hooking

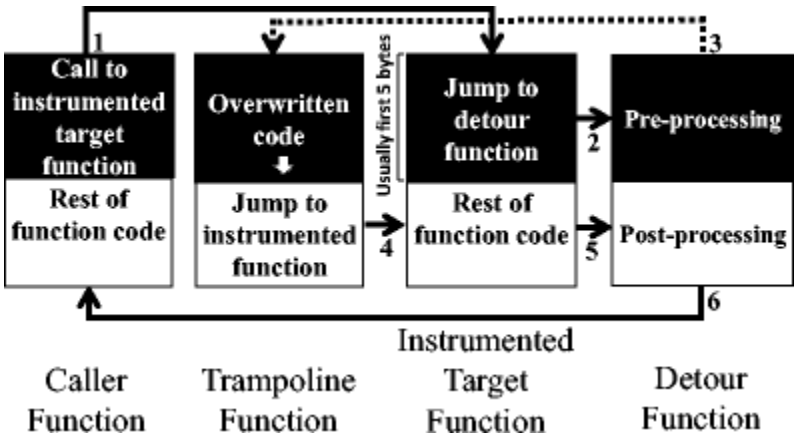
EAT Hooking nhằm can thiệp vào quá trình gọi hàm xuất (exported functions) của một DLL bằng cách thay đổi địa chỉ trong EAT để chuyển hướng sang hàm khác do attacker chỉ định.

EAT Hooking ảnh hưởng tới tất cả các module được load sau khi việc hooking diễn ra — tức là sửa EAT của user32.dll để MessageBoxA trỏ đến hàm giả thì mọi module sau đó gọi MessageBoxA qua user32.dll đều bị ảnh hưởng (kể cả plugin DLL được load sau đó).



Inline Hooking

Inline hooking ghi trực tiếp mã (patch) vào phần đầu của hàm trong DLL hợp lệ, thường thay bằng một lệnh nhảy (JMP) tới mã do attacker cung cấp.



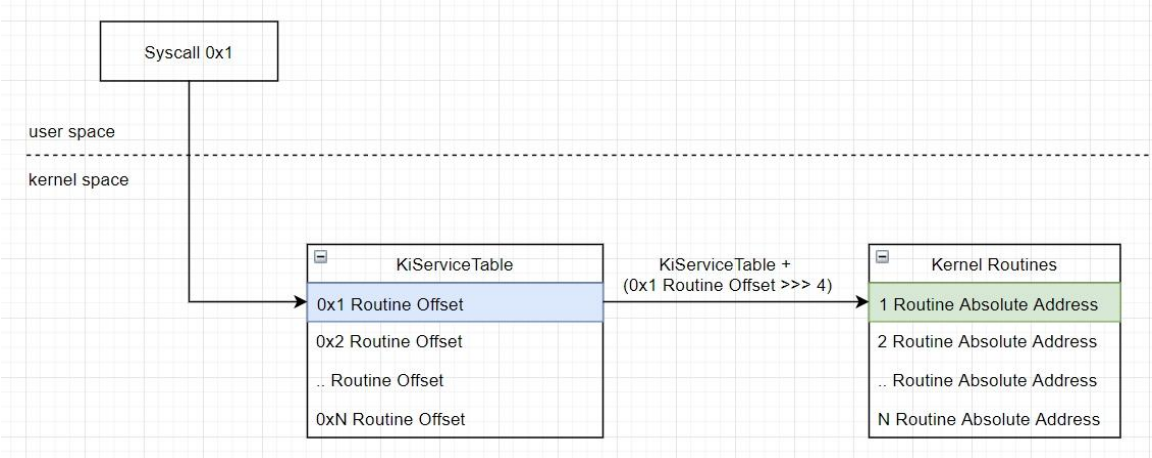
Thông thường attacker sẽ ghi đè khoảng 5 byte đầu tiên của hàm để đặt JMP; đoạn mã bổ sung sẽ thực thi các lệnh bị ghi đè rồi nhảy về phần còn lại của hàm gốc.

```
Inline Hooking (Illustration)
[Program calls DoSomething()]
[0x20001000] -> JMP 0x30002000   (jump to malicious code)
[0x30002000] malicious code executes
[0x30002010] replay overwritten bytes
[0x30002015] JMP 0x20001005    (return to original)
```

SSDT (System Service Descriptor Table) và SSDT Hooking

SSDT là cấu trúc ở cấp kernel chứa các entry cho system call. Khi một tiến trình user-mode gọi một hàm Win32 dẫn xuống kernel, hệ thống tham chiếu SSDT để xác định địa chỉ hàm kernel tương ứng.

Syscall chỉ đơn thuần là một index trong **System Service Dispatch Table (SSDT)**, bảng này chứa một mảng các pointer đối với hệ điều hành 32-bit (hoặc các relative offsets tới **Service Dispatch Table** đối với hệ điều hành 64-bit) trỏ đến tất cả các **critical system APIs** như **ZwCreateFile**, **ZwOpenFile**, v.v...



SSDT Hooking thay đổi entry trong SSDT để chuyển hướng system call đến hàm kernel do attacker (driver) cung cấp. Vì SSDT nằm ở kernel, kỹ thuật này yêu cầu quyền kernel và ảnh hưởng ở mức hệ thống.

Đặc điểm chính:

- Cần quyền kernel (driver).

- Ảnh hưởng toàn hệ thống (global).

- Rất nguy hiểm — có thể gây crash nếu sai.

- Dễ bị phát hiện bởi PatchGuard, Driver Signature Enforcement trên Windows x64.

- Trên Windows hiện đại, nhiều syscall sử dụng 'syscall' instruction và hook SSDT truyền thống bị làm khó.

```
SSDT Hooking (Illustration)
[SSDT]
+-------------------------------+
| NtCreateFile -> 0xFFFFF800... |  <-- original kernel function
+-------------------------------+
(Attacker driver modifies SSDT entry)
+-------------------------------+
| NtCreateFile -> 0xFFFFA000... |  <-- now points to attacker's kernel function
```

So sánh (IAT / EAT / Inline / SSDT) — tóm tắt

Kỹ thuật	Cấp độ tác động	Phạm vi ảnh hưởng	Yêu cầu đặc quyền	Khó phát hiện
IAT	user-mode	1 process /	user-mode	Trung bình

		module		
EAT	user-mode	modules load sau đó	user-mode	Trung bình
Inline	user-mode (patch)	module đang load	user-mode	Khó nếu obf/CRC
SSDT	kernel-mode	Toàn hệ thống (global)	kernel (driver)	Khó (cao)

Phát hiện & Ngăn chặn — chi tiết (defensive)

1) Ngăn chặn (Preventive / Hardening)

- Driver Signing & DSE (Driver Signature Enforcement): chỉ cho phép driver đã ký nạp vào kernel (Windows x64). Điều này ngăn SSDT hooking từ driver không ký.

- PatchGuard / Kernel Patch Protection: trên Windows x64, PatchGuard phát hiện và phản ứng khi kernel code hoặc data (như SSDT) bị sửa; giúp ngăn hoặc phát hiện SSDT/inline kernel patches.

- Secure Boot / VBS (Virtualization-based Security): Secure Boot + VBS (HVCI) tăng rào cản cho mã kernel trái phép.

- ASLR, DEP, CFG: giảm hiệu quả của inline patching và làm khó việc gọi mã dữ liệu.

- Code Signing cho user-mode & integrity checks: kiểm tra chữ ký file khi load; so sánh checksum vùng code với on-disk.

- Giảm quyền: chạy service/ứng dụng với least privilege, hạn chế quyền nạp DLL hoặc inject.

- Whitelisting application / HIPS / EDR: chỉ cho phép danh sách ứng dụng/drivers tin cậy.

Phát hiện (Detection / Forensics)

- IAT/EAT detection: duyệt Import Table của process và kiểm tra địa chỉ function có thực sự nằm trong module gốc.

- Inline detection: tính toán hash/CRC cho region code và so sánh định kỳ với on-disk hoặc copy an toàn.

- SSDT detection: so sánh pointer trong SSDT với giá trị 'expected' từ on-disk kernel or symbol server (yêu cầu quyền kernel để truy cập).

- ETW / Sysmon / Audit: theo dõi hành vi đáng ngờ như process load DLL bất thường, driver install, hoặc thay đổi hệ thống.

- Integrity monitors: giải pháp EDR / HIPS so sánh in-memory code vs on-disk và cảnh báo thay đổi.

Công cụ & Forensics (tổng quan)

- Process Explorer / Procmon: xem module base, handle, threads.
- WinDbg: kiểm tra SSDT entries, disasm, memory comparisons.
- Sigcheck / signtool: kiểm tra chữ ký.
- EDR / HIPS: phát hiện hook/injection, cảnh báo runtime.