Cyber Security is a critical field in our digital world, protecting systems, networks, and data from cyber threats. This course will introduce you to essential tools and techniques used in cybersecurity, including Linux, Bash scripting, C programming, Assembly language, and Python.

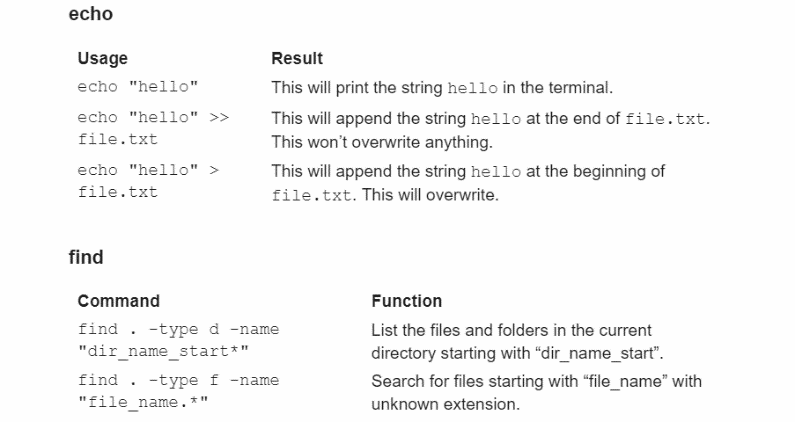
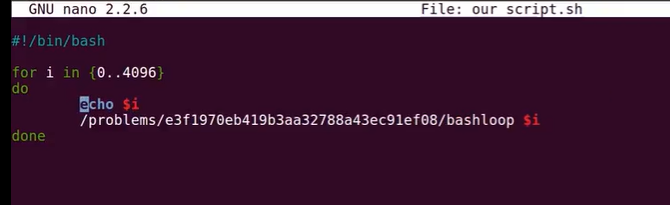
The resources, including the assignments and projects : <https://github.com/CSeCIITB/LS-2024-Tools-for-Cyber-Security>

**Timeline:**  
Week 1: 27 June - 7 July   
Week 2: 8 July - 14 July  
Week 3: 15 July - 21 July  
Week 4: 22 July - 28 July

* Week 1-3: Weekly content will be released along with corresponding assignments.
* Project statement will be released in 2 parts:
  + Part 1 will be released along with Week 2
  + Part 2 will be released after Week 3

There are other methods to solve that challenge (for example: <https://www.youtube.com/watch?v=1Drk5ZeGH14> ) but that video demonstrates how you can create a symbolic link instead of copying a file if you want to run the same file from a different place.

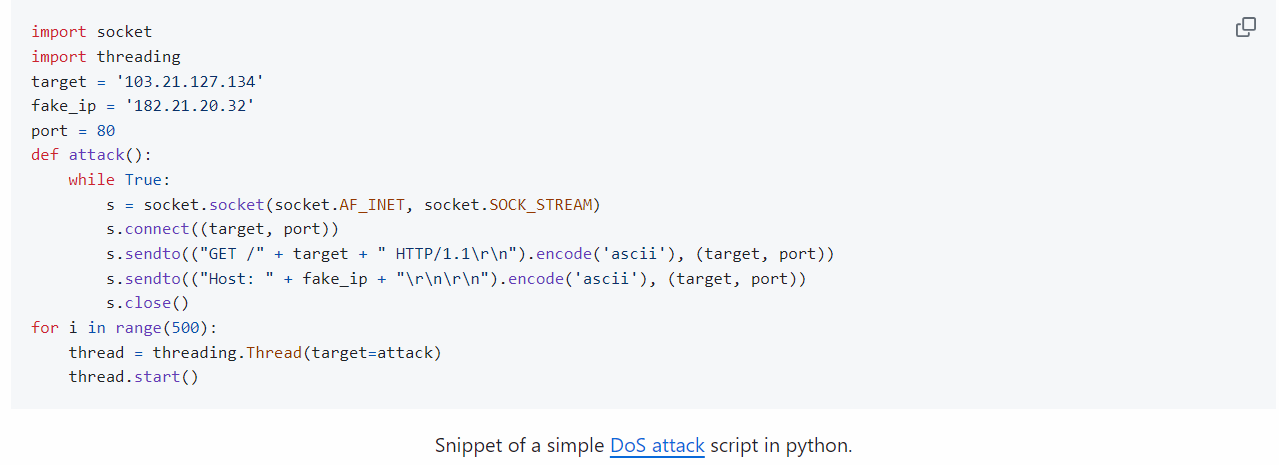
* Linux is an operating system just like Windows and MacOS. An operating system is a piece of code that enables the computer software to communicate and operate with the computer hardware easily. Unlike Windows and MacOS though, Linux is free and open-source. What this means is that you don't need to pay to use Linux and can even look at the [source code](https://github.com/torvalds/linux) yourself (unlike other operating systems where the source code is a trade secret).
* Shell scripting enables one to automate tasks that otherwise require [100s](https://www.youtube.com/watch?v=K19COVM5XuA) of mouse clicks and other kinds of interesting stuff as well.
* <https://www.youtube.com/watch?v=YA-3NI_Lfns>   
  <https://www.youtube.com/watch?v=s3ii48qYBxA>   
  <https://www.youtube.com/watch?v=PPQ8m8xQAs8>   
    
  Bash Scripting : [Bash conditions and loops - bi0s wiki](https://wiki.bi0s.in/basics/one_liners/)   
  [Bash Scripting Tutorial](https://ryanstutorials.net/bash-scripting-tutorial/)  
  Stdin, Stderr, Stdout: [Files - bi0s wiki](https://wiki.bi0s.in/basics/files/)
* Linux commands:
  + <https://www.geeksforgeeks.org/essential-linuxunix-commands/>
  + pwd, mkdir, touch, clear, man, whoami,
  + | //piping method: gives the std output into std input for the next cmd.
  + rm file1, rm -r dir1 , rm -f file1 , rm -rf dir1 // f = force deletion of read-only files.
  + cd, cd .. , cd dir\_path , cd ~ , cd /
  + ls, ls -a , ls -l , ls dir1 // l = detailed, a = all files(hidden too).
  + cp file1 file2 // If, dir2 and file2 are not present, it will create a folder or file resp.  
    cp -r dir1 dir2 // r = recursive
  + mv file1 dir1 , mv dir1 dir2 , mv file1 . // (.) = current directory   
    mv file1 file2 // Renaming a file ! (If file 2 is not present) (or else overwrites file2)
  + cat file1 //prints file contents in terminal  
    cat -n file1 //prints with line numbers.  
    less file1 //prints file contents but one page at a time.
  + find dir1 -name file\_name // Find inside the directory for that file name   
    find .
  + grep -rnw “text\_to\_be\_found” //Searches for the given string in any type of file.
  + cmd | grep “some\_string”   
    //Highlights and shows only the lines containing the given string, from the output of the command given.   
    cmd | grep -v “some\_string”  
    //Highlights and shows only the lines NOT containing the given string, from the stdIN.
  + rev // Reverses the std input given to it.
  + cut -d “ “ -f1 // First word(f1) of every line is the output.
  + tr -d “\n” // Merges all the lines into one (deletes \n from the stdIN)
  + sort combined.txt | uniq | wc -l
  + cmd > file1 // Redirects the stdout into the file and OVERWRITES it.
  + cmd >> file1 // Redirects the stdout into the file and appends it.
  + ./binaryfile1 x > /dev/null // x = 0(stdin), 1(stdout), 2(stderr) // dev/null = sort of trash can.



* Linux practice --   
  <https://overthewire.org/wargames/bandit/bandit0.html>   
    
  Use the following syntax to set up a connection with the server:  
  $ ssh <username>@bandit.labs.overthewire.org -p 2220  
  (For the username and passcode – see the website) (ctrl+D / type logout = to logout of the ssh)
* Secure Shell (SSH) = Securely connects to remote server.  
  🡪 ssh user\_name@host -p xxxx //host = IP or domain\_name //xxxx = port number
* Netcat (nc) =   
  🡪 nc host xxxx
* chmod +x file1.exe //
* sudo = In UNIX a superuser or root is a user which has unrestricted access to all the commands, files, folders and resources of the system.
* Ex. / CTFs
  + ssh as ctf-player to titan.picoctf.net at port 57470 to get the flag, with password 6dd28e9b  
    🡪 ssh ctf-player@titan.picoctf.net -p 57470
  + Connect to jupiter.challenges.picoctf.org at port 25103 to get the flag.  
    🡪 nc jupiter.challenges.picoctf.org 25103
* 1 challenge is left in Section 1 – “challenge1”

**# Section 2 (Python and pwntools lib.)**

* During the design of an application, you might need to decide whether to use a compiled language (C/Cpp/x86 assembly lang/ARM assembly lang) or an interpreted language (Python) for the application source code.   
  the decision to use an interpreted language is based on time restrictions on development or for ease of future changes to the program, trading for higher execution costs. Because each line of an interpreted program must be translated each time it is executed.   
  Interpreted programs are usually less efficient than compiled programs.   
  Python is an open source software like Linux.
* Programming languages such as REXX and Java, can be either interpreted or compiled.
* Keeping this in mind, we can see that it would make sense to use a compiled language for the intensive parts of an application (heavy resource usage), whereas interfaces (invoking the application), prototyping and less-intensive parts could be written in an interpreted language.
* Is Python itself written in C or C++? [An interesting read](https://softwareengineering.stackexchange.com/questions/20988/why-is-python-written-in-c-and-not-in-c) on this. Unlike C or C++, Python has [automatic memory management](https://www.geeksforgeeks.org/memory-management-in-python/) i.e. in Python memory allocation and deallocation method is automatic, since it has its own garbage collector, so that the user does not have to do manual garbage collection. Python is a [dynamically typed](https://www.geeksforgeeks.org/type-systemsdynamic-typing-static-typing-duck-typing/) programming language which makes it more succint.



* Installing pwntools : [Link](http://docs.pwntools.com/en/latest/install.html)  
     
  sudo apt-get install python3 python3-pip python3-dev git libssl-dev libffi-dev build-essential  
  python3 -m pip install --upgrade pip  
  python3 -m pip install --upgrade pwntools  
  # Adding ~/.local/bin to my $PATH environment variables.  
  gedit ~/.bashrc  
  # Add the following at the end of the script : export PATH="$HOME/.local/bin:$PATH"   
  sudo apt-get install python3-dev
* [Pwntools](http://docs.pwntools.com/en/latest/), an interesting framework, in itself is a huge package. It will get you familiar with writing exploit scripts in Python.  
  Some other important modules/libraries regularly used are [sys](https://docs.python.org/3/library/sys.html), [os](https://docs.python.org/3/library/os.html), [subprocess](https://docs.python.org/3/library/subprocess.html), all these are in-built python modules.
* [CSeC - Intro to Python Video](https://iitbacin.sharepoint.com/:v:/s/CSecClub/ETXm3oSRRoJKgKDCaKLiRbYBTh5fLAYkaow3MX59-5FTyQ?e=1rRkWg)
* string evaluation of input in python2 [HSCTF - Python Remote Code Execution](https://www.youtube.com/watch?v=gmaWOknsb2A)(5 min) | Read more about Python2.x [input vulnerability](https://www.geeksforgeeks.org/vulnerability-input-function-python-2-x/)  
  pwntools process interaction [GOOGLE CTF 2021](https://www.proggen.org/doku.php?id=security:ctf:writeup:google:2021:filestore) (good read)  
  pwntools in bash [TAMU CTF 2020](https://www.youtube.com/watch?v=fZ3mPRctbO0) (17 mins)  
  Request module usage [OverTheWire natas level4](https://www.youtube.com/watch?v=Sf63W1xXzNU) (11 mins)  
  Pyjail [Offshift 2021](https://www.youtube.com/watch?v=aK3b0PM1Fz8) (6 mins)

**# Section 3 : C programming and x86 assembly**

* Computers can understand and run only machine instructions, but programmers usually code in high–level programming languages, such as C, Python, or Javascript. This is because high–level programming languages are easier to work with and resemble human languages and mathematical notation.