## Preparation

|  |  |
| --- | --- |
| **Item & Usage** | **Download Link** |
| Filezilla (for those who lazy and hate scp) | <https://download.filezilla-project.org/client/FileZilla_3.51.0_win64_sponsored-setup.exe> |
| Visual Studio Code  \*add PHP Getters and Setters on marketplace if you are using PHP and restart after install | <https://code.visualstudio.com/docs/?dv=win> |
| Use Mobaxterm for SSH and File transfer alot easier :) <- ish good (tried and tested) | <https://download.mobatek.net/2062020111930940/MobaXterm_Portable_v20.6.zip> |

## Lab 5 Setting Up Jenkins

|  |  |
| --- | --- |
| **Description** | **Command** |
| Login to the server via SSH | ssh <username>@<ip address>  ssh ranger@192.168.11.130 |
| Change shell to /bin/bash for command history and auto complete | cat /etc/passwd (prob no need)  chsh -s /bin/bash |
| **Installing Docker** | |
| Update Ubuntu | sudo apt update |
| Install docker prerequisites | sudo apt-get install \  apt-transport-https \  ca-certificates \  curl \  gnupg-agent \  software-properties-common  -y |
| Add Docker’s GPG Key | curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add - |
| Something about keys? (Just do it) | apt-key fingerprint 0EBFCD88 |
| Use the following command to set up the stable repository. (Just do it) | sudo add-apt-repository \  "deb [arch=amd64] https://download.docker.com/linux/ubuntu \  $(lsb\_release -cs) \  stable" |
| Update Ubuntu | sudo apt update |
| Install Docker Engine | sudo apt-get install docker-ce docker-ce-cli containerd.io -y |
| Verify Docker engine is installed correctly | sudo docker run hello-world |
| The following should be seen when you run the above command: | |
| Install Git | sudo apt install git |
| Add some git settings | git config --global user.email "you@example.com"  git config --global user.name "Your Name" |
|  | |
| Install Jenkins on Docker  Red words: May need to change, take note of -- before for context | sudo docker run --detach -u root --name jenkins-tutorial --volume jenkins-data:/var/jenkins\_home --volume "$HOME":/home --volume /var/run/docker.sock:/var/run/docker.sock --publish 8080:8080 jenkinsci/blueocean  sudo docker run --name jenkins-lab -d -u root -v /var/jenkins\_home:/var/jenkins\_home -v /var/run/docker.sock:/var/run/docker.sock --publish 8080:8080 jenkinsci/blueocean |
| Get Jenkins InitalAdminPassword  Red words: name based on previous command   * Use the other command if it doesn’t show it | sudo docker exec jenkins-tutorial cat /var/jenkins\_home/secrets/initialAdminPassword  sudo docker logs jenkins-tutorial |
| Example results for sudo docker logs jenkins-tutorial command | |
| **Unlocking Jenkins** | |
| Navigate to port 8080 of your cloud instance | http://<ip address>:8080  http://192.168.11.130:8080 |
| Enter the long string from the logs command in the following screen    I think Install suggested plugins is recommended,      Advised to set up a user, else continue as admin and use the InitialAdminPassword    Then continue to setup until you see this screen | |

## All Jenkins Plugins

|  |
| --- |
| **Jenkins Plugins Installed** |
| Navigate to Manage Jenkins > Manage Plugins > Available (must ensure there’s a blue tick beside the entry)  **OWASP Dependency-Check**    **Warnings Next Generation**    **SonarQube Scanner** |
| **Combined Jenkinsfile** |
|  |

## 

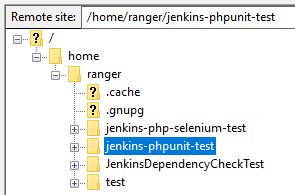
## Lab 5B OWASP Dependency-Check

|  |  |
| --- | --- |
| **Install OWASP Dependency-Check Plugin** | |
| Navigate to Manage Jenkins > Manage Plugins > Available (must ensure there’s a blue tick beside the entry)    To restart or not is up to you. If you choose to do so, remember to refresh occasionally as it does not auto refresh the page for you.    Navigate tool Configuration, scroll down to Dependency-Check and click Add Dependency-Check, set name to ‘Default’ and click save. | |
| **Adding OWASP Dependency Check into Jenkins Pipeline** | |
| For testing Jenkins Dependency Check to ensure it’s working | git clone <https://github.com/0xprime/JenkinsDependencyCheckTest.git> |
| Check current directory (Take note, useful later) | pwd |
| **Create a Pipeline** | |
| Click on New Item, enter a name (whatever you like), select pipeline and click ok.    Follow the config entered, under the Pipeline section. Test being whatever is your git repo folder name. Remember to untick Lightweight checkout. Enter the results of pwd into the Repository URL and remove your username.  Example:   * pwd: /home/ranger/xxx * Repo url: /home/xxx | |
| **Adding a Jenkins File to test OWASP Dependency Check** | |
| Add/Edit Jenkinsfile | vim Jenkinsfile |
| Paste the pipeline code that inside vim and press escape button and type :wq  pipeline {  agent any  stages {  stage('Checkout SCM') {  steps {  git '/home/JenkinsDependencyCheckTest'  }  }  stage('OWASP DependencyCheck') {  steps {  dependencyCheck additionalArguments: '--format HTML --format XML', odcInstallation: 'Default'  }  }  }  post {  success {  dependencyCheckPublisher pattern: 'dependency-check-report.xml'  }  }  } | |
| Commit the Jenkinsfile to local git repo | git add .  git commit -m “Add initial Jenkinsfile” |
| **Running the pipeline and checking the results of dependency check** | |
| Click build now and click on status. Oh donee \0-0/    (from the same page but extreme right?, please refresh your page)    Click on the completed build    In the build, navigate to Dependency-Check to view the exact results. | |
| **Suppressing False Positives in OWASP Dependency-Check** | |
| This is how to get to workspaces    The workspace shows the files for the build.    In the workspace, right click click on dependency-check-report.html and click save link as  For a false positive entry, click on suppress    A popup will appear, click Complete XML Doc and copy the XML into a file name it suppression.xml or something and save it in the same location as the Jenkinsfile      I used vim to create suppression.xml    Update OWASP DependencyCheck to the Jenkins file to include suppression.xml  pipeline {  agent any  stages {  stage('Checkout SCM') {  steps {  git '/home/JenkinsDependencyCheckTest'  }  }  stage('OWASP DependencyCheck') {  steps {  dependencyCheck additionalArguments: '--format HTML --format XML --suppression suppression.xml', odcInstallation: 'Default'  }  }  }  post {  success {  dependencyCheckPublisher pattern: 'dependency-check-report.xml'  }  }  } | |
| Commit the Jenkinsfile to local git repo | git add .  git commit -m “Added dependency-check suppression” |
| Click build now again, refresh the page once done and the dependency-check trend is expected to drop. | |

## Lab 6 Automated Unit Testing (PHP)

**Prerequisites**

* Go to xsite and download both zip files from lab 6.
* Paste the zip contents into the VM
  + I recommend using filezilla ? (ss attached)

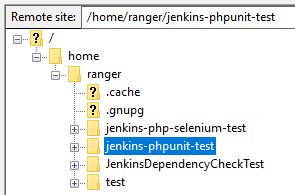


|  |  |
| --- | --- |
| **Description** | **Command** |
| Go to the jenkins-phpunit-test folder | cd ~  cd jenkins-phpunit-test |
| Initialize folder as git repo | git init |
| Commit the Jenkinsfile to local git repo | git add .  git commit -m “Add initial files” |
| **Create Jenkins Pipeline with Unit Testing** | |
| Click on New Item, enter a name (whatever you like), select pipeline and click ok.    Follow the config entered, under the Pipeline section. Test being whatever is your git repo folder name. Remember to untick Lightweight checkout. Enter the results of pwd into the Repository URL and remove your username.  Example:   * pwd: /home/ranger/xxx * Repo url: /home/xxx     Click save and click build now the results should be as shown below:    Update the provided Jenkinsfile, enhanced portions in red to include test reporting.  pipeline {  agent {  docker {  image 'composer:latest'  }  }  stages {  stage('Build') {  steps {  sh 'composer install'  }  }  stage('Test') {  steps {  sh './vendor/bin/phpunit --log-junit logs/unitreport.xml -c tests/phpunit.xml tests'  }  }  }  post {  always {  junit testResults: 'logs/unitreport.xml'  }  }  } | |
| Commit the edited Jenkinsfile to local git repo | git add .  git commit -m “Updated jenkinsfile to include reporting” |
| Click save and click build now the results should be as shown below (build 3):    Refresh the page to see the following at the extreme right section    Click on the build and click on Test Result (The below ss are from two different builds) | |

## Lab 6B Automated Integration and UI Testing (PHP)

**Prerequisites**

* Go to xsite and download both zip files from lab 6.
* Paste the zip contents into the VM
  + I recommend using filezilla ? (ss attached)



|  |  |
| --- | --- |
| **Description** | **Commands** |
| Go to the jenkins-php-selenium-test folder | cd ~  cd jenkins-php-selenium-test |
| Initialize folder as git repo | git init |
| Commit the Jenkinsfile to local git repo | git add .  git commit -m “Add initial files” |
| Change the access rights of the script files (for Jenkins) | sudo git update-index --chmod=+x jenkins/scripts/deploy.sh  sudo git update-index --chmod=+x jenkins/scripts/kill.sh |
| Change the access rights of the script files (for local Git) | cd ~  cd jenkins-php-selenium-test  sudo chmod +x jenkins/scripts/\* |
| Update deploy.sh | Update the provided deploy.sh in /jenkins/scripts, enhanced portions in red to correspond to the location of the git repo.  #!/usr/bin/env sh  set -x  docker run -d -p 80:80 --name my-apache-php-app -v /home/ranger/jenkins-php-selenium-test/src:/var/www/html php:7.2-apache  sleep 1  set +x  echo 'Now...'  echo 'Visit http://localhost to see your PHP application in action.' |
| Remove /r error IF YOU HAVE ENCOUNTER | sed -i $'s/\r$//' jenkins/scripts/kill.sh  sed -i $'s/\r$//' jenkins/scripts/deploy.sh |
| Replace the string at URL to vm IP address. File can be found at src/test/java/mycompany/app/AppTest.java |  |
| Commit the Jenkinsfile to local git repo | git add .  git commit -m “Changed permissions and updated deploy.sh” |
| **Create Jenkins Pipeline** | |
|  | |
| **Run Integration and UI test** | |
| Allow TCP connections to port 80 | sudo ufw allow 80/tcp |
| Click save and click build now the results should when you click one the “BlueOcean” thingy. | |
|  | |
| **Troubleshooting Guide** | |
| Check if your ports 80 (for UI Testing) and 8080 (for Jenkins) are opened | netstat -lntu |
|  | |
| Check if your docker containers are running, your one might have more or less, it depends. But make sure that one in RED is there, else continue to troubleshoot. | sudo docker container ls -a |
|  | |
| sudo ./jenkins/scripts/kill.sh | |

## Lab 7 Warnings Next Generation Plugin

|  |  |
| --- | --- |
| **Description** | **Commands** |
| Install “Warnings Next Generation” plugin | At your Jenkins Web:  Go go Manage Jenkins > Manage Plugins > Available > Type “warnings” > Select on the “Warnings Next Generation” checkbox > Click on “Download now and install after restart”.    Then select on the “Restart jenkins when installation is complete and no jobs are running” checkbox.  But to restart or not is up to you. If you choose to do so, remember to refresh occasionally as it does not auto refresh the page for you. |
| Download the project via zip file (your computer) **or** via git command line (in your vm) | [https://github.com/ScaleSec/vulnado](https://github.com/ScaleSec/vulnado.git)  git clone <https://github.com/ScaleSec/vulnado.git> |
| Unzip the project (ignore if downloaded using git clone in vm) | Right click > 7-Zip > Extract to “vulnado-master” |
| Use WinSCP and copy that project to your server (ignore if downloaded using git clone in vm) |  |
| Go to that project folder on your server | cd ~  cd vulnado-master / cd vulnado (git) |
| Initialize folder as git repo (ignore if downloaded using git clone in vm) | git init |
| Commit the Jenkinsfile to local git repo (ignore if downloaded using git clone in vm) | git add .  git commit -m “Add initial files” |
| Create a Jenkinsfile (git or nano?) | nano Jenkinsfile |
| **Paste the following into the Jenkinsfile:**  pipeline {  agent {  docker { image 'maven' }  }  stages {  stage ('Checkout') {  steps {  git branch:'master', url: 'https://github.com/ScaleSec/vulnado.git'  }  }  stage ('Build') {  steps {  sh 'mvn --batch-mode -V -U -e clean verify -Dsurefire.useFile=false -Dmaven.test.failure.ignore'  }  }  stage ('Analysis') {  steps {  sh 'mvn --batch-mode -V -U -e checkstyle:checkstyle pmd:pmd pmd:cpd findbugs:findbugs'  }  }  }  post {  always {  junit testResults: '\*\*/target/surefire-reports/TEST-\*.xml'  recordIssues enabledForFailure: true, tools: [mavenConsole(), java(), javaDoc()]  recordIssues enabledForFailure: true, tool: checkStyle()  recordIssues enabledForFailure: true, tool: spotBugs(pattern: '\*\*/target/findbugsXml.xml')  recordIssues enabledForFailure: true, tool: cpd(pattern: '\*\*/target/cpd.xml')  recordIssues enabledForFailure: true, tool: pmdParser(pattern: '\*\*/target/pmd.xml')  }  }  } | |
| Add the new file to local git | Git add .  Git commit -m “added Jenkinsfile” |
| **Create Jenkins Pipeline** | |
|  | |
|  | |
|  | |
|  | |
| This one need wait sibei long… so just be patient ok | |
|  | |

## Lab 8 SonarQube

|  |  |
| --- | --- |
| **Description** | **Commands** |
| Download the project via zip file (your computer) **or** via git command line (in your vm) | [https://github.com/OWASP/Vulnerable-Web-Application](https://github.com/OWASP/Vulnerable-Web-Application.git)  git clone <https://github.com/OWASP/Vulnerable-Web-Application.git> |
| Unzip the project (ignore if downloaded using git clone in vm) | Right click > 7-Zip > Extract to “Vulnerable-Web-Application-master” |
| Use WinSCP and copy that project to your server (ignore if downloaded using git clone in vm) |  |
| Go to that project folder on your server | cd ~  cd Vulnerable-Web-Application-master |
| Initialize folder as git repo (ignore if downloaded using git clone in vm) | git init |
| Commit the Jenkinsfile to local git repo (ignore if downloaded using git clone in vm) | git add .  git commit -m “Add initial files” |
| **Installing and Using SonarQube Docker** | |
| Install SonarQube by docker | sudo docker pull sonarqube |
| Create docker container for SonarQube | sudo docker run -d --name sonarqube -e SONAR\_ES\_BOOTSTRAP\_CHECKS\_DISABLE=true -p 9000:9000 sonarqube:latest |
| Login to your sonarqube  **Login Credentials**  **login=admin**  **password=admin** | http://<ip address>:9000 |
|  | |
| **login=admin**  **password=admin** | |
| Then select “<> Manually”. | |
| Key in any name you like or the name that is stated in the lab test instructions.  Then click “Set Up”. | |
| Key in any token name you like or the name that is stated in the lab test instructions.  Then click “Generate”. **\*Important, will be used in Jenkinsfile\*** | |
| **Copy the command \*Used for Jenkinsfile** | |
| **Install SonarQube Scanner Plugin**  Navigate to Manage Jenkins > Manage Plugins > Available (must ensure there’s a blue tick beside the entry)    **Add SonarQube to Global Tool Configuration**  Go to Jenkins > Manage Jenkins > Global Tool Configuration > Add SonarQube Scanner | |
|  | |
| **Add SonarQube to Configure System**  Go to Jenkins > Manage Jenkins > Configure System > Add SonarQube > Save | |
| Create a Jenkinsfile in the folder | |
| pipeline {  agent {  docker { image 'maven' }  }  stages {  stage ('Checkout') {  steps {  git branch:'master', url: 'https://github.com/OWASP/Vulnerable-Web-Application.git'  }  }  stage('Code Quality Check via SonarQube') {  steps {  script {  def scannerHome = tool 'SonarQube';  withSonarQubeEnv() {  sh "${scannerHome}/bin/sonar-scanner -Dsonar.projectKey=OWASP2 -Dsonar.sources=. -Dsonar.host.url=http://192.168.1.100:9000 -Dsonar.login=35f1d0206902a92f1409e4cc8fee83b32069aa33"  }  }  }  }  }  post {  always {  recordIssues enabledForFailure: true, tool: sonarQube()  }  }  } | |
| Add Jenkins to local git | git add .  git commit -m “added Jenkinsfile” |
| DO NOTE THAT: | |
| **Create Jenkins Pipeline** | |
| Click Jenkins > Click New Item > Type “Sonarqube\_lab” > Click Pipeline > Click Ok.  Click Pipeline > At definition dropdown bar, select “Pipeline script from SCM” > At SCM dropdown bar, select Git > Type “/home/Vulnerable-Web-Application-master” at Repository URL > Click Save. | |
| Press “Build Now” and then “SonarQube” to analyze the result. Then you slowly wait. | |
| You may click into the build or head to http://<ip address>:9000 to analyse the sonarqube result. Enjoy :) no enjoy ):  **In the Build**    **In SonarQube** | |

## Other Troubleshooting

\*Replace jenkins-tutorial with the name of your container

|  |  |
| --- | --- |
| **Question** | **What to do** |
| WTF MY CONTAINER DISAPPEARED !? | Run the following command:  sudo docker run --detach -u root --name jenkins-tutorial --volume jenkins-data:/var/jenkins\_home --volume "$HOME":/home --volume /var/run/docker.sock:/var/run/docker.sock --publish 8080:8080 jenkinsci/blueocean |
| HOW TO RESTART DOCKER?? | Run the following command:  sudo docker restart jenkins-tutorial |
| HOW TO START MY DOCKER CONTAINER?? | Run the following command:  sudo docker start jenkins-tutorial |
| HOW TO STOP MY DOCKER CONTAINER?? | Run the following command:  sudo docker stop jenkins-tutorial |