**Zurich API Penetration Testing Report V1.0**

Submitted to <Client-name>

Submitted to Zurich

By



**Wipro Ltd.**

**Aug 2021**

|  |  |
| --- | --- |
| **Item** | **Description** |
| Document Title: | Zurich API Penetration Testing Report |
| Version No: | 1.0 |
| Status: | Penetration Test Report |
| File Name: | Zurich API Penetration Testing Report V1.0 |
| Publish Date: | 12th Aug 2021 |

|  |  |  |
| --- | --- | --- |
| **Author(s)** | | |
| ***Name*** | ***Functional Section, Department*** | ***Signature/Date*** |
| Devi Satish K | CRS, Wipro Ltd | 13th Aug, 2021 |
| **Reviewed by** |  |  |
| ***Name*** | ***Functional Section, Department*** | ***Signature/Date*** |
| Neha Srivastava | CRS , Wipro Ltd | 13th Aug, 2021 |

|  |  |  |
| --- | --- | --- |
| **Approver Name** | **Role** | **Approval Received** |
| Steven Brindley | Solution Architect |  |
| Stuart Wallis | Information Security Consultant |  |
| Neil Brakewell | Project Manager |  |
| Steven Hunt | Programme QA Manager |  |

[1 Executive Summary 5](#_Toc78379505)

[1.1 Objective 5](#_Toc78379506)

[2 Scope 6](#_Toc78379507)

[2.1 In Scope 6](#_Toc78379508)

[2.2 Applications in Scope 6](#_Toc78379509)

[2.3 Penetration Testing Activities 6](#_Toc78379510)

[2.4 Out of scope 7](#_Toc78379511)

[3 Approach 7](#_Toc78379512)

[4 Findings and Recommendation 9](#_Toc78379513)

[4.1 Security Threats by Severity 9](#_Toc78379514)

[4.2 High Level Summary of Vulnerabilities Identified 9](#_Toc78379515)

[5 Detailed Findings 10](#_Toc78379516)

[5.1 Critical Severity Vulnerabilities 10](#_Toc78379517)

[5.1.1 No “Critical Severity” vulnerabilities were identified during the assessment. 10](#_Toc78379518)

[5.2 High Severity Vulnerabilities 10](#_Toc78379519)

[5.2.1 No “High Severity” vulnerabilities were identified during the assessment 10](#_Toc78379520)

[5.3 Medium Severity Vulnerabilities 10](#_Toc78379521)

[5.3.1 No “Medium Severity” vulnerabilities were identified during the assessment 10](#_Toc78379522)

[5.4 Informational Severity Vulnerabilities 10](#_Toc78379523)

[5.4.1 No “Informational Severity” vulnerabilities were identified during the assessment. 10](#_Toc78379524)

[5.5 Test Run Output for Step-1 11](#_Toc78379525)

[5.6 Test Run Output for Step 2 21](#_Toc78379526)

[6 Appendix 25](#_Toc78379527)

[6.1 Wipro’s Web Penetration Testing Assessment Methodology 25](#_Toc78379528)

[6.2 Threat Severity Levels 26](#_Toc78379529)

[6.3 Attack Complexity Level Definitions 27](#_Toc78379530)

[6.4 Risk Rating Matrix by Impact and Likelihood 27](#_Toc78379531)

[6.4.1 Likelihood 28](#_Toc78379532)

[6.4.2 Impact 28](#_Toc78379533)

[6.5 CVSS v3.1 Risk Ratings 29](#_Toc78379534)

[6.6 Details of OWASP Vulnerabilities 30](#_Toc78379535)

[6.7 Best Practices for Input / Output Data Validation 32](#_Toc78379536)

[7 Glossary 34](#_Toc78379537)

# Executive Summary

## Objective

This 2-step assessment was conducted with an aim to gradually bypassing the controls and access the API, WADL file and perform API testing as per OWASP standard. This penetration testing exercise was conducted against Zurich API, which was accessible https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/{2EFE0C43-F107-4D81-B5D9-44D66729A3F5}, https://uat-zeus-interfaces.zurich.com/b2b-api/v1/claims/80-52-725858/activities

The overall objective of step-1 exercise is to access the Zurich API with fake certificates. The testing was performed from 5th Aug-2021 till 11th Aug-2021.

Strengths of application revealed during the assessment were:

* API was not accessible with the fake certificates.

The overall objective of Step-2 exercise is to pull the data from the API and extract the WADL file. We had received 2 certificates (zurick-wipro-pentest.pem and ZFSIssuingCA1.pem) from the development team to perform this activity.

The testing was performed between 5th Aug-2021 till 11th Aug-2021 based on the schedule agreed.

Strengths of application revealed during the assessment were:

* WADL file is not extractable, and server responds with 404 file not found error.

# Scope

## In Scope

This section provides information on the applications and penetration testing activities in scope.

## Applications in Scope

For Step-1

|  |  |
| --- | --- |
| Application Name | Comments |
| Zurich API | Development team had not provided any certificate or file. |

For Step-2

|  |  |
| --- | --- |
| Application Name | Comments |
| Zurich API | Development team had provided CA and client certificates with key |

## Penetration Testing Activities

Testing activities are categorized under the below main categories:

For Step-1

* Creating the fake certificates with tools and importing.
* Access the API with the fake certificates.

For Step-2

* Importing the certificate.
* Configure pfx file in Microsoft store and access the API
* Extract WADL file

## Out of scope

* Social Engineering
* Secure Code Review
* Vulnerability Fixes
* Intrusive Tests and Exploitation
* Denial of Service

# Approach

For Step-1

We had created few fake client certificates through Openssl tool to access the API.

For Step-2

We had received 2 certificates from the development team to access the API.

|  |  |  |  |
| --- | --- | --- | --- |
| Descriptions | Objective | Procedure | Tools Used |
| Walkthrough  Zurich-b2b API | To gather information about b2b API and understand its functionality | * Manual walkthrough of the b2b API to identify application entry points * Used spiders / crawlers to capture resources related to the API * Analyzed the error codes returned by the API * Performed application fingerprinting to understand the technology stack and web server version | * Burp Suite Professional and ReadyAPI |
| Vulnerability  Identification | To discover vulnerabilities in API via simulation of attacks via the proxy tool | * Analyzed the application framework with known vulnerabilities. * Tested for privilege escalation through parameter manipulation * Performed test cases for insecure direct object reference * Analyzed for the input validation on every parameter * Carried out session related test cases * Tested for the business logic flaws * Checked for integrity issues in the application | * Burp Suite Professional, CURL and ReadyAPI |
| Vulnerability Exploitation | To demonstrate the existence of the discovered vulnerability | Performed a manual assessment based on the discovered identified vulnerabilities covering the following application security domains:   * Missing Access Control * Improper Input Validation * Session Management issues * Password Policy Validation * Injection Vulnerabilities * Insecure file Upload * Components with known vulnerabilities * Insufficient Logging and Monitoring * Security Misconfiguration * Sensitive Data Exposure | * Burp Suite Professional and ReadyAPI |
| Reporting | To document the identified vulnerabilities in the API | Create report documenting the vulnerabilities discovered in the test. | * Customized report |

# Findings and Recommendation

## Security Threats by Severity

Wipro team performed several tests and not able to access API and extract API WADL through the external network.

## High Level Summary of Vulnerabilities Identified

Wipro team performed several tests and were not able to access the API with fake certificates and also not able to extract WADL file as per step 1 and step 2 respectively.

**Vulnerability Distribution by Risk Rating**

Summary of vulnerability findings according to the severity levels and based on OWASP Top 10 are as shown below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **OWASP 2017 top 10** | | | | | | |
| **Description** | **Critical** | **High** | **Medium** | **Low** | **Informational** | **Total** |
| A1 – Injection | **0** | **0** | **0** | **0** | **0** | **0** |
| A2 – Broken Authentication | **0** | **0** | **0** | **0** | **0** | **0** |
| A3 – Sensitive Data Exposure | **0** | **0** | **0** | **0** | **0** | **0** |
| A4 – XML External Entities (XXE) | **0** | **0** | **0** | **0** | **0** | **0** |
| A5 – Broken Access Control | **0** | **0** | **0** | **0** | **0** | **0** |
| A6 – Security Misconfiguration | **0** | **0** | **0** | **0** | **0** | **0** |
| A7 – Cross Site Scripting (XSS) | **0** | **0** | **0** | **0** | **0** | **0** |
| A8 – Insecure Deserialization | **0** | **0** | **0** | **0** | **0** | **0** |
| A9 – Using Components with Known Vulnerabilities | **0** | **0** | **0** | **0** | **0** | **0** |
| A10 – Insufficient Logging & Monitoring | **0** | **0** | **0** | **0** | **0** | **0** |
| Business Logic By Pass | **0** | **0** | **0** | **0** | **0** | **0** |
| **Total** | **0** | **0** | **0** | **0** | **0** | **0** |

# Detailed Findings

## Critical Severity Vulnerabilities

# No “Critical Severity” vulnerabilities were identified during the assessment.

## High Severity Vulnerabilities

# No “High Severity” vulnerabilities were identified during the assessment

## Medium Severity Vulnerabilities

# No “Medium Severity” vulnerabilities were identified during the assessment

## Informational Severity Vulnerabilities

# No “Informational Severity” vulnerabilities were identified during the assessment.

## Test Run Output for Step-1

**Test Run Output**

The overall objective of this exercise is to access the Zurich API with fake certificates. The testing was performed from 6th-Aug-2021 till 15-Aug-2021.

We have created few fake client certificates through Openssl tool to access the API, but the API was not accessible from client end. Please refer the below screenshot for reference:

**Screenshot 1:**

1. We have used Openssl tool to generate an ECC Private Key and CSR (Certificate Signing Request).

2. During the generation of the CSR, you are prompted for several pieces of information. These are the X.509 attributes of the certificate.

3. We have inserted certificate attributes as shown in the snapshot below:

**Certificate Information**

**Graphical user interface, text, application

Description automatically generated**

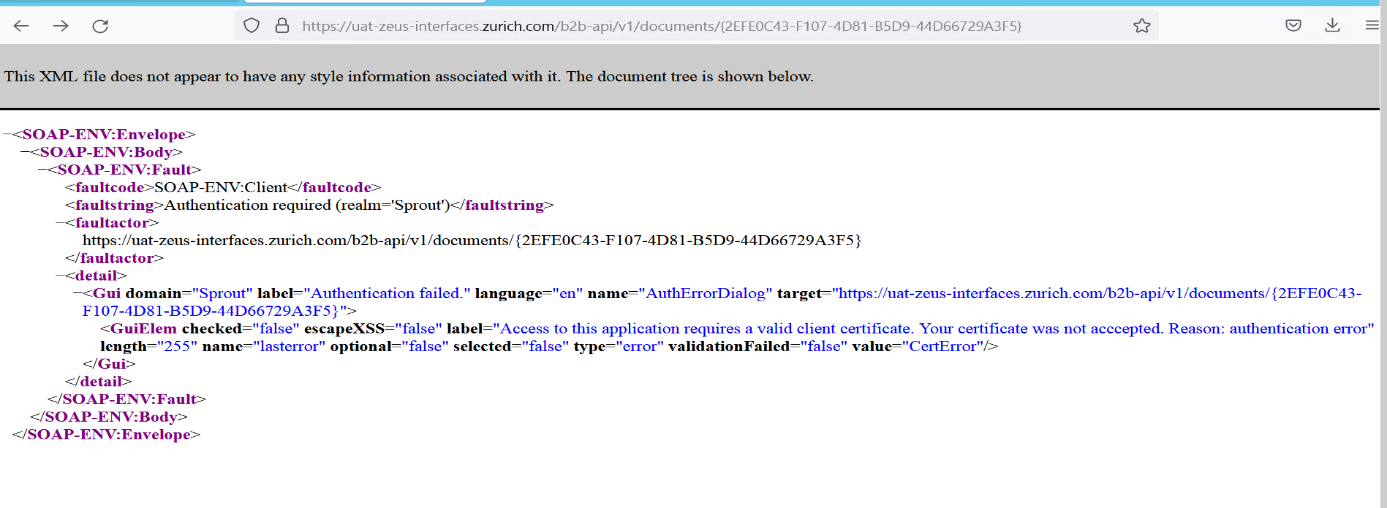
4. Now we have imported Client1.crt in the browser as shown in the snapshot below:

Graphical user interface, text, application, chat or text message, email

Description automatically generated

5. Browse the API URL: [https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/{2EFE0C43-F107-4D81-B5D9-44D66729A3F5}](https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/%7b2EFE0C43-F107-4D81-B5D9-44D66729A3F5%7d) and it is not accessible and triggered certificate error as shown in the snapshot below:

|  |  |
| --- | --- |
| Affected URLs | [https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/{2EFE0C43-F107-4D81-B5D9-44D66729A3F5}](https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/%7b2EFE0C43-F107-4D81-B5D9-44D66729A3F5%7d)  [https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/{EDF92D1D-E3CF-4941-B192-BE899D1600C8}](https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/%7b2EFE0C43-F107-4D81-B5D9-44D66729A3F5%7d)  <https://uat-zeus-interfaces.zurich.com/b2b-api/v1/claims/80-52-725858/activities>  <https://uat-zeus-interfaces.zurich.com/b2b-api/v1/claims/80-52-725858/activities/cc%3A7371083> |



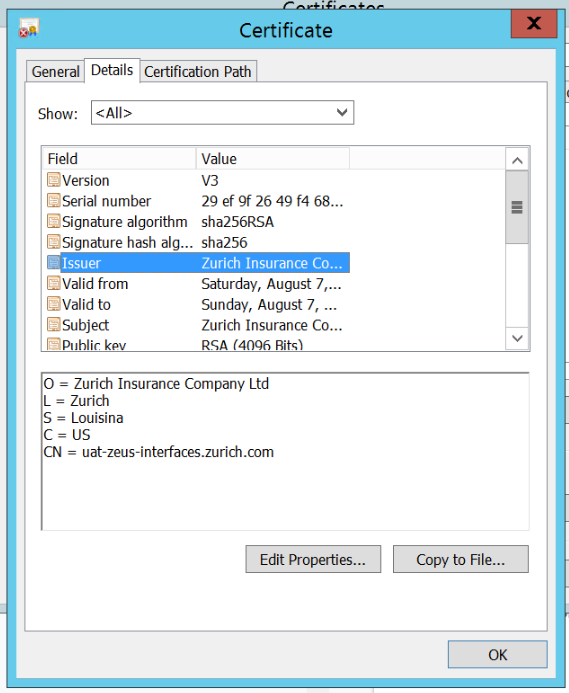
**Scenario 2:**

1. We have used Openssl tool to generate a RSA Private Key 4096 bits and CSR (Certificate Signing Request).

2. During the generation of the CSR, you are prompted for several pieces of information. These are the X.509 attributes of the certificate.

3. We have inserted certificate attributes as shown in the snapshot below:

**Certificate Information**



4. Now, we have imported Client2.crt in the browser as shown in the snapshot below:

Graphical user interface, text, application, chat or text message, email

Description automatically generated

5. Browse the API URL: [https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/{2EFE0C43-F107-4D81-B5D9-44D66729A3F5}](https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/%7b2EFE0C43-F107-4D81-B5D9-44D66729A3F5%7d) and it is not accessible triggered certificate error as shown in the snapshot below:

|  |  |
| --- | --- |
| Affected URLs | [https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/{2EFE0C43-F107-4D81-B5D9-44D66729A3F5}](https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/%7b2EFE0C43-F107-4D81-B5D9-44D66729A3F5%7d)  [https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/{EDF92D1D-E3CF-4941-B192-BE899D1600C8}](https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/%7b2EFE0C43-F107-4D81-B5D9-44D66729A3F5%7d)  <https://uat-zeus-interfaces.zurich.com/b2b-api/v1/claims/80-52-725858/activities>  <https://uat-zeus-interfaces.zurich.com/b2b-api/v1/claims/80-52-725858/activities/cc%3A7371083> |

Graphical user interface, text, application, email

Description automatically generated

**Scenario 3:**

1. We have used Openssl tool to generate an RSA Private Key 4096 bits and CSR (Certificate Signing Request) for **Multidomain certificate**.

2. During the generation of the CSR, you are prompted for several pieces of information. We have inserted 3 different domains in DNS Alternative name attribute of CSR. These are the X.509 attributes of the certificate

Graphical user interface, text, application

Description automatically generated

3. We have imported Client3.crt in the browser as shown in the snapshot below:

Graphical user interface, text, application, email

Description automatically generated

5. Browse the API URL: [https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/{2EFE0C43-F107-4D81-B5D9-44D66729A3F5}](https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/%7b2EFE0C43-F107-4D81-B5D9-44D66729A3F5%7d) and it is not accessible and triggered certificate error as shown in the snapshot below:

|  |  |
| --- | --- |
| Affected URLs | [https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/{2EFE0C43-F107-4D81-B5D9-44D66729A3F5}](https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/%7b2EFE0C43-F107-4D81-B5D9-44D66729A3F5%7d)  [https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/{EDF92D1D-E3CF-4941-B192-BE899D1600C8}](https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/%7b2EFE0C43-F107-4D81-B5D9-44D66729A3F5%7d)  <https://uat-zeus-interfaces.zurich.com/b2b-api/v1/claims/80-52-725858/activities>  <https://uat-zeus-interfaces.zurich.com/b2b-api/v1/claims/80-52-725858/activities/cc%3A7371083> |

Graphical user interface, text, application, email

Description automatically generated

**Screenshot 4:**

1. We have used Openssl tool to generate **Wildcard SSL certificate** with an RSA Private Key and CSR (Certificate Signing Request)

2. During the generation of the CSR, you are prompted for several pieces of information. These are the X.509 attributes of the certificate.

3. We have inserted certificate attributes as shown in the snapshot below:

**Certificate Information**

Text

Description automatically generated

4. We have imported Client4.crt in the browser as shown in the snapshot below:

Graphical user interface, text, application

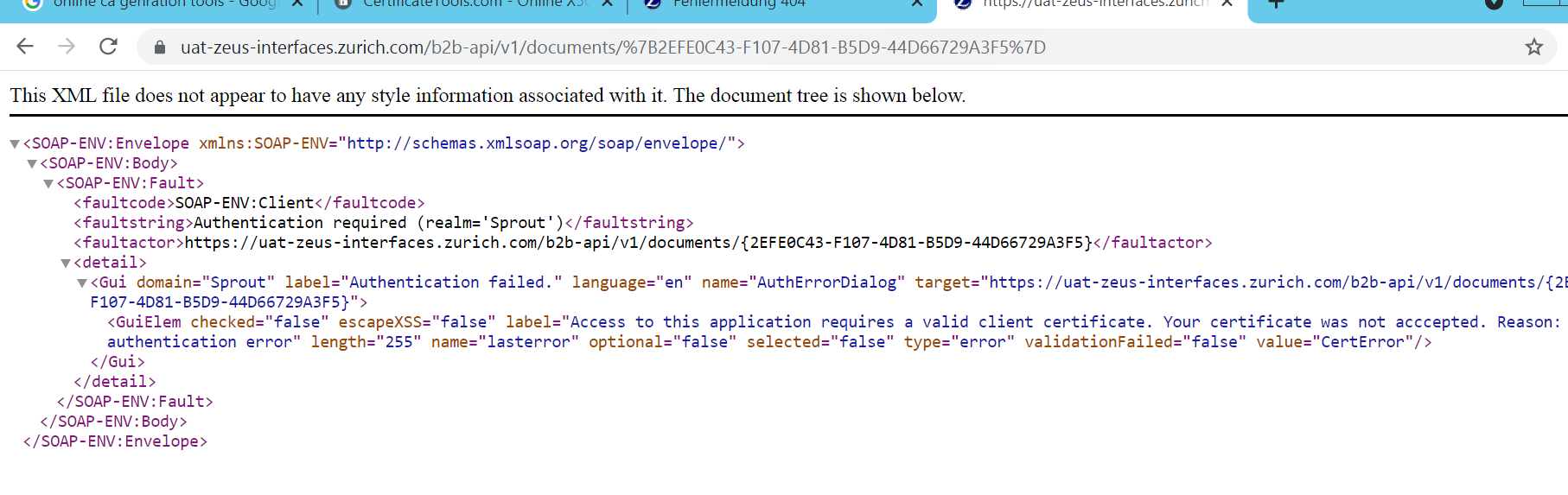
Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

5. Browse the API URL: [https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/{2EFE0C43-F107-4D81-B5D9-44D66729A3F5}](https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/%7b2EFE0C43-F107-4D81-B5D9-44D66729A3F5%7d) and it is not accessible and triggered certificate error as shown in the snapshot below:

|  |  |
| --- | --- |
| Affected URLs | [https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/{2EFE0C43-F107-4D81-B5D9-44D66729A3F5}](https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/%7b2EFE0C43-F107-4D81-B5D9-44D66729A3F5%7d)  [https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/{EDF92D1D-E3CF-4941-B192-BE899D1600C8}](https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/%7b2EFE0C43-F107-4D81-B5D9-44D66729A3F5%7d)  <https://uat-zeus-interfaces.zurich.com/b2b-api/v1/claims/80-52-725858/activities>  <https://uat-zeus-interfaces.zurich.com/b2b-api/v1/claims/80-52-725858/activities/cc%3A7371083> |



**Screenshot 5:**

1. We have used Openssl tool to generate **Wildcard SSL certificate with different domain** with an RSA Private Key and CSR (Certificate Signing Request)

2. During the generation of the CSR, you are prompted for several pieces of information. These are the X.509 attributes of the certificate.

3. We have inserted certificate attributes as shown in the snapshot below:

**Certificate Information**

Text

Description automatically generated

4. We have imported Client5.crt in the browser as shown in the snapshot below:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

5. Browse the API URL: [https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/{2EFE0C43-F107-4D81-B5D9-44D66729A3F5}](https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/%7b2EFE0C43-F107-4D81-B5D9-44D66729A3F5%7d) and it is not accessible and triggered certificate error as shown in the snapshot below:

|  |  |
| --- | --- |
| Affected URLs | [https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/{2EFE0C43-F107-4D81-B5D9-44D66729A3F5}](https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/%7b2EFE0C43-F107-4D81-B5D9-44D66729A3F5%7d)  [https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/{EDF92D1D-E3CF-4941-B192-BE899D1600C8}](https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/%7b2EFE0C43-F107-4D81-B5D9-44D66729A3F5%7d)  <https://uat-zeus-interfaces.zurich.com/b2b-api/v1/claims/80-52-725858/activities>  <https://uat-zeus-interfaces.zurich.com/b2b-api/v1/claims/80-52-725858/activities/cc%3A7371083> |

Graphical user interface, text, application, email

Description automatically generated

**Note: The fake client certificates are already shared separately.**

## Test Run Output for Step 2

**Screenshot 1:**

1. We had imported provided certificates both CA and client file by windows certificate store in **Ready API tool**
2. Now when we clicked on send button, it showed us 200 ok code as shown in the snapshot below:

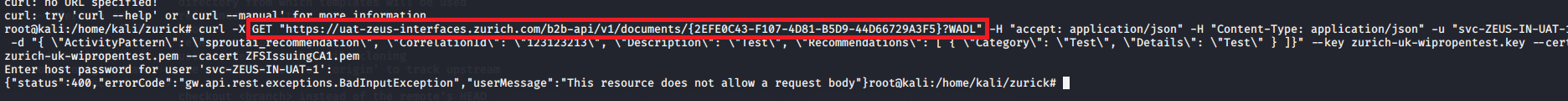
Graphical user interface, text, application, email

Description automatically generated

1. We have applied automated fuzzing on below two APIs with ReadyApi Pro tool but we have not able to extract WDASL file through this scenario

**Scenario 2:**

1. We have tried to access WADL file by amending **.asmx?WADL** in the end of the API URL:
2. https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/{2EFE0C43-F107-4D81-B5D9-44D66729A3F5}
3. https://uat-zeus-interfaces.zurich.com/b2b-api/v1/claims/80-52-725858/activities
4. Crafted CURL HTTP request and amend asmx?WADL in the URL, sent CURL HTTP request to server
5. We got 404 file not found error as a response as shown in the snapshot below:



**Scenario 3:**

1. We have tried to access WADL file by amending externalapis\_UAT.YAML in the end of the API URL:

A https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/{2EFE0C43-F107-4D81-B5D9-44D66729A3F5}

B https://uat-zeus-interfaces.zurich.com/b2b-api/v1/claims/80-52-725858/activities

1. Crafted CURL HTTP request and amend externalapis\_UAT.YAML in the URL, sent CURL HTTP request to server
2. We got 404 file not found error as a response as shown in the snapshot below:

**Request and Response:**

Text

Description automatically generated

**Scenario 4:**

1. We have tried to access WADL file by amending .svc in the end of the API URL:

A https://uat-zeus-interfaces.zurich.com/b2b-api/v1/documents/{2EFE0C43-F107-4D81-B5D9-44D66729A3F5}

B https://uat-zeus-interfaces.zurich.com/b2b-api/v1/claims/80-52-725858/activities

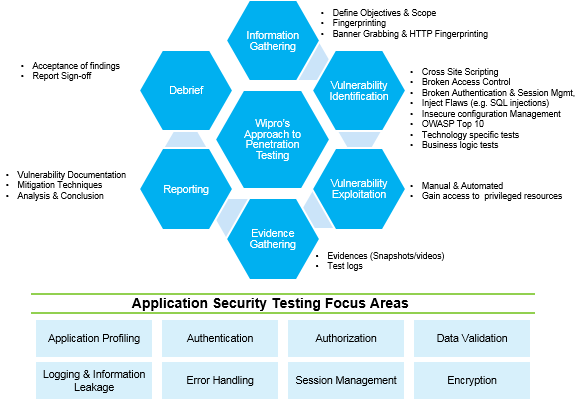
1. Crafted CURL HTTP request and amend b2b-api.svc, sent CURL HTTP request to server
2. We got 404 file not found error as a response as shown in the snapshot below:

Graphical user interface

Description automatically generated with medium confidence

# Appendix

## Wipro’s Web Penetration Testing Assessment Methodology



## Threat Severity Levels

|  |  |  |
| --- | --- | --- |
| **Severity** | **Description** | **Color Code** |
| **Critical** | Critical (Level 4) vulnerabilities provide remote intruders with remote root or administrator capabilities. With this level of vulnerability, hackers can compromise the entire host. Level 4 includes vulnerabilities that provide remote hackers full file system read and write capabilities, remote execution of commands as a root or administrator user. |  |
| **High** | High Risk (Level 3) vulnerabilities provide intruders with remote user, but not remote administrator or root user capabilities. Level 3 vulnerabilities give hackers partial access to file systems (for example, full read access without full write access). Vulnerabilities that expose highly sensitive information also qualify as level 3 vulnerabilities. |  |
| **Medium** | Medium Risk (Level 2) vulnerabilities provide hackers with access to specific information stored on the host, including security settings. This level of vulnerabilities could result in potential misuse of the host by intruders. Examples of level 2 vulnerabilities include partial disclosure of file contents, access to certain files on the host, directory browsing, disclosure of filtering rules and security mechanisms, susceptibility to denial of service (DoS) attacks, and unauthorized use of services (for example, mail relaying). |  |
| **Low** | Low Risk (Level 1) vulnerabilities expose some sensitive information from the host, such as precise versions of services. With this information, hackers could research potential attacks to try against a host. |  |
| **Information** | Informational are those vulnerabilities. Which may not be exploited but sometimes leak sensitive information |  |

## Attack Complexity Level Definitions

Some attacks against computer systems are more complicated than others; exploiting vulnerability in an application may involve merely inserting a "magic" character in form field, while other attacks may require a carefully coordinated series of interactions with obscure application services. Unfortunately, the complexity of an attack has more of an effect on the likelihood of it being defended against, rather than the likelihood of it being used by an attacker (who is probably wielding an arsenal of complex attacks to leverage against a computer system). Ironically, the most complex attacks are often the most popular.

**High:** Exploitation of this problem requires exploit code, which is difficult to write and may require access to specific types of computer systems. Actually using this tool may require specific knowledge of the vulnerability and the system on which it is present.

**Medium:** A special-purpose software tool is required to exploit this problem; this tool is probably quite easy to use and understand by a neophyte hacker, but exploitation of this problem may be out of the reach of individuals that are not familiar with the security community or the hacker underground.

**Low:** The attack can be executed by an unskilled attacker without any special tools (perhaps by using standard UNIX utilities, or by using their web browser). The problem may be obvious even to someone who is not familiar with the issues involved in computer security.

## Risk Rating Matrix by Impact and Likelihood

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Risk Rating** | | **Impact** | | | |
| **Minor** | **Moderate** | **Major** | **Extreme** |
| **Likelihood** | **Very Likely** | **Medium** | **High** | **Critical** | **Critical** |
| **Likely** | **Low** | **Medium** | **High** | **Critical** |
| **Unlikely** | **Low** | **Medium** | **Medium** | **High** |
| **Very Unlikely** | **Low** | **Low** | **Low** | **Medium** |

# Likelihood

| **Likelihood** | **Explanation** |
| --- | --- |
| **Very Likely** | The attack can be executed by an attacker without using any special tools or techniques. Vulnerability that can be obviously exploited where exploits are already available. |
| **Likely** | The attack is not too difficult to carry out. Specific tools may be used to carry out such attacks. |
| **Unlikely** | A special-purpose software tool is required to exploit this problem; this tool is probably quite easy to use and understand by a neophyte hacker, but exploitation of this problem may be out of the reach of individuals that are not familiar with the security community or the hacker underground. |
| **Very Unlikely** | Exploitation of this problem requires exploit code, which is difficult to write and may require access to specific types of computer systems. Actually using this tool may require specific knowledge of the vulnerability and the system on which it is present. |

# Impact

| **Impact** | **Explanation** |
| --- | --- |
| **Extreme/Critical** | The finding may result in a serious compromise of multiple assets of the Organization. |
| **Major/High** | The finding may result in a serious compromise of the system in question. This may imply an actual shell-level compromise (i.e. root or administrator) or a significant compromise of confidential information assets (i.e. database mining). |
| **Moderate/ Medium** | The finding may result in a significant compromise of the system in question. This may imply the theft of a user-credential, or the ability to access limited information assets on the system. |
| **Minor/Low** | The finding may result in a relatively mild compromise of the system in question. This may imply minor information disclosure or common misconfiguration issues. |

## CVSS v3.1 Risk Ratings

The Common Vulnerability Scoring System (CVSS) is a free and open industry standard for assessing the severity of computer system security vulnerabilities.

CVSS is composed of three metric groups, Base, Temporal, and Environmental, each consisting of a set of metrics. The Base metric group represents the intrinsic characteristics of a vulnerability that are constant over time and across user environments. It is composed of two sets of metrics: The Exploitability metrics and the Impact metrics.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Exploitability Metrics | Values | | | |
| Access Vectors | Local | Adjacent Network | | Network |
| Attack Complexity | High | Medium | | Low |
| Privilege Required | None | Low | | High |
| User Interaction | None | | Required | |
| Scope | Unchanged | | Changed | |

|  |  |  |  |
| --- | --- | --- | --- |
| Impact Metrics | Values | | |
| Confidentiality | None | Low | High |
| Integrity | None | Low | High |
| Availability | None | Low | High |

## Details of OWASP Vulnerabilities

|  |  |
| --- | --- |
| Injection | Injection flaws, such as SQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker’s hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data. |
| Broken Authentication | Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently. |
| Sensitive Data Exposure | Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser. |
| XML External Entity (XXE) | Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks. |
| Broken Access Control | Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc. |
| Security Misconfiguration | Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion. |
| Cross-Site Scripting (XSS) | XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites. |
| Insecure Deserialization | Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks. |
| Using Components with Known Vulnerabilities | Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts. |
| Insufficient Logging & Monitoring | Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring. |

For more details, please refer to the [official OWASP website.](http://www.owasp.org)

## Best Practices for Input / Output Data Validation

A server can receive input from a user and consequently provide data as well. Both of these forms of data can be malicious and controls should be implemented both on the client and server side to validate the data to close all vulnerabilities in an application. There are three main models to consider about when designing a data validation strategy. Irrespective of the model used, data validation methods should check Data Type, Syntax and Length.

* **Accept Only Known Valid Data - Recommended**

A character set may be defined for each field where input from the user is accepted. E.g. “A-Z, a-z, @,., 0-9, \_” is a character set for a field that accepts user email.

* **Reject Known Bad Data**

A character set of bad data may be defined for the site that has to be rejected. E.g. “CREATE, DROP, OR”

* **Sanitize Known Bad Data**

A character set of bad data is defined and any input field that has such a character is modified. E.g. “If there is a single quote (‘) in the data, it is replaced with two single quotes.”

Further all the allowed input/output data must be sanitized on the server side by replacing scripts tags, sent as part of user input/output, with appropriate representations. For example, “**<**” by **&lt**;“**>**” by**&gt;**“**(**“ by **&#40**etc. This would avoid scripts from being executed on the client side.

Client input must also be checked for URL encoded data. URL encoding, sometimes referred to as percent encoding, is the accepted method of representing characters within a URI that may need special syntax handling to be correctly interpreted. This is achieved by encoding the character to be interpreted with a sequence of three characters. This triplet sequence consists of the percentage character **“%”** followed by the two hexadecimal digits representing the byte code of the original character. For example, the [US-ASCII character set](http://en.wikipedia.org/wiki/ASCII) represents a space with byte code 32, or hexadecimal 20. Thus its URL-encoded representation is %20. Attackers can use this method of encoding to hide popularly known malicious strings like *“script”*, *“select”* etc.

Other common characters that can be used for malicious purposes:

|  |  |
| --- | --- |
| **Character** | **URL encoded** |
| ‘ | %27 |
| “ | %22 |
| ; | %3b |
| < | %3c |
| = | %3d |
| > | %3e |
| ) | %29 |
| ( | %28 |
| space | %20 |

All input validation checks should be completed after the data has been decoded and validated as acceptable content (e.g. maximum and minimum lengths, correct data type, does not contain any encoded data, textual data only contains the characters a-z and A-Z etc.)

# Glossary

|  |  |
| --- | --- |
| Abbreviation | Details |
| **ARIN** | American Registry of Internet Numbers. This is the primary governing body that regulates Internet IP addresses. Other similar registries include APNIC and RIPE. |
| **CVSS** | Common Vulnerability Scoring System |
| **CWE** | Common Weakness Enumeration |
| **DoS** | Denial of Service. DoS is a specific type of application attack which can make servers and/or routers crash and typically results in an application outage. |
| **DNS** | Domain Name System/Service. A protocol used on the Internet for translating hostnames into Internet addresses. For example, DNS is the service that would translate www.google.com into the IP address 216.239.57.104. DNS is basically a phone book for the Internet. |
| **Domain Name** | Strings of alphanumeric characters used to name/identify computers, applications, and organizations on the Internet. |
| **Exploit** | A vulnerability in software or computer configurations that can be used for breaking security or otherwise attacking an Internet host over the application. |
| **Fingerprint** | To identify by means of a distinctive mark or characteristic. For example, fingerprints are used to remotely identify which services, servers, operating systems, etc... That are running on any application. |
| **Hacker** | A person who enjoys exploring the details of programmable systems and how to stretch their capabilities, as opposed to most users, who prefer to learn only the minimum necessary. Many times the term is also used to describe a person who breaks into computer systems and/or applications. |
| **OWASP** | Open Web Application Security Project |
| **PT** | Penetration Test |
| **Port Scan** | The process of examining a group of ports on a computer to determine which ones are active. A port scan does not identify which applications/services are running on a computer, what any active ports are used for, or any security threats on the computer. It only determines which ports are active. |