Functional requirements Design and implement a Penetration testing laboratory based on the following requirements:

1.) Selection and usage of penetration testing tools (e.g. Kali Linux, nmap, Nessus, OpenVAS, Metasploit, Nikto, Dirbuster, BurpSuite, Sqlmap, THC-Hydra, etc)

2.) Usage of Automation tools for penetration testing (e.g. NSE (nmap scripting engine), Shell/Bash code, Windows PowerShell, etc.)

3.) Usage of custom exploits (e.g. C/Python/Perl scripts).

4.) Network infrastructure implementation (e.g. switch/hub, WEP/WPA/WPA2 on wireless AP, firewall)

5.) System infrastructure implementation: selection and implementation of Vulnerable Systems. Selection/Implementation should be based on multiple different operating systems including but not limited to the following: i) Microsoft Windows XP/7/8/10, Server 2003/2008/2012/2016 , etc. ii) Linux Ubuntu/Fedora/etc. However, you are not allowed to use preconfigured OS such as Metaspoiltable. iii) Any other windows or Linux server based operating systems. You are to use VMware workstation as your virtualization platform. Vulnerabilities to be exploited can include but not limited to the following:

i) Insecure user/file permissions such as Sudo , SUID and unquoted service paths

ii) Buffer Overflow (Using different methods such as the NOP Sled, JUMP ESP, SEH, etc.)

iii) Outdated plugins/software such as java, acrobat reader

iv) Kernel exploit v) Post-installed software

vi) Post-installed services (Such as the use of SSH tunnelling: local port , remote port, and dynamic port forwarding for pivoting purposes)

vii) Eternal Blue/Romance

viii) HeartBleed ix) ShellShock x) BlueKeep xi) DirtyCow xii) Misconfiguration of the OS such as enabling of RDC (Remote Desktop Connection), etc.

6.) Linux Web Server hosting a vulnerable PHP web application. The web application should contain vulnerabilities and be susceptible to attacks including, but not limited to the following: i) SQL injection (e.g. malicious read, write, delete operations on the database) ii) XSS (Cross Site Scripting) iii) XSRF (Cross Site Request Forgery) iv) Weak Authentication and session management v) Command injection vi) Local/Remote File Inclusion vii) Security misconfiguration viii) Etc.

7.) Detailed explanations together with relevant countermeasures for each of the identified vulnerabilities should be presented to the evaluators in a clear and concise manner.

8.) Full technical documentation (user & trainer) to be provided. Documentation should contain details such as: i) Network Diagram ii) Use Case Diagram, ERD Diagram for the Vulnerable Web Application iii) Detailed steps taken and results of all port scanning and vulnerability scanning activities. iv) Vulnerabilities of the implemented systems v) Exploit steps vi) Countermeasures to be applied to the vulnerabilities

9.) Implement a well-designed CTF (Capture the flag) with the provided solutions on a suitable platform (e.g. CTFd / Docker /DigitaOcean, etc) . Flags are to be designed to train participants in the areas of : i) General IT skills (e.g. Linux OS, common protocols such as SSH/SSL/HTTPS , common tools such as ncat/Wireshark , etc). ii) Cryptography/Encoding. iii) Web exploitation. iv) Binary Exploitation. v) Reverse engineering. - Tasks stated is assigned to only one student. For instance, each student should research on a different set of system or web application vulnerabilities. Each student is to be accountable for the individual contribution towards the team. - Every task MUST have a verifiable/measurable result. So “Research web vulnerabilities” is not enough. “Select SQL injection as chosen web vulnerability” would be a correct alternative, since this is closed with the decision “We are covering SQL injection in our vulnerable web app and XXX would be our relevant countermeasure”.