CS-745 Project Statement

Submission Deadline: 11 May 2024 (Saturday) 23:55 Hrs

Build a Transparent SSL-Proxy Server

SSL or TLS is an application layer protocol for communication security. It allows two communicating end-points to protect their communication from a third party. These two communicating parties set up an encrypted tunnel between them. The encryption/decryption of this communication tunnel is done using a symmetric key, which is negotiated using the SSL/TLS protocol. SSL/TLS protocol requires the exchange of digital certificates to ensure the authenticity of communicating parties to each other. Usually, a PKI is involved in distributing digital certificates. PKI can be skipped if the two parties know each other's digital certificates by direct introduction.

Experiment Setup

1. Install VirtualBox
2. Install a Linux VM (with a minimal package list, without a graphical interface but with an OpenSSH server) let us call this VM `vm-nox-vanilla`
3. Install a Linux VM (with graphical support, you require only a browser, other packages can be skipped from installing, but keep the OpenSSH server) let us call this VM `vm-gui-vanilla`
4. Clone `vm-nox-vanilla` into `vm-server-A` and `vm-proxy`
5. Clone `vm-gui-vanilla` into `vm-browser-1`
6. Now you have 3 VMs in your VirtualBox: one VM with Firefox, and a second VM with routing tables to direct Firefox traffic to the application installed in VM3. The second VM acts as a proxy between the browser and the web server
7. The experimental setup is depicted in Figure 1, where all communication between the browser and the server is passing through the proxy (You may need to tweak the NETWORK MANAGER interface of your VirtualBox software)

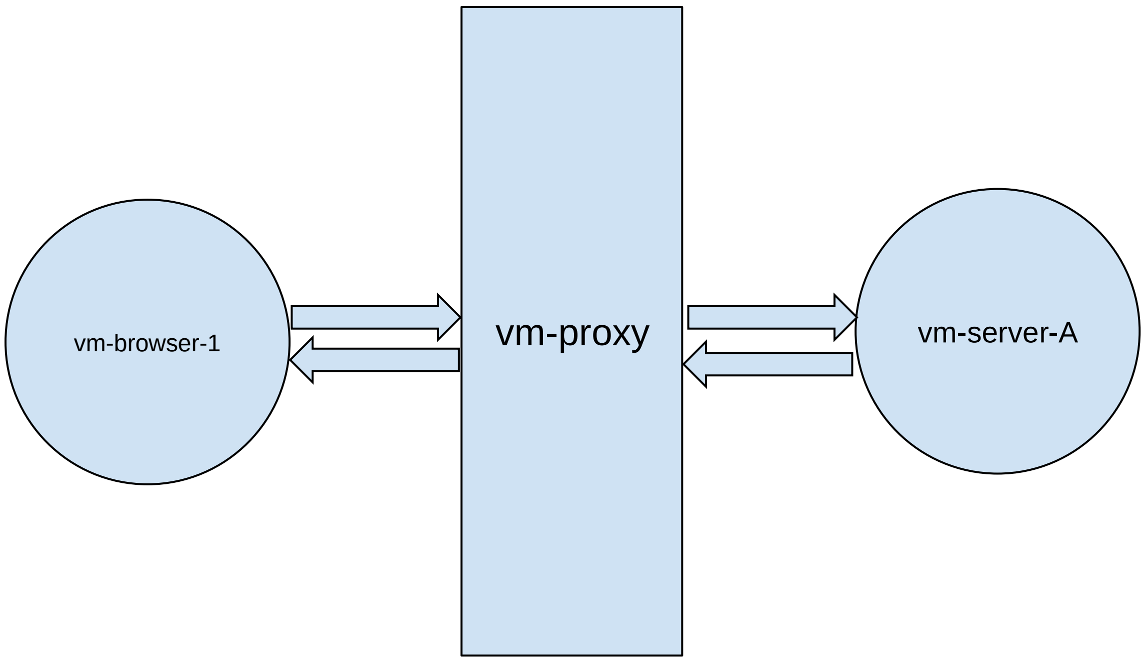

Figure 1

Task 1

1. Install Apache2, postgresql, phpbb on the webserver to start a bulletin board using phpbb. You may alternatively skip phpbb and write your own bulletin board application.
2. Start network traffic collection on `vm-proxy`
3. Configure your bulletin board in such a way that XSS attack is possible (you may need to clone `vm-gui-vanilla` into `vm-browser-2`) browser-2 will be the victim
4. Implement XSS mitigation technique on the web-server
5. Save the traffic as `traffic-task-1.pcap`

Task 2

1. Extend the setup in Figure 1 to show an example of CSRF attack (you may need to deploy one more web-server, clone it from `vm-nox-vanilla`)
2. Implement XSS mitigation technique
3. Continue capturing the network traffic and once mitigation is implemented, save it as `traffic-task-2.pcap`

Task 3

1. Install a Single-Sign-On authentication module on your web-applications (vm-server-A and vm-server-B)
2. Try to log into application-A and application-B using single-sign-on (either Google/Facebook/Twitter)
3. Continue capturing the network traffic and save it as `traffic-task-3.pcap`

Task 4

1. Install self-signed digital certificates on vm-server-A and vm-server-B (change Apache configuration file accordingly)
2. Check that traffic intercepted on vm-proxy is no more plaintext and is encrypted
3. Save traffic as `traffic-task-4.pcap`

Task 5

1. Disable the Single-Sign-On authentication module
2. Enable user authentication from browser-1 and browser-2 to the web applications using client-side digital certificates
3. Save traffic as `traffic-task-5.pcap`

Task 6

1. Modify the proxy to act as a transparent SSL-proxy. In other words, when the proxy becomes a SSL-proxy, it establishes two SSL tunnels on either side of it. That is: Browser wants to establish a secure (SSL/HTTPS) connection with the web server but the proxy intercepts this request between Browser and Web server and establishes one SSL connection with the Browser pretending to be the intended web-server and another SSL connection with the web server to pretend as Browser. So, both of the communicating entities (browser & web-server) do not know traffic interception.
2. Save traffic as `traffic-task-6.pcap`