CRY300 Lab 3: Heartbleed

SecureSet Academy

CORE: Crypto 300

October 1, 2020

CRY300 Lab 3: Heartbleed

Lab Overview

The Heartbleed vulnerability takes advantage of the connection between two servers/or server-client where one member sends a “health check” of the other during sessions of inactivity. This is done by a requestor sending a small 16-bit number and gets a reply with a similar payload. On vulnerable servers, the Heartbleed vulnerability is performed by requesting a much bigger payload (64KB) with the server responding with information to fit the bill, potentially containing sensitive information.

In this lab, we will execute this attack to retrieve a private key of the server with the intent to decrypt content. We will first install the compromised version of openssl (1.0.1) where this vulnerability can be exploited, then download the heartleech program that attacks this vulnerability after setting up a connection with itself (localhost). This vulnerability searches the dumped memory payload for p’s and q’s and divides them into the cert at the initial connection handshake (i.e. to find the private key).

Software Compilation and Installation

First, we download the unpatched openssl 1.0.1 source code from openssl.org. Without this, we will not be able to conduct the exploit. This version was installed into the /home/splash/src directory we made. We then modify the source code by applying a patch that fixes expired certs in compilation testing. We then compile the openssl source code dependencies before the main software (with “make” command). We move the compiled code to the home directory and out of the src directory. We need to make sure the home directory libraries are linked to the openssl binary we made. We created an openssl_heartbleed link in the home folder to make it convenient (Figure 1) and visual that we are using the vulnerable binaries.

I made an observation that if I typed “openssl version” I saw the patched version (1.1.1 from Sept 2018) so it was good to double check that my original openssl libraries were not compromised. That means the commands we typed to get the vulnerable openssl (Figure 2) did not alter the openssl libraries I work with not using the exploit. ***Using putty made it easier to copy/paste from the PDF document***

Figure 1

Linked vulnerable openssl libraries

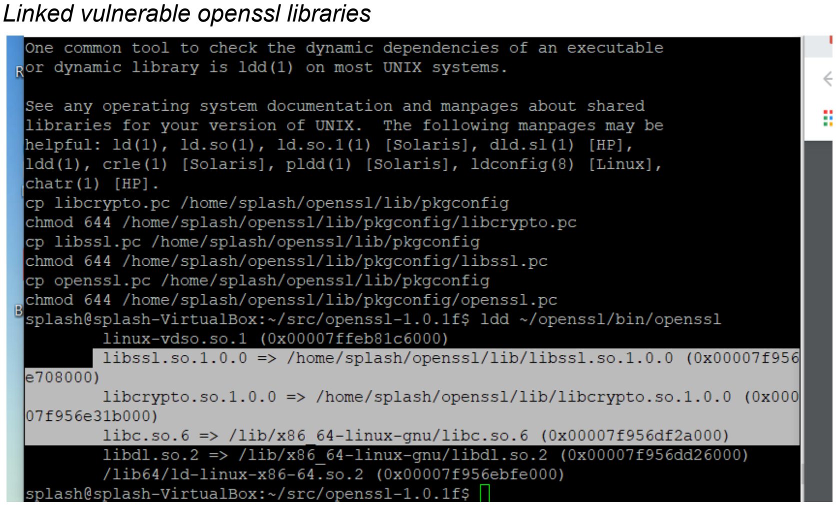
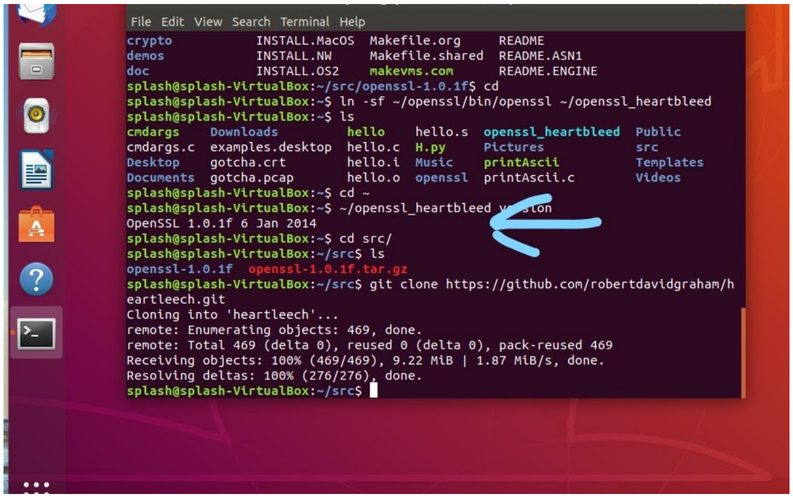


Figure 2

Vulnerable version was compiled



Proof of Concept

We have retrieved the heartleech program from GitHub, compiled it, and checked to make sure it was properly installed (Figure 3). Next, we generated a private key and created a self-signed cert called “root_ca.crt” (this was not made as a .cer extension, but a CEL ASCII file) with a Certificate Name of Heartbleed Root CA, it expires in a year, and with a sha256 algorithm to hash the cert. We make a CSR with another key from the SSL server to use in the SSL handshake, without it we could not set up a connection. To finish the CSR, we sign the cert with our root CA private key (Figure 4).

The exploit happens when the attacker uses the openssl heartbeat vulnerability to capture the root private key. We have to launch a listening server (“s_server” command) on port 8443 with its SSL key and crt available to legitimize the connection. ****The -www flag is crucial. According to the manpage, without it, it won’t send the status message (aka heartbeat) when the client connects!! Otherwise I would assume we’d have to wait for a heartbeat health check**** We open another separate terminal that will act as the attacker. After this connection is established, the attacker will use the heartleech attack on the localhost server listening on 8443 and grab the leeched.key (RSA private key from server, we renamed it) from the server’s -www flag forced health check heartbeat (Figure 5).

The heartleech program took advantage of the heartbeat sending a request for a 64KB response, the server responded, dumped some memory, and heartleech searched for the p’s and q’s from the dumped memory to grab the RSA private key and save it as leeched.key. We compared moduluses of both the heartbleed (server) key and the leeched key and they were the same, meaning we grabbed the private key from the buffer overread.

Figure 3

Heartleech installed

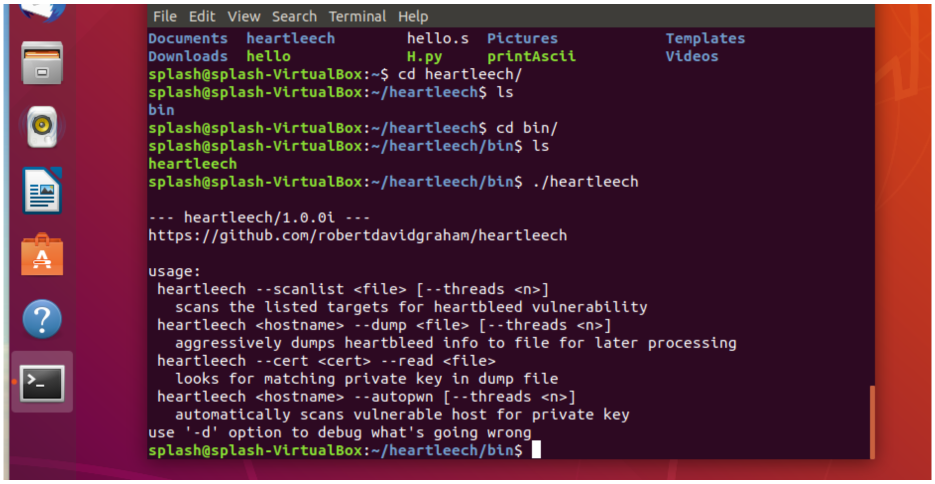


Figure 4

Signed Cert Request

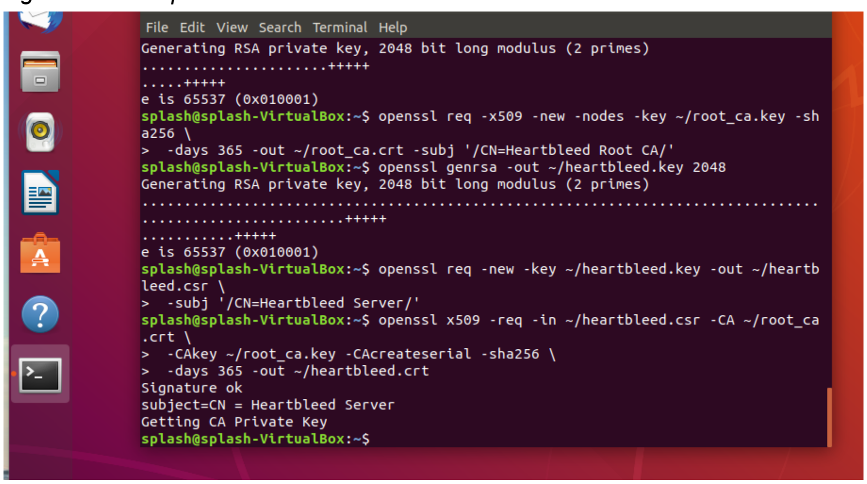
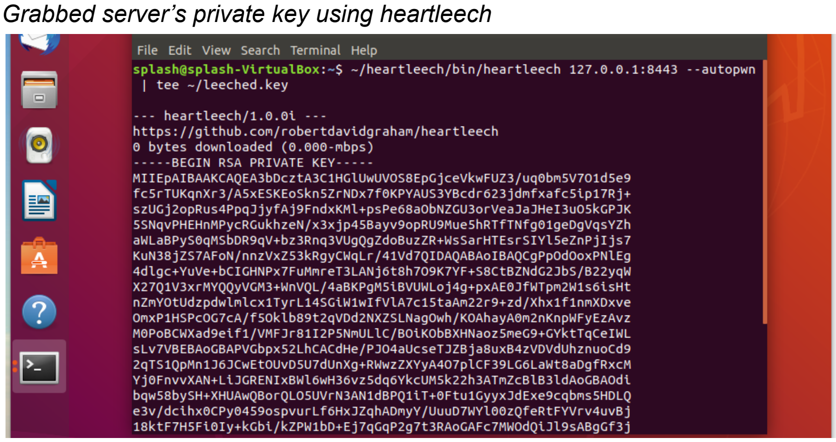


Figure 5

Grabbed server's private key using heartleech



Simulating Data Theft

This part of the lab shows us how we can use the grabbed key to decrypt data. We create some random hex text then use the web server's certificate (with public key) to encrypt the data. We took a look at the data in base64 encoding (Figure 6). With the leeched key (RSA private key we grabbed from the memory dump) we can successfully decipher the encrypted hex data we inputted, hence a data theft (Figure 7). Basically if we can grab some encrypted files heading to the server, we can use the private key we have to decipher them. We approached it like this to show that data can be vulnerable if we have the private key AND we can capture traffic encrypted with the public key. This can provide an essential MITM exploit if we can grab that traffic (i.e. you head to a site, send data/certs/DSA signatures encrypted with the public key) to decipher it.

Figure 6

Encrypted text file

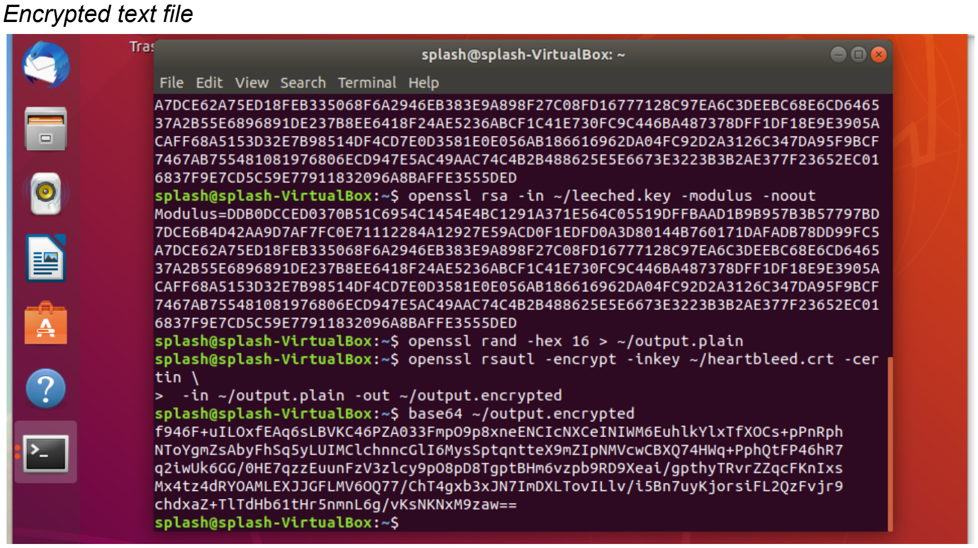
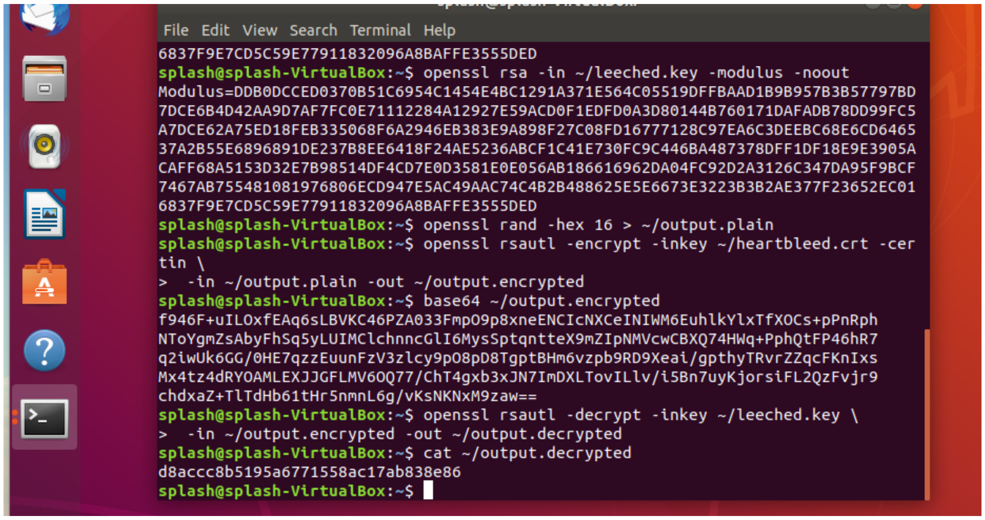


Figure 7

Deciphered data



Summary

Heartleech is a well crafted program used to take advantage of the heartbleed buffer overread. It takes advantage of this by searching for key values in the memory dump that could contain the RSA private key. If you can create a MITM between a client and the vulnerable server and you've already grabbed the RSA private key, you can see any encrypted traffic the client sends using the public key. Potentially, if you somehow have a backdoor into the server, you can reinstall the openssl vulnerability using the commands we inputted during the install.

References

“CRY 300 Heartbeat” PDF. SecureSet Academy. (2020). Canvas Portal: CRY300 Course Materials

“Heartleech.” Graham, Robert David. 2014. https://www.recordnotfound.com/heartleech-robertdavidgraham-31360

Ubuntu 18 manpages for “openssl,” “s_server”