Automating Reconnaissance

By Jackson Long, Robert Skelton, John Wallace

What is Digital Reconnaissance?

Digital Reconnaissance refers to the process of gathering information on a target through various means, such as social media, forums, and various scans. This is also known as:

Open-source Intelligence Gathering (OSINT)

In a legitimate context, digital recon can be used by security pros to identify and address vulnerabilities in a company's network, or by law enforcement to gather evidence for criminal investigations.

All In One!

- After doing penetration testing, we had to utilize many different command line tools manually for reconnaissance, but we felt as if this seemed a bit redundant.
- We wanted to automate this phase of pentesting and we wanted to ensure consistency.
- This script can also be standardized across an organization, ensuring that all members of the team are using the same process to perform reconnaissance, which can help reduce the risk of errors.
- The end goal of the recon88 script is to help streamline the reconnaissance phase and integrate multiple scans into one quickly accessible script.

What is Recon88.sh?

- Recon88 uses a multitude of command line tools, from aggressive scans to a soft server vulnerability finder
- It checks if commands are installed, if not, it installs what is needed.
- Then it resolves the URL to its IP addresses and runs various tools.
- Tools included: NMAP, Shodan, dirb, nikto, wpscan, and subfinder
- The results of each tool are saved into a new file and corresponding directory so you can quickly access your scan results.

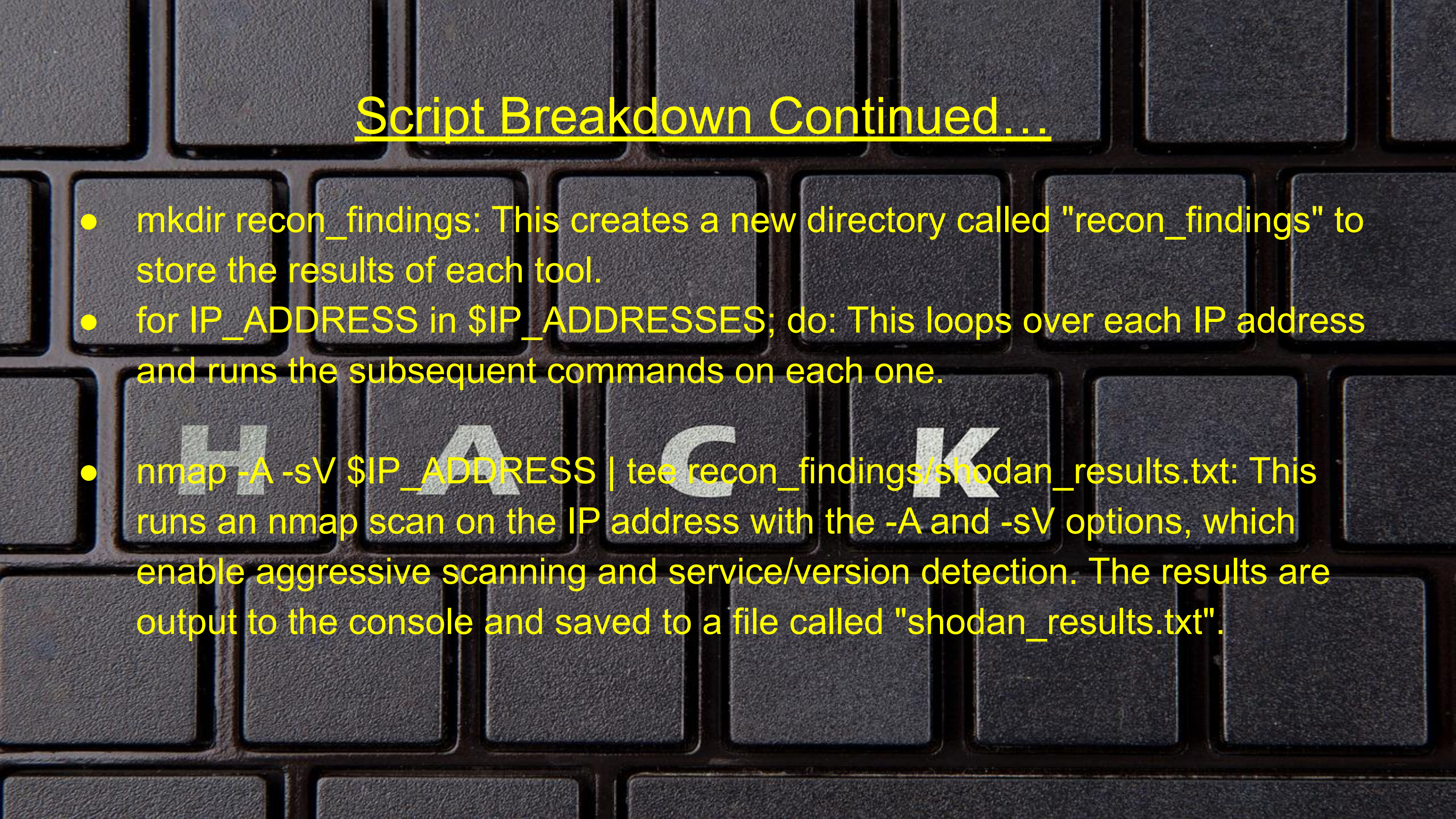
Summary of Tool Use

Overall, this script automates the reconnaissance phase of a web application penetration test and saves the results in separate files for easy analysis. The script also includes error handling for cases where the IP address cannot be resolved or the necessary commands cannot be installed. However, it should be noted that some of the tools used in this script (such as Nmap, Nikto, and Wpscan) may generate a significant amount of network traffic and should be used with caution on production systems.

Script Breakdown

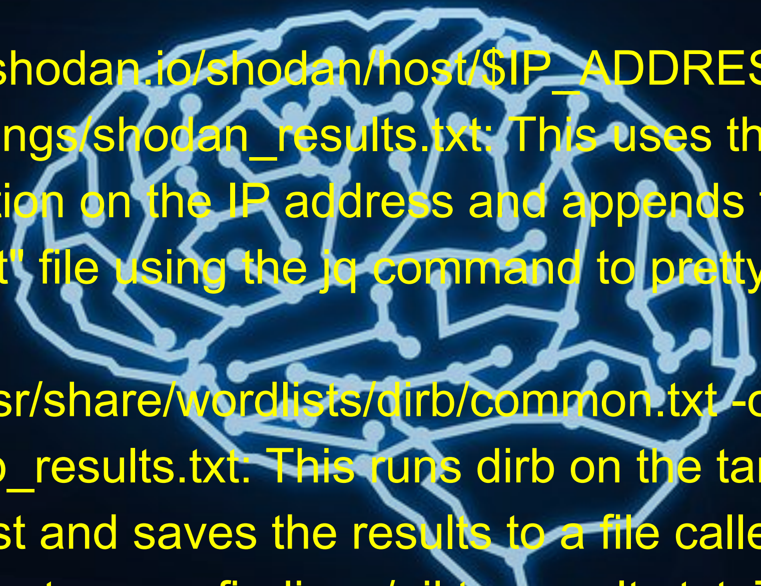
- set -e: This tells the script to exit immediately if any command exits with a non-zero status.
- if [$# -ne 2]: This checks if two arguments (URL and Shodan API key) are provided when the script is run.
- if ! command -v <command> &> /dev/null: This checks if a specific command (such as host or jq) is installed and, if not, installs it.
- IP_ADDRESSES=$(host $URL | awk '/has address/ { print $4 }'): This resolves the URL to its IP addresses using the host command and saves the result in a variable called IP_ADDRESSES

Script Breakdown Continued...

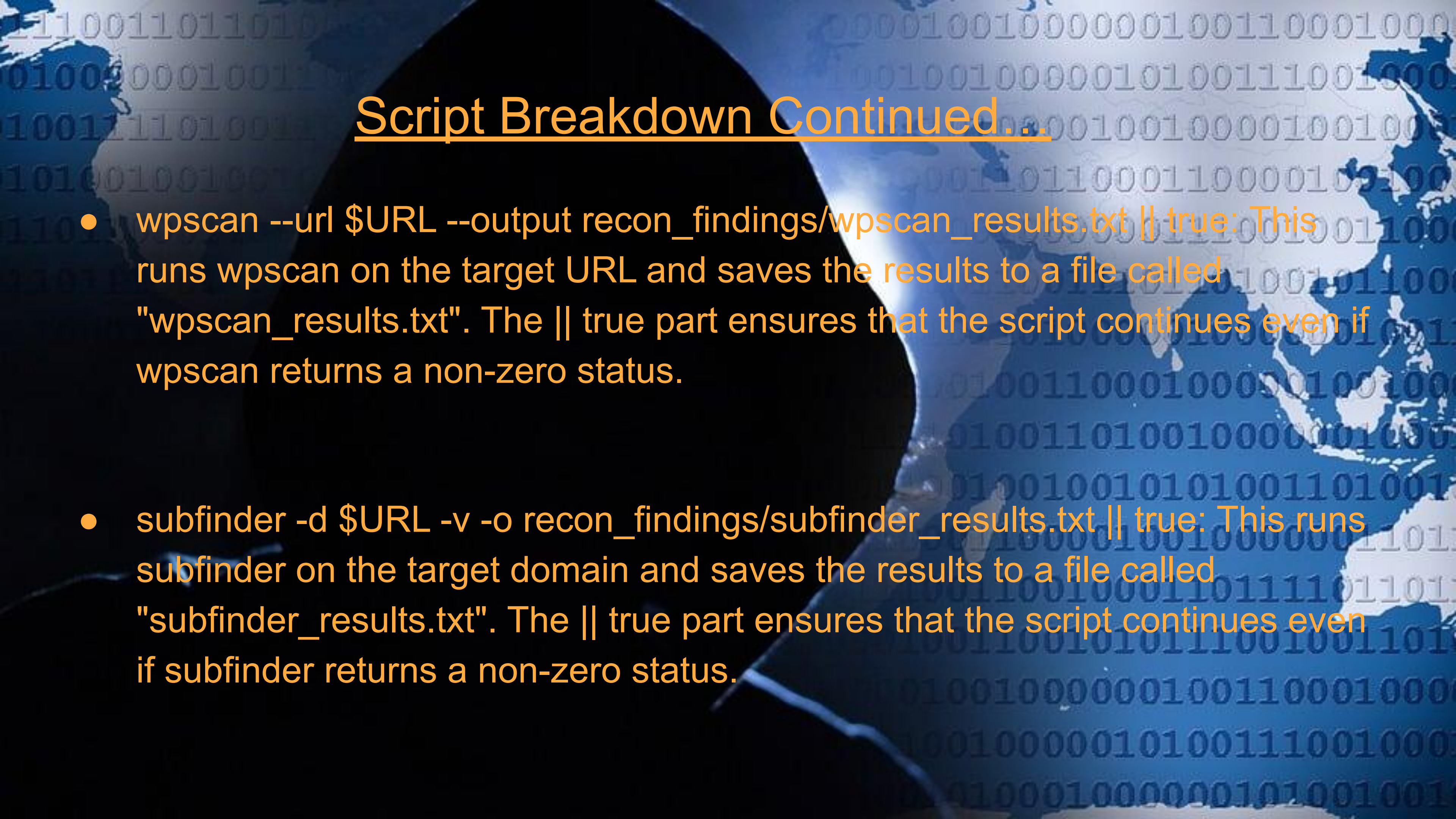


- mkdir recon_findings: This creates a new directory called "recon_findings" to store the results of each tool.
- for IP_ADDRESS in $IP_ADDRESSES; do: This loops over each IP address and runs the subsequent commands on each one.
- nmap -A -sV $IP_ADDRESS | tee recon_findings/shodan_results.txt: This runs an nmap scan on the IP address with the -A and -sV options, which enable aggressive scanning and service/version detection. The results are output to the console and saved to a file called "shodan_results.txt".

Script Breakdown Continued…



- curl -s "https://api.shodan.io/shodan/host/$IP_ADDRESS?key=$API_KEY" | jq '.' >> recon_findings/shodan_results.txt: This uses the Shodan API to search for information on the IP address and appends the results to the "shodan_results.txt" file using the jq command to pretty-print the JSON output.
- dirb http://$URL /usr/share/wordlists/dirb/common.txt -o recon_findings/dirb_results.txt: This runs dirb on the target URL using the common.txt wordlist and saves the results to a file called "dirb_results.txt".
- nikto -h $URL -output recon_findings/nikto_results.txt: This runs nikto on the target URL and saves the results to a file called "nikto_results.txt".



Script Breakdown Continued...

- wpscan --url $URL --output recon_findings/wpscan_results.txt || true: This runs wpscan on the target URL and saves the results to a file called "wpscan_results.txt". The || true part ensures that the script continues even if wpscan returns a non-zero status.

- subfinder -d $URL -v -o recon_findings/subfinder_results.txt || true: This runs subfinder on the target domain and saves the results to a file called "subfinder_results.txt". The || true part ensures that the script continues even if subfinder returns a non-zero status.

Steps Taken

- We used Github repositories and Kali.org for the tools inside our script.
- Syntax for the script was made by us (John, Tracy, Jackson), but we checked syntax for errors and issues with ChatGPT. ChatGPT allowed us to make sure that our syntax was mostly error free, we still had to manually error check and see where things were failing.
- Subfinder: https://www.kali.org/tools/subfinder/
- Dirb: https://github.com/v0re/dirb
- Nikto: https://github.com/sullo/nikto
- Wpscan: https://www.kali.org/tools/wpscan/

Script Demonstration!