Часть 1: проектирование

Ссылка на первую часть задания:
https://drive.google.com/file/d/1znqtVGQZL0wO5wbsUEY5aJfyr2F9HuH_/view?usp=sharing

1. Изучите архитектуру систему и сценарий ее работы по ссылке.
2. Скопируйте файл задания с помощью File > Make a Copy.
3. Используя сценарий работы, смоделируйте негативные сценарии с помощью https://plantuml.com/sequence-diagram или другого редактора UML-диаграмм.
4. Определите, в каких случаях какие цели безопасности могут быть нарушены, и какие меры можно предпринять.
5. Разработайте политику архитектуры – декомпозируйте систему и определите типы доменов безопасности.
6. Всё решение поместите в свою копию файла задания.

Цели безопасности

1. Только авторизованные пользователи могут взаимодействовать со спутником.
2. Пользователи могут выполнять в своих программах только разрешенные операции.
3. Спутник не делает фотографии запрещенных районов.
4. Орбита спутника всегда остается в рамках заданных ограничений.

Предположения безопасности

1. Оборудование спутника (симулятор) работает в штатном режиме.
2. Обслуживающий персонал системы благонадежен.

Часть 2: прототипирование

Скачайте проект:
https://disk.yandex.ru/d/Dx8mUsoSlkT2oQ

1. Изучите код в папке system и satellite_simulator, а также примеры использования в папке example и корне проекта. Установите необходимые библиотеки (matplotlib и numpy).
2. Реализуйте функционал недостающих модулей (№1-6) согласно примерам (example_1.py). Поместите разработанные модули в папку satellite_control_system, не редактируйте код симулятора или код в папке system.

   Для удобства уже реализованы классы OpticsControl и OrbitCotrol, но в них не хватает функционала. Доработайте его.

В каждом новом блоке нужно по аналогии с примерами реализовать методы run(self) и _check_events_q(self). Первый метод описывает логику процесса, а второй проверку сообщений от других процессов.

Также не забудьте вызвать конструктор базового класса с помощью super().__init__().

Обратите внимание, что в методе run необходимо вызвать 2 метода
_check_events_q() – проверка событий от других модулей.
_check_control_q() – проверка системных событий из базового класса.

Для логов используйте метод _log_message(), реализованный в базовом классе процесса.

3. Реализуйте меры защиты, которые позволят соблюсти цели безопасности.
4. Реализуйте монитор безопасности (см. example_2.py) и политики безопасности для контроля потоков данных согласно примеру.
5. Продемонстрируйте работу программы – попытайтесь задать данные, нарушающие цели ЦБ или скомпрометируйте работу недоверенных доменов. Система должна не допустить нарушения ЦБ.

Воспрещается редактировать код симулятор спутника и базовые классы, предоставленные в проекте. Необходимо наследовать базовые классы, предложенные в программе (BaseCustomProcess)

Синтаксис программ пользователя

Программы должны храниться в текстовом файле и могут иметь следующие команды:

- ORBIT <altitude> <raan> <inclination> – выйти на орбиту на высоту (метры), с RAAN (радианы) и с inclination (радианы). Спутник считает новую орбиту, перемещается на ближайшую точку на этой орбите и продолжает движение с нее.
- MAKE PHOTO – создать снимок текущей точки на поверхности земли. Информация о снимке помещается в хранилище данных и отображается на карте.
- ADD ZONE <id> <lat1> <lon1> <lat2> <lon2> – создать зону с ограничениями, в которой запрещены снимки. Id – уникальный идентификатор зоны, (lat1,lon1) – широта и долгота нижней левой точки зоны, (lat2,lon2) – широта и долгота верхней правой точки зоны. Id задается в int, а координаты во float. Если зона с таким id уже есть, то она не создается.
- REMOVE ZONE <id> – удалить зону с указанным id.

У пользователя может быть 3 вида прав:

- Создание снимков.
- Корректировка орбит.
- Редактирование ограничений на снимки.