Domain Mapper: Comprehensive Script Explanation

Introduction to Domain Mapper

Domain Mapper is an advanced cybersecurity script designed for comprehensive network reconnaissance and potential vulnerability assessment. This powerful tool automates the process of scanning, enumerating, and potentially exploiting network targets, making it invaluable for penetration testers and security professionals.

Key features of the script include:

- Modular design with distinct phases: Scanning, Enumeration, and Exploitation
- Multiple intensity levels for each phase, allowing for basic to advanced assessments
- Integration with popular security tools like Nmap, Masscan, CrackMapExec, and John the Ripper
- Automated installation of required tools
- User-friendly interface with color-coded output for easy interpretation
- Comprehensive PDF reporting for documentation and analysis

Note: This script is intended for authorized use only. Always ensure you have explicit permission before running security assessments on any network or system.

Detailed Function Breakdown

1. colors()

```
function colors(){
    RED='\033[0;31m'
    GREEN='\033[0;32m'
    YELLOW='\033[0;33m'
    NC='\033[0m' # No Color
}
```

This function defines ANSI color codes for output formatting, enhancing the readability of script messages throughout execution.

2. d_figlet()

```
function d_figlet() {
    if ! command -v figlet &> /dev/null 2>&1; then
        echo -e "${RED}[-]${NC} Figlet is not installed, start installing figlet."
        echo -e "${YELLOW}[!]${NC} Please be patient, It might take a while (2 minutes)"
        sudo apt update &> /dev/null 2>&1;
        sudo apt install figlet -y &> /dev/null 2>&1;
    fi
    figlet "Domain mapper"
    echo "[#] Hello! and welcome to the domain mapper"
}
```

This function checks for and installs the 'figlet' tool if not present, then uses it to display an ASCII art banner of the script's name, providing a visually appealing introduction.

3. root()

```
function root(){
    echo "[#] Please make sure you run this script with root account"
    if [[ ${id -u} != 0 ]]; then
        echo -e "${RED}[-]${NC} Please run the script with root account"
        exit 1
    else
        echo -e "${GREEN}[+]${NC} You will move forward to start scanning your target, Enjoy!"
    fi
}
```

This function ensures the script runs with root privileges, which are necessary for certain network operations and tool installations. It exits if not run as root.

4. folder+target()

```
function folder+target(){
    TS=${date +%H:%M}
    DM="Domain_mapper_results_$TS"
    mkdir -p $DM
    cd $DM
    report_file="$DM/audit_file.$TS.txt"

    validate_ip() {
        local ip=$1
        local cidr=32
        local ip_regex="^([0-9]{1,3}\.){3}[0-9]{1,3}$"
        local cidr_regex="^([0-9]{1,3}\.){3}[0-9]{1,3}/([0-9]|[1-2][0-9]|3[0-2])$"
        if [[ $ip =~ $ip_regex ]] || [[ $cidr =~ $cidr_regex ]]; then
            return 0
        else
            return 1
        fi
    }

    while true; do
        read -p "[?] Please Enter a valid IP address for your target [network/host]: " target
        if validate_ip "$target" "$target"; then
            echo -e "${GREEN}[+]${NC} Your target IP address is: $target"
            break
        else
            echo -e "${RED}[-]${NC} Your IP address input is NOT valid, please enter a valid IP addres
        fi
    done
}
```

This function creates a timestamped folder for storing results and prompts the user for a valid target IP address or CIDR range. It includes input validation to ensure a correct IP format.

5. Tool Installation Functions

Several functions follow a similar pattern to check for and install required tools:

```
function d_python() {
    if ! command -v python3 &> /dev/null 2>&1; then
        echo -e "${RED}[-]${NC} python3 is not installed"
        echo "[#] start installing python3"
        sudo apt install python3-impacket -y &> /dev/null 2>&1;
    else
        echo -e "${GREEN}[+]${NC} python3 is installed!"
    fi
}

# Similar functions: d_nmap(), d_masscan(), d_john(), d_enscript(), d_ghostscript(), e_crackmapexec()
```

These functions ensure all necessary tools (Python3, Nmap, Masscan, John the Ripper, Enscript, Ghostscript, and CrackMapExec) are installed before proceeding with the main operations.

6. scaning()

```
function scaning(){
    echo "[#] Choose the operation level for the scanning mode before any actions are executed."
    echo "[*] 1. Basic - scan with -Pn. "
    echo "[*] 2. Intermediate - scan with -p- (all ports). "
    echo "[*] 3. Advanced - Including UDP scan."
    read -p "[?] Select operation level for Scanning Mode (1-3): " scanning_choice

    if [ $scanning_choice == 1 ]; then
        echo "[#] Starting basic scan"
        nmap -Pn $target > $basic_scan_$TS
        Domain_ip=$(cat $basic_scan_$TS | grep -e "report for" -e "ldap" -e "kerberos" | grep -B 1 -e
    elif [ $scanning_choice == 2 ]; then
        echo "[#] Starting intermediate scan"
        nmap -Pn -p- $target > intermediate_scan_$TS
        Domain_ip=$(cat intermediate_scan_$TS | grep -e "report for" -e "ldap" -e "kerberos" | grep -
    elif [ $scanning_choice == 3 ]; then
        echo "[#] Starting advanced scan"
        ad=$(echo "$target" | grep -i "/")
        if [ "$ad" == "$target" ]; then
            echo "[#] Because you chose to scan more than one address, then Runs a scan with rate 100
            masscan -p0-65535,U:0-65535 $target --rate 1000000 > advanced_scan_1_$TS
        elif [ -z $ad ]; then
            echo "[#] Because you chose to scan one address, then Runs a scan with rate 2000"
            masscan -p0-65535,U:0-65535 $target --rate 2000 > advanced_scan_2_$TS
        fi
        Domain_ip=$(cat advanced_scan_*_$TS | grep -e "88" -e "139" | grep -Eo "\b([0-9]{1,3}\.){3}[0
    else
        echo -e "${RED}[-]${NC} You didn't choose a valid option!"
        exit
    fi

    echo "[#] scan completed"
    if [ -z "$Domain_ip" ]; then
        echo -e "${RED}[-]${NC} The Domain server not found"
    else
        echo -e "${GREEN}[+]${NC} The Domain server is at: $Domain_ip"
    fi
}
```

This function performs the scanning phase with three operation levels, using Nmap for basic and intermediate scans, and Masscan for advanced scans including UDP ports. It attempts to identify the domain controller IP based on the scan results.

7. Enumeration()

```
function Enumeration(){
    read -p "[?] Would you like also move to the Enumeration phase (Y/N): " enum
    if [ $enum == Y ] || [ $enum == y ]; then
        echo "[#] Choose the operation level for Enumeration Mode (1-3): "
        read -p "[?] Select operation level for Enumeration Mode (1-3): " enumeration_choice
        if [ $enumeration_choice == 1 ]; then
            echo "[#] Starting basic Enumeration"
            nmap -Pn -sV --script broadcast-dhcp-discover $Domain_ip > basic_enumeration_$TS
        elif [ $enumeration_choice == 2 ]; then
            echo "[#] Starting Intermediate Enumeration"
            nmap -Pn -sV --script broadcast-dhcp-discover,ldap-search,smb-enum-sessions $Domain_ip >
            nmap -p 139,445,22,21,3389,5986,5985,1433,636 -sV --open $Domain_ip > crack_$TS
            # ... (additional enumeration steps with CrackMapExec)
        elif [ $enumeration_choice == 3 ]; then
            echo "[#] Starting Advanced Enumeration"
            # ... (advanced enumeration steps, including comprehensive CrackMapExec usage)
        else
            echo -e "${RED}[-]${NC} You didn't choose a valid option!"
            exit
        fi
    else
        echo "[#] OK, You chose to not move on to the Enumeration phase"
        exit
    fi
}
```

This function performs enumeration with increasing levels of depth, using tools like Nmap and CrackMapExec. It includes options for basic DHCP discovery, intermediate enumeration of users and shares, and advanced enumeration including password policies and admin group members.

8. Exploitation()

```
function Exploitation(){
    if [ $enumeration_choice == 1 ]; then
        echo -e "${YELLOW}[!]${NC} You can't move to the exploitation, Because you don't have enough i
        read -p "[?] Would you like to At least do an Nmap scan with vulnerability script? (Y/N): " vn
        if [ $vul == Y ] || [ $vul == y ]; then
            nmap -Pn -sV --script=vuln $Domain_ip > vuln_scan_$TS
        fi
        exit
    fi

    read -p "[?] Would you like also move to the Exploitation phase (Y/N): " expl
    if [ $expl == Y ] || [ $expl == y ]; then
        echo "[#] Choose the operation level for Exploitation Mode (1-3): "
        read -p "[?] Select operation level for Exploitation Mode (1-3): " Exploitation_mode
        if [ $Exploitation_mode == 1 ]; then
            echo "[#] Starting Basic Exploitation"
            nmap -Pn -sV --script=vuln $Domain_ip > vuln_scan_$TS
        elif [ $Exploitation_mode == 2 ]; then
            echo "[#] Starting Intermediate Exploitation"
            nmap -Pn -sV --script=vuln $Domain_ip > vuln_scan_$TS
            # ... (password spraying with CrackMapExec)
        elif [ $Exploitation_mode == 3 ]; then
            echo "[#] Starting Advanced Exploitation"
            # ... (advanced exploitation steps, including secretsdump.py and John the Ripper)
        else
            echo -e "${RED}[-]${NC} You didn't choose a valid option!"
        fi
    else
        echo "[#] OK, You chose to not move on to the Exploitation phase"
        exit
    fi
}
```

This function attempts exploitation based on gathered information, with increasing levels of intensity. It includes vulnerability scanning, password spraying, and potential extraction and cracking of Kerberos tickets.

9. pdfile()

```
function pdfile(){
    echo "[#] Saving the Results in a PDF file (Results_$TS.pdf)"
    for_output=$(ls | grep -v -e top-1000000.txt -e top-1000.txt -e http_default_users.txt )
    cat $for_output > output
    enscript output -p output.ps 2>/dev/null
    ps2pdf output.ps Results_$TS.pdf 2>/dev/null
}
```

This function generates a comprehensive PDF report of all findings, facilitating easy sharing and archiving of results.

Main Execution Flow

```
#execute function by order
colors
d_figlet
root
folder+target
d_python
d_nmap
d_masscan
d_john
d_enscript
d_ghostscript
e_crackmapexec
scaning
Enumeration
Exploitation
pdfile
```