Proceedings of Student-Faculty Research Day, CSIS, Pace University, May 7th, 2010

Improving Web Security Education with Virtual Labs and Shared Course Modules

Lixin Tao, Li-Chiou Chen, and Chienting Lin
Pace University, ltao@pace.edu, lchen@pace.edu, clin@pace.edu

Abstract - **One challenge in web security education is its interdisciplinary and practical nature. Students need to have the basic knowledge and skills of a web developer to understand many of the web security topics, and some of them are normally covered in multiple advanced courses like Computer Networks and Network Security, or are absent from many existing undergraduate or graduate degree programs. This paper shares our experience of using VMware virtual machines in supporting hands-on web security education, and developing multiple virtual web security lab modules based on the virtual machines. The lab modules are part of our NSF SWEET (Secure WEb dEvelopment Teaching) project, and each of them contains (1) concepts in a nutshell; (2) lab objectives; (3) software setup; (4) detailed lab instructions; and (5) lab evaluations. Comprehensive lab modules have been developed to guide students to build virtual Ubuntu virtual machines with publicly available tools and install all necessary web servers, application servers and database servers on them so they can function as the foundation and platforms of the other course modules. The other covered course modules include cryptography, HTTP and HTTPS protocols, and introduction to Java web technologies.**

Index Terms - Network security, Virtual labs, Web technology, Web security.

INTRODUCTION

One challenge in web security education is its interdisciplinary and practical nature. Students need to have the basic knowledge and skills of a web developer to understand many of the web security topics, and some of them are normally covered in multiple advanced courses like *Computer Networks* and *Network Security*, or are absent from many existing undergraduate or graduate degree programs (like Ajax or web services). The students also need to actually practice to learn how to prevent, identify and resolve web security breaches, but limited university resources and few local web security domain experts sometimes limit the scope and scale of projects that students could practice in lab environments. If such labs are not designed properly, students could also cause web security problems when they explore security vulnerabilities of university or company public web sites.

This paper shares our experience of using VMware virtual machines in supporting hands-on web security education, and developing multiple virtual web security lab modules based on the virtual machines. The lab modules are part of our NSF SWEET (Secure WEb dEvelopment Teaching) project, and each of them contains (1) concepts in a nutshell; (2) lab objectives; (3) software setup; (4) detailed lab instructions; and (5) lab evaluations. Comprehensive lab modules have been developed to guide students to build virtual *Ubuntu* virtual machines with publicly available tools and install all necessary web servers, application servers and database servers on them so they can function as the foundation and platforms of the other course modules. The other covered course modules include cryptography, HTTP and HTTPS protocols, and introduction to Java web technologies. All the lab modules can be installed on portable USB thumbnail disks and run on any computer that installs the free VMware Player. The paper will also present our experience in adopting these course modules in multiple network and web security courses at Pace University.

SELECTION OF COMPUTER LAB VIRTUALIZATION TECHNOLOGIES

The virtualization of a computer means to run emulator software, like VMware Player [1] or Microsoft Virtual PC [2], on a computer (host computer or physical computer) to emulate another desired computer (virtual computer). A virtual computer is implemented by a folder of 2-8 GB files, and the emulator runs these files to emulate the virtual computer as a computer window or the complete computer desktop. To the users a virtual computer is just the same as the physical one. The virtual and host computers can have different operating systems, and share data and Internet access. The users can work on multiple virtual computers and the host computer at the same time. The users can install new applications on the virtual computer as on a physical one. A virtual computer can run most operating systems including all versions of Windows and Linux.

There are two virtualization technologies: (1) server-side virtualization for running the virtual computers on a remote server computer, and (2) client-side virtualization for running the virtual computers on users' own computers. While company/university IT infrastructure servers can use the server-side virtualization to improve resource utilization, client-side virtualization greatly reduces the pressure on the

servers and network bandwidth, and take advantage of faculty and student PCs' excessive computing power already available today. In the recent years we have studied many PC/lab outsourcing services based on server-side virtualization, including those based on IBM mainframes, and concluded that they all have recurring high costs or very limited flexibility and resources for the users (such remote virtual computers are at least ten times slower than a today's $400 PC in supporting interactive lab sessions, usually with no Internet access and less than one GB of disk space).

The main advantages of client-side computer virtualization for university computer labs include

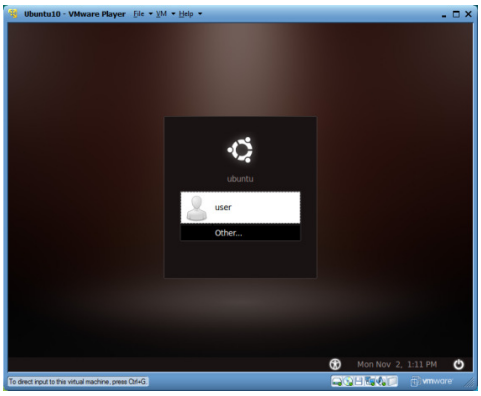
- There is no need of investment for a university to adopt virtual computing labs, and there are no recurring costs.
- There is no significant cost for distributing the virtual computers and lab modules to the faculty or students. The virtual computer's implementation folder can be distributed through web downloading, USB flash disks, or DVD disks. Typically a student can keep his/her virtual computers for different courses on a single thumb-size USB flash disk.
- If a person knows how to use a physical computer, he/she knows how to use a virtual computer. There is minimal learning curve for the users.
- New software can be installed on a virtual computer. If the user messes up a virtual computer, he/she just gets a fresh copy of the virtual computer's implementation folder.
- Any computer, no matter whether it runs Windows, Linux, or Mac OS, can install a free (around $70 for Mac OS) computer emulator and run the virtual computers. Therefore the students can work on a course lab anywhere. The course work can be started on one computer, and later resumed and completed on another. Therefore we could support the concepts of "coursework can be conducted anywhere on any computer" and "portable labs".
- If an online student has difficulties in the hands-on course work, the instructor can easily reproduce the problems and help resolve the problems since they are using the same working environment or virtual computer.
- The university has less pressure in updating its lab PCs and it needs less staff to maintain them. Any PC that has 1-4 GB memory and 100 GB disk space can support virtual computers in the coming years, and all university PCs of a specific model can share a single configuration image. If a PC has problems, just get it a fresh copy of the virtual computer's implementation folder (assuming important data are on university network disks or owner's disks).

For our web security virtual labs we choose VMware virtualization over Microsoft Virtual PC because the former can support virtual machines running any operating system including all flavors of Linux.

UBUNTU SECURITY LAB PLATFORM

A complete lab module has been developed for students and faculty who have no Linux background to develop a Ubuntu v9.10 (the latest version) virtual machine (VM). Instructions are also included to install and configure most necessary IT servers and tools needed for supporting network/web security and computing technologies. The tutorial module includes (1) a detailed lab manual "A Tutorial on Setting up Ubuntu Linux Virtual Machines" [3] detailing step-by-step guidance to achieve the above tasks; (2) 7z auto-extracting file for the completed basic Ubuntu v9.10 VM ready for software installation [4]; (3) 7z auto-extracting file for the completed Ubuntu v9.10 VM ready for lab use or distribution to the students [5]; and (4) a video tutorial for those who need further visual help[6].

If users choose to use the preconfigured Ubuntu v9.10 VM, they just need to download the file "ubuntu10.exe" and run it to extract VM folder "ubuntu10"; download and install the free VMware Player [1] for their operating systems; and double-click on file "ubuntu10/Ubuntu10.vmx" in a file explorer to launch the VM as shown below.



After entering "user" or "root" as the user name and "12345678" as the password, the users will see the following Ubuntu user interface:

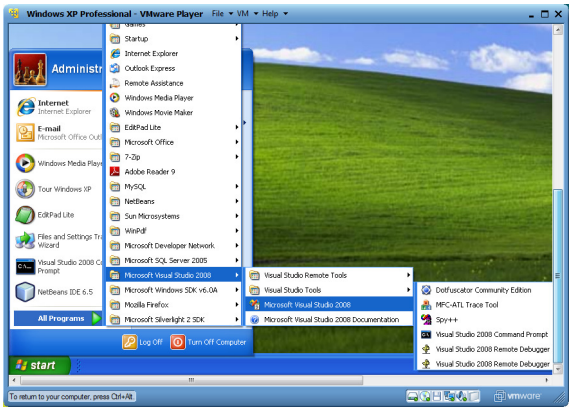
VM ubuntu10 has the installation of the following tools and servers:

- Gnu C++/C compiler
- Java JDK v6.6
- Tomcat web server v6.0.20
- Apache web server v2.2 with support for Perl, PHP and MySQL

- MySQL database server v5.1
- Eclipse IDE v1.2.1 (Galileo SR1)
- NetBeans IDE v6.7.1
- GlassFish application server v2.1 including Java EE web server)
- Derby database server v10.4.2.1
- Drupal web contents management system v6.14
- Java Tutorial 2009-09-23
- Java EE 5 Tutorial v1.0_05
- First Cup tutorial for Java EE v2.1
- A dozen of pre-deployed web applications on Tomcat for students to learn and experiment with web technologies.

The server installations have been configured for IT server production run. For example, the Tomcat and Apache web servers have been properly integrated so the Tomcat servlet container can run behind the Apache web server to generate HTML responses for Apache with Java technologies.

Most of our web security lab modules are built on top of this Ubuntu VM which is available for free distribution to the public. We have also developed a similar Windows XP VM, as shown below, with similar tools and servers plus the IIS web server, MS Office and Microsoft Visual Studio 2008 for students to study web security technologies on the .NET platform.



SECURITY LAB MODULE ON CRYPTOGRAPHY

Security lab module "Introduction to Cryptography" [7] covers fundamental concepts of symmetric secret key ciphers, public key ciphers, hash functions, digital signatures and digital certificates. The lab has the following objectives:

a. Learn and practice how to use MD5 and SHA1 to generate hash codes of strings or large files, and verify whether a downloaded file is valid;
b. Learn and practice how to use GPG to encrypt/decrypt files with symmetric algorithms;
c. Learn and practice how to use GPG to generate public/private key pairs and certificates, distribute the certificate with public key to a friend, let the friend encrypt a document with the public key, and let the key owner decrypt the document with the private key.

The lab starts with the installation of the GnuPG-Agent installation on the Ubuntu VM. The following is sample lab guide for practicing PGP concepts with GPG on the Ubuntu VM.

1. Create Linux Accounts for Alice and Mike

- Launch your *Ubuntu10* VM, and start a terminal window.
- Run command "sudo adduser alice" to create a Linux account for Alice. Use 123456 as password.
- Run command "sudo adduser mike" to create a Linux account for Mike. Use 123456 as password.
- Run command "sudo visudo" to launch file "/etc/sudoers.tmp" in a text editor, insert the following two lines at the end of the file, and then use Crtl+O to write out the revised contents, and use Ctrl+X to exit the editor. This step will enable Alice and Mike to use "sudo".

```
alice  ALL=(ALL) NOPASSWD: ALL
mike  ALL=(ALL) NOPASSWD: ALL
```

2. Run as Alice and Mike in two terminal windows

- In the terminal window, run "sudo login", and then login as Alice.
- Start a new terminal window, run "sudo login", and then login as Mike.

3. Generate keys for Alice

- In Alice's terminal window, run "gpg --gen-key" to generate her public and private keys. Enter "DSA and Elgamal" for key kind, 2048 for key size, "key does not expire" for key expiration date, "Alice" for real name, *alice@pace.edu* for email address, "Alice's keys" as comment, and "Alice's passphrase" for passphrase. You may need to type over 284 random keys to generate enough entropy so the keys could be created.

4. Generate keys for Mike

- In Mike's terminal window, run "gpg --gen-key" to generate his public and private keys. Enter "DSA and Elgamal" for key kind, 2048 for key size, "key does not expire" for key expiration date, "Michael" for real name, *mike@pace.edu* for email address, "Mike's keys" as comment, and "Mike's passphrase" for passphrase. You may need to type over 284 random keys to generate enough entropy so the keys could be created.

5. Export Alice's public key to Mike

- In Alice's terminal window, run "gpg --armor --output alice-pk --export alice@pace.edu" to dump Ali's public key in file "alice-pk". You can run "more alice-pk" to review the public key.
- Run "sudo cp alice-pk /home/mike" to copy Alice's public key file "alice-pk" to Mike's home folder.

- In Mike's terminal window, verify the existence of file "/home/mike/alice-pk" by running "ls" in Mike's home folder ~ (/home/mike).
- In the same Mike's terminal window, run "gpg --import alice-pk" to import Alice's public key into Mike's key store.
- In the same Mike's terminal window, run "gpg --edit-key alice@pace.edu" to enter the editing session for Alice's public key. Type sub-command "fpr" to review the fingerprint of Alice's public key. Type sub-command "sign" to sign this key with Mike's key. You will be asked to enter Mike's passphrase, which is "Mike's passphrase". Type sub-command "check" to review who is on the signature list of Alice's public key, and we will see Alice (self-signature) and Mike on the list to confirm the validity of the key. You type sub-command "quit" to exit the editing session, and confirm to save the changes.

6. Create and encrypt a message

- In Mike's terminal window, run "cat > msg-to-alice" followed by the ENTER key, type "Alice's secret message", and then type key combination Ctrl+D to close the file. You just created a new text file "msg-to-alice" with contents "Alice's secret message".
- In Mike's terminal window, run "gpg --recipient alice@pace.edu --output secret-to-alice --encrypt msg-to-alice" to generate a new file "secret-to-alice" containing the encrypted version file "msg-to-alice".
- In Mike's terminal window, run "more secret-to-alice" to review the encrypted version of the message.
- In Mike's terminal window, run "sudo cp secret-to-alice /home/alice" to copy file "secret-to-alice" to Alice's home folder "/home/alice".
- In Alice's terminal window, run "ls" in Alice's home folder ~ (/home/alice) to verify the existence of file "secret-to-alice".

7. Decrypt the message

- In Alice's terminal window, run command "gpg --output msg-from-mike --decrypt secret-to-alice" to decrypt the contents of file "secret-to-alice" and save the result in a new file "msg-from-mike".
- In Alice's terminal window, run "more msg-from-mike" to review the decrypted message from Mike.

As our other lab modules, this module also has a rich set of evaluation questions to check how well the students have understood the essence of the lab module, and guide them to creatively apply the learned concepts and skills in solving related questions. The following are some sample review questions for this lab module:

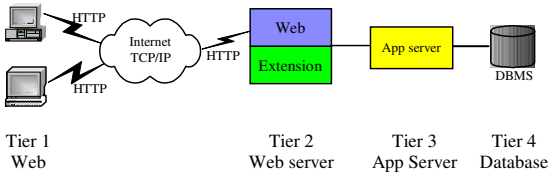
Question 1: Suggest some secure ways for distributing symmetric or public keys.
Question 2: Is email a secure way for distributing symmetric or public keys.
Question 3: Suppose Tom has the public key of Lisa. What is the best way for Tom to send his public key to Lisa?
Question 4: Suppose Tom has the public key of Lisa. What is the best way for Tom to send a secret message to Lisa?
Question 5: If a Word file is digitally signed, is the file also normally encrypted so it cannot be eavesdropped?
Question 6: Can you totally trust a company if that company has a digital certificate signed by VeriSign?
Question 7: Can technologies alone completely solve the network or web security problems?

SECURITY LAB MODULE ON WEB TECHNOLOGIES

Security lab module for introducing web technologies [8] covers the web architecture, uniform resource locators, HTML basics, the HTTP protocol, web session data management (cookies, hidden fields, query strings and server-side session objects). The lab has the following objectives:

a. Compare HTTP GET and HTTP POST requests;
b. Observe HTTP communications with proxy server Paros;
c. Experiment with cookies through web applications;
d. Compare web browser and web server interactions with HTML forms and with hyperlinks;
e. Learn how to use JavaScript to validate form data in the web browsers;
f. Learn how to create a static web site;
g. Learn how to create your first JSP web application on Tomcat;
h. Learn how to create your first servlet web application on Tomcat.

The clear tiered web architecture, as depicted below, shows the students the full landscape of modern web applications, justifies the separation of web servers and application servers, and puts all kinds of web technologies in perspective.



The lab has the following sub-sessions:

1) Comparing HTTP GET and HTTP POST Requests
2) Observing HTTP Communications with Paros
3) Working with Cookies
4) Submitting Data with HTML Form and Hyperlink

5) Validating Form Data with JavaScript
6) Creating Your First JavaServer Page Web Application
7) Creating Your First Servlet Web Application

Sample Lab for Creating a Simple Servlet Web Application

From the last exercise we know servelts are the cornerstone of Java web technologies. In this exercise you will develop a servlet web application with the same function as the last *welcomeJSP* web application so you can better compare the two technologies.

1. Launch the *ubuntu10* VM with username "user" and password 12345678.
2. In a file explorer, open "/home/user/tomcat/webapps". Right-click on any blank space in the explorer right pane and choose "Create Folder" to create new folder "/home/user/tomcat/webapps/welcomeServlet".
3. In the file explorer, open folder "~/tomcat/webapps/welcomeServlet". Right-click on any blank space in the explorer and choose "Create Document|Empty File" and create a new file with name "main.html".
4. Right-click on file "main.html" and choose menu item "Open With|gedit" to open file "main.html" in the gedit editor.
5. Type the following text into the file, and save the file.

```
<html>
<body>
<h2>My First Servlet Application</h2>
<form method="post" action="welcome">
Please enter your name: <input type="text"
name="name"/>
<br/>
<input type="submit" value="OK"/>
</form>
</body>
</html>
```

6. Create in folder "~/tomcat/webapps/welcomeServlet" a new folder "WEB-INF". All servlet based web applications have this folder for holding those files not directly accessible from web browsers.
7. Create in folder "~/tomcat/webapps/welcomeServlet/Web-INF" a new folder "classes". All Java classes to be run on the web server must be under this folder.
8. Create in folder "~/tomcat/webapps/welcomeServlet/Web-INF/classes" a new file "Welcome.java". Copy the following contents in this file and save the file. If a servlet needs to process HTTP POST requests, it needs to have a *doPost(request, response)* method. If a servlet needs to process HTTP GET requests, it needs to have a *doGET(request, response)* method. We put all logics in the *doPost(request, response)* method and call this method inside the *doGet(request, response)* method to avoid redundant code. Both of the two methods have two parameters: *request* representing all data submitted through the HTTP request, including data in the HTTP request entity body and query string; and *response* representing the data to be sent back to the remote web browser through an HTTP response. Method *request.getParameter("name")* is first called to retrieve the value that the user has typed in the *name* text field. After setting the output data type and retrieving the output object of *response*, the remainder of the code just prints out an HTML file piece by piece. This section of code must remind you of the similar code we reviewed for the Java code produced from file "Welcome.jsp" in the last exercise.

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class Welcome extends HttpServlet {
  public void doPost(HttpServletRequest request,
                     HttpServletResponse response)
      throws IOException, ServletException {
    String name = request.getParameter("name");
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<body>");
    out.println("<h2>Welcome, " + name + "</h2>");
    out.println("</body>");
    out.println("</html>");
  }

  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
    throws IOException, ServletException {
    doPost(request, response);
  }
}
```

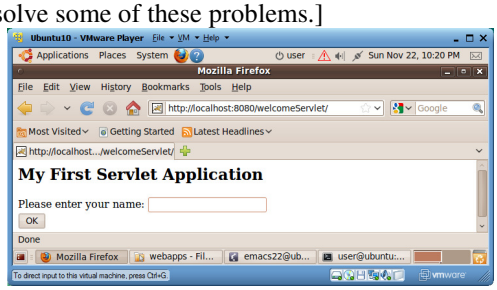
9. Create in folder "~/tomcat/webapps/welcomeServlet/Web-INF" a new file "web.xml" and copy the following contents into it. Each servlet web application needs this configuration file. You first declare that file "main.html" is the web application's welcome file: if a web browser visits this web application but not specifying which file to retrieve, the welcome file will be sent back by default. You then assign a name "welcome" to the servlet class "Welcome". In the last "servlet-mapping" element, you declare that if the URL of an HTTP request contains "/welcome", the request will be sent to the servlet named "welcome" for processing, which is "Welcome" in this case.

```
<web-app>
  <servlet>
    <servlet-name>welcome</servlet-name>
    <servlet-class>Welcome</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>welcome</servlet-name>
    <url-pattern>/welcome</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>main.html</welcome-file>
  </welcome-file-list>
</web-app>
```

10. Now you need to compile the servlet into a bytecode file "Welcome.class". Use the file browser to open folder "~/tomcat/webapps/welcomeServlet/Web-INF/classes". Right-click on any blank space in the right pane, and choose menu item "Open in Terminal". A new terminal window will start with "~/tomcat/webapps/welcomeServlet/Web-INF/classes" as its working folder. Run command "javac Welcome.java" to compile Java source file "Welcome.java" into "Welcome.class".
11. You have completed your first servlet web application. Use a web browser to visit *http://localhost:8080/welcomeServlet* and you will see a web browser view similar to the following one. [Sometimes *Tomcat* could get confused when you are developing web applications. Restart the VM may resolve some of these problems.]



12. JSP and servlet web applications are normally deployed as Web Archive (WAR) files. To make a WAR file "welcomeServlet.war" for the current web application, start a terminal window in folder "~/tomcat/webapps/welcomeServlet, and run command "jar cvf welcomeServlet.war *". To deploy this web application in a different Tomcat web server, you only need to drop file "welcomeServlet.war" in that Tomcat installation's "webapps" folder, and this WAR file will be automatically extracted into web application folder "welcomeServlet" and the web application will start to work right away, assuming that Tomcat is running.
13. By now you have successfully completed your first servlet web application. Congratulations!

Question 1: Where should Java servlet files be located in a servlet web application?
Question 2: Appletes are Java classes to be downloaded to web browsers and run in web browser sandboxes. Should applet files be put under folder "WEB-INF"?
Question 3: What is the main function of servlet methods doGet() and doPost()?
Question 4: What is the main function of configuration file "WEB-INF/web.xml"?
Question 5: What is URL pattern of a servlet?
Question 6: What is the relationship between a JSP page and a servlet class?

WEB SECURITY LABS ADOPTION EXPERIENCE

Most of our web security lab modules have been successfully incorporated in the undergraduate course "Overview of Computer Security" and graduate courses "Web and Internet Security" and "Concepts and Structures of Internet Computing". We collected students' feedback on the SWEET modules adopted in two classes of "Overview of Computer Security" and "Web and Internet Security" offered in Fall 2009. A web-based survey using 5-point Likert scale was conducted at the end of the semester. The survey included questions to elicit their feedback on the lecture materials, laboratory exercises, the mapping between the lecture and the lab and the overall impact of these modules on their learning. Our results show that the students had invested significant amount of time (2-4 hours per week on average) in completing hands-on exercises. However, they generally agreed that the course materials were planned well (average 4.1 for lecture category), the exercises had drawn their interests (average 4.1 for lab exercise category), the exercises had helped them in learning the course materials (average 4.1 for the mapping between labs and lecture), and they would be interested in pursuing further in the Information Assurance area (average 3.9 for overall category).

CONCLUSION

This paper describes selected web security lab modules of our NSF SWEET project. These modules are available for free sharing with our colleagues. The adoption of the virtualization technology by these modules makes them effective in improving hands-on exercises/projects and online delivery of web security/technology courses.

ACKNOWLEDGEMENT

This work was supported by the NSF CCLI 0837549. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the National Science Foundation or the U.S. government.

REFERENCES

[1] VMware Inc., "VMware Player", http://www.vmware.com/products/player

[2] Microsoft Inc., "Microsoft Virtual PC", http://www.microsoft.com/windows/downloads/virtualpc

[3] Tao, L, "A Tutorial on Setting up Ubuntu Linux Virtual machines", http://csis.pace.edu/lixin/ubuntu/LinuxTutorial.pdf, 2010

[4] Tao, L, "ubuntu10basic.exe", http://csis.pace.edu/lixin/ubuntu/ubuntu10basic.exe, 2010

[5] Tao, L, "ubuntu10.exe", http://csis.pace.edu/lixin/ubuntu/ubuntu10.exe, 2010

[6] Tao, L, "A Video Tutorial on Setting up Ubuntu Linux Virtual machines", http://csis.pace.edu/lixin/ubuntu904, 2009

[7] Tao, L, "Introduction to Cryptography", http://csis.pace.edu/lixin/download/lab-cryptography.pdf, 2010

[8] Tao, L, "Introduction to Web Technologies", http://csis.pace.edu/lixin/download/lab-introduce-web.pdf, 20