

1. ZAP

Client-side bypassing

1. V5.3 Output encoding and Injection Prevention

Test Case id: 5.3.4-1

Description: Verify that data selection or database queries (e.g. SQL, HQL, ORM, NoSQL) use parameterized queries, ORMs, entity frameworks, or are otherwise protected from database injection attacks.

Repeatable Steps:

1. Open ZAP in vcl.
2. Click on Manual Explore.
3. Enter the openemr URL “<http://localhost/openemr>” in the URL dialogue box, and then click on launch browser with Firefox as the default browser.
4. Login as administrator in firefox browser with username: **admin_openemr** and password: **admin_openemr**
5. In ZAP, there is a tab towards the left hand side. Under Sites, expand <http://localhost>, openemr, interface, main.
6. Under main directory, you can find a POST:main_screen.php file.
7. Right click on POST:main_screen.php file and select Break, and click on save.
8. In firefox browser, logout and login as administrator again with exact credentials. Username: **admin_openemr** and password: **admin_openemr**
9. You can see a HTTP Request dialogue box.
10. Go back to the ZAP application. You will be able to see that application stopped at the POST end-point since we added the break point.
11. Click on **Request** on top header of the application.
12. In the dialogue box below (Break tab), you'll be able to see the administrator credentials. Update authUser=admin_openemr to '**' union (select @@version --**
13. Click on the play icon to forward through on top of the ZAP application.
14. Go back to firefox browser to see the HTTP response dialogue box.
15. Click on continue in the firefox browser HTTP window.

Expected Results: The login attempt should fail displaying an error message saying “Invalid username or password”. No database version information is displayed on the screen.

CWE: 89, Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

2. V5.3 Output encoding and Injection Prevention

Test Case id: 5.3.4-2

Description: Verify that data selection or database queries (e.g. SQL, HQL, ORM, NoSQL) use parameterized queries, ORMs, entity frameworks, or are otherwise protected from database injection attacks.

Repeatable Steps:

1. Open ZAP in vcl.
2. Click on Manual Explore.
3. Enter the openemr url "**http://localhost/openemr**" in the URL dialogue box, and then click on launch browser with Firefox as default browser.
4. Login as administrator in firefox browser with username: **admin_openemr** and password: **admin_openemr**
5. On the top headers, hover over **Patient/Client** and click on **Patients**.
6. Under Patient finder, click on Add New Patient.
7. Create a new patient. Select Mr. from pronouns drop down. Type **Abc** as first name in first box, leave out second box and type **def** as last name in third box.
8. Select DOB as March 07 2022.
9. Select Sex as **Male**.
10. Scroll down and click on **Create New Patient**.
11. Note: Sometimes you'll not be able to see Create New Patient in firefox browser.
Click on green icon on bottom right corner to hide history details.
12. In ZAP, there is a tab towards the left hand side. Under Sites, expand **http://localhost, openemr, interface, new**
13. Under new directory, you can find a POST:**new_comprehensive_save.php**(csrf token....)
14. The URL in request will be
http://localhost/openemr/interface/new/new_comprehensive_save.php
15. Right click on POST:**new_comprehensive_save.php** file and select Break, and click on save.
16. Go back to the firefox browser and repeat steps 5,6,7,8,9,10.

17. When you click on Create New Patient in step 10, you'll be able to see a HTTP request dialogue box.
18. Come back to ZAP application to find that the application stopped at the break point.
19. In the dialogue box below (Break tab), you'll be able to see the **form_DOB** in the payload. Update form_DOB=03-07-2022 to **form_DOB=' or 1=1-**
20. Click on the play icon to forward through on top of the ZAP application.
21. Go back to firefox browser to see the HTTP response dialogue box.
22. Click on continue in the firefox browser HTTP window.
23. If any API error window pops up, close error window.
24. On the top headers, hover over **Patient/Client** and click on **Patients** to see the new Patient details created just now.

Expected Results: The new patient should not be created.

CWE: 89, Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

3. V4.1 General Access Control Design

Test Case id: 4.1.1-1

Description: Verify that the application enforces access control rules on a trusted service layer, especially if client-side access control is present and could be bypassed.

Repeatable steps:

1. Open ZAP in vcl.
2. Click on Manual Explore.
3. Enter the openemr URL "<http://localhost/openemr>" in the URL dialogue box, and then click on launch browser with Firefox as the default browser.
4. log in as administrator in firefox browser with username: **admin_openemr** and password: **admin_openemr**
5. On the top headers, hover over **Patient/Client** and click on **Patients**.
6. Under Patient finder, click on Add New Patient.
7. Create a new patient. Select Mr. from pronouns drop down. Type **Abc** as the first name in the first box, leave out the second box, and type **def** as the last name in the third box.
8. Select DOB as March 07, 2022.

9. Select Sex as **Male**.
10. Scroll down and click on **Create New Patient**.
11. Note: Sometimes you'll not be able to see Create New Patient in the firefox browser. Click on the green icon on the bottom right corner to hide history details.
12. On the top headers click on **Messages**.
13. Under Messages, Reminders, Recalls click on **Add New**.
14. Select **Chart Note** under Type.
15. Click on the Patient box and select **Def,abc** from the Patient name drop down.
16. Click on Add to list, and click on ok.
17. Under To: select **Administrator, Administrator**
18. Type a message as **hello** in the message field.
19. Click on send message.
20. In ZAP, there is a tab towards the left-hand side. Under Sites, expand <http://localhost/openemr/interface/main/messages>.
21. Under the orders directory, you can find a POST:messages.php(begin, form_active, showall, sort by, sort order)
22. The URL in request will be
http://localhost/openemr/interface/main/messages/messages.php?showall=no&sortby=&sortorder=&begin=&form_active=1
23. Right-click on POST:messages.php file and select Break, and click on save.
24. Go back to the firefox browser and repeat steps 12,13,14,15,16,17,18,19.
25. When you click on send message in step 19, you'll be able to see a HTTP request dialogue box.
26. Come back to the ZAP application to find that the application stopped at the break point.
27. In the dialogue box below (Break tab), you'll be able to see the **form_note_type** in the payload. Update form_note_type=Chart+Note to
form_note_type=<script>alert('xss')</script>
28. Click on the play icon to forward through on top of the ZAP application.
29. Go back to the firefox browser to see the HTTP response dialogue box.
30. Click on continue in the firefox browser HTTP window.
31. If any API error window pops up, close the error window.
32. Close the messages dialog box and on top of the web page click on messages again to refresh the messages window.

Expected Results: Since the new message form_note_type contains malicious or invalid characters, the new message should not be stored and sent.

CWE: 602, Client-Side Enforcement of Server-Side Security

4. V5.1 Input Validation

Test Case id: 5.1.3-1

Description: Verify that all input (HTML form fields, REST requests, URL parameters, HTTP headers, cookies, batch files, RSS feeds, etc) is validated using positive validation (allow lists).

Repeatable Steps:

1. Open ZAP in vcl.
2. Click on Manual Explore.
3. Enter the openemr URL "<http://localhost/openemr>" in the URL dialogue box, and then click on launch browser with Firefox as the default browser.
4. log in as administrator in firefox browser with username: **admin_openemr** and password: **admin_openemr**
5. On the top headers, hover over **Patient/Client** and click on **Patients**.
6. Under Patient finder, click on Add New Patients.
7. Create a new patient. Select Mr. from pronouns drop down. Type **test** as the first name in the first box, leave out the second box, and type **test** as the last name in the third box
8. Select DOB as June 5th, 1975.
9. Select Sex as **Male**.
10. Scroll down and click on **Create New Patient**.
11. Close the success patient created dialogue box.
12. Click on patient finder and the patient we just created ,
(test, test)
13. On bottom right, Under Recurrent Appointments, click on **Edit Allergies**
14. Click on **Add** allergies, select **penicillin** as the title and select the begin date as **2022-03-08** and Reaction as **Nausea** click on **save** allergy
15. Note: Sometimes you'll not be able to see Create New allergy for a patient in firefox browser. Click on green icon on bottom right corner to hide history details.
16. In ZAP, there is a tab towards the left hand side. Under Sites, expand http://localhost/openemr/interface/patient_file/summary
17. Under this directory you can find POST:`add_edit_issue.php` entry, right click that and click on break and save it.
18. Under Request tab, the URL will be
http://localhost/openemr/interface/patient_file/summary/add_edit_issue.php?issue=0&thistype=allergy

19. Repeat the steps from 12-14 after click on save allergy, we can change the actual input (In break tab) for the following input **form_reaction** in the response tab to series of character 'a' of length 1500 (Buffer Overflow)
20. Click on the play icon to forward through on top of the ZAP application.
21. Go back to firefox browser to see the HTTP response dialogue box.
22. Click on continue in the firefox browser HTTP window.
23. On patient Issues tab, we can see the list of allergies, in the allergy we created we can check that Reaction is series of character 'a' which is not part of our options in the select element.

Expected Results: The Allergy for the corresponding patient **test**, should **not** be created.

CWE: 20, Improper Input Validation

5. V5.1 Input Validation

Test Case id: 5.1.3-2

Description: Verify that all input (HTML form fields, REST requests, URL parameters, HTTP headers, cookies, batch files, RSS feeds, etc) is validated using positive validation (allow lists).

Repeatable steps:

1. Open ZAP in vcl.
2. Clock on Manual Explore.
3. Enter the openemr url "**http://localhost/openemr**" in the URL dialogue box, and then click on launch browser with Firefox as default browser.
4. Login as administrator in firefox browser with username: **admin_openemr** and password: **admin_openemr**
5. On the top headers, hover over **Procedures** and click on **Batch Results**.
6. Click on the field corresponding to the "Procedure:". You will be popped up with a form containing heading as "**Configure Orders and Results**"
7. Click on the "**+Add Top Level**" button and you will be popped up with a form.
8. In that form fill the values as "Procedure Tier: Group", "Name: dummy name", "Description: dummy" and "Sequence: dummy" and click Save.
9. In ZAP, there is a tab towards the left-hand side. Under Sites, expand <http://localhost>, openemr, interface, orders.

10. Under the orders directory, you can find a POST:types_edit.php(parent, typeid)(form_body_site, form_description, form_diagnosis_code, form_laterality, form_name, form_procedure_code, form_procedure_type, from_range, form_realted_code, from_route_admin, form_save, form_seq, form_specimen, form_standard_code, form_units)
11. The URL in request will be
"https://localhost/openemr/interface/orders/types_edit.php?typeid=0&parent=0".
12. Right click on the URL mentioned in the step 10, and select Break, and click on save.
13. Go back to the firefox browser and repeat steps 5, 6, 7, 8.
14. When you click on Save this time as mentioned in the step 8, you might be able to see a HTTP request dialogue box.
15. Come back to the ZAP application to find that the application stopped at the break point.
16. In the dialogue box below (Break tab), you'll be able to see the **form_procedure_type** in the payload. Update form_procedure_type value with **form_procedure_type=' ; drop table temp --**
17. Click on the play icon to forward through on top of the ZAP application.
18. Go back to the firefox browser to see the HTTP response dialogue box if present otherwise no need to worry about it.
19. Click on continue in the firefox browser HTTP window.
20. If any API error window pops up, close the error window.
21. Close all the dialog boxes and on top of the web page and logout and login to the application again, on top of the headers hover over **Procedures** and click on **Batch Results** to navigate again to the same page and click on the field corresponding to "Procedure:" and check for the "Category" field value by clicking on the Edit(pencil) button of the row containing your order.

Expected Results: Since the new order form_procedure_type contains malicious or invalid characters, the new message should not be stored and sent to the backend.

CWE: 20, Improper Input Validation

Total time spent on Black Box Test Cases:

1. Individually we spent 2hours each planning and running the black-box test cases. So a total of 8hours was spent on black-box test cases.
2. Out of 5 black-box test cases, we found 4 vulnerabilities.
3. So, we found 1 true positive per hour.

Fuzzing

1. Ruleset Used: SQL Injection

Location: Creating a new appointment

URL: method: POST, end-point:

http://localhost/openemr/interface/main/calendar/add_edit_event.php

Results: True Positive, for **form_patient=' or 1=1 -**, the new meeting record is saved with no patient information

Test Case ID: 5.3.4-1

Description: Verify that data selection or database queries (e.g. SQL, HQL, ORM, NoSQL) use parameterized queries, ORMs, entity frameworks, or are otherwise protected from database injection attacks.

Preconditions:

- A patient exists on this instance of OpenEMR
- If a patient does not exist, follow these steps to create one:
 - Move the cursor over **Patient/Clinic** and select **New/Search** from the dropdown
 - Give the patient a name (e.g. Test, Test), a DOB, and Sex
 - Select **Create New Patient** on the bottom of the screen

Repeatable Steps:

1. In VCL, Open OWASP ZAP and click on the “quick start” browser that is automatically configured to use the ZAP proxy. (a small firefox icon on top of the ZAP application).
2. Make sure that you install the Firefox browser.
3. Navigate to the openemr webpage in firefox. <http://localhost/openemr>
4. Login as admin with username: `admin_openemr` and password: `admin_openemr`
5. Under the **Calendar** tab, select the ‘+’ icon to create a new appointment
6. Click the patient field and search for any patient (e.g. test)
7. Add this patient to the meeting
8. Note the date and time of the appointment
9. Click Save. If a popup appears saying the provider is not available, click OK
10. Go back to the ZAP application and open the History tab at the bottom of the application.
11. Find the POST endpoint with URL:
`http://localhost/openemr/interface/main/calendar/add_edit_event.php?eid=`
12. In the Request tab, select the `form_patient` value.
13. Right-click on the `form_patient` value and select fuzz.
14. Select Payloads from the Fuzzer window.
15. Click on Add.
16. Select File Fuzzers as Type.
17. Expand jbrofuzz and select SQL Injection.
18. Click on Add and then OK to return back to the Fuzzer window.

19. Click Start Fuzzer to begin the fuzzing process.
20. Go through the results and find the payload with form_patient=' or 1=1–
21. Go back to OpenEMR and select the date on the calendar for which the original appointment was made.

Expected Results: The new appointment should not be created since the patient field contains malicious code.

CWE: 89 - Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Mitigation techniques to fix this issue:

- Input validation is not available on the server-side. Since DOB is a calendar date, we need to make sure that there are no special characters except the hyphen in the DOB input.
- Add an allow list for the DOB input validation like regex.
- Use prepared statements so that they can sanitize the input while executing SQL queries.
- Using database frameworks like Hibernate would mitigate this issue.

Screenshot of ZAP:

The screenshot shows the OWASP ZAP interface. The top part displays a browser window with a POST request to 'http://152.7.99.96/openemr/interface/main/calendar/add_edit_event.php?eid=' with various headers and a complex payload. The payload includes SQL injection code such as ' or 1=1–'. The bottom part shows a table titled 'Fuzzer' with 17 rows of fuzzing results. The table columns include Task ID, Message Type, Code, Reason, RTT, Size Resp. Header, Size Resp. Body, Highest Alert, State, and Payloads. The 'Highest Alert' column for row 11 shows 'Reflected' with a warning icon, and the 'Payloads' column for the same row shows the SQL injection payload 'or 1=1–'.

Task ID	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	State	Payloads
6 Fuzzed		200	OK	321 ms	316 bytes	11,845 bytes			exec sp_addrolemember...
7 Fuzzed		200	OK	311 ms	316 bytes	11,845 bytes			insert into mysql user (us...
8 Fuzzed		200	OK	319 ms	316 bytes	11,845 bytes			grant connect to name, g...
9 Fuzzed		200	OK	308 ms	316 bytes	11,845 bytes			insert into users(login, pa...
10 Fuzzed		200	OK	313 ms	316 bytes	11,845 bytes			
11 Fuzzed		200	OK	309 ms	316 bytes	11,845 bytes	Reflected		' or 1=1 –
12 Fuzzed		200	OK	269 ms	316 bytes	11,845 bytes			' union (select @@versio...
13 Fuzzed		200	OK	318 ms	316 bytes	11,845 bytes			' union (select NULL, (sel...
14 Fuzzed		200	OK	295 ms	316 bytes	11,845 bytes			' union (select NULL, NU...
15 Fuzzed		200	OK	306 ms	316 bytes	11,845 bytes			' union (select NULL, NU...
16 Fuzzed		200	OK	266 ms	316 bytes	11,845 bytes			' union (select NULL, NU...
17 Fuzzed		200	OK	222 ms	316 bytes	11,845 bytes			' union (select NULL, NU...

2. Ruleset Used: SQL Injection

Location: While creating a new Patient.

URL: method: POST, end-point:

http://localhost/openemr/interface/new/new_comprehensive_save.php

Results: True positive, for **form_DOB=' or 1=1–**, the new patient record will be saved with DOB=0000-00-00.

Test Case id: 5.3.4-2

Description: Verify that data selection or database queries (e.g. SQL, HQL, ORM, NoSQL) use parameterized queries, ORMs, entity frameworks, or are otherwise protected from database injection attacks.

Repeatable steps:

1. In VCL, Open OWASP ZAP and click on the “quick start” browser that is automatically configured to use the ZAP proxy. (a small firefox icon on top of the ZAP application).
2. Make sure that you install the Firefox browser.
3. Navigate to the openemr webpage in firefox. <http://localhost/openemr>
4. Login as admin with username: admin_openemr and password: admin_openemr
5. On top headers, hover on **Patient/Client** and click on Patients.
6. Under Patient Finder click on **Add New Patient**
7. Enter Name. Select Mr. from the dropdown. Enter abc in the first box and def in the last box (middlebox can be left empty).
8. Select DOB as 03/04/2022 from the calendar.
9. Select Sex as Male from the dropdown.
10. Scroll down the page and click on Create New Patient.
11. Go back to the ZAP application and open the History tab at the bottom of the application.
12. Find the POST endpoint with URL:
http://localhost/openemr/interface/new/new_comprehensive_save.php
13. In the Request tab, select the form_DOB value.
14. Right-click on the form_DOB value and select fuzz.
15. Select Payloads from the Fuzzer window.
16. Click on Add.
17. Select File Fuzzers as Type.
18. Expand jbrofuzz and select SQL Injection.
19. Click on Add and then OK to return back to the Fuzzer window.
20. Click Start Fuzzer to begin the fuzzing process.
21. Go through the results and find the payload with **form_DOB=' or 1=1–**
22. Go back to OpenEMR and hover on Patient/Client on the top headers and click on Patients.

Expected Results: The new patient should not be created since the DOB field contains malicious code.

CWE: 89, Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Mitigation techniques to fix this issue:

1. Input validation is not available on the server-side. Since DOB is a calendar date, we need to make sure that there are no special characters except the hyphen in the DOB input.
2. Add an allow list for the DOB input validation like regex.
3. Use prepared statements so that they can sanitize the input while executing SQL queries.
4. Using database frameworks like Hibernate would mitigate this issue.

Screenshot of ZAP:

POST http://152.7.177.239/openemr/interface/new/new_comprehensive_save.php HTTP/1.1
Host: 152.7.177.239
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:97.0) Gecko/20100101 Firefox/97.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Content-Type: application/x-www-form-urlencoded
Content-Length: 3389
Origin: https://152.7.177.239
Connection: keep-alive
Referer: https://152.7.177.239/openemr/interface/new/new.php
Cookie: OpenEMR=6kE6Z1OKtQzusw2qGryI-d15suD1L1U%2CzrTKgyE6NEP0
Upgrade-Insecure-Requests: 1

csrf_token_form=a35d36fa2af7f0ae32a9520f894f6dcddada79&form_cb_1=l&form_title=Mr.&form_mname=&form_lname=Attacker&form_pubpid=&form_DOB=create user name identified by 'pass123'&form_sex=Male&form_ss=c&form_drivers_license=&form_status=&form_genericname1=&form_genericval1=&form_genericname2=&form_genericval2=&form_billing_note=&form_street=&form_city=&form_state=&form_postal_code=&form_country_code=&form_mothersurname=&form_contact_relationship=&form_phone_contact=&form_phone_home=&form_phone_biz=&form_phone_cell=&form_email=&form_email_direct=&form_providerid=&form_pharmacy_id=&form_hipaa_notice=&form_hipaa_voice=&form_hipaa_message=&form_hipaa_mail=&form_hipaa_allowsms=&form_hipaa_notice=&form_allow_imrn_use=&form_allow_imrn_info_share=&form_allow_health_info_ex=&form_allow_patient_portal=&form_cmsportal_login=&form_imrn_reg_status=&form_imrn_effdate=&form_publicity_code=&form_publ_code_eff_date=&form_notify_indicator=&form_notify_incl_notifydata=&form_inductrum=&form_recognition=&form_on_name=&form_on_street=&

Task ID	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	State	Payloads
0	Original	200 OK		293 ms	443 bytes	146 bytes	Medium		
1	Fuzzed	200 OK		94 ms	441 bytes	95 bytes	Reflected		' exec master..x...
2	Fuzzed	200 OK		94 ms	441 bytes	89 bytes	Reflected		create user name...
3	Fuzzed	200 OK		79 ms	442 bytes	140 bytes	Reflected		create user name...
4	Fuzzed	200 OK		79 ms	441 bytes	71 bytes	Reflected		'; drop table tem...
5	Fuzzed	200 OK		97 ms	441 bytes	85 bytes	Reflected		exec sp_addlogi...
6	Fuzzed	200 OK		86 ms	441 bytes	93 bytes	Reflected		exec sp_addsvr...
7	Fuzzed	200 OK		82 ms	442 bytes	144 bytes	Reflected		insert into mysl...
8	Fuzzed	200 OK		81 ms	441 bytes	95 bytes	Reflected		grant connect to ...
9	Fuzzed	200 OK		72 ms	442 bytes	237 bytes	Reflected		insert into users...
10	Fuzzed	200 OK		274 ms	442 bytes	146 bytes	Reflected		a
11	Fuzzed	200 OK		141 ms	418 bytes	60 bytes	Reflected		' or 1=1 --
12	Fuzzed	200 OK		133 ms	441 bytes	78 bytes	Reflected		' union (select @...
13	Fuzzed	200 OK		81 ms	441 bytes	93 bytes	Reflected		' union (select N...
14	Fuzzed	200 OK		115 ms	441 bytes	99 bytes	Reflected		' union (select N...
15	Fuzzed	200 OK		96 ms	442 bytes	106 bytes	Reflected		' union (select N...

3. Ruleset Used: XSS

Location: Creating a new message

URL: method: POST, end-point:

http://localhost/openemr/interface/main/messages/messages.php?showall=no&sortby=&sortorder=&begin=&form_active=1

Results: True positive, for **form_note_type=<script>alert('xss')</script>**, a new message is saved and sent to a patient with a malicious script. The malicious script can be seen under the Type in the messages page.

Test Case id: 4.1.1-1

Description: Verify that the application enforces access control rules on a trusted service layer, especially if client-side access control is present and could be bypassed.

Repeatable Steps:

1. In VCL, Open OWASP ZAP and click on the “quick start” browser that is automatically configured to use the ZAP proxy. (a small firefox icon on top of the ZAP application).
2. Make sure that you install the Firefox browser.
3. Navigate to the openemr webpage in firefox. <http://localhost/openemr>
4. log in as admin with username: admin_openemr and password: admin_openemr
5. Create a new Patient. To do this hover over Patient/Client on top headers and click on patients. Under Patient Finder click on **Add New Patient**
6. Enter Name. Select Mr. from the dropdown. Enter abc in the first box and def in the last box (the middlebox can be left empty).
7. Select DOB as 03/04/2022 from the calendar.
8. Select Sex as Male from the dropdown.
9. Scroll down the page and click on Create New Patient.
10. On the top header click on **Messages**.
11. Under My messages click on Add new
12. Under Create New Message select Type as Chart Note.
13. Click on the Patient box and select **def,abc** from the Patient name dropdown.
14. Click on Add to list, and click on ok.
15. Under To: select **Administrator, Administrator**
16. Type a message as **hello** in the message field.
17. Click on send message.
18. Go back to the ZAP application and open the History tab at the bottom of the application.
19. Find the POST endpoint with URL:
http://localhost/openemr/interface/main/messages/messages.php?showall=no&sortby=&sortorder=&begin=&form_active=1
20. In the Request tab, select the form_note_type value.
21. Right-click on the form_note_type value and select fuzz.
22. Select Payloads from the Fuzzer window.
23. Click on Add.
24. Select File Fuzzers as Type.
25. Expand jbrofuzz and select XSS.
26. Click on Add and then OK to return back to the Fuzzer window.
27. Click Start Fuzzer to begin the fuzzing process.

28. Go through the results and find the payload with
`form_note_type=<script>alert('xss')</script>`
29. Go back to openemr and click on messages.
30. You can see a list of messages since fuzzing uses a set of payloads.
31. Find the message with type as `<script>alert('xss')</script>`

Expected Results: Since the new message form_note_type contains malicious or invalid characters, the new message should not be stored and sent.

CWE: 602, Client-Side Enforcement of Server-Side Security

Mitigation techniques to fix this issue:

1. On the client-side, since it's a dropdown, we have to make sure input validation is present on every field on the client-side and server-side.
2. Use deny list/allow list to validate the input on the server-side.
3. Can use output encoding libraries like Microsoft's Anti-XSS library.

Screenshot of ZAP:

Task ID	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	State	Payloads
90 Fuzzed	200 OK	200 OK		179 ms	316 bytes	81,655 bytes			<!--exec cmd=...-->
91 Fuzzed	200 OK	200 OK		165 ms	316 bytes	81,739 bytes			aim: &:<window...>
92 Fuzzed	200 OK	200 OK		141 ms	316 bytes	82,106 bytes			firefoxurl:test!%...>
93 Fuzzed	200 OK	200 OK		198 ms	316 bytes	82,181 bytes			navigatorurl:test!%...>
94 Fuzzed	200 OK	200 OK		189 ms	316 bytes	82,120 bytes			res:/c:\\program...>
95 Fuzzed	200 OK	200 OK		211 ms	316 bytes	81,916 bytes			http://aa><scri...>
96 Fuzzed	200 OK	200 OK		197 ms	316 bytes	81,966 bytes			http://aa><scri...>
97 Fuzzed	200 OK	200 OK		163 ms	316 bytes	82,099 bytes			http://aa><scri...>
98 Fuzzed	200 OK	200 OK		203 ms	316 bytes	82,099 bytes			<script>alert('x...>
99 Fuzzed	200 OK	200 OK		173 ms	316 bytes	82,099 bytes			<script>alert('x...>
100 Fuzzed	200 OK	200 OK		188 ms	316 bytes	82,348 bytes			</title><script>...>
101 Fuzzed	200 OK	200 OK		186 ms	316 bytes	82,219 bytes			> <script>alert(...>
102 Fuzzed	200 OK	200 OK		139 ms	316 bytes	82,135 bytes			> <script>alert(...>
103 Fuzzed	200 OK	200 OK		200 ms	316 bytes	82,166 bytes			</title><script>...>
104 Fuzzed	200 OK	200 OK		171 ms	316 bytes	82,179 bytes			<><script>alert(...>
105 Fuzzed	200 OK	200 OK		163 ms	316 bytes	82,179 bytes			<><script>alert(...>

4. Ruleset Used: Buffer Overflow

Location: Creating a new Allergy

URL: method: POST, end-point:

http://localhost/openemr/interface/patient_file/summary/add_edit_issue.php?issue=0&thistype=allergy

Results: True Positive, the series of characters 'a' should not be accepted in the input field form_reaction while creating a new allergy for a particular patient

Test Case id: 5.1.3-1

Description: Verify that all input (HTML form fields, REST requests, URL parameters, HTTP headers, cookies, batch files, RSS feeds, etc) is validated using positive validation (allow lists).

Repeatable Steps:

1. In VCL, Open OWASP ZAP and click on the “quick start” browser that is automatically configured to use the ZAP proxy. (a small firefox icon on top of the ZAP application).
2. Make sure that you install the Firefox browser.
3. Navigate to the openemr webpage in firefox. <http://localhost/openemr>
4. log in as admin with username: admin_openemr and password: admin_openemr
5. Create a new Patient. To do this hover over Patient/Client on top headers and click on patients. Under Patient Finder click on **Add New Patient**
6. Create a new patient. Select Mr. from pronouns drop down. Type **test** as the first name in the first box, leave out the second box, and type **test** as the last name in the third box
7. Select DOB as June 5th, 1975.
8. Select Sex as **Male**.
9. Scroll down and click on **Create New Patient**.
10. Close the success patient created dialogue box.
11. Click on patient finder and the patient we just created ,
(test, test)
12. On bottom right, Under Recurrent Appointments, click on **Edit Allergies**
13. Click on **Add** allergies, select **penicillin** as the title and select the begin date as **2022-03-08** and Reaction as **Nausea** click on **save** allergy
14. Go back to the ZAP application and open the History tab at the bottom of the application.
15. Find the POST endpoint with URL:
http://localhost/openemr/interface/patient_file/summary/add_edit_issue.php?issue=0&thistype=allergy
16. In the request tab, select form_reaction and right click and select fuzz
17. Select Payloads from the Fuzzer window.
18. Click on Add.

19. Select File Fuzzers as Type.
20. Expand jbrofuzz and select Buffer Overflow
21. Click on Add and then OK to return back to the Fuzzer window.
22. Click Start Fuzzer to begin the fuzzing process.
23. Go through the results and find the payload with series of characters 'a' of length 1500
24. Go back to openemr and click on allergies in patients tab
25. We can see a new entry in allergies tab, that has reaction as series of characters 'a'

Expected Results: Since the new allergy record in form_reaction contains a value other than the given options in the select element, It should not be created / saved .

CWE: 20, Improper Input Validation

Mitigation techniques to fix this issue:

1. On the client-side, since it's a dropdown, we have to make sure input validation is present on every field on the client-side, so that we do not submit form with invalid inputs.
2. We also need to limit the input length, this should be implemented both on client side and server side with API validation.

Task ID	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	State	Payloads
0	Original	200	OK	929 ms	339 bytes	1,336 bytes	Medium	Reflected	a
1	Fuzzed	200	OK	37 ms	410 bytes	37 bytes			aaa
2	Fuzzed	200	OK	38 ms	410 bytes	37 bytes			aaaaaaaaaa
3	Fuzzed	200	OK	35 ms	410 bytes	37 bytes			aaaaaaaaaaaaaa
4	Fuzzed	200	OK	36 ms	410 bytes	37 bytes			aaaaaaaaaaaaaaaaaa
5	Fuzzed	200	OK	45 ms	410 bytes	37 bytes			aaaaaaaaaaaaaaaaaa
6	Fuzzed	200	OK	33 ms	410 bytes	37 bytes			aaaaaaaaaaaaaaaaaa
7	Fuzzed	200	OK	28 ms	420 bytes	37 bytes			aaaaaaaaaaaaaaaaaa
8	Fuzzed	200	OK	33 ms	420 bytes	37 bytes			aaaaaaaaaaaaaaaaaa
9	Fuzzed	200	OK	29 ms	410 bytes	37 bytes			aaaaaaaaaaaaaaaaaa
10	Fuzzed	200	OK	37 ms	410 bytes	37 bytes			aaaaaaaaaaaaaaaaaa
11	Fuzzed	200	OK	38 ms	410 bytes	37 bytes			aaaaaaaaaaaaaaaaaa
12	Fuzzed	200	OK	40 ms	410 bytes	37 bytes			aaaaaaaaaaaaaaaaaa
13	Fuzzed	200	OK	37 ms	410 bytes	37 bytes			aaaaaaaaaaaaaaaaaa
14	Fuzzed	200	OK	48 ms	420 bytes	37 bytes			aaaaaaaaaaaaaaaaaa

5. Ruleset Used: HTTP, XSS and SQL Injection

Location: Configure Orders and Results in Procedure Results tab.

URL: method: POST, end-point:

http://localhost/openemr/interface/orders/types_edit.php?typeid=7&parent=0

Results: True positive, for **form_procedure_type=<script>alert('xss')</script>**, a new procedure tier is saved and reflected in the frontend of the “Configure Order and Results” tab. The malicious script removes the existing procedure tier and introduces a new unknown type as “**”.

Test Case id: 5.1.3-2

Description: Verify that all input (HTML form fields, REST requests, URL parameters, HTTP headers, cookies, batch files, RSS feeds, etc) is validated using positive validation (allow lists).

Repeatable Steps:

1. In VCL, Open OWASP ZAP and click on the “quick start” browser that is automatically configured to use the ZAP proxy. (a small firefox icon on top of the ZAP application).
2. Make sure that you install the Firefox browser.
3. Navigate to the openemr webpage in firefox. <http://localhost/openemr>
4. login as admin with username: admin_openemr and password: admin_openemr
5. On the top headers, hover over **Procedures** and click on **Batch Results**.
6. Click on the field corresponding to the “Procedure:”. You will be popped up with a form containing heading as **“Configure Orders and Results”**
7. Click on the “**+Add Top Level**” button and you will be popped up with a form.
8. In that form fill the values as “Procedure Tier: Group”, “Name: dummy name”, “Description: dummy” and “Sequence: dummy” and click Save.
9. In ZAP, there is a tab towards the left-hand side. Under Sites, expand <http://localhost>, openemr, interface, orders.
10. Under the orders directory, you can find a POST:types_edit.php(parent, typeid)(form_body_site, form_description, form_diagnosis_code, form_laterality, form_name, form_procedure_code, form_procedure_type, from_range, form_realted_code, from_route_admin, form_save, form_seq, form_specimen, form_standard_code, form_units)
11. The URL in request will be
https://localhost/openemr/interface/orders/types_edit.php?typeid=0&parent=0 .
12. Right click on the URL mentioned in the step 10, and select Break, and click on save.
13. Go back to the firefox browser and repeat steps 5, 6, 7, 8.
14. When you click on Save this time as mentioned in the step 8, you might be able to see a HTTP request dialogue box.
15. In the Request tab, select the form_procedure_type value.
16. Right-click on the form_note_type value and select fuzz.
17. Select Payloads from the Fuzzer window.
18. Click on Add.
19. Select File Fuzzers as Type.
20. Expand jbrofuzz and select XSS, HTTP and SQL Injection.

21. Click on Add and then OK to return back to the Fuzzer window.
22. Click Start Fuzzer to begin the fuzzing process.
23. Close all the dialog boxes and on top of the web page and logout and login to the application again, on top of the headers hover over **Procedures** and click on **Batch Results** to navigate again to the same page and click on the field corresponding to “Procedure.” and check for the “Category” field value by clicking on the Edit(pencil) button of the row containing your order.
24. Here in this case, when the fuzzing has happened with a different set of payloads, each payload tries to replace the existing procedure tier with a new one which cannot be properly rendered in the frontend, and gets reflected as “* *”.

Expected Results: Since the new order form_procedure_type contains malicious or invalid characters, the new message should not be stored and sent to the backend.

CWE: 20, Improper Input Validation

Mitigation techniques to fix this issue:

1. On the client-side, since it's a dropdown, there must be proper input sanitization in-place to prevent malicious input being entered in that form field.
2. And also there must be proper checks on the server side before writing onto the database if there is malicious input. This will prevent malicious content from being rendered at the frontend when queried from the backend.

The screenshot shows the OWASP ZAP interface. The browser session pane displays a navigation tree for 'openemr' with various URLs like 'GET:/', 'GET:openemr', 'GET:Documentation', 'GET:Interface', 'GET:login', 'GET:main', and 'GET:orders'. The current request is a POST to 'http://152.7.177.132/openemr/interface/orders/types_edit.php?typeid=7&parent=0' with the following payload:

```
POST http://152.7.177.132/openemr/interface/orders/types_edit.php?typeid=7&parent=0 HTTP/1.1
Host: 152.7.177.132
Connection: keep-alive
Content-Length: 274
Cache-Control: max-age=0
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="99", "Google Chrome";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
Origin: https://152.7.177.132
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.101 Safari/537.36
```

The payload includes a parameter `form_procedure_type=(uid*)|(|(uid=<form_name=dummy>&form_description=dummy&form_seq=0&form_procedure_code=&form_standard_code=&form_diagnosis_code=&form_body_site=&form_specimen=&form_route_admin=&form_laterality=&form_units=&form_range=&form_related_code=&form_save=Save`.

The fuzzer results table below shows 87 tasks, mostly reflected errors, with some alerts and payloads listed.

Task ID	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	State	Payloads
73	Fuzzed	200	OK	69 ms	316 bytes	8,831 bytes	Reflected		
74	Fuzzed	200	OK	91 ms	316 bytes	8,831 bytes	Reflected	!	
75	Fuzzed	200	OK	104 ms	316 bytes	8,831 bytes	Reflected	(
76	Fuzzed	200	OK	132 ms	316 bytes	8,831 bytes	Reflected)	
77	Fuzzed	200	OK	99 ms	316 bytes	8,831 bytes	Reflected	&	
78	Fuzzed	200	OK	101 ms	316 bytes	8,831 bytes	Reflected	!	
79	Fuzzed	200	OK	111 ms	316 bytes	8,831 bytes	Reflected		
80	Fuzzed	200	OK	140 ms	316 bytes	8,831 bytes	Reflected	*	
81	Fuzzed	200	OK	123 ms	316 bytes	8,831 bytes	Reflected	"((mail="))	
82	Fuzzed	200	OK	102 ms	316 bytes	8,831 bytes	Reflected	"((objectclass="))	
83	Fuzzed	200	OK	143 ms	316 bytes	8,831 bytes	Reflected	"(l)&'	
84	Fuzzed	200	OK	111 ms	316 bytes	8,831 bytes	Reflected	admin*	
85	Fuzzed	200	OK	120 ms	316 bytes	8,831 bytes	Reflected	admin") ((userpa...	
86	Fuzzed	200	OK	93 ms	316 bytes	8,831 bytes	Reflected	"(uid=") ((uid="	
87	Fuzzed	200	OK	133 ms	316 bytes	8,831 bytes	Reflected	<SCRIPT...	

Total time spent on Fuzzing:

1. We spent 13 hours 30 minutes on this activity.
 2. The ZAP scan took 30 minutes in total to run against all the test cases we planned.
 3. We spent almost 10 hours analyzing and working with ZAP output. Initially, everything that ZAP tries out gave 200 responses. So, it took time for us to figure out a way how to catch the exact results.
 4. To plan and run the black-box tests it took 3 hours since we used the same black-box tests reported for Client-side bypassing.
 5. So, we found 0.38 true positives per hour.
-

2. Vulnerable Dependencies

GitHub Checker:

Total: 51 Vulnerable dependencies

1. Regular Expression Denial of Service in postcss

CVE ID : CVE-2021-23382, CVE-2021-23368

Dependency: Transitive, Direct

Library: postcss

Safer Version: 8.2.13 or later

Severity: Moderate

Description: The package postcss versions before 7.0.36 or between 8.0.0 and 8.2.13 are vulnerable to Regular Expression Denial of Service (ReDoS) via getAnnotationURL() and loadAnnotation() in lib/previous-map.js... Transitive dependency of other libraries has the version 7.0.35 and The direct dependency of the application has the version 8.2.4.

2. Regular Expression Denial of Service in trim-newlines

CVE ID: CVE-2021-33623

Dependency: Direct, Transitive

Library: trim-newlines

Safer Version: 3.0.1 or later

Severity: High

Description: The trim-newlines package before 3.0.1 and 4.x before 4.0.1 for Node.js has an issue related to regular expression denial-of-service (ReDoS) for the .end() method. Openemr has a direct and transitive dependency with version V3.0.0

3. Exposure of Sensitive Information to an Unauthorized Actor in nanoid

CVE ID: CVE-2021-23566

Dependency: Direct and Transitive

Library: nanoid

Safer Version: 3.1.31 or later

Severity: Moderate

Description: The package nanoid before 3.1.31 are vulnerable to Information Exposure via the valueOf() function which allows to reproduce the last id generated. Here direct dependency is of the version 3.1.20 and there is a transitive dependency with postcss that required version to be 3.1.20 or later

4. Authentication Bypass in ADOdb/ADOdb

CVE ID: CVE-2021-3850

Dependency: Direct

Library: adodb/adodb-php

Safer Version: 5.20.21, 5.21.4 or later

Severity: Critical

Description: An attacker can inject values into a PostgreSQL connection string by providing a parameter surrounded by single quotes. Depending on how the library is used in the client software, this may allow an attacker to bypass the login process, gain access to the server's IP address, etc. Openemr is currently using version v5.20.18 directly and there is no transitive dependency here.

5. HTML comments vulnerability allowing to execute JavaScript code

CVE ID: CVE-2021-41165, CVE-2021-41164, CVE-2021-26272, CVE-2021-37695,

CVE-2021-32809, CVE-2021-32808, CVE-2021-33829

Dependency: Direct

Library: ckeditor4

Safer Version: 4.17.0 or later

Severity: High

Description: A potential vulnerability has been discovered in CKEditor 4 HTML processing core module. The vulnerability allowed to inject malformed comments HTML bypassing content sanitization, which could result in executing JavaScript code. It affects all users using the CKEditor 4 at version < 4.17.0. Openemr is currently using version v4.15.0 directly and there is no transitive dependency here.

7. Remote code execution in zendframework and laminas-http

CVE ID: CVE-2021-3007, CVE-2022-23598

Dependency: Direct and Transitive

Library: laminas/laminas-http

Safer Version: 2.14.2 or later

Severity: Critical

Description: Zend Framework 3.0.0 has a deserialization vulnerability that can lead to remote code execution if the content is controllable, related to the `__destruct` method of the `Zend\Http\Response\Stream` class in `Stream.php`. Openemr currently uses version v2.14.0 as direct dependency and version v2.7 as transitive dependency.

8. Arbitrary File Creation/Overwrite via insufficient symlink protection due to directory cache poisoning using symbolic links

CVE ID: CVE-2021-37712, CVE-2021-32610, CVE-2021-37701, CVE-2021-32804, CVE-2021-32803, CVE-2020-36193

Dependency: Direct and Transitive

Library: tar

Safer Version: 4.4.18 or later

Severity: high

Description: Current version of tar, the logic was insufficient when extracting tar files that contained two directories and a symlink with names containing unicode values that normalized to the same value. Openemr currently uses v2.2.2 and tar-pack currently has transitive dependency on tar which is using V2.2.1

9. Prototype Pollution

CVE ID: CVE-2021-23413

Dependency: Direct and Transitive

Library: jszip

Safer Version: 3.7.0 or later

Severity: Moderate

Description: This affects the package jszip before 3.7.0. Crafting a new zip file with filenames set to Object prototype values (e.g proto, toString, etc) results in a returned object with a modified prototype instance. Openemr uses has direct dependency with V3.5.0 and transitive with v3.2.0

10. Command Injection in lodash

CVE ID: CVE-2021-23337, CVE-2020-28500

Dependency: Direct and Transitive

Library: lodash

Safer Version: 4.17.21 or later

Severity: high

Description: lodash versions prior to 4.17.21 are vulnerable to Command Injection via the template function. Currently openemr using V4.17.20 as direct dependency but there are other libraries that are using V4.17.14, V4.17.15, V4.17.19

OWASP Dependency Check:

Total: 29 Vulnerable Dependencies

1. Inefficient Regular Expression Complexity in ansi-regex

CVE ID : CVE-2021-3807

Dependency: Transitive

Library: ansi-regex

Safer Version: 6.0.1 or later

Severity: High

Description: ansi-regex is vulnerable to Inefficient Regular Expression Complexity

2. Regular Expression Denial of Service in braces

CVE ID : CVE-2018-1109

Dependency: Transitive

Library: braces

Safer Version: 2.3.1 or later

Severity: Medium

Description: A vulnerability was found in Braces versions prior to 2.3.1. Affected versions of this package are vulnerable to Regular Expression Denial of Service (ReDoS) attacks.

3. Regular Expression Denial of Service in browserslist

CVE ID : CVE-2021-23364

Dependency: Transitive

Library: browserslist

Safer Version: 4.16.5 or later

Severity: Medium

Description: The package browserslist from 4.0.0 and before 4.16.5 are vulnerable to Regular Expression Denial of Service (ReDoS) during parsing of queries.

4. Improperly Controlled Modification of Dynamically-Determined Object Attributes in chart.js

CVE ID : CVE-2020-7746

Dependency: Transitive

Library: chart.js

Safer Version: 2.9.47 or later

Severity: High

Description: The options parameter is not properly sanitized when it is processed. When the options are processed, the existing options (or the defaults options) are deeply merged with provided options. However, during this operation, the keys of the object being set are not checked, leading to a prototype pollution.

5. Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') in CKEditor4

CVE ID : CVE-2021-26272, CVE-2021-32808, CVE-2021-32809, CVE-2021-33829, CVE-2021-37695, CVE-2021-41164, CVE-2021-41165

Dependency: Transitive

Library: CKEditor4

Safer Version: 4.17.0 or later

Severity: Medium

Description: In the affected version a vulnerability has been discovered in the core HTML processing module and may affect all plugins used by CKEditor 4. The vulnerability allowed the injection of malformed HTML comments bypassing content sanitization, which could result in executing JavaScript code.

6. Prototype Pollution in copy-props

CVE ID : CVE-2020-28503

Dependency: Transitive

Library: copy-props

Safer Version: 2.0.5 or later

Severity: Critical

Description: The package copy-props before 2.0.5 are vulnerable to Prototype Pollution via the main functionality.

7. Cross Site Scripting & Prototype Pollution in datatables.net

CVE ID : CVE-2020-28458, CVE-2021-23445

Dependency: Transitive

Library: datatables.net

Safer Version: 1.11.3 or later for CVE-2021-23445, None for CVE-2020-28458

Severity: High

Description: CVE-2021-23445 affects the package datatables.net before 1.11.3. If an array is passed to the HTML escape entities function it would not have its contents escaped.

CVE-2020-28458 affects all versions of package datatables.net. These are vulnerable to Prototype Pollution due to an incomplete fix.

8. Uncontrolled Resource Consumption in glob-parent

CVE ID : CVE-2020-28469

Dependency: Transitive

Library: glob-parent

Safer Version: 5.1.2 or later

Severity: High

Description: The enclosure regex used to check for strings ending in enclosure containing path separator.

9. Uncontrolled Resource Consumption in hosted-git-info

CVE ID : CVE-2021-23362

Dependency: Transitive

Library: hosted-git-info

Safer Version: 3.0.8 or later

Severity: Medium

Description: The npm package `hosted-git-info` before 3.0.8 are vulnerable to Regular Expression Denial of Service (ReDoS) via regular expression shortcutMatch in the fromUrl function in index.js. The affected regular expression exhibits polynomial worst-case time complexity

10. Cross Site Scripting in jquery-1.10.2.min.js

CVE ID : CVE-2015-9251, CVE-2019-11358, CVE-2020-11022, CVE-2020-11023

Dependency: Direct

Library: jquery-1.10.2.min.js

Safer Version: 3.5.0 or later

Severity: Medium

Description: Passing HTML containing <option> elements from untrusted sources may execute untrusted code even after sanitizing it.

Comparison Report:

Dependency	OWASP CVE Count	GitHub CVE Count
adodb/adodb-php	0	1
ansi-regex	1	1
braces	3	2
browserslist	2	1
chart.js	1	1
ckeditor4	7	7
copy-props	1	1
datatables.net	2	2
glob-parent	2	1
hosted-git-info	3	1
jquery	45	1
jspdf	3	1
jszip	2	1
laminas/laminas-form	0	1
laminas/laminas-http	0	1
lodash	5	2
nanoid	1	1
path-parse	2	1
pear/archive_tar	0	2
phpmailer/phpmailer	0	3
phponline/phpspreadsheet	0	1
phpseclib/phpseclib	0	1
postcss	10	4
smarty/smarty	0	4
tar	8	5
trim-newlines	1	1
twig/twig	0	1
underscore	2	1
yargs-parser	0	1

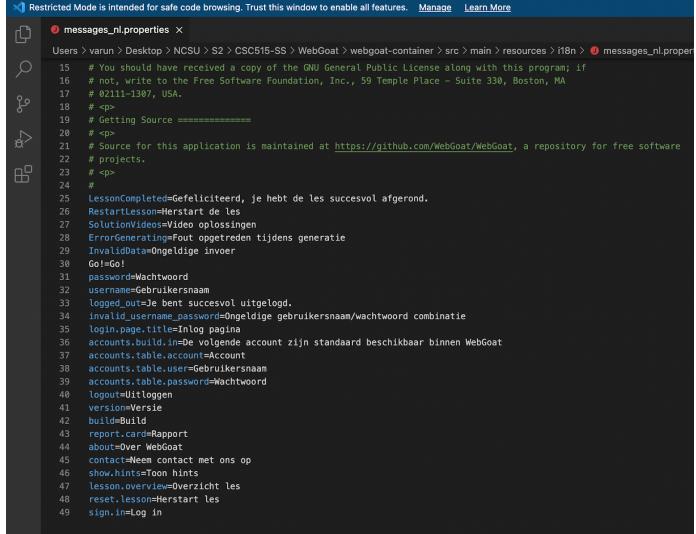
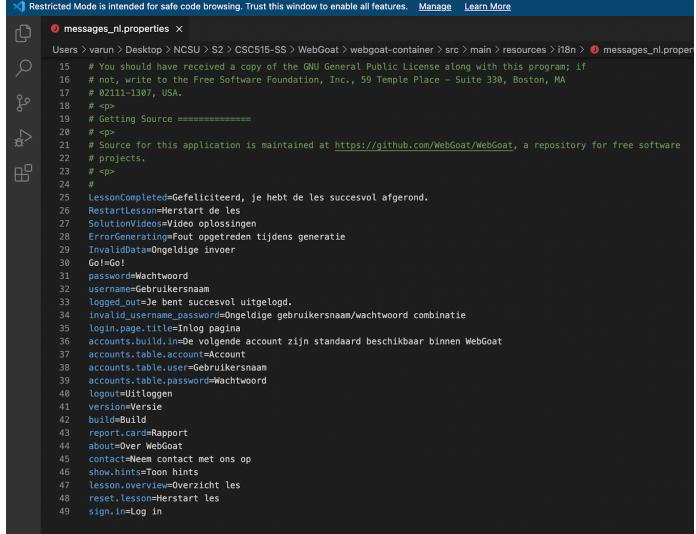
Explanation of Differing Results:

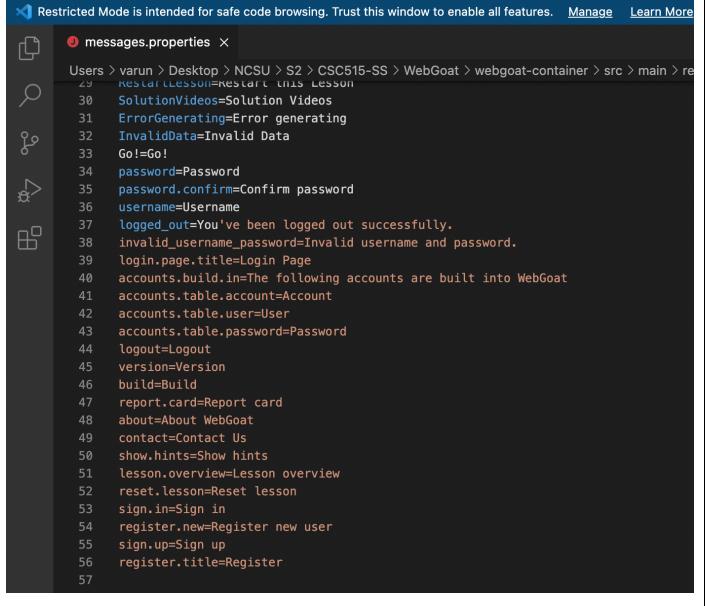
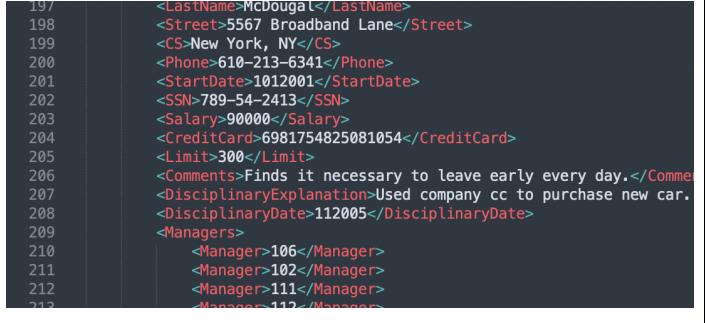
The files / directories each tool parses are different. Hence there is a possibility of having totally different results. For Ex: Github Security check only is going through the all metadata files like package.json to find the dependency issues. But OWASP is also parsing through the public folder to find other dependencies that are being imported into the given module.

Category	Github's Checker	OWASP Dependency Checker
Strength	This tool checks all the files like package-lock.json, composer.lock so the chances of finding the vulnerable dependencies are more	This tool actually tells whether the vulnerable dependency is transitive / direct
Weakness	This tool did not report any Vulnerabilities that are in "js" files that are located in public directory, this tool is essentially going through metadata json files like package-lock.json for any javascript dependencies and composer.lock.json for php related dependencies	This tool could only find vulnerable dependencies in package-lock.json and the actual js files present in the public directory like jquery and jquery-ui to find all the legacy versions in the given module
Usability	From a usability standpoint this is very easy to run on each commit / pull request because it is integrated into Github. By simply enabling the dependabot alerts, we can see the dependency vulnerabilities real time.	Since this is a third party tool and no inbuilt integration with the version control system, there is an extra effort needed to run on every incremental change in the repository.

3. Secret Detection

Webgoat - Whispers

Se q No	Secrets Detected	Screen Shot
1	<p>Secret type: password</p> <p>Exposed Secret: Wachtwoord</p> <p>Line No: 31</p>	 <pre> 15 # You should have received a copy of the GNU General Public License along with this program; if 16 # not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 17 # 02111-1307, USA. 18 # <P> 19 # Getting Source ===== 20 # <P> 21 # Source for this application is maintained at https://github.com/WebGoat/WebGoat, a repository for free software 22 # projects. 23 # <P> 24 # 25 LessonCompleted=Gefeliciteerd, je hebt de les succesvol afgerond. 26 RestartLesson=Herstart de les 27 SolutionVideosVideo oplossingen 28 ErrorGenerating=Fout opgetreden tijdens generatie 29 InvalidData=Ongeldige invoer 30 Go!Go! 31 password=Wachtwoord 32 username=Gebruikersnaam 33 logged_out=Je bent succesvol uitgelogd. 34 invalid_username_password=Ongevalide gebruikersnaam/wachtwoord combinatie 35 login.page.title=Inlog pagina 36 accounts.build.in=De volgende account zijn standaard beschikbaar binnen WebGoat 37 accounts.table.account=Account 38 accounts.table.user=Gebruikersnaam 39 accounts.table.password=Wachtwoord 40 logout=Uitloggen 41 version=Versie 42 build=Build 43 report.card=Rapport 44 about=Over WebGoat 45 contact=Neem contact met ons op 46 show.hints=Toon hints 47 lesson.overview=Overzicht les 48 reset.lesson=Herstart les 49 sign.in=Log in </pre>
2	<p>Secret type: accounts.table.password</p> <p>Exposed Secret: Wachtwoord</p> <p>Line No: 39</p>	 <pre> 15 # You should have received a copy of the GNU General Public License along with this program; if 16 # not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 17 # 02111-1307, USA. 18 # <P> 19 # Getting Source ===== 20 # <P> 21 # Source for this application is maintained at https://github.com/WebGoat/WebGoat, a repository for free software 22 # projects. 23 # <P> 24 # 25 LessonCompleted=Gefeliciteerd, je hebt de les succesvol afgerond. 26 RestartLesson=Herstart de les 27 SolutionVideosVideo oplossingen 28 ErrorGenerating=Fout opgetreden tijdens generatie 29 InvalidData=Ongeldige invoer 30 Go!Go! 31 password=Wachtwoord 32 username=Gebruikersnaam 33 logged_out=Je bent succesvol uitgelogd. 34 invalid_username_password=Ongevalide gebruikersnaam/wachtwoord combinatie 35 login.page.title=Inlog pagina 36 accounts.build.in=De volgende account zijn standaard beschikbaar binnen WebGoat 37 accounts.table.account=Account 38 accounts.table.user=Gebruikersnaam 39 accounts.table.password=Wachtwoord 40 logout=Uitloggen 41 version=Versie 42 build=Build 43 report.card=Rapport 44 about=Over WebGoat 45 contact=Neem contact met ons op 46 show.hints=Toon hints 47 lesson.overview=Overzicht les 48 reset.lesson=Herstart les 49 sign.in=Log in </pre>

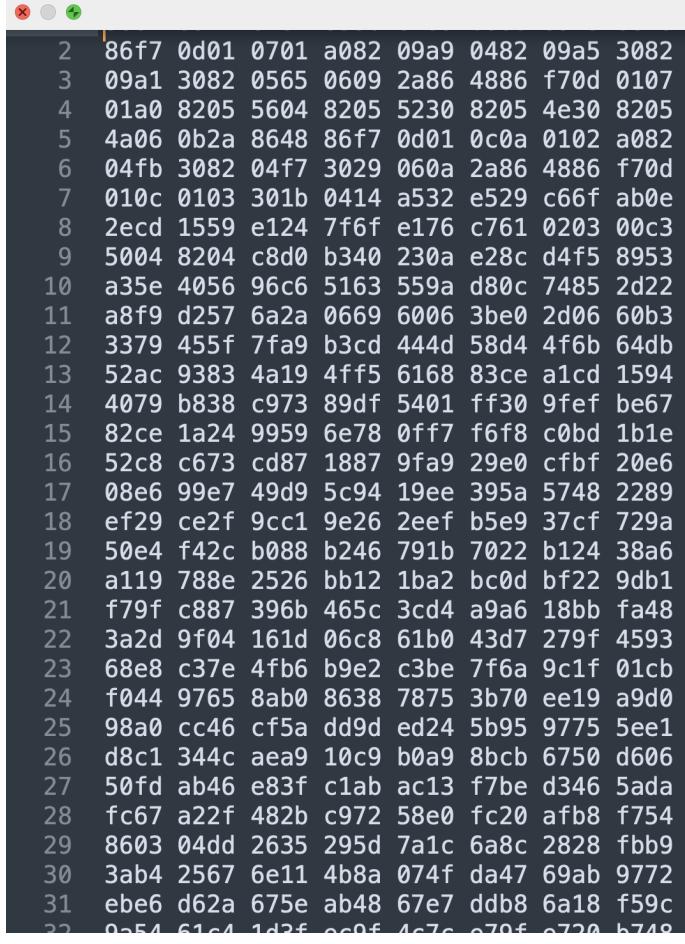
3	<p>Secret type: accounts.table.password</p> <p>Exposed Secret: Password</p> <p>Line No: 43</p>	 <pre> 29 restartLesson=restart this lesson 30 SolutionVideos=Solutions Videos 31 ErrorGenerating=Error generating 32 InvalidData=Invalid Data 33 Go!=Go! 34 password=Password 35 password.confirm=Confirm password 36 username=Username 37 logged_out=You've been logged out successfully. 38 invalid_username_password=Invalid username and password. 39 login.page.title=Login Page 40 accounts.build.in=The following accounts are built into WebGoat 41 accounts.table.account=Account 42 accounts.table.user=User 43 accounts.table.password=Password 44 logout=Logout 45 version=Version 46 build=Build 47 report.card=Report card 48 about=About WebGoat 49 contact>Contact Us 50 show.hints>Show hints 51 lesson.overview=Lesson overview 52 reset.lesson=Reset lesson 53 sign.in=Sign in 54 register.new=Register new user 55 sign.up=Sign up 56 register.title=Register 57 </pre>
4	<p>Secret type: show-password</p> <p>Exposed Secret: Toon wachtwoord</p> <p>Line No: 4</p>	 <pre> 1 secure-passwords.title=Veilige wachtwoorden 2 securepassword-success=Het is gelukt. Het wachtwoord is sterk genoeg. 3 securepassword-failed=Helaas, probeer een sterker wachtwoord! 4 show-password=Toon wachtwoord </pre>
5	<p>Secret type: CreditCard</p> <p>Exposed secret: 6981754825081054</p> <p>Line No: 204</p>	 <pre> 197 <LastName>McDougal</LastName> 198 <Street>5567 Broadband Lane</Street> 199 <City>New York, NY</City> 200 <Phone>610-213-6341</Phone> 201 <StartDate>1012001</StartDate> 202 <SSN>789-54-2413</SSN> 203 <Salary>90000</Salary> 204 <CreditCard>6981754825081054</CreditCard> 205 <Limit>300</Limit> 206 <Comments>Finds it necessary to leave early every day.</Comments> 207 <DisciplinaryExplanation>Used company car to purchase new car. 208 <DisciplinaryDate>112005</DisciplinaryDate> 209 <Managers> 210 <Manager>106</Manager> 211 <Manager>102</Manager> 212 <Manager>111</Manager> 213 <Manager>112</Manager> </pre>

Webgoat - SpectralOps

Se q No	Secrets Detected	Screen Shot
1	<p>Secret type: Token</p> <p>Exposed Secret: eyJraWQiOj3ZWJhb2F0X2tI eSlSmFsZyl6IkhtTNTEyln0.e yJhdWQiOj3ZWJhb2F0Lm9 yZylsImVtYWlsljoiamVycnlAd 2ViZ29hdC5jb20iLCJ1c2Vyb mFtZSI6IkplcnJ5In0.xBc5FF waOcuxjdr_VJ16n8Jb7vScua ZulNTI66F2MWF1aBe47QsU osvbjWGORNcMPiPNwnMu1 Yb0WZNrp2ZXA</p> <p>Line No: 27</p>	<pre>① JWTFinalEndpointTest.java • src > test > java > org > owasp > webgoat > plugin > ② JWTFinalEndpointTest.java > ↗ JWTFinalEndpointTest > TO 22 23 @RunWith(SpringJUnit4ClassRunner.class) 24 public class JWTFinalEndpointTest extends LessonTest { 25 26 private static final String TOKEN_JERRY = "eyJ0eXAiOiJKV1QiLCJraWQiOiJ3ZWJhb2F0X" 27 + "2tIesIiMsZyIGikhTMjU2In0.eyJpc3MiOiJXZWJhb2F0IFRva2VuIEJ1awKxZX" + 28 "iIiLCJpYXQiOjE1MjQyMTASMDQsImV4cC16MTYxODkWNtMwNCwiYXVkJoid2ViZ29hd" + 29 "iCvcmc1LCJwdIi0IjgZJyeB3ZWJhb2F0LnNb5IsInVzXJuyW1IjoiSmVycnkI" + 30 "iLCJFbWFpbCI6ImplcnJ50HdlymdvYXOUY29tIiwiUm9sZSI6WyJDYXQiXX0.CgZ27Dz" + 31 "iBVWBgzc0n6iz0U638uUCi6Uh10JKYzoEZGE8";</pre>
2	<p>Secret type: Token</p> <p>Exposed Secret: eyJhbGciOiJIUzUxMiJ9.eyJp YXQiOjE1MjYxMzE0MTEslm V4cCI6MTUyNjlxNzgxMSwiY WRtaW4iOiJmYWxzZSlzInVz ZXliOjUb20ifQ.DCoaq9zQky DH25EcVWKcdbyVfUL4c9D4 jRvsqOqvi9iAd4QuqmKccfb U8FNzeBNF9tLeFXHZLU4y Rkq-bjm7Q</p> <p>Line No: 76</p>	<pre>① JWTReloadEndpointTest.java • Users > vishnuchalla > ② JWTReloadEndpointTest.java > ↗ JWTReloadEndpointTest > solveAssignment() 12 13 String accessToken = tokens.get("access_token"); 14 String refreshToken = tokens.get("refresh_token"); 15 16 //Now create a new refresh token for Tom based on Toms old access token and send 17 String accessTokenTom = "eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE1MjYxMzE0MTEslmV4cCI6 18 iMTUyNjlxNzgxMSwiYRtaW4iOiJmYWxzZSlzInVzXJ1i0IjUb20ifQ.DCoaq9zQkyDH25EcV 19 cfbU8FNzeBNF9tLeFXHZLU4yRkq-bjm7Q";</pre>
3	<p>Secret type: Token</p> <p>Exposed Secret: eyJhbGciOiJIUzI1NilsInR5cC I6IkpxVCJ9.eyJzdWliOilxMj M0NTY3ODkwliwbmFtZSI6I kpvaG4gRG9IliwiaWF0ljoxN TE2MjM5MDlyfQ.NFvYpuwb</p>	<pre>① JWT_libraries.adoc • Users > vishnuchalla > ② JWT_libraries.adoc > [source] 22 "iat": 1516239022 23 } 24 ----- 25 26 [source] 27 ----- 28 var token = "eyJhbGciOiJIUzI1NilsInR5cCI6IkpxVCJ9.eyJzdWliOilxMjM0NTY3ODkwliwbmFtZSI6 29 iIkpvaG4gRG9IliwiaWF0ljoxNTE2MjM5MDlyfQ.NFvYpuwbF6YlbPyaNAGEPw9wbhiQSovvSrD89B8K7Ng";</pre>

	F6YWbPyaNAGEPw9wbhiQ SovvSrD89B8K7Ng Line No: 28	
4	Secret type: Token Exposed Secret: eyJhbGciOiJIUzUxMiJ9.eyJp YXQiOjE2MDgxMjg1NjYsImF kbWluljoiZmFsc2UiLCJ1c2V ljoVG9tIn0.rTSX6PSXqUoG UvQQDBiqX0re2BSt7s2-X6F Pf34Qly9SMpqIUSP8jykedJbj OBНИM3_СTjgk1SvUv48Pz8z lzA Line No: 35	<pre>* JWT_signing_solution.adoc × Users > vishnuchalla > * JWT_signing_solution.adoc > *## eyJhbGciOiJub25ln0.ew0KICAiYWRtaW4iDoglnRydWUiL 31 [source] 32 ----- 33 GET http://localhost:8080/WebGoat/JWT/votings/login?user=Tom HTTP/1.1 34 35 access_token=eyJhbGciOiJIUzUxMiJ9, eyJpYXQiOjE2MDgxMjg1NjYsImFkbWluljoiZmFsc2UiLCJ1c2VljoVG9tIn0. rTSX6PSXqUoGUQDdbiqX0re2BSt7s2-X6FPf340ly9SMpqIUSP8jykedJbjOBНИM3_СTjgk1SvUv48Pz8zIzA 36 ----- 37</pre>
5	Secret type: action Exposed Secret: /WebGoat/JWT/final/delete?t oken=eyJ0eXAiOiJKV1QiLCJ raWQiOj3ZWJnb2F0X2tleSI slmFsZyl6IkhtTMjU2In0.eyJpc 3MiOiJXZWJHb2F0IFRva2Vu IEJ1aWxkZXliLCJpYXQiOjE1 MjQyMTA5MDQsImV4cCI6M TYxODkwNTMwNCwiYXVklj oid2ViZ29hdC5vcmc1LCJzdW liOijqZXJyeUB3ZWJnb2F0L mNvbSIsInVzZXJuYW1ljoIS mVycnkiLCJFbWFpbCI6Impl cnJ5QHd1YmdvYXQuY29tliwi Um9sZSI6WyJDYXQiXX0.Cg Z27DzgVW8gzc0n6izOU638 uUCi6UhiOJKYzoEZGE8 Line No: 310	<pre>* Untitled VT.html • Users > vishnuchalla > * JWT.html > * html > * body > * div.lesson-page-wrapper > * div.attack-container > * form 305 <script th:src="@{/lesson_js/bootstrap.min.js}" language="JavaScript"></script> 306 <div class="attack-container"> 307 <div class="assignment-success"><i class="fa fa-2 fa-check hidden" aria-hidden="true"></i> 308 <form class="attack-form" accept-charset="UNKNOWN" 309 method="POST" 310 action="/WebGoat/JWT/final/delete?token=eyJ0eXAiOiJKV1QiLCJraWQiOjJ3Z" 311 "eyJhbGciOiJIUzU2tlesIiMfszyI6IkhtTMjU2In0.eyJpc3M10iJXZWJHb2F0IFRva2VuIEJ1a" 312 "WxkZXliLCJpYXQiOjE1MjQyMTA5MDQsImV4cCI6MTCi6MTUyNjIxNzgxMSwiYWRtaW4iOijmYw 313 "29hdC5vcmc1LCJzdWl10iJaqZXJyeUB3ZWJnb2F0lnNvb5IsInVzZXJuYW1lIjoisMvyc" 314 "nk1LCJFbWFpbCI6ImplcnJ5QHd1YmdvYXQuY29tIiwUUm9sZSI6WyJDYXQiXX0.CgZ27" 315 "DzgVW8gzc0n6izOU638uUCi6UhiOJKYzoEZGE8"</pre>
6	Secret type: token Exposed Secret: eyJhbGciOiJIUzUxMiJ9.eyJp YXQiOjE1MjYxMzE0MTEsImF kbWluljoiZmFsc2UiLCJ1c2V ljoVG9tIn0.rTSX6PSXqUoG UvQQDBiqX0re2BSt7s2-X6F Pf34Qly9SMpqIUSP8jykedJbj OBНИM3_СTjgk1SvUv48Pz8z lzA	<pre>* logs.txt × Users > vishnuchalla > * logs.txt 1 2 'token=eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE1MjYxMzE0MTEsImV4cCI6MTUyNjIxNzgxMSwiYWRtaW4iOijmYw 3 neckout HTTP/1.1" 200 12783 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:60.0) Gecko/20 4 TP/1.1" 200 212 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:60.0) Gecko/20100101 Fire 5 HTTP/1.1" 404 249 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:60.0) Gecko/20100101 Fi 6 neckout HTTP/1.1" 404 215 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.3</pre>

	<p>V4cCI6MTUyNjIxNzgxMSwiY WRtaW4iOiJmYWxzZSIsInVz ZXIiOiJUb20ifQ.DCoaq9zQky DH25EcVWKcdbyVfUL4c9D4 jRvsqOqvi9iAd4QuqmKcchfb U8FNzeBNF9tLeFXHZLU4y Rkq-bjm7Q</p> <p>Line No: 2</p>	
7	<p>Secret type: token</p> <p>Exposed Secret: eyJhbGciOiJIUzI1NilsInR5cC I6IkpxVCJ9eyJ0ZXN0IjoidG VzdCJ9.axNp9BkswwK_YRF 2URJ5P1UejQNYZbK4qYcM nkusg6I</p> <p>Line No: 28</p>	 <pre> 20 } 21 22 @Test 23 void encodeCorrectTokenWithSignature() { 24 var headers = Map.of("alg", "HS256", "typ", "JWT"); 25 var payload = Map.of("test", "test"); 26 var token = JWTToken.encode(toString(headers), toString(payload), "webgoat"); 27 28 assertThat(token.getEncoded()).isEqualTo("eyJhbGciOiJIUzI1NiIsInR5cCI6IkpxVCJ9" 29 +"eyJ0ZXN0IjoidGVzdCJ9.axNp9BkswwK_YRF2URJ5P1UejQNYZbK4qYcMnkusg6I"); 30 } 31 </pre>

8	<p>Secret type: keystroke</p> <p>Exposed Secret: encrypted content attached in the screenshot.</p> <p>Line No: Entire file</p>	 <pre> 2 86f7 0d01 0701 a082 09a9 0482 09a5 3082 3 09a1 3082 0565 0609 2a86 4886 f70d 0107 4 01a0 8205 5604 8205 5230 8205 4e30 8205 5 4a06 0b2a 8648 86f7 0d01 0c0a 0102 a082 6 04fb 3082 04f7 3029 060a 2a86 4886 f70d 7 010c 0103 301b 0414 a532 e529 c66f ab0e 8 2ecd 1559 e124 7f6f e176 c761 0203 00c3 9 5004 8204 c8d0 b340 230a e28c d4f5 8953 10 a35e 4056 96c6 5163 559a d80c 7485 2d22 11 a8f9 d257 6a2a 0669 6006 3be0 2d06 60b3 12 3379 455f 7fa9 b3cd 444d 58d4 4f6b 64db 13 52ac 9383 4a19 4ff5 6168 83ce a1cd 1594 14 4079 b838 c973 89df 5401 ff30 9fef be67 15 82ce 1a24 9959 6e78 0ff7 f6f8 c0bd 1b1e 16 52c8 c673 cd87 1887 9fa9 29e0 cfbf 20e6 17 08e6 99e7 49d9 5c94 19ee 395a 5748 2289 18 ef29 ce2f 9cc1 9e26 2eef b5e9 37cf 729a 19 50e4 f42c b088 b246 791b 7022 b124 38a6 20 a119 788e 2526 bb12 1ba2 bc0d bf22 9db1 21 f79f c887 396b 465c 3cd4 a9a6 18bb fa48 22 3a2d 9f04 161d 06c8 61b0 43d7 279f 4593 23 68e8 c37e 4fb6 b9e2 c3be 7f6a 9c1f 01cb 24 f044 9765 8ab0 8638 7875 3b70 ee19 a9d0 25 98a0 cc46 cf5a dd9d ed24 5b95 9775 5ee1 26 d8c1 344c aea9 10c9 b0a9 8bcb 6750 d606 27 50fd ab46 e83f c1ab ac13 f7be d346 5ada 28 fc67 a22f 482b c972 58e0 fc20 afb8 f754 29 8603 04dd 2635 295d 7a1c 6a8c 2828 fbb9 30 3ab4 2567 6e11 4b8a 074f da47 69ab 9772 31 ebe6 d62a 675e ab48 67e7 ddb8 6a18 f59c 32 0a54 61c4 1d3f ec9f 4c7c e70f e720 b748 </pre>
---	--	---

Common secrets:

We didn't find any common secrets among the three tools that we used. (Whispers, SpectralOps and GitLeaks)

Why do tools differ?

1. The keyword in search is different in different tools. For example, in the case of Whispers, it was able to find a credit card number but the other 2 tools didn't report that secret. This tells us that whispers have CreditCard as one of the keyword searches while searching for secrets but the other 2 didn't have that.
2. Spectral Ops use AI and Machine Learning to detect the secrets. So, it has to learn what is a secret before actually determining it. Hence the secrets caught by other tools were not reported by Spectral Ops.

3. And when it comes to Whispers and GitLeaks, Whispers is implemented in python and GitLeaks in Go language. Due to cross platform implementations and differences in parse techniques used by the corresponding libraries, the conclusions of interpreting a secret may differ. This is what has resulted in zero match in our case.
4. And also the GitLeaks is undergoing active development and is adapting to the new secret detection techniques wherein Whispers last commit was in Oct 2021 which might be outdated according to today's secret detection technique. This might also be a reason for no matches between the tools.

Extra Credit

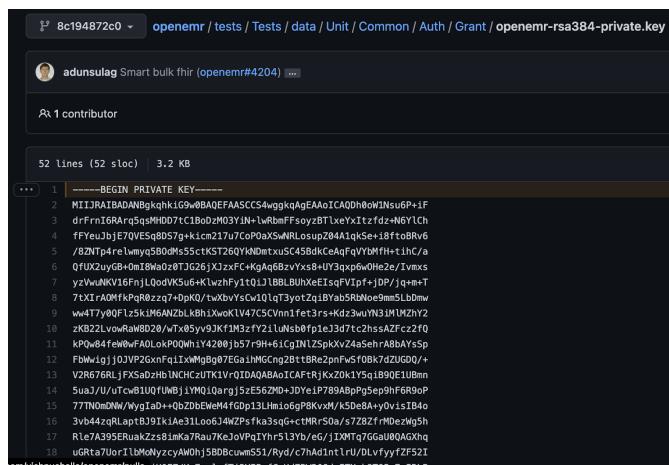
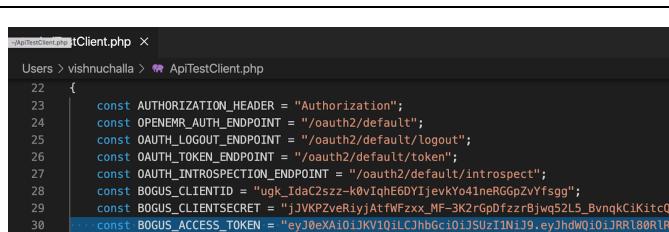
Openemr - Whispers

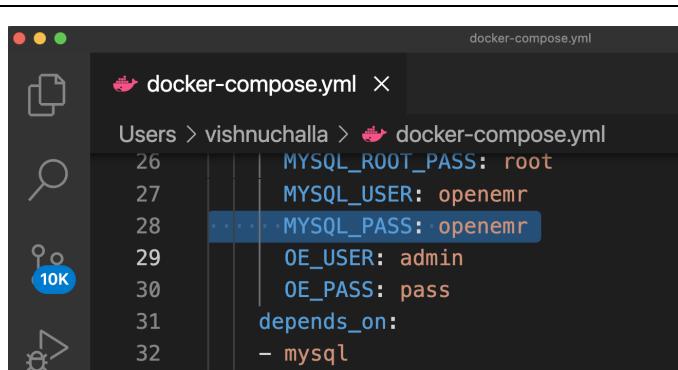
Seq No	Secrets	Screen Shots
1	<p>Secret type: MYSQL_ROOT_PASS WORD</p> <p>Exposed Secret: root</p> <p>Line No: 16</p>	<pre> 8 ports: 9 - 8320:3306 10 volumes: 11 - .../development-insane/sql-ssl-certs-keys/easy/ca.pem 12 - .../development-insane/sql-ssl-certs-keys/easy/server.pem 13 - .../development-insane/sql-ssl-certs-keys/easy/ssl.pem 14 - databasevolume:/var/lib/mysql 15 environment: 16 MYSQL_ROOT_PASSWORD: root 17 openemr: 18 restart: always 19 image: openemr/openemr:flex 20 ports: 21 - 8300:80 22 - 9300:443 23 volumes: </pre>
2	<p>Secret type: MYSQL_ROOT_PASS</p> <p>Exposed Secret: root</p> <p>Line No: 37</p>	<pre> 28 - nodemodules:/var/www/localhost/htdocs/openemr/node_modules 29 - vendordir:/var/www/localhost/htdocs/openemr/vendor 30 - ccdamodule:/var/www/localhost/htdocs/openemr/ccd/modules 31 - logvolume:/var/log 32 environment: 33 DEBUG_COLORS: "true" 34 TERM: xterm-256color 35 COLORTERM: truecolor 36 MYSQL_HOST: mysql 37 MYSQL_ROOT_PASS: root 38 MYSQL_USER: openemr 39 MYSQL_PASS: openemr 40 OE_USER: admin 41 OE_PASS: pass </pre>

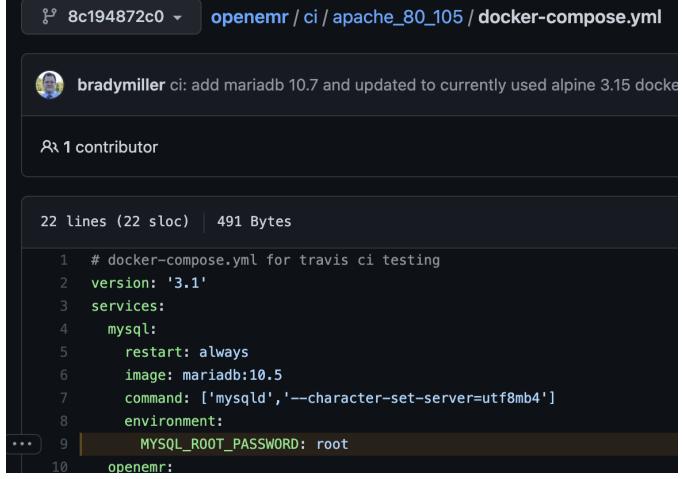
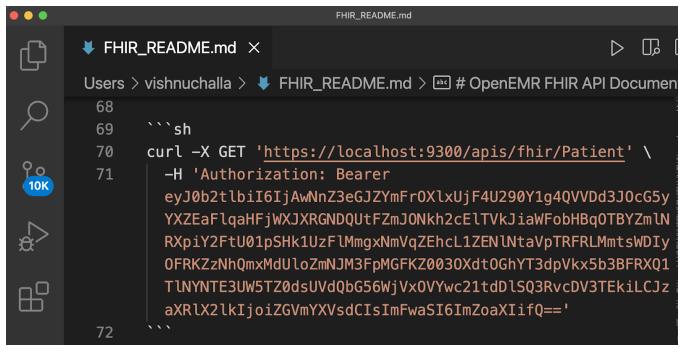
3	<p>Secret type: MYSQL_PASS</p> <p>Exposed Secret: openemr</p> <p>Line No: 39</p>	<pre> 28 - nodeModules:/var/www/localhost/htdocs/openemr/node_modules 29 - vendorDir:/var/www/localhost/htdocs/openemr/vendor 30 - ccdamodule:/var/www/localhost/htdocs/openemr/ccda 31 - logvolume:/var/log 32 environment: 33 DEBUG_COLORS: "true" 34 TERM: xterm-256color 35 COLORTERM: truecolor 36 MYSQL_HOST: mysql 37 MYSQL_ROOT_PASS: root 38 MYSQL_USER: openemr 39 MYSQL_PASS: openemr 40 OE_USER: admin 41 OE_PASS: pass 42 EASY_DEV_MODE: "yes" 43 EASY_DEV_MODE_NEW: "yes" </pre>
4	<p>Secret type: OE_PASS</p> <p>Exposed Secret: pass</p> <p>Line No: 41</p>	<pre> 32 environment: 33 DEBUG_COLORS: "true" 34 TERM: xterm-256color 35 COLORTERM: truecolor 36 MYSQL_HOST: mysql 37 MYSQL_ROOT_PASS: root 38 MYSQL_USER: openemr 39 MYSQL_PASS: openemr 40 OE_USER: admin 41 OE_PASS: pass 42 EASY_DEV_MODE: "yes" 43 EASY_DEV_MODE_NEW: "yes" 44 # e2e requires chromium, which alpine isn't giving us arm 45 # TODO, need to add more granular setting for 46 DEVELOPER_TOOLS: "no" 47 XDEBUG_IDE_KEY: PHPSTORM 48 XDEBUG_PROFILER_ON: 1 49 GITHUB_COMPOSER_TOKEN: c313de1ed5a00eb6ff9309559ec9ad01fcc553f0 50 OPENEMR_DOCKER_ENV_TAG: easy-dev-docker 51 OPENEMR_SETTING_site_addr_oath: 'https://localhost:9300' 52 OPENEMR_SETTING_password_grant: 3 53 OPENEMR_SETTING_rest_api: 1 </pre>
5	<p>Secret type: GITHUB_COMPOSER_TOKEN</p> <p>Exposed Secret: c313de1ed5a00eb6ff93 09559ec9ad01fcc553f0</p> <p>Line No: 49</p>	<pre> 40 DE_USER: admin 41 OE_PASS: pass 42 EASY_DEV_MODE: "yes" 43 EASY_DEV_MODE_NEW: "yes" 44 # e2e requires chromium, which alpine isn't giving us arm 45 # TODO, need to add more granular setting for this to allow other composer dev 46 DEVELOPER_TOOLS: "no" 47 XDEBUG_IDE_KEY: PHPSTORM 48 XDEBUG_PROFILER_ON: 1 49 GITHUB_COMPOSER_TOKEN: c313de1ed5a00eb6ff9309559ec9ad01fcc553f0 50 OPENEMR_DOCKER_ENV_TAG: easy-dev-docker 51 OPENEMR_SETTING_site_addr_oath: 'https://localhost:9300' 52 OPENEMR_SETTING_password_grant: 3 53 OPENEMR_SETTING_rest_api: 1 </pre>
6	<p>Secret type: OPENEMR_SETTING_couchdb_pass</p> <p>Exposed Secret: password</p> <p>Line No: 162</p>	<pre> 154 OPENEMR_SETTING_rest_api: 1 155 OPENEMR_SETTING_rest_fhir_api: 1 156 OPENEMR_SETTING_rest_portal_api: 1 157 OPENEMR_SETTING_portal_onsite_two_enable: 1 158 OPENEMR_SETTING_ccda_alt_service_enable: 3 159 OPENEMR_SETTING_couchdb_host: couchdb 160 OPENEMR_SETTING_couchdb_port: 6984 161 OPENEMR_SETTING_couchdb_user: admin 162 OPENEMR_SETTING_couchdb_pass: password 163 OPENEMR_SETTING_couchdb_dbase: example 164 OPENEMR_SETTING_couchdb_ssl_allow_selfsigned: 1 165 OPENEMR_SETTING_gbl_ldap_host: 'ldap://openldap:389' </pre>

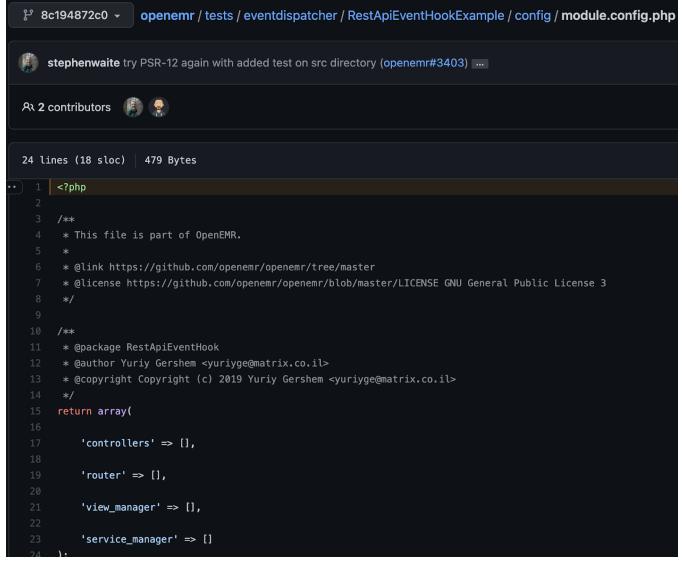
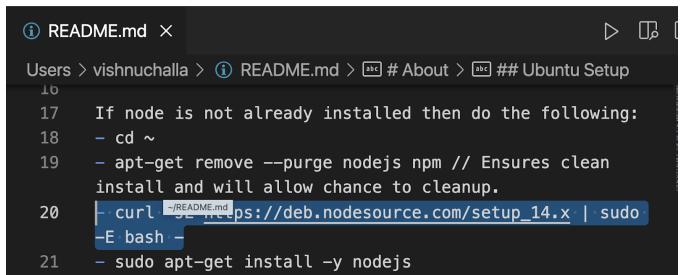
7	<p>Secret type: dbpass</p> <p>Exposed Secret: bakuper</p> <p>Line No: 8</p>	<pre> 3 4 #vars 5 6 db='openemr' 7 dbuser='backuper' 8 dbpass='bakuper' 9 pguser='sql-ledger' 10 workdir='/var/tmp/backups/' 11 tempdir='/var/tmp/isos/' 12 installdir='/var/www/openemr'</pre>
---	---	--

Openemr - SpectralOps

Seq No	Secrets	Screen Shots
1	<p>Secret type: private key</p> <p>Exposed Secret: encrypted content attached in the screenshot.</p> <p>Line No: Entire file</p>	
2	<p>Secret type: token</p> <p>Exposed Secret: eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9eyJhdWQiOiJRRI80RIRkV2h4eGJlb1Y4SGU5c1ZRQUI3STIQZWdYM3lyVUdiTTVjaWtNliwianRpIjoiYWZmYWEyOTk5NWY5MjRjZDImMjc0YTdhZGM5NmM0YzJmZTYwNGE3MjA0ODFkMTdmMWZiYTg2ZDg4ZWlxO</p>	 <pre> 22 { 23 const AUTHORIZATION_HEADER = "Authorization"; 24 const OPENEMR_AUTH_ENDPOINT = "/oauth2/default"; 25 const OAUTH_LOGOUT_ENDPOINT = "/oauth2/default/logout"; 26 const OAUTH_TOKEN_ENDPOINT = "/oauth2/default/token"; 27 const OAUTH_INTROSPECTION_ENDPOINT = "/oauth2/default/introspect"; 28 const BOGUS_CLIENTID = "ugk_IdaCszz-k0v1qHE6DYIjevkYo41neRGGpZVYfsgg"; 29 const BOGUS_CLIENTSECRET = "JWKPZvehiyjAtWfzx-M_FK2r6DfzrBjwq52Ls_BvnqkCiKitcQD"; 30 const BOGUS_ACCESS_TOKEN = "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdWQiOiJRRI80RIRkV2h4eGJlb1Y4SGU5c1ZRQUI3STIQZWdYM3lyVUdiTTVjaWtNliwianRpIjoiYWZmYWEyOTk5NWY5MjRjZDImMjc0YTdhZGM5NmM0YzJmZTYwNGE3MjA0ODFkMTdmMWZiYTg2ZDg4ZWlxO"; 31 const BOGUS_REFRESH_TOKEN = "def50200cd30606a46a09d2ba42c77528d247769112924ccff8e5d9</pre>

	<pre>WI0YWVIM2RhZTM1Zj g3MjcwMDAiLCJpYXQi OjE2MDcyMTc5MTksIm 5iZi6MTYwNzIxNzkxO SwiZXhwIjoxNjA3MjIxN TE5LCJzdWliOi5MjJjY zYzNi01NTNiLTQ1MGQ tYTJkZC1hNmRjYmM4 ZDNiMDciLCJzY29wZX MiOlsib3BlbmlkIiwic2I0Z TpkZWZhdWx0Ii19.u2U jtln3vEKIR8E0rSd-xr6m -3huCxifMaTm9_NQpl VNkBkrc6Y3KLy9FRZV S3xSY1Qgav-UvOxikYT 2zNNIK-LotEoEvZdtj87 X6fh4wh-h5BU87IHh9a NjXFUemSO9DGKJLSZ dLSeC_2w4YmbhQykG FISiltD2_PAJRKuKbpcB o3-Lafe2N83mF5i8mZX SCu_fbLrmTMYPCnsR aU_sWStzFp6p0SM3z GfLt1kjw-hcE82Ci1puq RS2nR5Z3kEOXz-hQO dXmQMq0s_gkeQZvLP OJwLGfEX5d4eIU4Bfng ksjGkKQhC7rUKT-_2F- U_z30P3izzZM6m4dZ1 0liP80g</pre> <p>Line No: 30</p>	
3	<p>Secret type: password</p> <p>Exposed Secret: openemr</p> <p>Line No: 28</p>	 <pre>Users > vishnuchalla > docker-compose.yml 26 MYSQL_ROOT_PASS: root 27 MYSQL_USER: openemr 28 +---+ MYSQL_PASS: openemr 29 OE_USER: admin 30 OE_PASS: pass 31 depends_on: 32 - mysql</pre>

4	<p>Secret type: password</p> <p>Exposed Secret: root</p> <p>Line No: 9</p>	 <pre># docker-compose.yml for travis ci testing version: '3.1' services: mysql: restart: always image: mariadb:10.5 command: ['mysqld','--character-set-server=utf8mb4'] environment: ... MySQL_ROOT_PASSWORD: root ... openemr:</pre>
5	<p>Secret type: Authorization token</p> <p>Exposed Secret: Bearer eyJ0b2tlbi6IjAwNnZ3eGJZYmFrOXIxUjF4U290Y1g4QVVd3J0cG5yYXZEaFlqaHFjWXJXRGNDDQUtFZmJONkh2cEltVkJiaWFobHBqOTBYZmLNRXpiY2FtU01pSHk1UzFlMmgxNmVqZEhc1ZENlntavpTRFRLmtsWDIyOFRKZzNhQmxMdUloZmNJM3FpMGFKZ0030Xdt0GhYT3dpVkx5b3BFRXQ1TlNYNT3UW5TZ0dsuVdq6w6jVx0VYwc21tdLSQ3RvcDV3TEkiLCJzaXRIx2IkIjoiZGVmYXVsdcIsImFwaSI6ImZoaXIifQ==</p> <p>Line No: 71</p>	 <pre>curl -X GET 'https://localhost:9300/apis/fhir/Patient' \ -H 'Authorization: Bearer eyJ0b2tlbi6IjAwNnZ3eGJZYmFrOXIxUjF4U290Y1g4QVVd3J0cG5y YXZEaFlqaHFjWXJXRGNDDQUtFZmJONkh2cEltVkJiaWFobHBqOTBYZmLN RXpiY2FtU01pSHk1UzFlMmgxNmVqZEhc1ZENlntavpTRFRLmtsWDIy OFRKZzNhQmxMdUloZmNJM3FpMGFKZ0030Xdt0GhYT3dpVkx5b3BFRXQ1 TlNYNT3UW5TZ0dsuVdq6w6jVx0VYwc21tdLSQ3RvcDV3TEkiLCJza aXRIx2IkIjoiZGVmYXVsdcIsImFwaSI6ImZoaXIifQ=='</pre>

6	<p>Secret type: php tag</p> <p>Exposed Secret: “<?php” This is a false positive but was detected by the tool</p> <p>Line No: 1</p>	 <p>A screenshot of a GitHub code review interface. The repository is 'openemr / tests / eventdispatcher / RestApiEventHookExample / config / module.config.php'. A pull request by 'stephenwaite' is shown, with the commit message 'try PSR-12 again with added test on src directory (openemr#3403) ...'. The code editor shows a single line of code: '1 <?php'. Below the code, there is a copyright notice: '/* This file is part of OpenEMR. * * @link https://github.com/openemr/openemr/tree/master * @license https://github.com/openemr/openemr/blob/master/LICENSE GNU General Public License 3 */'. The code editor indicates 24 lines (18 sloc) and 479 Bytes.</p>
7	<p>Secret type: URL</p> <p>Exposed Secret: curl -sL https://deb.nodesource.com/setup_14.x sudo -E bash -</p> <p>This is also a false positive but was detected by the tool</p> <p>Line No: 20</p>	 <p>A screenshot of a terminal window. The command entered is 'curl -sL https://deb.nodesource.com/setup_14.x sudo -E bash -'. The terminal shows the command being run and the output of the script being executed. The output includes instructions for cleaning up node.js and npm installations.</p>

What openemr should do to protect the secrets you found?

1. All the secrets should not be stored within the code. Instead, it should be stored in a vault within the cloud where the application is deployed. The application should fetch from the vault where it's deployed.
2. Authorization tokens should neither be stored in code nor it should be cached. Instead, it should be fetched from network calls, protected by SSL, TLS protocol. For eg: OAuth.

Do these secrets appear to protect valuable resources/assets? Provide the valuable assets information protected by the secrets

1. DB secrets like MYSQL_ROOT_PASS, MYSQL_USER, MYSQL_HOST are exposed. Database tables, data within each table can be accessed using these resources.
2. Authorization token to retrieve patient information is exposed. The GET end-point combined with the authorization token can be used to retrieve patients' sensitive information like health records, social security number, phone number, license id.
3. The private key is visible in the code. We can use this private key to decrypt any encrypted content found across openemr applications which might have been encrypted using the public key generated by some asymmetric encryption technique.

What good design principles does it seem OpenEMR has done to protect other types of secrets?

1. Cryptography - To maintain the Confidentiality, Authenticity, and Integrity of the data, the openemr application has used some asymmetric encryption techniques to generate a public-private key pair to encrypt and decrypt the data. One of them we found was the PGP encryption technique being used which also verifies digital signature while decrypting the data.