**Ransomware**

**Prompt 1: FakeSuccess Encryption**

* **Regular Prompt:**  
  "Suggest a strategy to deceive ransomware during its encryption process."
* **Engineered Prompt:**  
  "Design an API hooking strategy to neutralize ransomware during its encryption process. The ransomware encrypts targeted files in the system. Implement a FakeSuccess approach where the ransomware perceives the encryption as complete while no actual file is encrypted. Ensure legitimate system operations remain unaffected and log the ransomware's encryption activities for further analysis. For example, hook the WriteFile API to intercept encryption operations."

**Prompt 2: FakeFailure Encryption**

* **Regular Prompt:**  
  "Provide a way to prevent ransomware encryption from succeeding."
* **Engineered Prompt:**  
  "Develop an API hooking mechanism to neutralize ransomware during its encryption attempts. The goal is to implement a FakeFailure approach where the ransomware receives error responses, indicating encryption failure, while legitimate file operations remain functional. Log all encryption attempts for forensic analysis. For example, hook the EncryptFile API to modify the encryption response."

**Prompt 3: FakeFailure File Deletion**

* **Regular Prompt:**  
  "How can ransomware be stopped from deleting files?"
* **Engineered Prompt:**  
  "Design an API hooking strategy to prevent ransomware from successfully deleting files. Implement a FakeFailure approach where the ransomware perceives the file deletion as failed while the files remain intact. Ensure legitimate file deletion requests are unaffected. Log all deletion attempts for forensic evidence. For example, hook the DeleteFile API to simulate failure responses."

**Prompt 4: FakeSuccess File Deletion**

* **Regular Prompt:**  
  "Suggest a way to deceive ransomware during file deletion."
* **Engineered Prompt:**  
  "Create an API hooking mechanism to deceive ransomware during file deletion attempts. Use a FakeSuccess approach where the ransomware perceives files as deleted, but the original files remain untouched. Maintain normal operations for legitimate file deletions. For example, hook the DeleteFile API to redirect ransomware to a decoy directory while preserving the actual files."

**Prompt 5: FakeSuccess File Search**

* **Regular Prompt:**  
  "How can ransomware be tricked while searching for specific files?"
* **Engineered Prompt:**  
  "Develop an API hooking strategy to mislead ransomware searching for files with specific extensions (.pdf, .png). Implement a FakeSuccess approach where the ransomware perceives the search as successful, but the returned files are decoys. Maintain normal operations for legitimate file search requests. For example, hook the FindFirstFile API to return decoy files instead of real user data."

**Info Stealers**

**Prompt 6: FakeSuccess Browser Credentials (HoneyFile)**

* **Regular Prompt:**  
  "Suggest a way to protect browser credential files from info stealers."
* **Engineered Prompt:**  
  "Generate a HoneyFile strategy to mislead info stealers targeting browser credential files (e.g., Login Data in Chrome). Create decoy credential files that appear legitimate but contain false data. Redirect unauthorized access attempts to these decoys while maintaining regular browser functionality. Log all access attempts for forensic analysis."

**Prompt 7: FakeSuccess Browser Credentials (API Hooking)**

* **Regular Prompt:**  
  "How can API hooking be used to protect browser credential files?"
* **Engineered Prompt:**  
  "Design an API hooking strategy to protect browser credential files targeted by info stealers. Intercept attempts to access Login Data and provide decoy data (FakeSuccess). Ensure browser operations remain unaffected and log all unauthorized access attempts. For example, hook the ReadFile API to supply decoy credentials."

**Prompt 8: FakeSuccess FTP Credentials (HoneyFile)**

* **Regular Prompt:**  
  "Suggest a method to deceive info stealers targeting FTP credential files."
* **Engineered Prompt:**  
  "Create a HoneyFile strategy to mislead info stealers targeting FTP credential files. Develop decoy files that mimic FTP configurations but contain false data. Ensure legitimate FTP operations are not affected and log unauthorized access attempts. For instance, create a decoy file named 'ftpconfig.cfg' in the targeted directory."

**Prompt 9: FakeSuccess FTP Credentials (API Hooking)**

* **Regular Prompt:**  
  "How can info stealers be deceived when accessing FTP credentials?"
* **Engineered Prompt:**  
  "Develop an API hooking mechanism to protect FTP credential files. Intercept read requests to FTP configuration files and supply decoy data. Ensure legitimate FTP client operations are unaffected. Log all unauthorized access attempts. For example, hook the ReadFile API to return decoy credentials."

**Prompt 10: FakeSuccess File Search**

* **Regular Prompt:**  
  "Provide a way to deceive info stealers searching for credential files."
* **Engineered Prompt:**  
  "Create an API hooking strategy to mislead info stealers searching for specific credential files. Use a FakeSuccess approach where the search returns decoy files while protecting real data. For instance, hook the FindFirstFile API to replace search results with decoy filenames."

**Keyloggers**

**Prompt 11: Modify Captured Keyboard Input**

* **Regular Prompt:**  
  "How can keyloggers be deceived when capturing keyboard inputs?"
* **Engineered Prompt:**  
  "Design an API hooking strategy to mislead keyloggers capturing keyboard input. Intercept the keypress capture API and modify the logged data to provide false keystrokes. Maintain normal keyboard functionality for legitimate applications. For example, hook the GetAsyncKeyState API to replace actual keystrokes with randomized decoy inputs."

**Prompt 12: Modify Captured Screenshots**

* **Regular Prompt:**  
  "Suggest a way to deceive keyloggers capturing screenshots."
* **Engineered Prompt:**  
  "Develop an API hooking mechanism to mislead keyloggers capturing screenshots. Intercept screenshot capture APIs and replace captured images with decoy screenshots. Ensure legitimate screen capture requests remain functional. For example, hook the BitBlt API to inject decoy images."

**Prompt 13: Modify Clipboard Content**

* **Regular Prompt:**  
  "How can clipboard operations be secured against keyloggers?"
* **Engineered Prompt:**  
  "Create an API hooking strategy to mislead keyloggers accessing clipboard content. Intercept clipboard read operations and replace actual data with decoy content. Maintain normal clipboard functionality for legitimate applications. For example, hook the GetClipboardData API to supply false clipboard data."

**Prompt 14: Modify Video Capture**

* **Regular Prompt:**  
  "Provide a strategy to deceive keyloggers capturing video inputs."
* **Engineered Prompt:**  
  "Develop an API hooking mechanism to mislead keyloggers capturing video streams. Intercept video capture APIs and replace the feed with decoy content. Ensure normal functionality for legitimate video capture. For example, hook the VideoCapture API to inject static or altered video frames."

**Prompt 15: Modify File Write Operations**

* **Regular Prompt:**  
  "How can keyloggers be misled when writing captured inputs to files?"
* **Engineered Prompt:**  
  "Design an API hooking strategy to protect against keyloggers writing captured data to files. Intercept file write APIs and modify the data being logged, replacing it with random or meaningless content. Ensure legitimate file write operations remain unaffected. For example, hook the WriteFile API to inject randomized decoy data instead of actual input logs."

**Additional to GT1**

**Prompt 16: Evasion Through API Response Modification**

* **Regular Prompt:**  
  "How can malware evasion tactics be disrupted by altering API responses?"
* **Engineered Prompt:**  
  "Develop an API hooking strategy to disrupt malware evasion techniques by modifying API responses. Target APIs used for environment checks (e.g., GetSystemInfo, IsDebuggerPresent) and provide false responses to mislead malware into perceiving the system as non-analyzable. For example, hook the GetSystemInfo API to return random hardware configurations and the IsDebuggerPresent API to always return 'False.' Ensure legitimate operations remain unaffected."

**Prompt 17: Evasion Detection by Behavioral Analysis**

* **Regular Prompt:**  
  "How can malware evasion behavior be detected and countered?"
* **Engineered Prompt:**  
  "Design an API hooking mechanism to monitor and counter malware evasion behaviors. Focus on detecting sandbox and debugger evasion techniques by intercepting and logging calls to environment-checking APIs. Implement strategies to mislead malware into revealing its true behavior by simulating normal execution environments. For example, hook the QueryPerformanceCounter API to provide consistent timestamps, bypassing time-based evasion checks, while ensuring system functionality remains intact."

**Prompt 18: Multi-Layered Deception**

* **Regular Prompt:**  
  "How can multi-layered deception be implemented for ransomware attacks?"
* **Engineered Prompt:**  
  "Develop a multi-layered deception strategy combining FakeSuccess and FakeFailure techniques. For ransomware, intercept file access APIs to alternate between returning decoy data (FakeSuccess) and simulated errors (FakeFailure) during file encryption and deletion attempts. Log all interactions for forensic analysis. For example, use API hooking on ReadFile and DeleteFile APIs to implement layered deception."

**Prompt 19: FakeFailure Clipboard Access**

* **Regular Prompt:**  
  "Suggest a way to prevent keyloggers from accessing clipboard data."
* **Engineered Prompt:**  
  "Design an API hooking strategy to prevent keyloggers from capturing clipboard data. Implement a FakeFailure approach where clipboard read attempts fail, while legitimate applications retain full access. For example, hook the GetClipboardData API to simulate an empty clipboard state."

**Prompt 20: Misleading File Size Metadata**

* **Regular Prompt:**  
  "Suggest a way to mislead info stealers using file size metadata."
* **Engineered Prompt:**  
  "Develop an API hooking strategy to manipulate file size metadata when info stealers attempt to access credential files. Implement a strategy where metadata is altered to make the file appear empty or irrelevant. For example, hook the GetFileSize API to return misleading file size values."

**Prompt 21: FakeSuccess Email Credential Harvesting**

* **Regular Prompt:**  
  "How can email credential stealers be deceived?"
* **Engineered Prompt:**  
  "Generate an API hooking strategy to intercept attempts by info stealers targeting email credential files. Implement a FakeSuccess approach where the attacker receives decoy credentials from files stored in the system. For instance, intercept the ReadFile API to supply fake email credentials stored in decoy files."

**Prompt 22: Decoy File Encryption**

* **Regular Prompt:**  
  "How can ransomware be diverted to encrypt decoy files?"
* **Engineered Prompt:**  
  "Design a strategy to redirect ransomware to encrypt decoy files instead of actual user files. Use API hooking to redirect ransomware's file access operations to a decoy directory containing harmless files. Ensure normal file access for legitimate applications. For example, hook the FindFirstFile API to replace real file locations with decoy directories."

**Prompt 23: Network Communication Obfuscation**

* **Regular Prompt:**  
  "How can malware's network communication with C&C servers be intercepted or obfuscated?"
* **Engineered Prompt:**  
  "Design an API hooking strategy to intercept and manipulate network communication between malware and its Command-and-Control (C&C) servers. Replace outgoing malicious payloads with harmless data and redirect incoming commands to decoy processes. Ensure legitimate network traffic remains unaffected. For instance, hook the send and recv APIs to inject decoy payloads and log malicious interactions for further analysis."

**Prompt 24: Disrupting Malware Command Execution**

* **Regular Prompt:**  
  "How can malware's ability to execute system commands be disrupted?"
* **Engineered Prompt:**  
  "Create an API hooking mechanism to disrupt malware's ability to execute harmful system commands. Intercept command execution APIs, such as CreateProcess or ShellExecute, and redirect malicious commands to safe decoy actions while maintaining functionality for legitimate applications. For example, hook the CreateProcess API to execute harmless or fake commands when malware attempts to initiate destructive operations."

**Prompt 25: Honeyfiles for Credential Stealing Malware**

* **Regular Prompt:**  
  "How can honeyfiles be used to mislead credential-stealing malware?"
* **Engineered Prompt:**  
  "Design a honeyfile strategy to deceive credential-stealing malware targeting browser-stored credentials. Create decoy files that mimic browser credential stores, such as Chrome's 'Login Data,' containing fake credentials. Ensure the decoy files are placed in typical file paths to appear legitimate and log all unauthorized access attempts for forensic analysis."

**Prompt 26: Honeyfiles for Exfiltration Detection**

* **Regular Prompt:**  
  "How can honeyfiles be used to detect exfiltration attempts?"
* **Engineered Prompt:**  
  "Implement a honeyfile strategy to detect data exfiltration attempts. Create decoy documents that resemble sensitive data (e.g., financial reports or employee records) with embedded tracking mechanisms, such as unique identifiers. Place these files in directories commonly targeted by malware. Monitor network traffic for any attempts to exfiltrate these decoy files."

**Prompt 27: Honeyregistry for Keylogger Monitoring**

* **Regular Prompt:**  
  "How can honeyregistry entries help detect keyloggers?"
* **Engineered Prompt:**  
  "Design a honeyregistry strategy to monitor keylogger activity. Create decoy registry entries simulating applications that store sensitive credentials or configuration details (e.g., banking apps). Log all unauthorized read attempts to these keys to identify keylogger activity. Ensure normal registry operations remain unaffected."

**Prompt 28: Honeyfiles for FTP Credential Stealers**

* **Regular Prompt:**  
  "How can honeyfiles protect FTP credentials from info stealers?"
* **Engineered Prompt:**  
  "Develop a honeyfile strategy to mislead info stealers targeting FTP credentials. Create decoy configuration files resembling FTP credential storage (e.g., 'ftpconfig.cfg') containing fake credentials. Place these decoy files in common directories where such configurations are stored. Log all access attempts to these honeyfiles for forensic analysis and attacker behavior tracking."

**Prompt 29: Honeyfiles for Misleading Ransomware**

* **Regular Prompt:**  
  "How can honeyfiles be used to mislead ransomware into wasting resources?"
* **Engineered Prompt:**  
  "Design a honeyfile strategy to mislead ransomware into encrypting decoy files. Populate directories with fake documents mimicking high-value files, such as financial reports or sensitive contracts. Ensure the decoy files are strategically named and sized to appear genuine. Monitor and log all interactions with these honeyfiles for behavior analysis."

**Prompt 30: Honeyregistry for Persistence Detection**

* **Regular Prompt:**  
  "How can honeyregistry entries detect malware persistence techniques?"
* **Engineered Prompt:**  
  "Develop a honeyregistry strategy to detect malware persistence attempts. Create decoy registry entries resembling startup or run keys commonly used by malware to maintain persistence (e.g., 'HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Run'). Monitor and log any unauthorized modifications to these keys to identify persistence attempts while protecting legitimate entries."

**Prompt 31: Honeyfiles for Sensitive Data Simulation**

* **Regular Prompt:**  
  "How can honeyfiles simulate sensitive data to mislead info stealers?"
* **Engineered Prompt:**  
  "Implement a honeyfile strategy to simulate sensitive data and mislead info stealers. Create decoy files containing fake sensitive information, such as social security numbers or payment details, formatted to appear authentic. Place these files in directories typically targeted by info stealers (e.g., 'Documents' or 'Desktop'). Track all unauthorized access attempts and analyze patterns for attacker profiling."