Python Flask Vulnerabilities

1. **Cross-Site Scripting (XSS)**

Stored XSS Attack: Malicious content is injected to the server of the application by inputting scripts in somewhere like search box. Every time the user opens the website, the malicious content will be executed.

Reflected XSS Attack: The attacker can add the malicious content to the URL and send the fake link to the users via e-mails, messages or some websites. When the user opens the link and submits the specially crafted contents, the malicious script can steal the input and call particular interface to attack the server.

For DOM-Based and Reflected XSS, I think it is executed firstly by the client or the browser, thus doesn’t belong to python flask vulnerability.

1. **Server-Side Template Injection (SSTI)**

I think this is similar to XSS, but only happens to the servers instead of attacking the client first. It happens when the user input is entered to the template directly without being filtered. As a result, the attacker can input some malicious commands to manipulate the templates so that the whole server can be controlled directly. This kind of attack can be fatal to the server, which can cause Remote Code Execute (RCE) and taking advantage of other flaws on the server for further attacks.

1. **Denial of Service (DoS), Distributed Denial of Service (DDoS)**

DoS attack aims to make the web services unavailable to the users instead of taking use of security flaws. DDoS is a typical DoS attack which can generate a huge amount of traffic from many machines to make the server crash.

There’re 2 common types of DoS vulnerabilities:

1. High CPU/Memory Consumption. The attacker can send crafted requests that could cause the system to take disproportionate resources to process.
2. Crash. The attacker can send specially crafted value to make the server crash.
3. **Arbitrary Code Execution (ACE)**

When we use flask-ipban package to protect against DoS through IP banning, this attack can happen via yaml.load in the code. To fix this vulnerability, we need to upgrade the package to the versions higher than 0.2.2.

I didn’t find too much useful information about this. Sorry.

1. **Access Restriction Bypass**

Flask-AppBuilder is a development framework built on top of Flask. Owing to an improper authentication vulnerability in the REST API, it allows for a malicious actor with a carefully crafted request to successfully authenticate and gain access to existing protected REST API endpoints. This only affects non-database authentication types and new REST API endpoints. Users should upgrade to Flask-AppBuilder 3.3.4 to receive a patch.

1. **Deserialization of Untrusted Data**

Deserialization of untrusted data is when the application deserializes untrusted data without sufficiently verifying that the resulting data will be valid, letting the attacker to control the state or the flow of the execution. An attacker just needs to identify a piece of software that has both a vulnerable class on its path, and performs deserialization on untrusted data. Then all they need to do is send the payload into the deserializer, getting the command executed.

1. **CVE-2021-33026**

The Flask-Caching extension up to and including 1.10.1 for Flask relies on Pickle for serialization, which may lead to remote code execution or local privilege escalation. If an attacker gains access to cache storage like filesystem, they can construct a crafted payload, poison the cache, and execute Python code.

1. **CVE-2015-5306**

OpenStack Ironic Inspector (aka ironic-inspector or ironic-discovered), when debug mode is enabled, might allow remote malicious users to access the Flask console and execute arbitrary Python code by triggering an error.

There’re still a number of vulnerabilities like Cross-site Request Forgery, Dictionary Traversal, Improper Input Validation, Incorrect Default Permissions and Open Redirect. However, I didn’t do too much study on them limited by time.

As for the vulnerabilities that happens most frequently, I think they should be XSS, SSTI and DoS.