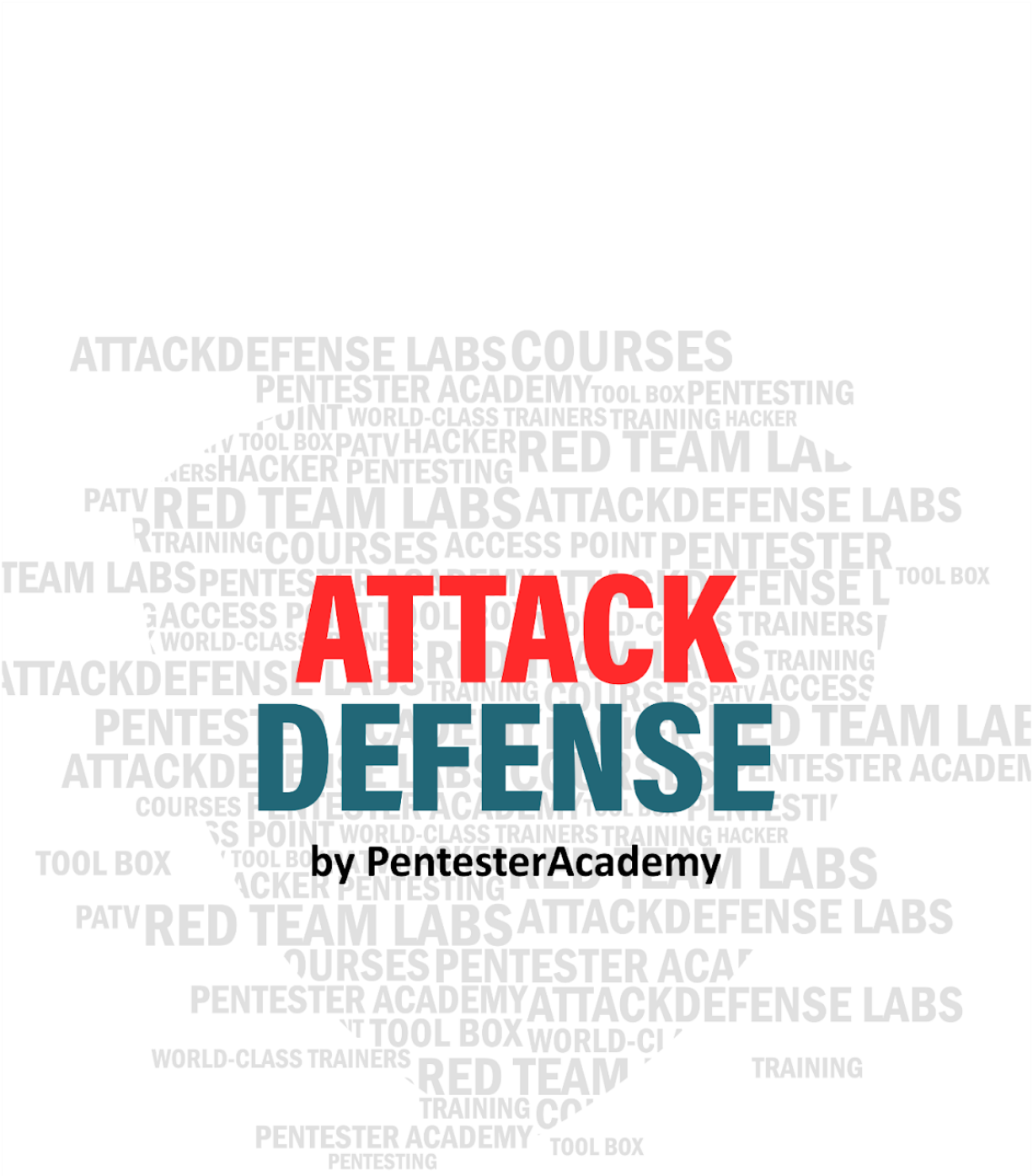
ATTACK
DEFENSE
by PentesterAcademy

Name	DevSkim: Code Security Review
URL	https://www.attackdefense.com/challengedetails?cid=2048
Type	DevSecOps Basics: Automated Code Review

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Challenge Description

DevSkim is a framework of IDE extensions and language analyzers that performs analysis on the applications and reports the vulnerabilities.

A Kali CLI machine (kali-cli) is provided to the user with Devskim installed on it. The source code for three sample web applications is provided in the home directory of the root user.

Objective: Use Devskim to find issues in the code!

Instructions:

- The source code of web applications is provided at /root/github-repos

Solution

Step 1: Check the available options of the devskim tool

Command: devskim --help

```
root@attackdefense:~# devskim  --help
Microsoft DevSkim Command Line Interface 0.1.9

Usage: devskim [options] [command]

Options:
  -?|-h|--help  Show help information
  -v|--version  Show version information

Commands:
  analyze    Analyze source code
  catalogue  Create csv file catalogue of rules
  pack       Pack rules into a single file
  test       Run tests for rules
  verify     Verify integrity and syntax of rules

Use "devskim [command] --help" for more information about a command.

root@attackdefense:~#
```

Step 2: Check the provided web applications.

Command: ls -l github-repos

```
root@attackdefense:~# ls -l github-repos/
total 12
drwxrwxr-x 7 root root 4096 Sep 15 05:42 node-app-template
drwxrwxr-x 4 root root 4096 Sep 15 05:42 python-hashes
drwxrwxr-x 3 root root 4096 Sep 15 05:42 wifi-password-cli
root@attackdefense:~#
```

We will take one example at a time and run the tool on that.

Example 1: Node App Template

Step A: Navigate to the repository and check its contents.

Commands:
cd github-repos/node-app-template
ls

```
root@attackdefense:~# cd github-repos/node-app-template/
root@attackdefense:~/github-repos/node-app-template#
root@attackdefense:~/github-repos/node-app-template# ls
app.js  bin  config.js  LICENSE.md  package.json  public  README.md  routes  views
root@attackdefense:~/github-repos/node-app-template#
```

Step B: Run the Devskim tool and analyze the source code of the application. Pass the path of the repository as an argument

Command: devskim analyze .

```
root@attackdefense:~/github-repos/node-app-template# devskim analyze .
file:./app.js
        region:41,23,41,28 - DS137138 [Moderate] - Insecure URL

file:./.git/config
        region:7,8,7,13 - DS137138 [Moderate] - Insecure URL

Issues found: 2 in 2 files
Files analyzed: 11
Files skipped: 116
root@attackdefense:~/github-repos/node-app-template#
```

Issues Detected

- Insecure URL found in multiple files

Example 2: Python Hashes

Step A: Navigate to the repository and check its contents.

Commands:
cd ~/github-repos/python-hashes
ls

```
root@attackdefense:~/github-repos# cd python-hashes/
root@attackdefense:~/github-repos/python-hashes#
root@attackdefense:~/github-repos/python-hashes# ls
hashes  LICENSE  README.md  setup.py
root@attackdefense:~/github-repos/python-hashes#
```

Step B: Run the Devskim tool and analyze the source code of the application. Pass the path of the repository as an argument

Command: devskim analyze . -s critical

```
root@attackdefense:~/github-repos/python-hashes# devskim analyze . -s critical
file:./hashes/bloom.py
        region:10,6,10,11 - DS126858 [Critical] - Weak/Broken Hash Algorithm
        region:50,49,50,54 - DS126858 [Critical] - Weak/Broken Hash Algorithm

Issues found: 2 in 1 files
Files analyzed: 8
Files skipped: 35
root@attackdefense:~/github-repos/python-hashes#
```

Issues Detected

- Weak Hash Algorithm is used in the application.

Example 3: Wifi Password CLI

Step A: Navigate to the repository and check its contents.

Commands:
cd ~/github-repos/wifi-password-cli
ls

```
root@attackdefense:~/github-repos# cd wifi-password-cli/
root@attackdefense:~/github-repos/wifi-password-cli#
root@attackdefense:~/github-repos/wifi-password-cli# ls
cli.js  license  package.json  readme.md  test.js
root@attackdefense:~/github-repos/wifi-password-cli#
```

Step B: Run the Devskim tool and analyze the source code of the application. Pass the path of the repository as an argument

Command: devskim analyze .

```
root@attackdefense:~/github-repos/wifi-password-cli# devskim analyze .
file:./cli.js
        region:5,14,5,95 - DS117838 [Critical] - Do not store tokens or keys in source code.

file:./.git/config
        region:7,8,7,13 - DS137138 [Moderate] - Insecure URL

Issues found: 2 in 2 files
Files analyzed: 3
Files skipped: 35
root@attackdefense:~/github-repos/wifi-password-cli#
```

Issues Detected

- Hardcoded token or keys in the source code
- Insecure URL found in the git configuration.

Learnings

Perform Automated code review using the Devskim tool.