Augury

Test Document

Team 07

Kaila Bryant

Ian Chorne

Nahod Habte

Liam Ison

Garrett McGuire

Date: (04/20/2025)

Department: CCSWE

Course: SWE 4724

Professor: Yan Huang

**1. Testing Strategy**

**1.1 Overall strategy**

Unit testing will be conducted by the authors of scripts, as they have the most knowledge about the workings of said scripts. Test results will be discussed as a team so we can determine the best course of action for correcting failed tests.

**1.2 Test Selection**

Unit testing is the main level of testing we will complete. Each test will be for a specific function within a script.

**1.3 Adequacy Criterion**

For testing our parser functions, we can achieve full coverage by using a full JSON response from FAKEula for each function.

**1.4 Bug Tracking**

Bugs and enhancement requests will be tracked via our Kanban board. Once a bug is identified, a new ticket will be created so we can successfully track our progress with fixing it.

**1.5 Technology and Tools**

For testing our Golang scripts, we created a separate test script within the appropriate directory and ran a command prompt instruction. For example, for the parser test script, we created a script in /backend/parser.

For testing the front end, we used the automated testing technology Cypress. This technology allowed us to test React scripts in a clean and organized manner.

**2. Test Cases**

***Parser Test Table***

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Name** | **Input** | **Expected Output** | **Purpose** |
| *ParseOIL\_Azure* | *A full JSON response from querying FAKEula directly through the Azure keyword using “1.2.3.4.” There are no missing keywords.*  [*http://localhost:7000/oil/azure/1.2.3.4*](http://localhost:7000/oil/azure/1.2.3.4) | *All sources besides OIL listed as “nil.”*    *Structure type should be OIL and source should be Client.*      *Each piece of data in the JSON response properly displayed in the console.* | *To test the functionality of the parseOIL function when JSON data containing Azure information is passed through.* |
| *ParseOIL\_Helios* | *A full JSON response from querying FAKEula directly through the Helios keyword using “1.2.3.4.” There are no missing keywords.*  [*http://localhost:7000/oil/helios/1.2.3.4*](http://localhost:7000/oil/helios/1.2.3.4) | *All sources besides OIL listed as “nil.”*    *Structure type should be OIL and source should be Helios.*      *Each piece of data in the JSON response properly displayed in the console.* | *To test the functionality of parseOil when data containing Helios information is passed through.* |
| *ParseOIL\_Netflow* | *A full JSON response from querying FAKEula directly through the NetFlow keyword using “1.2.3.4.” There are no missing keywords.*  [*http://localhost:7000/oil/netflow/1.2.3.4*](http://localhost:7000/oil/netflow/1.2.3.4) | *All sources besides OIL listed as “nil.”*    *Structure type should be OIL and source should be Netflow.*      *Each piece of data in the JSON response properly displayed in the console.* | *To test the functionality of parseOil when data containing Netflow information is passed through.* |
| *ParseOIL\_Okta* | *A full JSON response from querying FAKEula directly through the Okta keyword using “1.2.3.4.” There are no missing keywords.*  [*http://localhost:7000/oil/okta/1.2.3.4*](http://localhost:7000/oil/okta/1.2.3.4) | *All sources besides OIL listed as “nil.”*    *Structure type should be OIL and source should be Okta.*      *Each piece of data in the JSON response properly displayed in the console.* | *To test the functionality of parseOil when Okta authentication logs are passed through.* |
| *ParseOIL\_Prisma* | *A full JSON response from querying FAKEula directly through the Prisma keyword using “1.2.3.4.” There are no missing keywords.*  [*http://localhost:7000/oil/prisma/1.2.3.4*](http://localhost:7000/oil/prisma/1.2.3.4) | *All sources besides OIL listed as “nil.”*   *Structure type should be OIL and source should be Prisma.*  *Each piece of data in the JSON response properly displayed in the console.* | *To test the functionality of parseOil when data prisma VPN logs are passed through.* |
| *ParseOilMissingKeyword* | *The same JSON response as ParseOil\_Azure, but with the “timestamp” keyword and data removed.* | *All sources besides OIL listed as “nil” and each piece of data in the JSON response properly displayed, besides “timestamp” which should just display “”.* | *Tests how the ParseOil function behaves when an expected keyword of data is missing.* |
| *ParseProcess* | *A full JSON response from FAKEula using “1.2.3.4.”*  *http://localhost:7000/cbr/1.2.3.4* | *All sources besides Process listed as “nil” and all the JSON data displayed properly.* | *Tests the function to parse the first type of CBR responses, processes.* |
| *ParseHost* | *A full JSON response from FAKEula using “host1.”*  *http://localhost:7000/cbr/sensor/host1* | *All sources bseides Host listed as “nil” and all the JSON data displayed properly.* | *Tests parsing function of second CBR response, Host. This is the base test case, testing using all the sample data given by querying FAKEula directly.* |
| *ParseBinary* | *A full JSON response from FAKEula using “F88ADB10AB5313D4FA33416F6F5FB4FF.”*  *http://localhost:7000/cbr/binary/F88ADB10AB5313D4FA33416F6F5FB4FF* | *All sources bseides Binary listed as “nil” and all the JSON data displayed properly.* | *Tests parsing function for the third CBR response, Binary.* |
| *ParseAsset* | *A full JSON response from FAKEula using “server.”*  *http://localhost:7000/asset/server* | *All sources besides Asset listed as “nil” and all JSON data displayed properly.* | *Tests the Asset parser.* |
| *ParseGeo* | *A full JSON response from FAKEula using “1.2.3.4.”*  *http://localhost:7000/geo/1.2.3.4* | *All sources besides GeoIP listed as “nil” and all JSON data displayed properly.* | *Tests the GEO parser.* |
| *ParseLDAP* | *A full JSON response from FAKEula using “abob.”*  *http://localhost:7000/ldap/abob* | *All sources besides LDAP listed as “nil” and all JSON data displayed properly.* | *Tests the LDAP parser.* |
| *ParsePDNS* | *A full JSON response from FAKEula using “1.2.3.4.”*   *http://localhost:7000/pdns/1.2.3.4* | *All sources besides PDNS listed as “nil” and all JSON data displayed properly.* | *Tests the PDNS parser.* |
| *Cypress\_UI\_Elements* | *Load http://localhost:8080* | *Page should load with visible elements: <h1>AUGURY</h1>, <h2>IOC Intelligence</h2>, input field, and buttons (e.g., "Search", "Switch to Light Mode").* | *To verify that key UI elements are present and visible when the page loads.* |
| *Cypress\_Input\_Click* | *Cypress selects and clicks the username input (#username)* | *Input should be focusable and clickable.* | *To ensure input field is interactive.* |
| *Cypress\_Save\_Button* | *Cypress finds and verifies visibility of button with text Save Username* | *Button should be visible and clickable.* | *To confirm that the Save Username button renders correctly.* |
| *Cypress\_Light\_Mode\_Button* | *Cypress finds the button with text Switch to Light Mode* | *The toggle button should be visible and contain the correct text.* | *To validate that the theme switch button is visible and ready for interaction.* |
| *Cypress\_Saved\_Label* | *Cypress checks visibility of <p> with text Saved Username:* | *The label should be visible when the page loads.* | *To confirm UI feedback label appears on load.* |
| **ExtractIOCsFromResponse\_Basic** | A map[string]interface{}{ "data": []interface{}{ map[string]interface{}{ "threat": map[string]interface{}{ "indicator": map[string]interface{}{ "description":"ioc1" }}}}} | []string{"ioc1"} | Verify extractIOCsFromResponse pulls out the "description" fields correctly. |
| **MD5FromCBR\_Valid** | json.RawMessage([]byte(\{"data":[{"process":{"hash":{"md5":"abcd1234"}}}]} `))` | "abcd1234" | Ensure md5FromCBR unmarshals the nested CBR response and returns the MD5 when present. |
| **TestQueryPDNS\_Success** | Call controllers.QueryPDNS via GET /pdns?ioc=example.com with FAKEULA\_API\_URL pointed at your fake‑server that returns {"data":[]} | HTTP 200 and a JSON body containing a top‐level "data" array (parsed via FormatFakeulaResponse) | Verify that QueryPDNS returns 200 and the expected JSON structure when an IOC is provided. |
| **TestQueryPDNS\_MissingIOC** | Call controllers.QueryPDNS via GET /pdns without the ioc parameter | HTTP 400 Bad Request | Ensure that QueryPDNS rejects requests missing the required ioc query parameter. |
| **TestExtractFromText\_Integration** | Call controllers.ExtractFromText via POST /extract with body []byte("…malicious.com…"), environment set so extractor stub returns that IOC, and AUGURY\_SKIP\_DB=1 | HTTP 200 and a JSON body {"data": {"malicious.com": {…}}} | End‑to‑end integration of text extraction → IOC parsing → FAKEula queries (all stubbed) → final JSON payload. |

**3. Test Results**

|  |  |  |
| --- | --- | --- |
| **Test Name** | **Actual Output** | **Pass/Fail** |
| *ParseOIL\_Azure* | *Source: :*  *"client"*  *Structure Type: :*  *"oil"*  *Entry 1: {Oil:0xc000122008 Client:0xc00001eab0 Process:<nil> Host:<nil> Binary:<nil> Asset:<nil> Geo:<nil> LDAP:<nil> PDNS:<nil>}*  *Oil Data: :*  *{*  *"timestamp": "2025-01-23T21:15:51.439Z",*  *"userPrincipalName": "alice.bob@example.com",*  *"displayName": "laptop1",*  *"clientIp": "1.2.3.4",*  *"clientAsOrg": "ASN-ACME",*  *"displayMessage": ""*  *}*  *Client Data: :*  *{*  *"as\_org": "ASN-ACME",*  *"asn": 1234,*  *"ip": "1.2.3.4"*  *}*  *PASS* | *Pass* |
| *ParseOil\_Netflow* | *Source: :*  *"netflow"*  *Structure Type: :*  *"oil"*  *Entry 1: {Oil:0xc000124008 Client:<nil> Process:<nil> Host:<nil> Binary:<nil> Asset:<nil> Geo:<nil> LDAP:<nil> PDNS:<nil>}*  *Oil Data: :*  *{*  *"clientIp": "10.0.0.1",*  *"displayMessage": "",*  *"sourcePort": "53015",*  *"transport": "UDP",*  *"eventStart": "2025-01-23T21:05:00Z",*  *"eventEnd": "2025-01-23T21:09:59Z",*  *"destinationIP": "1.2.3.4",*  *"destinationPort": "25762"*  *}*  *PASS* | *Pass* |
| *ParseOil\_Helios* | *Source: :*  *"helios"*  *Structure Type: :*  *"oil"*  *Entry 1: {Oil:0xc000122008 Client:<nil> Process:<nil> Host:<nil> Binary:<nil> Asset:<nil> Geo:<nil> LDAP:<nil> PDNS:<nil>}*  *Oil Data: :*  *{*  *"timestamp": "2025-01-23T21:12:09.000Z",*  *"clientIp": "1.2.3.4",*  *"message": "Security Alert",*  *"displayMessage": "",*  *"observerHostname": "sensor1",*  *"suricataSignature": "1234567",*  *"sourcePort": "46971",*  *"sourceThreatClassification": "Residential Proxy",*  *"sourceThreatService": "Unknown",*  *"destinationThreatClassification": "Unclassified",*  *"destinationThreatService": "UNCLASSIFIED",*  *"pipeline": "megaoil\_helios",*  *"tags": [*  *"megaoil\_helios"*  *],*  *"networkProtocol": "UDP",*  *"sourceASNOrg": "ASN-ACME",*  *"sourceASN": "1234",*  *"sourceCountry": "US",*  *"sourceCity": "Atlanta",*  *"destinationIP": "5.6.7.8",*  *"destinationPort": "161"*  *}*  *PASS* | *Pass* |
| *ParseOilMissingKeyword* | *Oil Data: {*  *Timestamp:*  *UserPrincipal:alice.bob@example.com*  *DisplayName:laptop1 ClientIP:1.2.3.4 ClientASNOrg:ASN-ACME EventType: Outcome: Message:*  *}*  *Client Data: {* *AsOrg:ASN-ACME*  *ASN:1234*  *IP:1.2.3.4}* | *Pass* |
| *ParseProcess* | Source: process  Structure Type: process  Entry 1: {Oil: Client: Process:0xc000124000 Host: Binary: Asset: Geo: LDAP: PDNS:}  Process Data:: {  "name": "java", "command\_line": "/usr/local/java/java\_base/bin/java -Dp=executionserver -server -d64 -verbose:gc", "entity\_id": "00003094-0000-13ad-01d8-6a476fb04ab2", "executable": "/bw/local/java/jdk1.8.0\_312/bin/java", "pid": 5037, "start": "2022-05-17T23:40:05.647Z",  "uptime": 53818, "parent\_name": "bash", "parent\_pid": 5028, "parent\_entity\_id": "00003094-0000-13a4-01d8-6a476faec8c6-000000000001",  "user\_name": "alice", "host\_name": "host1", "host\_type": "workstation", "host\_ips": [ "192.168.0.1" ], "host\_os": "linux", "code\_signed": false,  "url": "[https://cbr.example.com/#/analyze/00003094-0000-13ad-01d8-6a476fb04ab2/1652830876949?cb.legacy\_5x\_mode=false"](https://cbr.example.com/#/analyze/00003094-0000-13ad-01d8-6a476fb04ab2/1652830876949?cb.legacy_5x_mode=false%22) } | *Pass* |
| *ParseBinary* | *Source: :*  *"binary"*  *Structure Type: :*  *"binary"*  *Entry 1: {Oil:<nil> Client:<nil> Process:<nil> Host:<nil> Binary:0xc00010e0e0 Asset:<nil> Geo:<nil> LDAP:<nil> PDNS:<nil>}*  *Binary Data: :*  *{*  *"md5": "F88ADB10AB5313D4FA33416F6F5FB4FF",*  *"sha256": "",*  *"filename": "ysoserial.exe",*  *"accessed": "2022-04-27T11:50:32.029Z",*  *"hosts": [*  *"host1"*  *],*  *"codeSigned": false,*  *"url": "https://cbr.example.com/#/binary/F88ADB10AB5313D4FA33416F6F5FB4FF"*  *}*  *PASS* | *Pass* |
| *ParseAsset* | *Source: :*  *"asset"*  *Structure Type: :*  *"asset"*  *Entry 1: {Oil:<nil> Client:<nil> Process:<nil> Host:<nil> Binary:<nil> Asset:0xc000120120 Geo:<nil> LDAP:<nil> PDNS:<nil>}*  *Asset Data: :*  *{*  *"name": "SERVER.EXAMPLE.COM",*  *"ip": "10.0.0.1",*  *"platformName": "Security Investigator",*  *"platformOwner": "Alice",*  *"executive": "Bob",*  *"stackName": "Cyber Defense",*  *"stackOwner": "Charlie",*  *"created": "2024-05-07T00:00:00Z",*  *"updated": "2025-01-15T00:00:00Z"*  *}*  *PASS* | *Pass* |
| *ParseGeo* | *Source: :*  *"geo"*  *Structure Type: :*  *"geo"*  *Entry 1: {Oil:<nil> Client:<nil> Process:<nil> Host:<nil> Binary:<nil> Asset:<nil> Geo:0xc000118140 LDAP:<nil> PDNS:<nil>}*  *Geo Data: :*  *{*  *"countryCode": "AU",*  *"countryName": "Australia",*  *"asNumber": "",*  *"asOrg": "",*  *"ip": "1.2.3.4"*  *}*  *PASS* | *Pass* |
| *ParseLDAP* | *Source: :*  *"ldap"*  *Structure Type: :*  *"ldap"*  *Entry 1: {Oil:<nil> Client:<nil> Process:<nil> Host:<nil> Binary:<nil> Asset:<nil> Geo:<nil> LDAP:0xc000068320 PDNS:<nil>}*  *LDAP Data: :*  *{*  *"email": "alice.bob@example.com",*  *"fullName": "Alice Bob",*  *"name": "abob",*  *"title": "CISO",*  *"companyName": "Example Corp",*  *"phone": "+1 (555) 555-5555",*  *"mobile": "+1 (555) 777-7777",*  *"created": "2015-01-01 20:00:00+00:00",*  *"manager": "CN=Bob Charlie (Example-Atlanta) bcharlie,OU=Users,OU=Standard Users,OU=Users and Computers,OU=Atlanta,OU=Example,DC=DOMAIN,DC=EXAMPLE,DC=com",*  *"age": "8692"*  *}*  *PASS* | *Pass* |
| *ParsePDNS* | *Source: :*  *"pdns"*  *Structure Type: :*  *"pdns"*  *Entry 1: {Oil:<nil> Client:<nil> Process:<nil> Host:<nil> Binary:<nil> Asset:<nil> Geo:<nil> LDAP:<nil> PDNS:0xc000008180}*  *PDNS Data: :*  *{*  *"answers": [*  *{*  *"data": "1.2.3.4",*  *"name": "a.internal-test-ignore.biz",*  *"type": "A",*  *"count": 1346,*  *"start": "2019-11-06T22:54:18Z",*  *"end": "2025-01-23T00:23:21Z"*  *},*  *{*  *"data": "1.2.3.4",*  *"name": "b.internal-test-ignore.biz",*  *"type": "A",*  *"count": 1346,*  *"start": "2019-11-06T22:54:18Z",*  *"end": "2025-01-23T00:23:21Z"*  *},*  *{*  *"data": "1.2.3.4",*  *"name": "ns1.37cw.com",*  *"type": "A",*  *"count": 1,*  *"start": "2023-03-11T22:50:20Z",*  *"end": "2023-03-11T22:50:20Z"*  *},*  *{*  *"data": "1.2.3.4",*  *"name": "ns2.37cw.com",*  *"type": "A",*  *"count": 1,*  *"start": "2023-03-11T22:50:20Z",*  *"end": "2023-03-11T22:50:20Z"*  *},*  *{*  *"data": "1.2.3.4",*  *"name": "ns1.45mov.com",*  *"type": "A",*  *"count": 1,*  *"start": "2023-03-11T22:50:20Z",*  *"end": "2023-03-11T22:50:20Z"*  *}*  *]*  *}*  *PDNS Answers: %d records*  *:*  *5*  *PASS* | *Pass* |
| *Cypress\_UI\_Elements* | *Page loaded successfully; <h1>, <h2>, and input field confirmed visible via Cypress assertions.* | *Pass* |
| *Cypress\_Input\_Click* | *Cypress successfully focused and clicked into #username input field.* | *Pass* |
| *Cypress\_Save\_Button* | *"Save Username" button was located and confirmed to be visible.* | *Pass* |
| *Cypress\_Light\_Mode\_Button* | *Button with text "Switch to Light Mode" was found and visible on page.* | *Pass* |
| *Cypress\_Saved\_Label* | *Paragraph with text "Saved Username:" was successfully located and confirmed visible using cy.contains("p", "Saved Username:").* | *Pass* |
| ExtractIOCsFromResponse\_Basic | []string{"ioc1"} | Pass |
| MD5FromCBR\_Valid | "abcd1234" | Pass |
| TestQueryPDNS\_Success | HTTP 200 + {"data":[]} | Pass: stub injected via FAKEULA\_API\_URL |
| TestQueryPDNS\_MissingIOC | HTTP 400 Bad Request | Pass |
| TestExtractFromText\_Integration | HTTP 200 + {"data":{"malicious.com":{…}}} | Pass: uses stub servers and AUGURY\_SKIP\_DB=1 flag to skip db queries for testing |