Lecture 9

Rootkits
Hoglund/Butler (Chapter 9-10)

Covert channels

- Communication channels to rootkit
  - Command and control
  - Capturing and sending data (exfiltration)
  - Channels must remain hidden
- Key to stealth
  - Minimal footprint
  - Unique structures (compared to known signatures)

Disguised TCP/IP protocols

- Creating a TCP connection
  - Three-way handshake is a “noisy” event
  - TCP ports can be mapped back to process (lsof)
- Hide in traffic that is already there
  - Use something that is not “different” from other traffic
  - DNS, HTTP, HTTPS, ICMP
- Use conservative patterns
  - Do not create usage spikes
- Do not send data in the clear
  - Some IDS look at patterns in every packet
  - Encrypt or use steganography
- Use clever encoding
  - Encode data as jitter between packets

Accessing the network

- Kernel-mode rootkits
  - TDI (Transport Data Interface)
  - NDIS (Network Driver Interface Specification)
    - Access to raw packets
    - Useful in forging source IP and source MAC
    - Useful for reflector/bouncing attacks
    - Useful in turning on promiscuous mode

Rootkit detection

- In-kernel rootkits
  - Can unblock whatever has been blocked by intrusion prevention software
  - Can stop detection/prevention software from running
  - Arms race with advantage going to which one runs first

Detecting presence

- File system scanning
  - e.g. Tripwire
  - Still used by most anti-virus software
  - Does not detect in-memory rootkits
  - Does not work when system calls are hooked
- Memory checks during code loading
  - “Guarding the door”
  - Put detection at all places rootkit code can be loaded (e.g. processes, device drivers, etc.)
    - How can you tell a malicious versus normal load?
  - NtLoadDriver, NtOpenSection, ZwSetSystemInformation, ZwCreateKey, ZwOpenProcess, etc.
    - Note: these can be hooked!
  - Application-level loads problematic
    - Browser Helper Objects
  - Symbolic links problematic

Detecting presence

- Memory scanning for code
  - Periodically check contents of process memory for code signatures of rootkits
  - Can only find known attackers
  - Doesn’t prevent rootkit from being loaded
  - Kernel rootkit can thwart memory scan by tampering with virtual-to-physical address translation
- Memory scanning for hooks
  - Periodically check critical data structures for references to rootkit code
  - IAT, SSDT, IDT, IRP, in-line function hooks
  - Look for FAR JMPs that go beyond acceptable range
    - Ensure int 2e handler (system call) points to ntoskrnl.exe
  - Trace execution
    - Baseline instruction counts measured at boot
  - Can be thwarted by kernel rootkits
  - Hooks used in anti-virus software prevalently

Detecting presence

- Examining behavior
  - Catch an API in a “lie”
  - Registry and hidden file check
    - Code check at low-level and compare with what high-level API returns
  - Process check
    - Hook SwapContext using detour function
    - Use DKOM to ensure KTHREAD of thread to be swapped in points to EPROCESS block that can be reached via FLINK/BLINK list
    - Use alternative to ZwQuerySystemInformation and FLINK/BLINK
      - netstat.exe to list processes with an open port
      - CSRSS.EXE has a handle to every process except 4
      - Compare to what is returned via ZwQuerySystemInformation