Lecture 7

Rootkits
Hoglund/Butler (Chapter 5-6)

Avoiding detection

- Two ways rootkits can avoid detection
  - Modify execution path of operating system to hide rootkit presence
  - Modify data that stores information about processes, files, etc. that would reveal presence of rootkit
- Last chapter
  - Modifying execution path via “hooking”
- This chapter
  - Modifying execution path via run-time patching

Patching

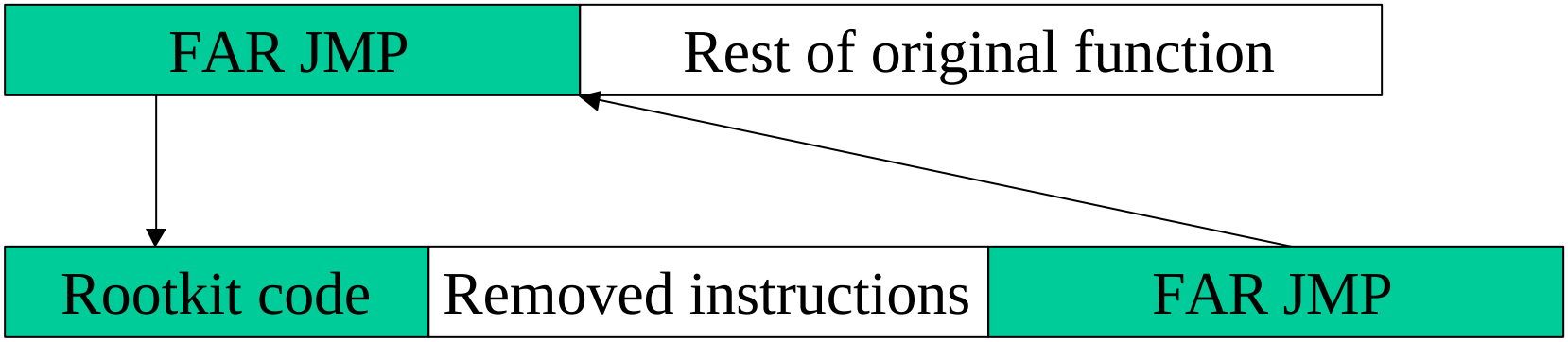
- Source-code
  - Modify source and recompile
- Binary
  - Modify result of compilation
- Memory
  - Modify code in memory as program executes “direct code-byte patch”
  - Hardest to detect
  - Often combined with low-level hardware manipulation such as page-table management
  - Must be able to read/write memory where functions reside (i.e. be within kernel)
  - Previously covered (in-line function hooking)

Run-time patching: Detours

- Detour patching
  - Call hooks modify tables and can be detected by anti-virus/anti-rootkit technology
  - Insert jump into function directly
    - Functions in multiple tables handled in one step
  - Example: MigBot
    - Detours two kernel functions: NtDeviceIoControlFile and SeAccessCheck
    - Both are exported and have entries in the PE header
    - Issues
      - Instruction alignment (leaving crumbs)
        - » Add 1 byte NOPs
      - Overwriting important code

Run-time patching: Detours

- Overwriting important code
  - Must know which OS is being used to ensure you know what code is overwritten
  - Must also ensure no one else has tampered or patched the function already
  - Must save the instructions being removed by adding the jump



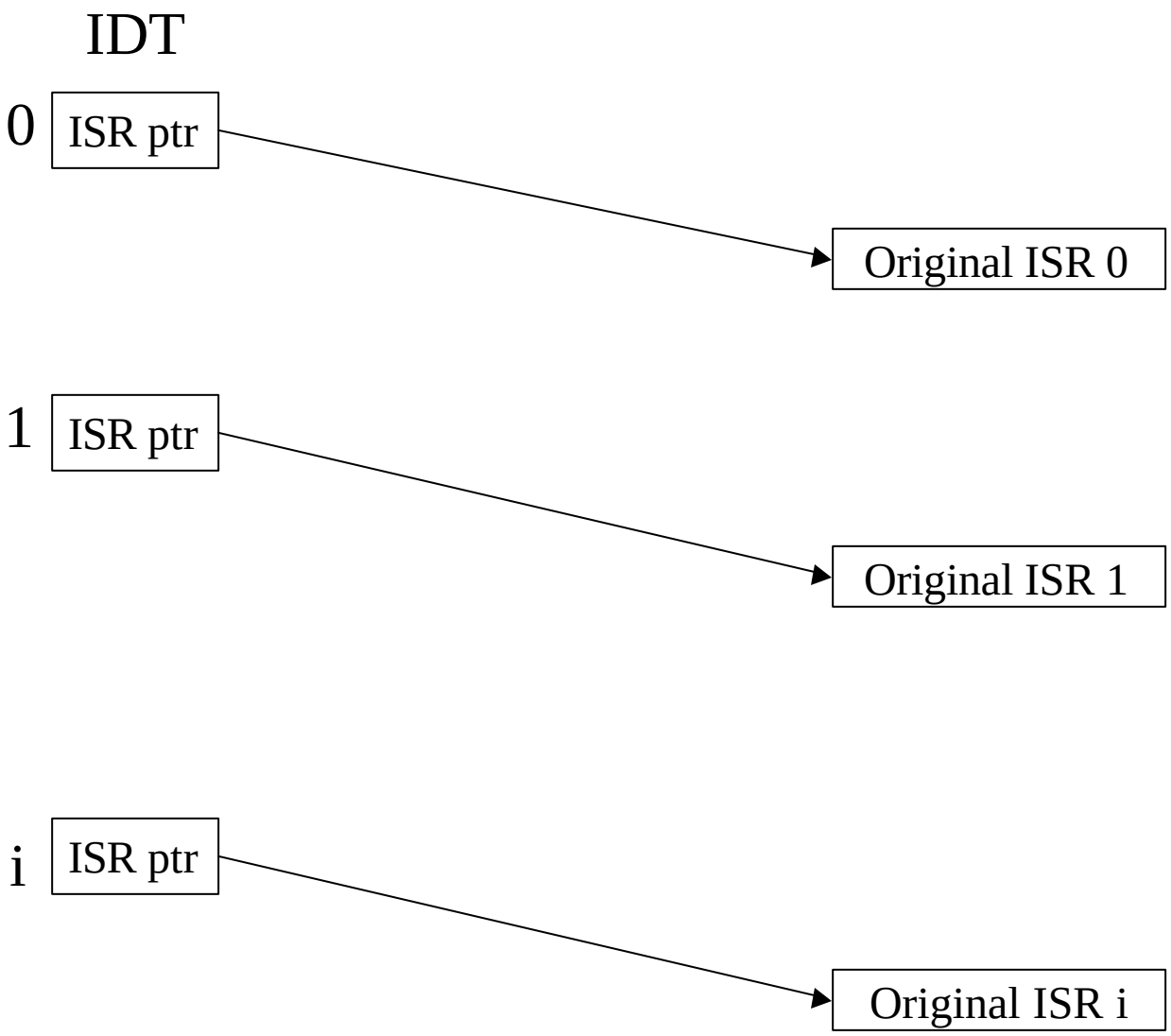
Run-time patching: Detours

- Other issues
  - Using NonPagedPool memory
    - Rootkit code resides in driver memory that can be paged out
    - Place code in non-paged memory
    - Allows driver itself to be unloaded so that it can no longer be detected
    - Rootkit driver only loaded long enough to apply patch
  - Patching addresses
    - Relative FAR JMP instruction target calculated at run-time
    - Need to patch this with desired offset at run-time

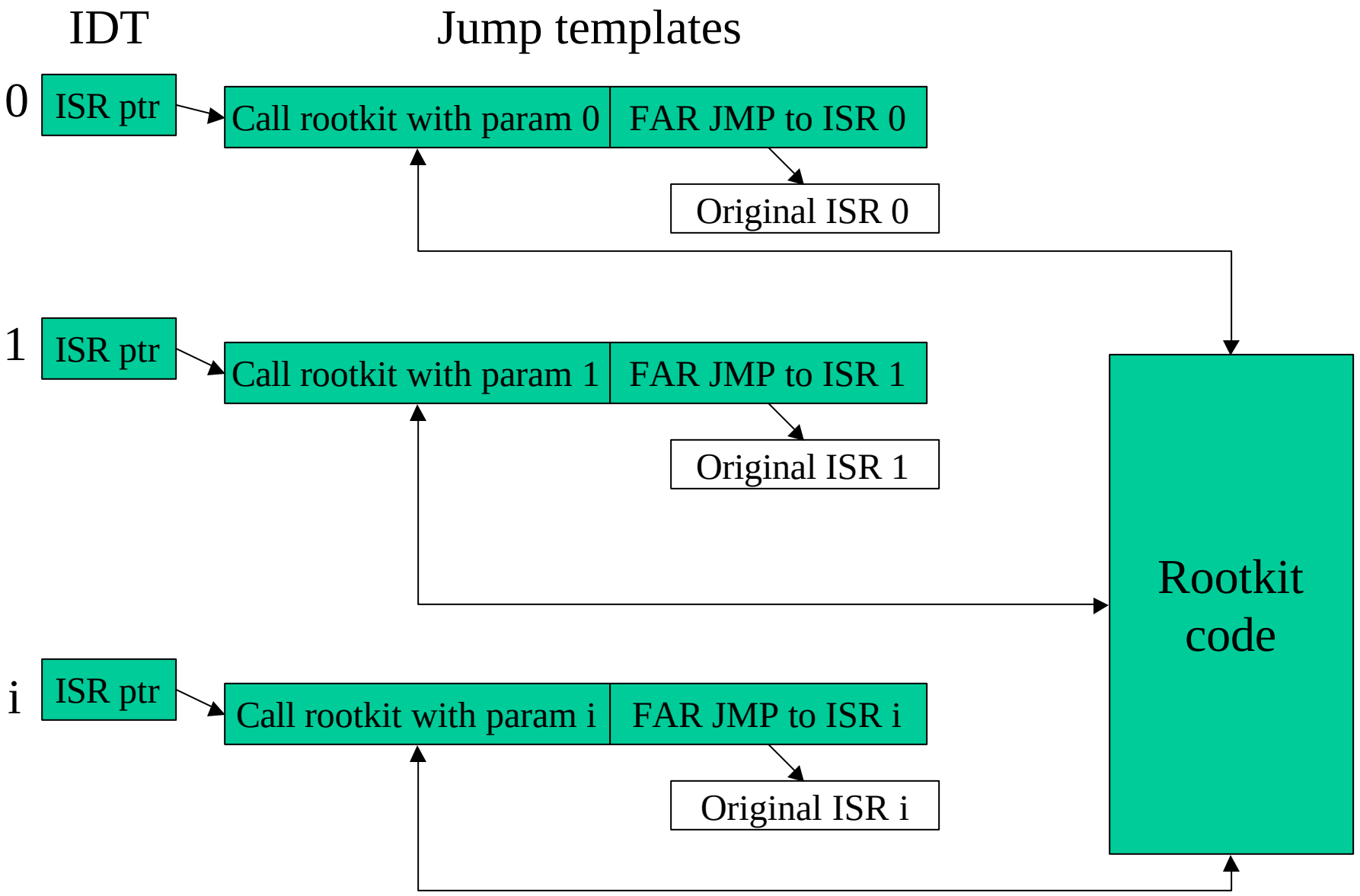
Run-time patching: Jump Templates

- Example: Hooking the Interrupt Descriptor Table (IDT)
  - Patch all entries in IDT with same detour code
  - Easier than patching every interrupt service routine (ISRs)
    - Each ISR at a different address
    - Hook every IDT entry, but include unique jump details to call back to original ISR
    - See code in book

Run-time patching: Original IDT



Run-time patching: Jump Templates



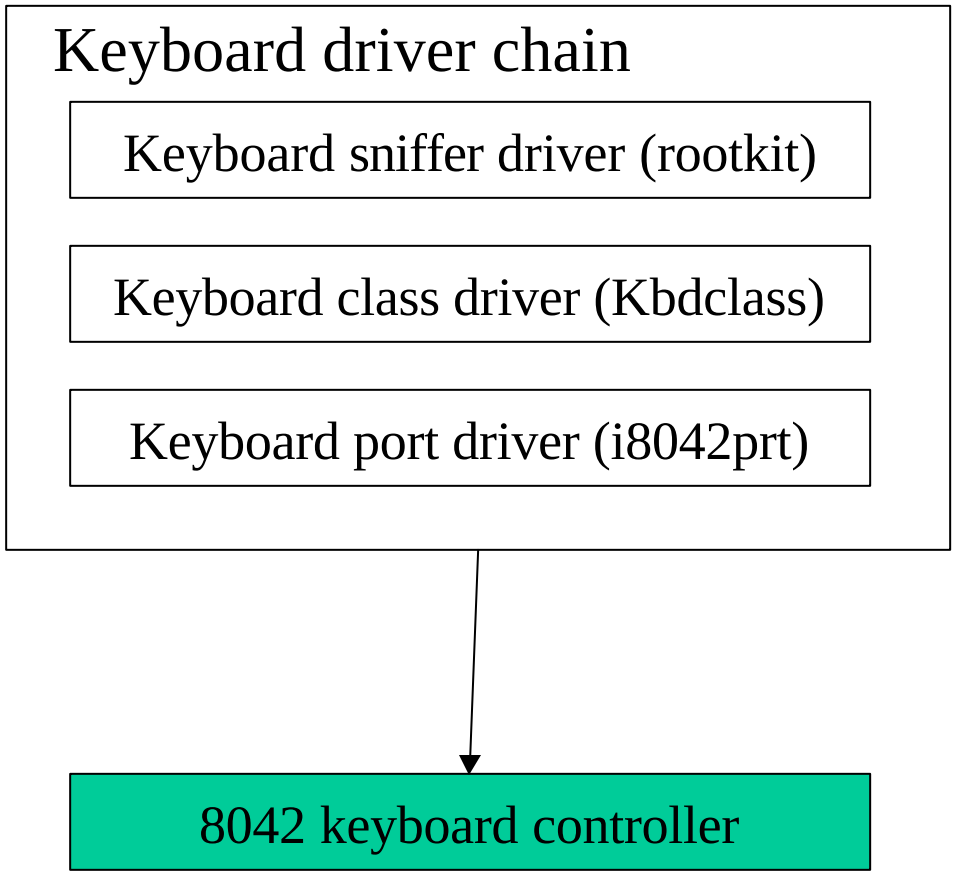
Variations

- Patching typically done at entry point
  - Easy to detect if placed in well-known place
  - Rootkit detection software often checks first 20 bytes of a function only
  - Solution: patch deep into function
- Good locations
  - Unique code byte strings (no false hits)
  - Within authentication functions
  - Kernel functions
    - Integrity-checking functions
    - Loader program that loads the kernel itself
    - Network functions
    - Firmware and BIOS

Layered drivers

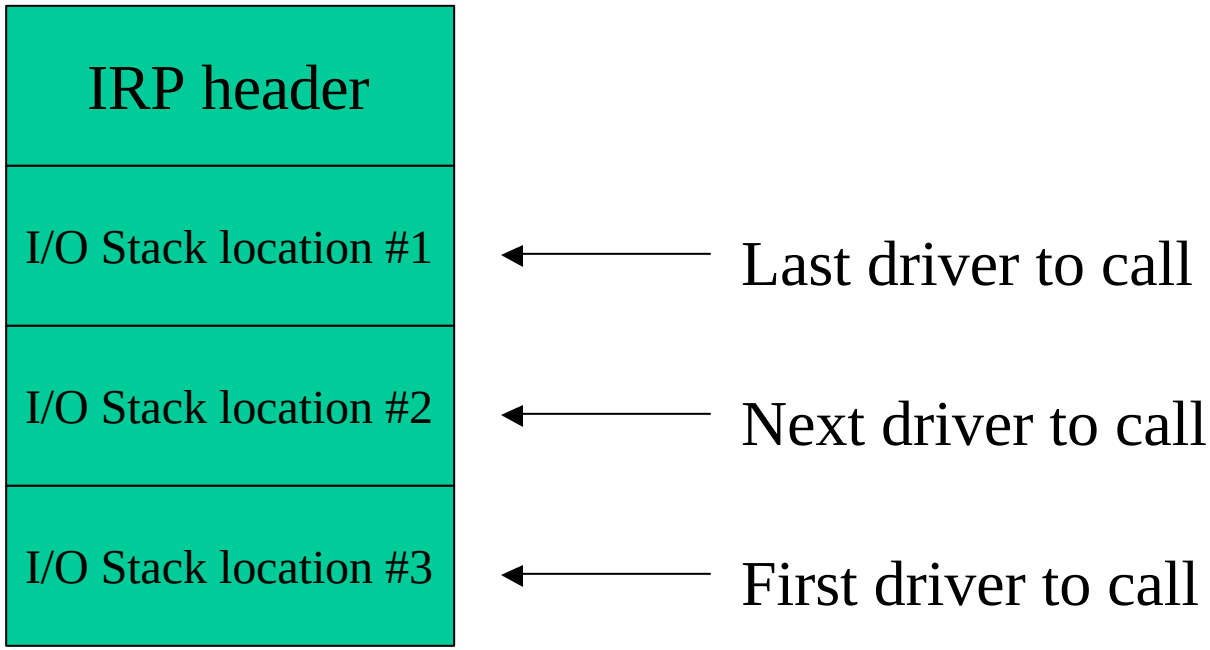
- Ability to chain multiple drivers to avoid re-implementing functions that can be shared
- Example chain: keyboard drivers
  - Lowest-level driver deals with direct access to bus and hardware device
  - Next level deals with data formatting and error codes
  - Each level intercepts data from lower level, modifies it, and passes it on to higher level
  - Perfect for rootkits!

Keyboard chain example



Details

- IRP (I/O request packet)
  - Contains stack specifying routines of the driver chain
  - I/O manager creates IRP and fills in IRP based on number of drivers in driver chain
  - Inject keyboard sniffer in chain, IRP automatically updated
  - Example: KLOG



File filters

- Used for stealth
  - Rootkits store files in file system that must remain hidden
- Common approach
  - Hooking SSDT to hide local files
  - Does not hide files mounted via SMB
- Use layered file system drivers to hide all rootkit files
  - Install hook on all available drive letters (HookDriveSet in book)
  - Rootkit parses file name in QueryDirectory.FileInformationClass QueryBuffer
    - Deletes entries associated with rootkit