# Patching & Remediation

All of the efforts we have made so far were intended to identify previously unknown vulnerabilities and then patch them. So, this last step is critical, as our whitebox pentesting exercise would only be complete with it. We must end our whitebox pentesting exercise with a thorough review containing our findings and knowledge and, most importantly, detailed code patches to remediate the identified vulnerabilities.

## Patching

Before writing our report, we should patch all identified vulnerabilities in our test environment. Since we have an excellent understanding of how each function works and how the vulnerability is occurring, it should be clear what we should change to prevent it. Of course, this can only be done where possible, as certain applications and vulnerabilities would make the patching very complicated and outside the scope of our testing.

We can apply these patches and then re-test our PoC exploit to ensure the application is no longer vulnerable. We must also go through the Local Testing process to ensure the vulnerability is remediated at every stage. If, at any stage, we notice that the vulnerability still exists (e.g. payload not being filtered properly), we should update our patches and start the testing again. Most importantly, we should ensure that our patches do not break any original functionality.

We have already established that the code reviewing and vulnerability analysis processes are different for each application, and so is the patching process. Throughout the various whitebox pentesting modules in HackTheBox Academy, including this one, we will provide different examples of patching and testing our code to ensure we deliver the best results.

## Reporting

Like any other pentesting exercise, we end our pentest by writing a full report detailing the steps necessary for exploitation and how to use the PoC script. We should also include a detailed review of each function and point out any potential issues we identified. We should provide the code patches we tested with the exact details of what we modified so senior software engineers can review our changes and ensure they comply with their standards and do not break anything.

If we could patch the vulnerability and maintain the full functionality of the application, we can add a note saying that the patch has been verified as working. Patching certain major vulnerabilities may take more work as they affect multiple other functions. If this is the case, we can suggest a patch and note that it has not yet/fully been tested.

It is also recommended that we provide some secure coding tips to avoid introducing similar vulnerabilities in the future. These would be similar to the ones we provide in our various whitebox pentesting and secure coding modules. Of course, these tips would only be highlights, so software engineers are always recommended to learn secure coding directly, as that would be the best way to reduce the flaws and vulnerabilities in their applications.