

S/N	CODES	FUNCTIONS
1	<pre> # This function is for user to choose which section of the script do they want to run. function userinterface () { # This command is to display the options available for user and request for their choice. read -p "What would you like to do? (A) Update and upgrade your system. (B) Install tools for the script (C) Install Nipe. (D) Check if connection is secure. (E) Access remote server to run nmap, whois and masscan commands. (F) Do everything in the script. (G) Exit Script." executions  # This command is to navigate the script accordance to what was input by the user. case \$executions in  # This command is when a user choose (A) and the script will just run the update and upgrading of the system. a   A)     updateupgrade     userinterface     ;;  # This command is when a user choose (B) and the script will just run the installation of required tools. b   B)     installtools     userinterface     ;;  # This command is when a user choose (C) and the script will just run the installation of Nipe services. c   C)     installnipe     userinterface     ;;  # This command is when a user choose (D) and the script will just run to check if the connection is secure or not. d   D)     checkconn     userinterface     ;;  # This command is when a user choose (E) and the script will just run for access to the remote server and its functions. e   E)     checkconn     remoteaccess     userinterface     ;;  # This command is when a user choose (F) and the script will run all its function from the beginning f   F)     updateupgrade &amp;&amp; installtools &amp;&amp; installnipe &amp;&amp; checkconn &amp;&amp; remoteaccess     userinterface     ;;  # This command is when a user choose (G) and the script end. g   G)     exit     ;;  esac } </pre>	<ul style="list-style-type: none"> <li>When a user runs the script, this is the user interface that the user will first see.</li> <li>This will allow users to decide which part of the script do they want to run.             <ol style="list-style-type: none"> <li>Update and Upgrade</li> <li>Install Tools</li> <li>Install Nipe</li> <li>Check Connection</li> <li>Access remote server</li> <li>Do All</li> <li>Exit Script</li> </ol> </li> <li>Some users might already have nipe service installed, some already have the tools required, some have everything and just wants to access remote server and run the commands.</li> <li>User can choose to just access a remote server and run the nmap, whois and masscan commands. But this choice will include the 'check connection' functions to make sure that the user IP address is always masked before entering the remote server.</li> <li>For new user with a fresh system who wants to install everything, they can choose choice (F). The script includes all the functions available and runs in order of its intended execution phase.</li> </ul>

S/N	CODES	FUNCTIONS
2	<pre> # This function updates and upgrades the current system to the latest version. function updateupgrade () {      echo '----- Update and Upgrade System -----'      # This command will get all the information on the latest version of packages that are     # available for the user's system.     sudo apt-get -y update      # This command will download and install all the latest version of the required packages     # to upgrade the user's system to the latest version available.     sudo apt-get -y upgrade      echo '----- Upgrade and Upgrade System Completed -----'  } </pre>	<ul style="list-style-type: none"> <li>• This function is used to update and upgraded the user's system to the latest version possible to run the script and its tools.</li> </ul>
3	<pre> # This function download and installs all the tools required to run the script. function installtools () {      echo '----- Installation of Tools -----'      # This command will install geany onto the system in the event there is a need for user to     # amend certain commands to meet their needs.     sudo apt-get -y install geany      # This command will install sshpass capabilities for connection to remote servers.     sudo apt-get -y install sshpass      echo '----- Installation of Tools Completed -----'  } </pre>	<ul style="list-style-type: none"> <li>• This function installs the required tools that is needed for the script.</li> <li>• Installation of geany function is for user to amend the script if required for their own needs.</li> <li>• Installation of sshpass is done as that is the main method used to access the remote server for this script.</li> </ul>
4	<pre> # This function install Nipe services onto the system. function installnipe () {     # This command is to navigate to the users Desktop everytime it needs to install Nipe     # services.     cd ~/Desktop      echo '----- Installation of Nipe -----'      # This command clone the repository from GitHub onto the system.     git clone https://github.com/htrgouvea/nipe      # This command is to navigate into the Nipe directory before executing the next command.     cd ~/Desktop/nipe      # This command installs the libraries and dependencies required to run Nipe services.     # Learned how to auto response "yes" response when installaing programs from     # https://stackoverflow.com/questions/7642674/how-do-i-script-a-yes-response-for-installing-prog-     # rams     yes   sudo cpan install Try::Tiny Config::Simple JSON      # This command installs the Nipe dependencies onto the user's system.     sudo perl nipe.pl install      echo '----- Installation of Nipe Completed -----'  } </pre>	<ul style="list-style-type: none"> <li>• This function installs the nipe services onto the user's system that is used to mask their external IP address.</li> <li>• Any user who runs this function will have the nipe directories placed at the Desktop.</li> </ul>

S/N	CODES	FUNCTIONS
5	<pre> # This function checks the Nipe services and make sure the connection of the user is masked. function checkconn () {     # This command is to navigate into the Nipe directory before executing the next command.     cd ~/Desktop/nipe      echo '----- Checking Nipe Connection -----'      # This command is to stop the Nipe services (if it is already started).     sudo perl nipe.pl stop      # This command stores the original external IP address into the variable "myip" (for reference).     myip=\$(curl -s ifconfig.io/ip)      # This command displays the user's original external IP address.     echo "The original External IP is : \$myip"      # This command starts the Nipe services on the system.     sudo perl nipe.pl start      # This command checks the Nipe services status and prints out result.     sudo perl nipe.pl status      # This command stores the result of the Nipe services status (to look specifically for the word "activated.") into the variable "nipestatus"     nipestatus=\$(sudo perl nipe.pl status  grep activated.  awk -F: '{print \$2}')      # This command stores the current External IP (after starting Nipe services) into the variable "newip".     newip=\$(curl -s ifconfig.io/ip)      # This if command checks if the Nipe services status has been activated.     if [ \$nipestatus == activated. ]     then         # This command executes when the first 'if' condition is met.         # This if command checks if the current External IP (stored in "newip") is NOT the same as the original External IP (stored in "myip").         if [ \$newip != \$myip ]         then             # This command executes when the second 'if' conditions is met.             then                 # This command will check to see what is the country code for the current network, and save it under the variable "CC".                 CC=\$(curl -s ifconfig.io/country_code)                  # This command displays the newly masked external IP address of the user.                 echo "The current Masked IP is : \$newip"                  # This command displays the newly masked country code of the user.                 echo "The current country code of network is : \$CC"                  echo '----- Connection is secured -----'                  # This command executes when the first 'if' condition is met but the second 'if' condition is not met.             else                 echo '----- Connection is not secured -----'                  # This command will restart and run the whole function again from the start.                 checkconn             fi         fi         # This command executes when the first 'if' condition is not met.     else         echo '----- Connection is not secured -----'          # This command will restart and run the whole function again from the start.         checkconn     fi } </pre>	<ul style="list-style-type: none"> <li>• This function runs and make sure that the network connection is masked.</li> <li>• The function will first store the original External IP address so that it can be used as a reference. It also prints out to show the user.</li> <li>• The function then starts the nipe services and do a check to ensure that the services is activated and the new External IP address is not the same as the original External IP address.</li> <li>• Sometimes the nipe services takes awhile to establish a connection and it prints out an error message to users. The function will continue to restart and try to see if it can eventually establish a connection.</li> <li>• Once a connection is confirmed, the function then prints out the new External IP address and country code just for user's knowledge and informs that the connection is secure.</li> </ul>

S/N	CODES	FUNCTIONS
6	<pre> # This function will execute the command required to: # (a) access into the remote server, # (b) execute the nmap and whois command from the remote server for any ip as input by user, # (c) copy the output of the nmap and whois command and send back to the host computer # (d) remove the output files at the remote server.  function remoteaccess () {     # This command request user to input the IP address of the remote server and save the input in the variable     "RSIP"     echo 'Input Remote server IP address:' &amp;&amp; read RSIP      # This command request user to input the user ID of the remote server and save the input in the variable "RSID"     echo 'Input Remote server ID:' &amp;&amp; read RSID      # This command request user to input the user password of the remote server and save the input in the     variable "RSPW"     echo 'Input Remote server PW:' &amp;&amp; read RSPW      # This command request user to input the IP address that they want to run the nmap and whois function on and     save the input in the variable "IPadd"     echo 'Please input IP address that you would want to scan: ' &amp;&amp; read IPadd      echo '----- Connecting into Server -----'      # Learned how to ssh access from https://www.redhat.com/sysadmin/ssh-automation-sshpas     # This command will ssh into the intended remote server using the saved details (user password, user ID, IP     address) and execute the "nmap" command on the IP address input by the user and save the output at the remote     server.     sshpass -p \$RSPW ssh -o StrictHostKeyChecking=no \$RSID@\$RSIP "nmap \"\$IPadd\" -oG \"\$IPadd\"_nmap.scan"      # This command will ssh into the intended remote server using the saved details (user password, user ID, IP     address) and execute the "whois" command on the IP address input by the user and save the output at the remote     server.     sshpass -p \$RSPW ssh -o StrictHostKeyChecking=no \$RSID@\$RSIP "whois \"\$IPadd\" &gt; \"\$IPadd\"_whois.scan"      # This command will ssh into the intended remote server using the saved details (user password, user ID, IP     address) and execute the "masscan" command on the IP address input by the user and save the output at the remote     server.     # Learned how to input the password for sudo without user input from     https://askubuntu.com/questions/470383/how-to-avoid-being-prompted-for-a-password-by-sudo     sshpass -p \$RSPW ssh -o StrictHostKeyChecking=no \$RSID@\$RSIP "echo \"\$RSPW\"   sudo -S masscan \"\$IPadd\" -p20-80     -oG \"\$IPadd\"_mass.scan"      echo '----- Scan Completed -----'     # This command will delete a directory with the scanned IP address as the name (if any).     rm -r ~/Desktop/\$IPadd/*      # This command will create a directory with the scanned IP address as the name.     mkdir ~/Desktop/\$IPadd      # Learned how to scp via SSH without the need to input password from     https://www.atlantic.net/dedicated-server-hosting/how-to-pass-password-to-scp-command-in-linux/     # This command will ssh into the intended remote server using the saved details (user password, user ID, IP     address) and copy the saved "nmap" output back to the user computer at the specified location.     sshpass -p \$RSPW scp \$RSID@\$RSIP:~/ "\$IPadd\"_nmap.scan ~/Desktop/\$IPadd      # This command will ssh into the intended remote server using the saved details (user password, user ID, IP     address) and copy the saved "whois" output back to the user computer at the specified location.     sshpass -p \$RSPW scp \$RSID@\$RSIP:~/ "\$IPadd\"_whois.scan ~/Desktop/\$IPadd      # This command will ssh into the intended remote server using the saved details (user password, user ID, IP     address) and copy the saved "masscan" output back to the user computer at the specified location.     sshpass -p \$RSPW scp \$RSID@\$RSIP:~/ "\$IPadd\"_mass.scan ~/Desktop/\$IPadd      # This command will ssh into the intended remote server using the saved details (user password, user ID, IP     address) and remove the "nmap" output file from the server.     sshpass -p \$RSPW ssh -o StrictHostKeyChecking=no \$RSID@\$RSIP rm "\$IPadd\"_nmap.scan      # This command will ssh into the intended remote server using the saved details (user password, user ID, IP     address) and remove the "whois" output file from the server.     sshpass -p \$RSPW ssh -o StrictHostKeyChecking=no \$RSID@\$RSIP rm "\$IPadd\"_whois.scan      # This command will ssh into the intended remote server using the saved details (user password, user ID, IP     address) and remove the "masscan" output file from the server.     sshpass -p \$RSPW ssh -o StrictHostKeyChecking=no \$RSID@\$RSIP rm "\$IPadd\"_mass.scan      # This command will check how many files is inside the newly created directory for the scanned IP address.     Checkfiles=\$(cd ~/Desktop/\$IPadd &amp;&amp; ls  wc -l)      # This if command is to check if the number of files in the directory is 1 or more.     if [ \$Checkfiles -ge 1 ]      # This command executes when the 'if' condition is met.     then         echo "----- \$Checkfiles files saved into folder -----"      # This command executes when the 'if' command is NOT met.     else         echo '----- 0 file saved into folder -----'      fi } </pre>	<ul style="list-style-type: none"> <li>This function runs the nmap, whois and masscan commands automatically once the user provides the required information to the script.             <ol style="list-style-type: none"> <li>Remote Server IP</li> <li>Remote Server Login ID</li> <li>Remote Server Password</li> <li>IP address that user wants to scan with nmap, whois and masscan.</li> </ol> </li> <li>The function will then access the remote server and runs the scans and save the output on the remote server.</li> <li>For every scan, the function will remove any directories and result previously saved (if any) for the user input IPaddress. This is to ensure no duplicates of directories.</li> <li>The function will then copy all the output files from the remote server back into the user's desktop. The function will also remove all the output files from the remote server so as not to leave any trace behind that activities are on going.</li> <li>The function will then inform user how many files are saved. So if the user see the number is less than 3, it means some scans are not done properly.</li> </ul>