Vulnerability Assessment & Penetration Testing Report

Of

AVN’s API Application

July 2022

**About L&T Technology Services:**

L&T Technology Services Limited (LTTS) is a global leader in Engineering and R&D (ER&D) services. With 399 patents filed for 51 of the Global Top 100 ER&D spenders. Our innovations speak for itself – World’s 1st Autonomous Welding Robot, Solar ‘Connectivity’ Drone, and the Smartest Campus in the World, to name a few. LTTS expertise in engineering design, product development, smart manufacturing, and digitalization touches every area of our lives. With 49 Innovation and R&D design centres globally, we specialize in disruptive technology spaces such as 5G, Artificial Intelligence, Collaborative Robots, Digital Factory, and Autonomous Transport. **LTTS is a publicly listed subsidiary of Larsen & Toubro Limited, the $18 billion Indian conglomerate operating in over 30 countries.**

Document History:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Var. Rel. No.** | **Release Date** | **Prepared By.**  **Prepared. Dt.** | **Reviewed By**  **Reviewed Dt.** | **Approved By**  **Approval Dt.** | **Remark/Revision Details** |
| 1.0 | Feb 2021 | Rohit Shetty  Pallavi Tomar  Nikhil Ravi | Vishal S R |  |  |
| 11th July 2022 | 11th July 2022 | 11th July 2022 |

Table of Contents

[1. Overview of the project 5](#_Toc108439855)

[2. Vulnerabilities explained in detail 7](#_Toc108439856)

[**2.1 No Input Validation** 7](#_Toc108439857)

[**2.2 AVN-SEND-DATE not validated at server-side** 8](#_Toc108439858)

[**2.3 Content-Type can be changed** 9](#_Toc108439859)

[**2.4 Database details disclosed after decoding the response** 10](#_Toc108439860)

[**2.5 Sensitive Information disclosure after decoding the response** 12](#_Toc108439861)

[**2.6 One character can be added in request body** 13](#_Toc108439862)

[**2.7 Text Injection** 14](#_Toc108439863)

[3. Abbreviation 16](#_Toc108439864)

[4. Appendix 16](#_Toc108439865)

# Overview of the project

L&T Technology Services (LTTS) security team has conducted Security Assessment for MG i-Smart API application. The purpose of the assessment is to evaluate the security posture of the web application against common vulnerabilities.

**Objective of the security assessment:**

As a part of this engagement a holistic approach was taken to conduct the Vulnerability Assessment and Penetration Testing on MG i-Smart API application. During the engagement High, Medium, and Low severity issues were identified with respect to Thales Drone web application.

**Approach**

The following approach was taken to make sure the target was assessed against known vulnerabilities from all possible security perspectives:

* Manual Vulnerability Assessment and Penetration Testing using OWASP TOP 10 for web application.

Some of the tools which were used are listed below:

|  |  |
| --- | --- |
| Target Application | MG i-Smart |
| Browser | Chrome, Firefox |
| Tools | BURP, Nmap, Wireshark, Whatweb |

* + 2. Key Security Policies

OWASP top 10 listed vulnerabilities ware used as a reference framework. The following key security aspects were checked:

1. Broken Access Control

2. [Cryptographic Failures](https://owasp.org/Top10/A02_2021-Cryptographic_Failures/)

3. [Injection](https://owasp.org/Top10/A03_2021-Injection/)

4. [Insecure Design](https://owasp.org/Top10/A04_2021-Insecure_Design/)

5. [Security Misconfiguration](https://owasp.org/Top10/A05_2021-Security_Misconfiguration/)

6. [Vulnerable and Outdated Components](https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/)

7. [Identification and Authentication Failures](https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/)

8. [Software and Data Integrity Failures](https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/)

9. [Security Logging and Monitoring Failures](https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/)

10. [Server-Side Request Forgery](https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/)

**Summary of Findings**

The graph below shows a summary of the number of vulnerabilities found for each impact level for the Application Security Assessment. Vulnerabilities found are addressed according to priority, findings, analysis, and recommendations from the assessment.

Table

Description automatically generated

# Vulnerabilities explained in detail

|  |  |  |  |
| --- | --- | --- | --- |
| **2.1 No Input Validation** | | | |
| **Impact** | Medium | **Risk Rating** | Low |
| **Ease of Exploit** | Difficult |
| **Likelihood** | Low |
| **Category** | CWE-20: Improper Input Validation | | |
| **URL/Impacted system** |  | | |
| **Description** | | | |
| While performing vulnerability assessment and penetration testing, it was observed that the API does not validate the parameter "AVN Version” before returning success message. This parameter can be used for further injections and exploitations to gain excessive data. | | | |
| **Impact** | | | |
| API data response without parameter validation. | | | |
| **Recommendation** | | | |
| Whitelist to restrict user input to safe values. Characters that should be blocked include (); , \* | & =, <>, ‘, “and whitespace. If implementing a whitelist, then essentially blacklisted everything out there in the universe except the stuff that’s in the list. The input should content strings. | | | |
| **How to recreate the Security defect** | | | |
|  | | | |
| **Evidence** | | | |
| **Graphical user interface, text, application  Description automatically generated** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **2.2 AVN-SEND-DATE not validated at server-side** | | | |
| **Impact** | Low | **Risk Rating** | Info |
| **Ease of Exploit** | Moderate |
| **Likelihood** | Low |
| **Category** | CWE-602: Client-Side Enforcement of Server-Side Security | | |
| **URL/Impacted system** |  | | |
| **Description** | | | |
| Random value set for avn-send-date and no server checks in place to authenticate them. | | | |
| **Impact** | | | |
| Authentication bypass vulnerability could allow attackers to perform various malicious operations by bypassing the device authentication mechanism. | | | |
| **Recommendation** | | | |
| • It is suggested to not expose authentication protocol in the client-side web browser script.  • It is recommended to validate all user input on the server side.  • It is best to have a secure and strong authentication policy in place | | | |
| **How to recreate the Security defect** | | | |
|  | | | |
| **Evidence** | | | |
| **Graphical user interface, text  Description automatically generated**  **Graphical user interface, text  Description automatically generated** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **2.3 Content-Type can be changed** | | | |
| **Impact** | Low | **Risk Rating** | Info |
| **Ease of Exploit** | Moderate |
| **Likelihood** | Low |
| **Category** | CWE-732: Incorrect Permission Assignment for Critical Resource | | |
| **URL/Impacted system** |  | | |
| **Description** | | | |
| CRLF refers to the special character elements "Carriage Return" and "Line Feed." These elements are embedded in HTTP headers and other software code to signify an End of Line (EOL) marker. Exploits occur when an attacker can inject a CRLF sequence into an HTTP stream. By introducing this unexpected CRLF injection, the attacker can maliciously exploit CRLF vulnerabilities to manipulate the web application's functions. A more formal name for CRLF injection is Improper Neutralization of CRLF Sequences. Because CRLF injection is frequently used to split HTTP responses, it can also be designated as HTTP Response Splitting or Improper Neutralization of CRLF Sequences in HTTP Headers. | | | |
| **Impact** | | | |
| CRLF results in an attacker gaining unauthorized access to modify application data, compromising integrity and enabling the exploitation of the vulnerabilities such as: XSS, Proxy and web server cache poisoning, website defacement, Hijacking the session etc | | | |
| **Recommendation** | | | |
| * Always follow the rule of never trusting user input. * Sanitize and neutralize all user-supplied data or properly encode output in HTTP headers that would otherwise be visible to users to prevent the injection of CRLF sequences and their consequences. Always follow the rule of never trusting user input. * Sanitize and neutralize all user-supplied data or properly encode output in HTTP headers that would otherwise be visible to users to prevent the injection of CRLF sequences and their consequences. | | | |
| **How to recreate the Security defect** | | | |
|  | | | |
| **Evidence** | | | |
| **Graphical user interface, text, application  Description automatically generated**  **Graphical user interface, text, application  Description automatically generated** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **2.4 Database details disclosed after decoding the response** | | | |
| **Impact** | Low | **Risk Rating** | Info |
| **Ease of Exploit** | Easy |
| **Likelihood** | Low |
| **Category** |  | | |
| **URL/Impacted system** | <https://thalesdrone-test.ltts.com/> | | |
| **Description** | | | |
| Information with respect to SQL query for tables creation is disclosed in the response. | | | |
| **Impact** | | | |
| Application package includes source code/ libraries which are not in use and can be exploited by an attacker. | | | |
| **Recommendation** | | | |
| It is highly recommended not to display the SQL databases queries containing information in the response. | | | |
| **How to recreate the Security defect** | | | |
|  | | | |
| **Evidence** | | | |
| **Graphical user interface, text, application, Word  Description automatically generated**  **Graphical user interface, text, application, email  Description automatically generated** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **2.5 Sensitive Information disclosure after decoding the response** | | | |
| **Impact** | Low | **Risk Rating** | Info |
| **Ease of Exploit** | Moderate |
| **Likelihood** | Low |
| **Category** | Information Disclosure | | |
| **URL/Impacted system** |  | | |
| **Description** | | | |
| The exposes of sensitive information to an actor that is not explicitly authorized to have access to that information. | | | |
| **Impact** | | | |
| Attacker can steal the sensitive information and can get full access as an admin into the application. | | | |
| **Recommendation** | | | |
| * It is highly recommended that there should be restriction on admin Account details and sensitive information such as username, user-id, phone number should not be stored in the application. * It is highly recommended that the sensitive information should not be disclosed as attacker can manipulate the information and get access to the account. * Make sure that the web application does not pass any sensitive information in URLs. | | | |
| **How to recreate the Security defect** | | | |
|  | | | |
| **Evidence** | | | |
| **Graphical user interface, text  Description automatically generated**  **Text  Description automatically generated** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **2.6 One character can be added in request body** | | | |
| **Impact** | Low | **Risk Rating** | Info |
| **Ease of Exploit** | Moderate |
| **Likelihood** | Low |
| **Category** | CWE-20: Improper Input Validation | | |
| **URL/Impacted system** |  | | |
| **Description** | | | |
| SQL injection is one of the most common attack mechanisms utilized by attackers to steal sensitive data from organizations. While SQL Injection can affect any data-driven application that uses a SQL database, it is most often used to attack web sites. SQL Injection is a code injection technique that hackers can use to insert malicious SQL statements into input fields for execution by the underlying SQL database. This technique is made possible because of improper coding of vulnerable web applications. | | | |
| **Impact** | | | |
| A successful SQL injection attack can result in unauthorized access to sensitive data such as tables name. | | | |
| **Recommendation** | | | |
| * The most basic SQL injections can be prevented by filtering input. However, it will not stop more complex attacks. * To prevent maximum SQL injection damage, it is recommended to separate your sensitive data and store it in multiple databases. * Customize your error messages so they don't give away any information about the structure of your database to a potential intruder. | | | |
| **How to recreate the Security defect** | | | |
|  | | | |
| **Evidence** | | | |
| **Graphical user interface, application, Word  Description automatically generated**  **Graphical user interface, text, application  Description automatically generated** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **2.7 Text Injection** | | | |
| **Impact** | Low | **Risk Rating** | Info |
| **Ease of Exploit** | Moderate |
| **Likelihood** | Low |
| **Category** | CWE-94: Improper Control of Generation of Text ('Text Injection') | | |
| **URL/Impacted system** | <https://thalesdrone-test.ltts.com/> | | |
| **Description** | | | |
| Text injection is a basically injection in which user input is reflected as it is in the application response as plaintext. This is one of the ways to perform content spoofing or virtual defacement which can be used in phishing attacks. | | | |
| **Impact** | | | |
| An attacker can use text injection vulnerability to present a customized message on the application that can phish users into believing that the message is legitimate. The intent is typical to tick victims, although sometimes the actual purpose may be to simply misrepresent the organization or an individual. | | | |
| **Recommendation** | | | |
| * The application should only accept the values and types that are defined for parameters and should be checked at the server-side whether there is change content, if there is change, then the application should reject that request. * Never construct and send messages via URL in the page response. Prefer using messages predefined in a property file. | | | |
| **How to recreate the Security defect** | | | |
|  | | | |
| **Evidence** | | | |
| **Graphical user interface, text, application, Word  Description automatically generated** | | | |

# Abbreviation

|  |  |
| --- | --- |
| APP | Application |
| HTML | Hyper Text Mark-up Language |
| HTTP(S) | Hypertext transfer protocol (Secured) |
| Pg. | Page |
| TLS | Transport Layer Security |
| SSL | Secure Sockets Layer |
| IP | Internet Protocol |
| LTTS | Larsen & Toubro Technology Services |
| SOP | Same Origin Policy |
| OWASP | Open Web Application Security Project |
| VAPT | Vulnerability Assessment and Penetration testing |
| CSP | Content Security Policy |
| CORS | Cross Origin Resource Sharing |
| IDOR | Insecure direct object references |
| MFA | No Multifactor Authentication |
| URL | Uniform Resource locator |
| XSS | Cross-Site Scripting |
| XXE | XML External Entities |
| SQL | Structured Query Language |

# Appendix

Vulnerability scan reports.

|  |
| --- |
|  |

|  |
| --- |
|  |