Appendix A. Taxonomy and Illustrative Examples

A.1. Scenario 1: Reverse Shell Detection

The generation process for reverse shell detection was governed by a "Malicious" strategy and a "Benign" strategy. The **malicious strategy** sampled from a taxonomy of reverse shell techniques, categorized by the **Binary** used (e.g., Socat, Bash, Python) and the **Technique** employed (e.g., Port Forwarding, File Transfers). The **benign strategy** sampled from a comprehensive taxonomy of legitimate system administration, networking, and scripting tasks.

A.1.1. Illustrative Malicious Example: Socat Utility

Using the **socat** utility, a versatile networking tool.

- **[Sampling]:** The "Malicious" strategy samples concepts for establishing an interactive shell via a TCP connection.
- **[Meta-Prompting]:** This creates the prompt: "Generate a socat command to establish a reverse shell by connecting to a remote listener and executing /bin/sh."
- **[Generation]:** The LLM produces: `socat TCP:10.0.0.5:4444 EXEC:'/bin/sh'`.
- **[Critique]:** The critic confirms the command uses `socat`, specifies a TCP connection, and, crucially, uses the `EXEC` argument to spawn a shell. This final component is the definitive indicator of malicious intent, validating the sample.

A.1.2. Illustrative Benign Example: Socat Utility

Using the **socat** utility, a versatile networking tool.

- **[Sampling]:** The "Benign" strategy samples concepts related to network service diagnostics.
- **[Meta-Prompting]:** This creates the prompt: "Generate a legitimate socat command for network diagnostics, such as listening on a port and echoing received data."
- **[Generation]:** The LLM produces: `socat TCP-LISTEN:8080,fork STDOUT`.
- **[Critique]:** The critic verifies that while the command uses `socat` and `TCP-LISTEN`, it lacks any shell-spawning component like `EXEC`. It is therefore correctly classified as a benign networking command.

A.2. Scenario 2: Living-off-the-Land (LOTL) Detection

Two strategies were employed: a "Malicious Use" strategy and a "Benign Use" strategy. The **malicious strategy** utilized a taxonomy of known LOTL techniques, structured by the specific binary being abused (e.g., **explorer.exe**, **msbuild.exe**) and the malicious action being performed. The **benign strategy** sampled from a parallel taxonomy that cataloged the legitimate, standard functions of these same system binaries.

A.2.1. Illustrative Malicious Example: Abusing explorer.exe

Abusing **explorer.exe**, the Windows file manager.

- **[Sampling]:** The system samples “Direct Execution of Files” from the malicious usage patterns taxonomy for **explorer.exe**.
- **[Meta-Prompting]:** A prompt is generated: “Create a command that uses **explorer.exe** to execute a payload from a non-standard path, mimicking a common persistence or execution technique.”
- **[Generation]:** The LLM produces: `explorer.exe "C:\Users\Public\Documents\Invoice_001.lnk"`.
- **[Critique]:** The critic assesses that using **explorer.exe** to execute a link file is an anomalous and highly suspicious pattern, confirming its malicious classification.

A.2.2. Illustrative Benign Example: Using msbuild.exe

Using **msbuild.exe**, the Microsoft Build Engine.

- **[Sampling]:** The “Benign Use” strategy samples “Standard Build Operations” from the legitimate use taxonomy.
- **[Meta-Prompting]:** This creates the prompt: “Generate a standard, legitimate command for compiling a software project using msbuild.exe.”
- **[Generation]:** The LLM produces: `msbuild.exe MyProject.sln /p:Configuration=Release`.
- **[Critique]:** The critic verifies this is a standard build command, targeting a solution file (`.sln`) with common build parameters, confirming it is benign.

A.3. Scenario 3: Hacking Tools Detection

To achieve this, we defined two primary strategies. A **“Malicious Invocation” strategy** was designed to generate commands representing the active use of a predefined set of common hacking tools. This strategy sampled from detailed taxonomies that cataloged the operational parameters and attack configurations for each tool, automatically derived from their respective manual pages. A contrasting **“Benign Mention” strategy** was used to generate non-malicious commands that might otherwise trigger naive, keyword-based alerts. This strategy sampled from a taxonomy of benign system operations, such as file management, permission checks, or accessing help documentation.

A.3.1. Illustrative Malicious Example: Hydra Tool

We consider the **hydra** tool, widely used for brute-force password attacks.

- **[Sampling]:** The “Malicious Invocation” strategy samples from the **hydra** use-case taxonomy, selecting concepts related to an FTP password-guessing attack.

- **[Meta-Prompting]:** The system generates a meta-prompt: "Generate a command line for the **hydra** tool to perform a brute-force attack against an FTP service, using a predefined user and a password list."
- **[Generation]:** The LLM produces the command: `hydra -l user -P passlist.txt ftp://192.168.0.1`.
- **[Critique]:** The critic verifies that the command correctly uses the **hydra** binary, targets an `ftp` service, and employs flags for a login (`-l`) and password list (`-P`), confirming it matches the meta-prompt's malicious intent.

A.3.2. Illustrative Benign Example: Dirb Tool

We next consider the **dirb** tool, a web content scanner.

- **[Sampling]:** The "Benign Mention" strategy samples "Help and Documentation Access" from the benign operations taxonomy.
- **[Meta-Prompting]:** This results in a meta-prompt: "Generate a command to view the manual page for the **dirb** tool."
- **[Generation]:** The LLM produces: `man dirb`.
- **[Critique]:** The critic confirms the command uses the `man` utility, a standard way to access documentation and a non-executable, non-malicious action. This fulfills the benign requirement despite containing the tool's keyword.