Сложный способ не работать(триггеры FM своими руками)

Знаем много о сети, однако NOC пока только коллекционирует события, информацию об оборудовании... Иногда залазим и запускаем массовое изменение настройки. Знакомо?

Исправим это!

Дано: Железки, которые глючат, но у них хватает самодиагностики на то, чтобы об этом сказать. Например UniFi - точка доступа для WiFi хотспота. Периодически может отвалиться wifi-контроллер, о чем точка расскажет в сислог: **Nov 22 15:09:56 kernel: [977222.991000] NETDEV WATCHDOG: wifi0: transmit timed out**

Отлично, точка при этом доступна на управление по меди, но перестает всех пускать по вафле.

Лечение: перезагрузка точки доступа.

Требования:

1. NOC про точку должен знать.
2. Точка должна слать сислоги NOC-у.
3. Событие о глюке должно классифицироваться.

Первые два пункта реализуются просто, не будем на них смотреть. Смотрим на третий.

Опачки, а классифицировать не можем, нет класса событий. Ну мы не деревенские дурачки, создадим.

Идем в **NOC/fm/collections/eventclasses/**

Лезем в каталог Vendor и создаем там свою иерархию. Точка производится Ubiquiti, платформа UniFi, класс Network. Да, это имена каталогов, которые надо создать. Вот собственно в **Vendor/Ubiquiti/UniFi/Network** мы положим файлик **WIFI.json** следующего содержания:

```
[
    {
        "name": "Vendor | Ubiquiti | UniFi | Network | WiFi controller hung",
        "description": "WiFi controller hung on AP",
        "action": "A",
        "vars": [
            {
                "name": "port",
                "description": "WiFi port",
                "type": "str",
                "required": false
            }
        ],
        "text" : {
            "en": {
                "subject_template": "UniFi WiFi controller hung",
                "body_template": "In some situations WiFi controller of the AP hangs.",
                "symptoms": "Specific messages in syslog from AP, AP works, but noone can connect to its wifi
networks",
                "probable_causes": "No specific causes found",
                "recommended_actions": "Reboot the AP either automaticaly or manualy"
            }
        }
    }
]
```

Перезапускаем NOC.

У нас появился класс, отклассифицируем событие(расписывать не буду), перезапускаем ***noc-classifier***(можно прибить процесс, ланчер его запустит снова).

Правило назовем **Ubiquiti | AirOS | Vendor | Ubiquiti | UniFi | Network | WiFi controller hung #1 (SYSLOG)**

Дальше идем в вебморде в *FM/Setup/Event Triggers*

Попутно заглядываем в *Sa/Setup/Object selectors* и создаем селектор для тестовой точки, назовем селектор **TestReboot**.

Итак, триггер. Нету? создаем. Назовем его **UniFi AP Stuck trigger**

Заполняем поля...

Триггер активен - **ДА**

Event Class RE: WiFi controller hung

Condition: true

MO Selector: TestReboot

pyRule: Reboot_UnifiAP

Стоп... Какая pyRule?? Надо ее создать!

Лезем в *Main/Setup/pyRule* и создаем эту pyRule!

Код там простой, если не занимаешься программированием... 🙂 Главное - не ошибиться в имени и поставить правильный интерфейс - **IEventTrigger**

```
# -*- coding: utf-8 -*-
##----------------------------------------------------------------------
## Reboot UniFi
##----------------------------------------------------------------------
## INTERFACE: IEventTrigger
##----------------------------------------------------------------------
## DESCRIPTION:
## Reschedule device configuration to fetch
## Used in event trigger for Config | Config Changed event class
##----------------------------------------------------------------------
## Copyright (C) 2007-2011 The NOC Project
## See LICENSE for details
##----------------------------------------------------------------------

## Python modules
import datetime
## NOC modules
from noc.sa.models import ReduceTask

TO = 300 # TimeOut for reboot script

@pyrule
def reboot_unifi(event):
    # Check managed object is managed
    if not event.managed_object.is_managed:
        return
    # Create MRT
    t = ReduceTask.create_task(event.managed_object.name,"pyrule:mrt_result","","commands",{"commands":
["reboot"]}, TO)
    t.get_result()
    return
```

Что сделает этот скрипт? да ничего особенного. Создаст MRT со скриптом commands, которому отдаст единственную команду - reboot.

Все, следим за точкой доступа. Как только наблюдаем событие, проверяем через 5 минут в контроллере аптайм точки. Убеждаемся, что все работает как надо.

Что дальше? пара пустяков - изменяем селектор, чтоб в него попадали ВСЕ точки UniFi.

Идем пить минералку. Изредка проверяем, что все ребутится как надо, потому что эта гадость может и зависнуть при ребуте.