Reporte: Modificaciones del extractor CICFlowmeter con GOOSE

Santiago Rios Guiral
santiago.riosg@udea.edu.co

Estudiante de maestría en ingeniería de telecomunicaciones
Grupo de investigación GITA
Universidad de Antioquia
2022

1 Introducción

En este reporte se presenta de forma general, las actividades realizadas para el modificar el extractor de tráfico CICFlowMeter [1] con el objetivo de capturar tráfico de tipo GOOSE (Generic Object Oriented Substation Event) definido por el estándar IEC61850, el cual regulariza el proceso de automatización en las subestaciones eléctricas. Esta nueva implementación del extractor GOOSE permite convertir el tráfico GOOSE proveniente de un archivo pcap a un archivo CSV. La posibilidad de contar con el tráfico capturado se presta para desarrollar un estudio del tráfico en las subestaciones electricas. Para alcanzar el objetivo se tomo como base el extractor CICFlowMeter [2] y la aplicación Goosestalker [3] que establece un formato de paquete GOOSE.

2 Tráfico GOOSE

IEC 61850 es un estándar para la automatización de subestaciones eléctricas. En este estándar se define el tráfico GOOSE. Los eventos de subestación orientados a objetos genéricos (GOOSE), son un mecanismo de modelo controlado en el que cualquier formato de datos (estado, valor) se agrupa en un conjunto de datos y se transmite dentro de un período de tiempo definido.

- Los datos de GOOSE se integran directamente en la trama Ethernet y funcionan en el mecanismo de publicador-suscriptor donde el tráfico GOOSE se envía desde un publicador a varios suscriptores con una dirección MAC multidifusión.
- GOOSE utiliza VLAN de acuerdo al estándar IEEE 802.1Q. De esta forma se separan los dominios del tráfico GOOSE dentro de una red física y así establece el nivel de prioridad de mensaje apropiado.

3 CICFlowMeter

CICFlowMeter es una herramienta desarrollada por el instituto de ciberseguridad de Canadá. Permite generar y analizar los flujos de tráfico en una red de computadores. Está orientado al análisis y la clasificación de tráfico con el objetivo de ofrecer herramientas capaces de discriminar el tráfico normal de las anomalías o ataques. CICFlowMeter tiene 2 modalidades de trabajo, la primera es mediante la lectura de archivos pcap

y la segunda mediante la captura en vivo. El archivo de salida esta formado por los flujos extraídos con las características establecidas en la implementación.

3.1 Instalación del extractor

```
https://github.com/SantiagoGuiral/CICflowmeter-goose.git
cd cicflowmeter
python3 setup.py install
```

3.2 Ejecución del extractor

CICFlowMeter permite capturar y analizar el tráfico en 2 modalidades, mediante la lectura de capturas de tráfico o desde una interfaz de red.

```
usage: cicflowmeter [-h] (-i INPUT_INTERFACE | -f INPUT_FILE) [-c] [-u URL_MODEL] output

positional arguments:
  output                output file name (in flow mode) or directory (in sequence mode)

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT_INTERFACE    capture online data from INPUT_INTERFACE
  -f INPUT_FILE         capture offline data from INPUT_FILE
  -c, --csv, --flow     output flows as csv
```

Para analizar paquetes provenientes de un archivo pcap se utiliza el siguiente comando:

```
cicflowmeter -f example.pcap -c flows.csv
```

Para capturar paquetes en la interfaz de red, se debe utilizar el siguiente comando:

```
cicflowmeter -i eth0 -c flows.csv
```

3.3 Análisis del extractor y modificaciones

Inicialmente se realizó un análisis del extractor y que tipo de paquetes permite capturar. Se encontró que este utiliza la librería de Python `Scapy` para capturar el tráfico de red y el énfasis se centra en tráfico que trabaja con los protocolos TCP y UDP de la capa de transporte. En el caso del proyecto de ciberseguridad, se requiere trabajar con tráfico GOOSE, el cual trabaja a nivel de capa de capa enlace, por lo que se requieren varias modificaciones para capturar un tráfico en especifico (GOOSE), y a su vez definir los flujos que se van a almacenar en el archivo csv.

4 Goosestalker

Goosestalker es una aplicación que tienen como objetivo analizar e interactuar con tráfico tipo GOOSE definido por el estándar IEC61850. La aplicación esta compuesta por diversos módulos que se encargan de analizar el tráfico de la red para extraer las comunicaciones del estándar IEC61850 y ofrecer un mecanismo para interpretar con mayor facilidad este tipo de tráfico de red. De los diferentes archivos que componen Goosestalker, el interés se centra en el modulo llamado `goose_parser`, el cual se encarga de analizar el paquete y extraer la información del PDU de GOOSE.

5 Combinando CICFlowMeter con GOOSE

En el proyecto de ciberseguridad del grupo de investigación GITA se requiere la capacidad de capturar tráfico GOOSE y convertirlo a flujos con el objetivo de analizar el comportamiento del tráfico. Para esto se busca la combinación de un extractor, en especifico CICFlowMeter y Goosestalker. En esta sección se explica

como se utilizaron ambas aplicaciones para desarrollar un extractor que trabaja en capa de enlace, extrae la información del PDU de GOOSE y genera unos parámetros del tráfico para su posterior análisis.

En la figura 1 se presenta el diagrama definido para el extractor modificado. La idea se centra en leer el tráfico desde un archivo pcap y agregar los paquetes GOOSE en flujos. CICFlowMeter se utiliza para leer el archivo pacap, generar los flujos, extraer diferentes características del tráfico y construir el archivo csv. Goosestalker se utiliza para extraer la información del PDU de GOOSE y agregarlas dentro de los flujos.

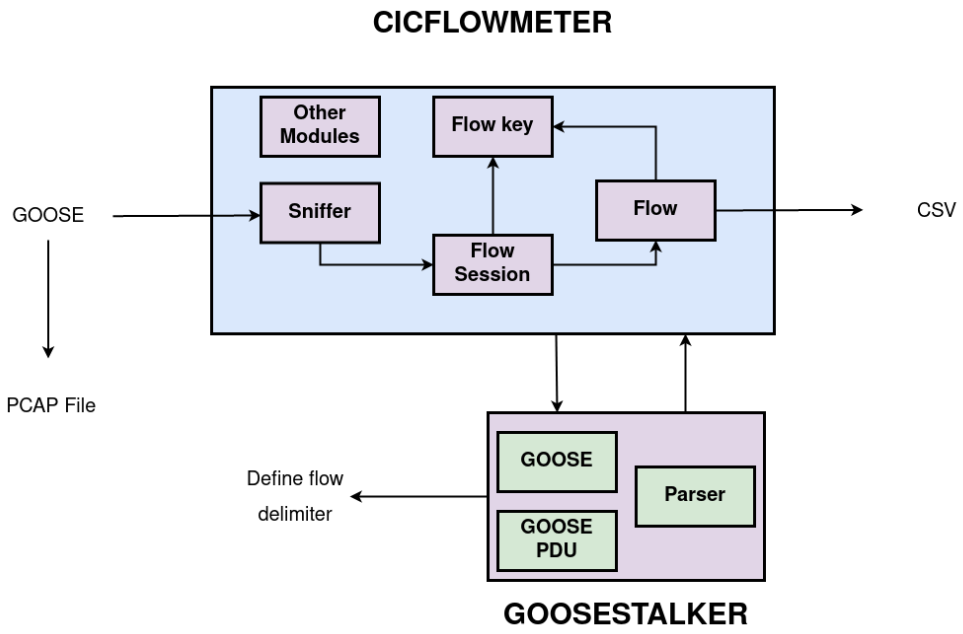

Figure 1: Diagrama del extractor CICFlowMeter con GOOSE

Inicialmente se realizó un análisis profundo del comportamiento del extractor CICFlowMeter y Goosestalker. A partir de esté estudio, se definió el proceso a seguir para combinar su implementación. A continuación se presentan un resumen de las etapas trascendentales que se siguieron para obtener un extractor con la capacidad de analizar tráfico GOOSE.

- En el archivo `sniffer` se modifico el filtro BPF utilizado por Scapy para capturar el tráfico. El filtro se modifico de tal forma que permita la captura de paquetes identificador con el tipo GOOSE en la trama Ethernet. EL filtro queda así: `(ether proto 0x88b8) or (vlan && ether proto 0x88b8)`.
- En el archivo `flow_session` se eliminaron todas las restricciones que impedían trabajar con paquetes GOOSE.
- Se definió un valor para el tamaño de la ventana de tiempo para definir los flujos de tráfico.
- En el archivo `packet_flow_key` Se definieron los parámetros del paquete GOOSE que serán utilizados para identificar el paquete en un flujo de red.
- El identificador de flujo esta basado en los valores de: MAC origen, MAC destino, appid, gocbRef, stNum. Estos son valores que se extraen del paquete GOOSE.
- En el archivo `flow` se eliminaron todas las características que están relacionadas a la capa de transporte y la capa de red ya que GOOSE opera en capa de enlace.

- En `flow` también se eliminaron las características del flujo referentes a la comunicación unidireccional en reversa. Esto debido a que el estándar IEC61850 para GOOSE define un modelo de operación publicador-subscriptor.
- En `flow` se agregaron las características que se pueden extraer del PDU de GOOSE para que estan se guarden en el flujo dentro del archivo csv.
- Se agrego un directorio llamado `goose` el cual esta conformado por los siguientes archivos: `goose`, `goose_pdu` y `goose_data`.
- El archivo `goose` se encarga de estructurar el paquete GOOSE con los campos establecidos dentro del estándar. Para esto se utiliza el formato creado en Goosestalker y la librería Scapy.
- El archivo `goose_pdu` estructura los campos que se encuentran en la carga útil del paquete GOOSE. Estos campos del PDU, son utilizados para establecer el identificador de flujo y para almacenarlos en el archivo csv.
- El archivo `goose_data` se creo con el objetivo de extraer la información del paquete GOOSE. se crearon varias funciones que permiten obtener los campos del PDU y obtener valores de la carga útil.
- Además, en el archivo `flow` se organizo la clase `Flow` para trabajar con el tráfico GOOOSE. Para esto se modifico el constructor, al agregar parámetros de los campos del PDU y se agregaron estructuras para guardar características del flujo.
- Se agregaron parámetros estadísticos para complementar los parámetros calculados por CICFlowMeter.

6 Resultados

En las modificaciones del extractor CICFlowMeter se logró obtener capturar y extraer el tráfico GOOSE. En la figura 2 se presenta una muestra del archivo csv generado. Se logra observar como se obtuvo la identificación de los flujos a partir de la lectura de un archivo pcap construido a partir de tráfico GOOSE generado en la Universidad Nacional sede Medellín.

7 Referencias

[1] CICflowmeter-goose. Repositorio de Github. Disponible en: CICflowmeter-goose .

[2] Python CICFlowMeter. Repositorio de Github. Disponible en: Python CICFlowMeter

[3] GooseStalker. Repositorio de Github. Disponible en: GooseStalker

[4] Instituto de ciberseguridad de Canada. CICFlowMeter (formerly ISCXFlowMeter). Disponible en: CICFlowMeter (formerly ISCXFlowMeter)

src_mac	dst_mac	appid	gocbRef	stNum
00:21:c1:24:90:a0	01:0c:cd:01:00:01	2	AA1J1Q01A2LD0/LLN0GOgcbGOOSEFROM_615	3
00:21:c1:24:85:86	01:0c:cd:01:00:02	2	AA1J1Q01A1LD0/LLN0GOgcbprueba620	3
00:21:c1:24:85:86	01:0c:cd:01:00:02	2	AA1J1Q01A1LD0/LLN0GOgcbprueba620	4
00:21:c1:24:90:a0	01:0c:cd:01:00:01	2	AA1J1Q01A2LD0/LLN0GOgcbGOOSEFROM_615	4
00:21:c1:24:85:86	01:0c:cd:01:00:02	2	AA1J1Q01A1LD0/LLN0GOgcbprueba620	5
00:21:c1:24:90:a0	01:0c:cd:01:00:01	2	AA1J1Q01A2LD0/LLN0GOgcbGOOSEFROM_615	5
00:21:c1:24:85:86	01:0c:cd:01:00:02	2	AA1J1Q01A1LD0/LLN0GOgcbprueba620	6
00:21:c1:24:90:a0	01:0c:cd:01:00:01	2	AA1J1Q01A2LD0/LLN0GOgcbGOOSEFROM_615	6
00:21:c1:24:85:86	01:0c:cd:01:00:02	2	AA1J1Q01A1LD0/LLN0GOgcbprueba620	7
00:21:c1:24:90:a0	01:0c:cd:01:00:01	2	AA1J1Q01A2LD0/LLN0GOgcbGOOSEFROM_615	7
00:21:c1:24:85:86	01:0c:cd:01:00:02	2	AA1J1Q01A1LD0/LLN0GOgcbprueba620	8
00:21:c1:24:90:a0	01:0c:cd:01:00:01	2	AA1J1Q01A2LD0/LLN0GOgcbGOOSEFROM_615	8
00:21:c1:24:85:86	01:0c:cd:01:00:02	2	AA1J1Q01A1LD0/LLN0GOgcbprueba620	9
00:21:c1:24:90:a0	01:0c:cd:01:00:01	2	AA1J1Q01A2LD0/LLN0GOgcbGOOSEFROM_615	9
00:21:c1:24:85:86	01:0c:cd:01:00:02	2	AA1J1Q01A1LD0/LLN0GOgcbprueba620	10
00:21:c1:24:90:a0	01:0c:cd:01:00:01	2	AA1J1Q01A2LD0/LLN0GOgcbGOOSEFROM_615	10
00:21:c1:24:85:86	01:0c:cd:01:00:02	2	AA1J1Q01A1LD0/LLN0GOgcbprueba620	11
00:21:c1:24:90:a0	01:0c:cd:01:00:01	2	AA1J1Q01A2LD0/LLN0GOgcbGOOSEFROM_615	11
00:21:c1:24:85:86	01:0c:cd:01:00:02	2	AA1J1Q01A1LD0/LLN0GOgcbprueba620	12
00:21:c1:24:90:a0	01:0c:cd:01:00:01	2	AA1J1Q01A2LD0/LLN0GOgcbGOOSEFROM_615	12
00:21:c1:24:85:86	01:0c:cd:01:00:02	2	AA1J1Q01A1LD0/LLN0GOgcbprueba620	13
00:21:c1:24:90:a0	01:0c:cd:01:00:01	2	AA1J1Q01A2LD0/LLN0GOgcbGOOSEFROM_615	13

Figure 2: Muestra de los flujos de tráfico GOOSE extraidos y guardados en un archivo csv