

Message-Level Security WS 2025/26

Exercise 01

Ruhr University Bochum Faculty of Computer Science
Chair for Network and Data Security

Beginning of Exercise: Monday 13th October, 2025, 14:00
Processing Time: 14 days
Submission Deadline: Monday 27th October, 2025, 14:00

Remarks

- ▶ The new exercises will be published on **every second Monday** before the lecture.
- ▶ The solutions must be submitted via Moodle. Please answer the questions on the first page of the Moodle test.
- ▶ You may work together in groups with **two** other persons. Each person in the group must submit the Moodle test **individually** and provide the answers to the first page of the Moodle test. Within each group, only **one** person submits the solutions of the exercise in the Moodle test.
- ▶ Please note the announcements and forums via Moodle due to possible postponements and failures of the lecture or exercise.

Exercises

[30 P]	Task 1: Web APIs: IP Address to Weather	2
[10 P]	Task 2: JSON: Parsing	2
[5 P]	Task 3: JSON: Web Tokens	4
[5 P]	Task 4: JSON: Buggy JWTs	4
[30 P]	Task 5: JSON: Parsing with JavaScript	5
[15 P]	Task 6: ID Token: JWT Headers	5
[28 P]	Task 7: ID Token: JSON Web Signature Security	5

Task 1 Web APIs: IP Address to Weather..... Σ 30 P

Your task is to use existing REST APIs to retrieve the current weather at your location, based on your IP address. You must use the list of REST APIs on Github¹.

- You must use Bruno², which is supported by all common operating systems and architectures (i.e., ARM and x86). Please export and submit your collection in the **Postman format**³. You must work with collection variables⁴ to pass data between requests.
- Alternatively, you can use Postman⁵ and submit your Postman collection⁶. In industry, Postman is considered as the de-facto standard tool for API testing. Note that, unfortunately, Postman requires you to register a free account. Thus, using Postman is optional and at your own discretion. You must work with collection variables⁷ to pass data between requests.

Your goal is to fulfill the following tasks:

- (10 P) (a) **Retrieve IP Address:** Create a REST call to retrieve your computer's IP address. Store the IP address in a collection variable: `ip_address`.
- (10 P) (b) **IP Address to Geolocation:** Create a REST call to retrieve geolocation information (latitude, longitude) for your IP address. Store the geolocation in two collection variables: `geo_ip_lat` and `geo_ip_lon`.
- (10 P) (c) **Geolocation to Weather:** Create a REST call to retrieve the current weather information for the geolocation. You must use the variables `geo_ip_lat` and `geo_ip_lon` to retrieve the information from the OpenMeteo API⁸.

Rules:

- You must provide all REST calls in your collection.
- Use the REST APIs on <https://github.com/public-apis/public-apis>.
- No other tools are allowed to be used.
- All variable names must match.
- You must use a different service (i.e., domain) in each step of the exercise.
- You are not allowed to setup your own REST API servers.
- All REST calls must be *free* and no API key must be required.

Task 2 JSON: Parsing..... Σ 10 P

Decide whether the following code is *valid* JSON. In case that it is invalid, provide an additional reason why it is invalid. There is no need to provide a reason if it is valid.

¹<https://github.com/public-apis/public-apis>

²<https://www.usebruno.com/>

³<https://docs.usebruno.com/get-started/import-export-data/export-collections>

⁴<https://docs.usebruno.com/scripting/vars>

⁵<https://www.postman.com/>

⁶<https://www.postman.com/collection/>

⁷<https://learning.postman.com/docs/sending-requests/variables/variables/>

⁸<https://open-meteo.com/en/docs>

(1 P) (a) Is the following code *valid* JSON?

```
1 { "type": "exercise", "running": True, "tasks": [1, 2, 3]}
```

☐ Yes. ☐ No. ☐ Reason: _____

(1 P) (b) Is the following code *valid* JSON?

```
1 [ null ]
```

☐ Yes. ☐ No. ☐ Reason: _____

(1 P) (c) Is the following code *valid* JSON?

```
1 'test'
```

☐ Yes. ☐ No. ☐ Reason: _____

(1 P) (d) Is the following code *valid* JSON?

```
1 "test" // this is a comment
```

☐ Yes. ☐ No. ☐ Reason: _____

(1 P) (e) Is the following code *valid* JSON?

```
1 -1
```

☐ Yes. ☐ No. ☐ Reason: _____

(1 P) (f) Is the following code *valid* JSON?

```
1 "\u003A\u0044"
```

☐ Yes. ☐ No. ☐ Reason: _____

(1 P) (g) Is the following code *valid* JSON?

```
1 1.337E-0
```

☐ Yes. ☐ No. ☐ Reason: _____

(1 P) (h) Is the following code *valid* JSON?

```
1 01.337e-1
```

☐ Yes. ☐ No. ☐ Reason: _____

(1 P) (i) Is the following code *valid* JSON?

1 [{ } , { }]


☐ Yes. ☐ No. ☐ Reason: _____

(1 P) (j) Is the following code *valid* JSON?

1 [[{1337: "https://www.youtube.com/watch?v=dQw4w9WgXcQ"}]]

☐ Yes. ☐ No. ☐ Reason: _____

Task 3 JSON: Web Tokens Σ 5 P


The following JWT is given in `json-web-token.jwt` . Decode and analyze the JWT to answer the following questions:


(2 P) (a) Which cryptographic algorithm identifier is used to sign/verify the JWT? Explain the meaning behind it.

(3 P) (b) You know that after the verification of the JWT's signature, the implementation (business logic) ignores the `gid` parameter. Is it possible to change it without invalidating the signature? Justify your answer.

Task 4 JSON: Buggy JWTs Σ 5 P

The following JWTs are given. During the signing process something went wrong. Decode and analyze the JWT to find the error in each of the tokens.

(2 P) (a) Analyze the given JWT in `buggy-jwt-1.jwt`  and describe the security issue.

(3 P) (b) Analyze the given JWT in `buggy-jwt-2.jwt`  and describe the security issue.

Task 5 JSON: Parsing with JavaScript Σ 30 P

- (30 P) (a) Implement and submit an HTML file with inline JavaScript that:
- ▶ Downloads the JSON document deployed on <https://accounts.google.com/.well-known/openid-configuration> with the Fetch API.
 - ▶ Parses the JSON document with a proper JavaScript function.
 - ▶ Prints out the `authorization_endpoint` URL, all supported scopes, and all supported claims in the developer console.

Task 6 ID Token: JWT Headers..... Σ 15 P

Please answer the following questions regarding JSON Web Tokens. Good starting points for your analysis might be the JWT RFC, JWS RFC, JWK RFC, and JWA RFC.

- (2 P) (a) Name the symmetric algorithm for integrity protection with the highest security that is supported by JSON Signature.
- (a) _____
- (2 P) (b) You want to use the ECDSA algorithm with P-384 and SHA-384. To which value do you have to set the “alg” parameter?
- (b) _____
- (3 P) (c) Provide **all** seven names of the JWS Header parameters that can be used for key identification.
- (c) _____
- (3 P) (d) Explain the use of the “crit” Header in your own words.
- _____
- _____
- _____
- _____
- (5 P) (e) You want to include a JWS *Unprotected* Header in your JWT. Which JWS Serialization do you have to choose and why?

Task 7 ID Token: JSON Web Signature Security Σ 28 P

A JWT verification service is deployed on:

- ▶ <https://json.e-hacking.de/json-sec/sigenc/>
(Tab: Attacking JSON Signature)

We implemented different JWS verifier that you have to bypass. You find a signed JWT on this website. Select the correct verifier for each task.

Useful Tools: JWTF⁹, JWT.io¹⁰, JWT Encoder/Decoder¹¹ (less restrictive), PEM-JWK Converter¹², JWK-PEM Converter¹³, Online RSA Key Generator¹⁴, Base64Url Encoder/Decoder¹⁵.

- (a) **Bypassing JWT Signature Validation:** Your goal is to manipulate the given JWT and bypass the verifiers by changing:

► the iban to DE66 6666 6666 6666 66

You know that the JSON Web Key  used to sign the message is:

```

1 {
2   "kty": "RSA",
3   "e": "AQAB",
4   "use": "sig",
5   "kid": "ced2a065-9f53-4c22-b872-702da906fd04",
6   "n": "iYJqkyG6j8AtavEVg8UDWhal3ICVxchgWWVBgoVdnz_xiftm-YDXR61D
7       rYb-FJvub1gy2ctpwG-NUU8TzKPJ2PgFL3YG3tanldXm-jLooSgLPsgD
8       G2MsEksWTo_iqjVJJ8oGXVec5HKQU90gcqBQPPMVjZ38BD9fkbpaYN-X
9       C6T0Cb5N5Kb04kiZvuf6dtrMEtlrThqtUsFexPNWz0GBSq9ZMEiZ5n8L
10      c4vUV2tDXzo0nTVAc_GIyGkUWQQ0go9cEUqoEQ693Kq89Fk2Wk_dCgS0
11      8n_9NQnFixcJucs9JND2j5jHJ5jmOfcJv0aG1H1J0aV9kHchHP77xQt1
12      u_p2Qw"
13 }
```

You see a flag once the task is solved and if the JWT is successfully verified despite the manipulations made. Provide the JWT you used to bypass the corresponding verifier.

- (3 P) i. Solution to bypass Verifier 2.

- (5 P) ii. Solution to bypass Verifier 3.

- (10 P) iii. Solution to bypass Verifier 5.

⁹<https://jwt.wtf/> – JWTF is operated by Hackmanit and unifies the functionality of multiple existing tools into a single interface. It also offers a set of preconfigured attacks for convenient testing. We therefore recommend using JWTF over the other tools. However, as the project is still in an early stage, some bugs or open features may remain. If you encounter any issues, please consider reporting them.

¹⁰<https://jwt.io/>

¹¹<https://irrte.ch/jwt-js-decode/index.html>

¹²<https://irrte.ch/jwt-js-decode/pem2jwk.html>

¹³<https://8gwifi.org/jwkconvertfunctions.jsp>

¹⁴<https://travistidwell.com/jsencrypt/demo/>

¹⁵<https://www.rfctools.com/base64url-encoder/>

(10 P)

iv. Solution to bypass Verifier 6.
