

# PARSER WIRESHARK PER CHRONICLE

Gruppo 4

Federica Di Giaimo  
Filippo Lucchesi  
Mariano Mancini



## PROBLEMA

- ? I dati di rete catturati con Wireshark sono in formati complessi
- ? Comunicazione con Google SecOps (ex Google Chronicle)
- ? Identificazione minacce nel flusso di pacchetti

## SOLUZIONE

- ! Sviluppo di un parser per convertire i dati Wireshark in JSON-UDM
- ! Integrazione efficiente con Google SecOps
- ! Selezione dei campi essenziali per l'analisi delle minacce

# WORKFLOW



# WIRESHARK



## PCAP

### Caratteristiche

- **Binario**
- Contiene tutti i dettagli dei pacchetti
- Salvataggio anche da **GUI**

### Limiti

- Utilizzabile solo con Software compatibili
- Non direttamente leggibile

## JSON

### Caratteristiche

- Formato **standard**
- Pronto da elaborare
- Salvataggio da **tshark**

### Limiti

- Salvataggio solo da **CLI**
- Di maggiori dimensioni



```
...dataflowkit.org/persons/0",  
"Lacinis Dis Parturient Institute",  
"ligula@diam.org",  
of 100 Merrill X. Dyer",  
"1",  
[  
  {  
    "id": 2323  
  },  
  {  
    "id": 2323  
  }  
]  
  
ps://scrape.dataflowkit.org/persons/1",  
": {  
  "Risus Donec Company",  
  "quet odio euerosNam.edu",  
  "00 Kato V. Weaver",  
  "6510"  
}
```

# GOOGLE SECOPS



- Servizio cloud di Google
- Analisi di numerosi dati da una serie di origini
- Rilevazione delle minacce

- Invio alla piattaforma grazie all'Ingestion API di Google
- Compatibile con formato JSON-UDM



# ESEMPIO FILE ACCETTATO DA CHRONICLE



Parametri predefiniti per specificare cattura con **Wireshark**

```
{  
  "event": {  
    "type": "NETWORK_CONNECTION",  
    "vendor_name": "Wireshark",  
    "product_name": "Wireshark PacketCapture",  
    "event_timestamp": "2025-01-27T15:14:29.464000+00:00"  
  },  
  "network": {  
    "eth": {  
      "source_mac": "00:0C:29:AB:CD:EF",  
      "destination_mac": "ff:ff:ff:ff:ff:ff"  
    },  
    "arp": {  
      "source_mac": "00:0C:29:AB:CD:EF",  
      "source_ipv4": "192.168.178.67",  
      "destination_mac": "00:00:00:00:00:00",  
      "destination_ipv4": "192.168.178.1"  
    },  
    "frame": {  
      "timestamp": "2025-01-27T15:14:29.464000+00:00",  
      "length": "42",  
      "protocols": "eth:ethertype:arp"  
    }  
  }  
},  
}
```

Conversione approfondita degli **indirizzi** IPv4, IPv6, MAC e dei **protocolli** TCP, UDP, ARP, TLS/SSL, HTTP, ICMP, DNS e MDNS.  
Sono ignorati i campi **vuoti**

Timestamp convertito in **RFC 3339**

Sono considerati i campi più importanti per l'**identificazione** di minacce

Informazioni aggiuntive

# SVILUPPO DEL PARSER



## Parser JSON → JSON-UDM

- Strada **principale**
- Alleggerisce il compito dello script Python  
→ Sola riorganizzazione struttura interna input



## Parser PCAP → JSON-UDM

- Strada alternativa
- Richiede molte risorse di computazione e temporali
- Disponibili varie librerie Python per PCAP

# ATTACCHI DA IDENTIFICARE



VULNERABILITÀ DEI  
SISTEMI E INJECTION



INTERCETTAZIONE E  
MANIPOLAZIONE DI DATI



DOS

**PARSER**  
**JSON → JSON-UDM**

# TIMESTAMP

## Conversione in RFC 3339

\*\* per **omettere** campi non presenti nell'input

Restituisce una **stringa** del tipo:  
2025-01-27T15:14:29.464000

- 2025-01-25: data (anno-mese-giorno)
- T: separatore obbligatorio
- Ora, minuti, secondi con frazione di secondo

Impostazione fuso orario  
come **UTC** (+00:00)

Conversione in **ISO 8601**  
(compatibile con RFC 3339)

```
**{"frame": {  
    **({"timestamp": convert_timestamp(frame.get("frame.time_utc"))}  
        if frame.get("frame.time_utc") else {}),  
    **({"length": frame.get("frame.len")}  
        if frame.get("frame.len") else {}),  
    **({"protocols": frame.get("frame.protocols")}  
        if frame.get("frame.protocols") else {}),  
} } if frame else {},
```

```
# Convert timestamp to RFC 3339  
def convert_timestamp(timestamp_str):  
    try:  
        dt = datetime.strptime(timestamp_str[:25], "%b %d, %Y %H:%M:%S.%f")  
        dt = dt.replace(tzinfo=timezone.utc)  
        iso_timestamp = dt.isoformat()  
    return iso_timestamp
```

# INDIRIZZI



## Indirizzi MAC (eth) e IP (IPv4 e IPv6)

```
Window
**({"eth": {
    **({"source_mac": eth.get("eth.src")} if eth.get("eth.src") else {}),
    **({"destination_mac": eth.get("eth.dst")} if eth.get("eth.dst") else {}),
}} if eth else {}),
```

I campi di **sorgente** e di **destinazione** sono utili per identificare attacchi **DoS**, nel caso in cui un indirizzo abbia inviato un numero **anomalo** di pacchetti

Un **TTL** basso potrebbe identificare un possibile attacco **DoS**

Riportare i campi della sorgente per identificare **IP spoofing** o **MAC spoofing**

```
Window
**({"ip": {
    **({"source": ip.get("ip.src")} if ip.get("ip.src") else {}),
    **({"destination": ip.get("ip.dst")} if ip.get("ip.dst") else {}),
    **({"ttl": ip.get("ip.ttl")} if ip.get("ip.ttl") else {}),
}} if ip else {}),

**({"ipv6": {
    **({"source": ipv6.get("ipv6.src")} if ipv6.get("ipv6.src") else {}),
    **({"destination": ipv6.get("ipv6.dst")} if ipv6.get("ipv6.dst") else {}),
}} if ipv6 else {}),
```

# PROTOCOLLI DI TRASPORTO

## Protocollo UDP e TCP

### Campi delle porte di sorgente e destinazione

Come nel caso degli indirizzi, questi campi sono controllati per individuare eventuali **DoS**



Window

```
**({"udp": {  
    **({"source_port": udp.get("udp.srcport")}) if udp.get("udp.srcport") else {},  
    **({"destination_port": udp.get("udp.dstport")}) if udp.get("udp.dstport") else {},  
}}) if udp else {}},
```

Anche **tcp.flags** può essere importante per individuare un attacco **DoS**, ad esempio controllando se il flag è SYN (potrebbe trattarsi di **SYN flood**) o un flag RST



Window

```
**({"tcp": {  
    **({"source_port": tcp.get("tcp.srcport")}) if tcp.get("tcp.srcport") else {},  
    **({"destination_port": tcp.get("tcp.dstport")}) if tcp.get("tcp.dstport") else {},  
    **({"flags": tcp.get("tcp.flags")}) if tcp.get("tcp.flags") else {},  
}}) if tcp else {}},
```

# PROTOCOLLI APPLICATIVI

# DNS e MDNS

- Permette a Google SecOps di filtrare i domini ed esaminare quelli più soggetti a **spoofing**

**TTL** molto bassi potrebbero essere sintomo di interventi malevoli, dove si obbliga la richiesta continua della risoluzione del nome

Indica se il pacchetto è una richiesta o una risposta.  
Permette di rilevare risposte DNS non richieste, possibile segno di **DNS spoofing**, dove un attaccante devia il client verso un servizio malevolo

```
Window
  dns["dns.flags_tree"].items(),
  dns.flags.response"})
  | print_dns(dns["dns.flags_tree"].items(),
  |           "dns.flags.response") is not None else {}),
  ].items(), "dns.qry.type"})
  | dns(dns["Queries"].items(),
  |       "dns.qry.type") is not None else {}),
  "Answers", "Answers", "dns.flags_tree"])) else {}),
```

Indica il tipo di query che il client sta richiedendo al server DNS. Potrebbe individuare **DNS tunneling**

# PROTOCOLLI APPLICATIVI

## print\_dns

L'accesso di alcuni campi **DNS** è meno diretto: è necessario iterare tra i sottocampi di alcune chiavi (nel parser: Answers, Queries e dns.flag\_tree)

```
Window
def print_dns(items, key):
    results = []
    for k, v in items:
        if isinstance(v, dict):
            result = v.get(key)
            if result is not None:
                results.append(result)
    return results if results else None
```

## DNS vs MDNS

La gestione dei campi MDNS è del tutto analoga a quella DNS. MDNS tuttavia non necessita del campo **dns.flags.response**:

- **DNS** è strutturato su un modello client-server, quindi ha la necessità di sapere se il pacchetto inviato è una richiesta o una risposta
- **MDNS** non ha una struttura centralizzata che gestisce le risposte, perciò non è essenziale distinguere i due tipi di pacchetti

# PROTOCOLLI APPLICATIVI

## Protocollo TLS

La versione del protocollo al livello record

È la versione **TLS** negoziata nell'handshake iniziale, quella realmente usata nella sessione.

Se entrambe le versioni sono obsolete, potrebbe esserne stato forzato l'uso per un **Downgrade**

**Attack**

```
Window
"tls": {
    **({ "record_version": tls["tls.record"].get("tls.record.version") }
        if "tls.record" in tls
        and tls["tls.record"].get("tls.record.version") is not None else {}),
    **({ "handshake": {
            **({ "version": print_handshake(tls["tls.record"], "tls.handshake.version") }
                if "tls.record" in tls
                and print_handshake(tls["tls.record"], "tls.handshake.version") is not None else {}),
            } if "tls.handshake" in tls["tls.record"] else {}),
        } if tls else {}),
    }
```

## Protocollo HTTP

```
Window
**({ "http": {
    **({ "host": http.get("http.host") } if http.get("http.host") else {}),
    **({ "file_data": http.get("http.file_data") } if http.get("file_data") else {}),
} } if http else {}),
```

Contiene il nome del dominio a cui il client sta facendo la richiesta **HTTP**. Utile per il monitoraggio del traffico

Contiene il payload del corpo dei messaggi inviati. Può essere utile per l'individuazione di **Cross-Site Scripting** (XSS) e **SQL Injection**

# PROTOCOLLI DI RETE

## Protocollo ARP

I campi considerati possono segnalare casi di **ARP spoofing** in caso di associazioni anomale tra indirizzi **MAC** e **IP**

```
**({"arp": {  
    **({"source_mac": arp.get("arp.src.hw_mac")}  
        if arp.get("arp.src.hw_mac") else {}),  
    **({"source_ip4": arp.get("arp.src.proto_ip4")}  
        if arp.get("arp.src.proto_ip4") else {}),  
    **({"destination_mac": arp.get("arp.dst.hw_mac")}  
        if arp.get("arp.dst.hw_mac") else {}),  
    **({"destination_ip4": arp.get("arp.dst.proto_ip4")}  
        if arp.get("arp.dst.proto_ip4") else {}),  
} } if arp else {}),
```

## Protocollo ICMP

Specifica il tipo di pacchetto **ICMP** ◦



```
**({"icmp": {  
    **({"type": icmp.get("icmp.type")}) if icmp.get("icmp.type") else {}),  
    **({"code": icmp.get("icmp.code")}) if icmp.get("icmp.code") else {}),  
} } if icmp else {}),
```

Specifica i dettagli del pacchetto. I due campi possono aiutare a capire se si è soggetti a **PING flood**

# PARSER

## PCAP → JSON-UDM

# PARSING DA PCAP

Scelta della libreria

QUALITÀ	PYSHARK	SCAPY	DPKT
FACILITÀ D'USO	MOLTO SEMPLICE, SINTASSI INTUITIVA ✓	PIÙ COMPLESSA, MA MOLTO FLESSIBILE ✗	SEMPLICE, MA MENO VERSATILE ✓
VELOCITÀ E PERFORMANCE	PIÙ LENTO (WRAPPER PER TSHARK) ✗	PRESTAZIONI INFERIORI RISPETTO A DPKT ⚠	MOLTO VELOCE, IDEALE PER GRANDI FILE PCAP ✓
FUNZIONALITÀ AVANZATE	LIMITATO ALL'ANALISI E AL PARSING DEI PACCHETTI ✗	CREAZIONE, MANIPOLAZIONE E INVIO DI PACCHETTI ✓	SOLO ANALISI, NESSUNA CREAZIONE DI PACCHETTI ✗

# PARSER CON PYSHARK

Carica il file in memoria e fornisce un **iteratore** per scorrere i pacchetti con un ciclo for

```
Window
cap = pyshark.FileCapture(pcap_file)
extracted_data = []
```

```
Window
if "ETH" in packet:
    eth_layer = packet.eth
    packet_info["source"]["mac"] = getattr(eth_layer, "src", None)
    packet_info["destination"]["mac"] = getattr(eth_layer, "dst", None)

if "IP" in packet:
    ip_layer = packet.ip
    packet_info["network"]["protocol"] = getattr(ip_layer, "version", None)
    packet_info["source"]["ip"] = getattr(ip_layer, "src", None)
    packet_info["destination"]["ip"] = getattr(ip_layer, "dst", None)
```

Ogni pacchetto è un oggetto con **attributi** specifici.  
Si verifica la presenza del livello richiesto e, se disponibile, si **estraggono** i campi con setattr

Aggiunge il **dizionario** del pacchetto appena elaborato alla lista di pacchetti processati

```
Window
extracted_data.append(packet_info)
current_size += event_size
```

# CREAZIONE OUTPUT

# CREAZIONE OUTPUT

- Necessità di suddividere l'output in più file di dimensione minore a **1 MB** per essere inviati a Google Chronicle

Serializzazione degli **eventi** e calcolo della dimensione in byte

Controlla se aggiungere l'evento corrente supererà il limite **massimo** del file

Se superato, si procede a produrre il file in **output**

Se non ha ancora raggiunto il **limite**, aggiunge l'evento corrente alla lista che sarà scritta nel file

```
for event in events:  
    try:  
        event_json = json.dumps(event, indent=4)  
        event_size = len(event_json.encode("utf-8"))  
  
        if current_size + event_size > max_size_bytes:  
            output_file = f"{base_output_file}_{current_file_index}.json"  
            with open(output_file, "w") as f:  
                json.dump(current_events, f, indent=4)  
  
            current_size = 0  
            current_events = []  
  
        current_size += event_size  
        current_events.append(event)
```

```
current_events.append(event)  
current_size += event_size
```

# SINGOLO FILE <=1MB

- Dopo l'iterazione principale sugli eventi, potrebbero essercene ancora in memoria (**current\_events**). Questo codice si assicura che tali eventi siano scritti in un file JSON prima della fine dell'esecuzione

Verifica se la lista di eventi contiene elementi da salvare

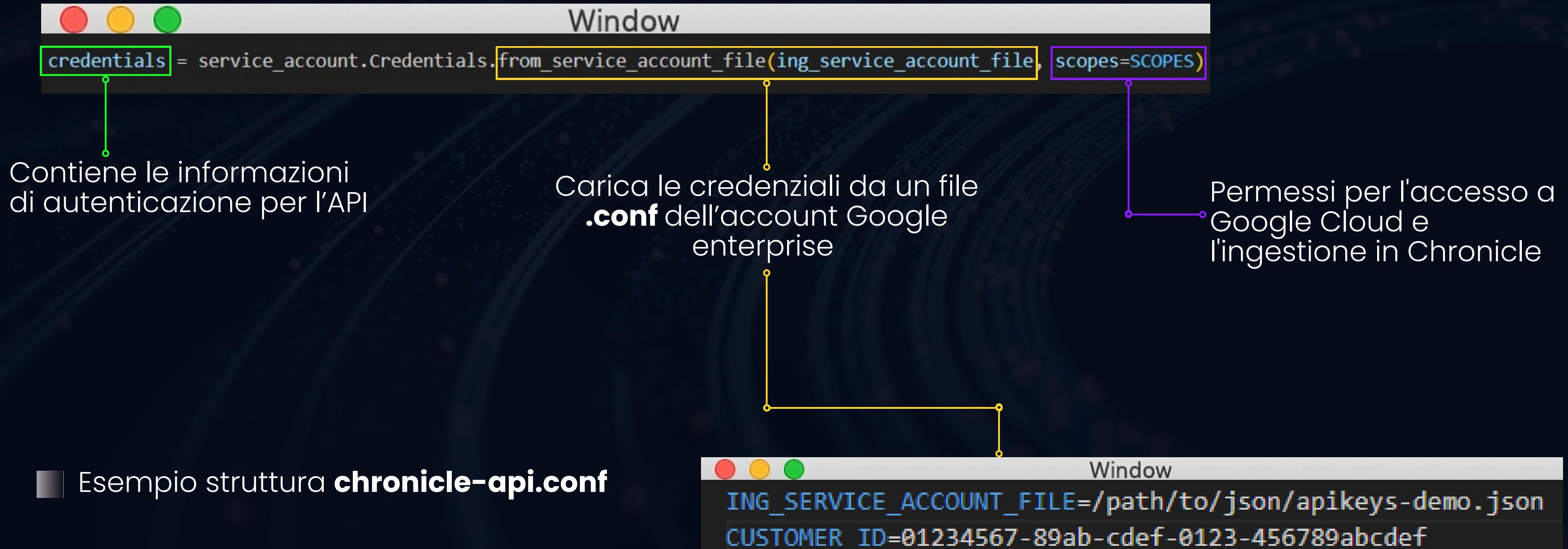
Scrive la lista **current\_events** in file JSON con indice crescente

Se l'operazione non va a buon fine regista un log di errore

```
Window
if current_events:
    try:
        output_file = f"{base_output_file}_{current_file_index}.json"
        with open(output_file, "w") as f:
            json.dump(current_events, f, indent=4)
            logging.info(f"Saved {len(current_events)} events to {output_file}.")
    except IOError as e:
        logging.error(f"Error writing the final file: {e}")
```

# INUIO A CHRONICLE

# INTERAZIONE CON API



# COSTRUZIONE REQUEST

■ URL base che indica al sistema dove inviare le richieste per l'ingestione dei dati

```
Window
INGESTION_API = "https://europe-west12-malachiteingestion-pa.googleapis.com"
```

```
Window
http_session = requests.AuthorizedSession(credentials)

url = f"{INGESTION_API}/v2/udmevents:batchCreate"

body = {
    "customerId": customer_id,
    "events": json.loads(json_events),
}
response = http_session.request("POST", url, json=body)
```

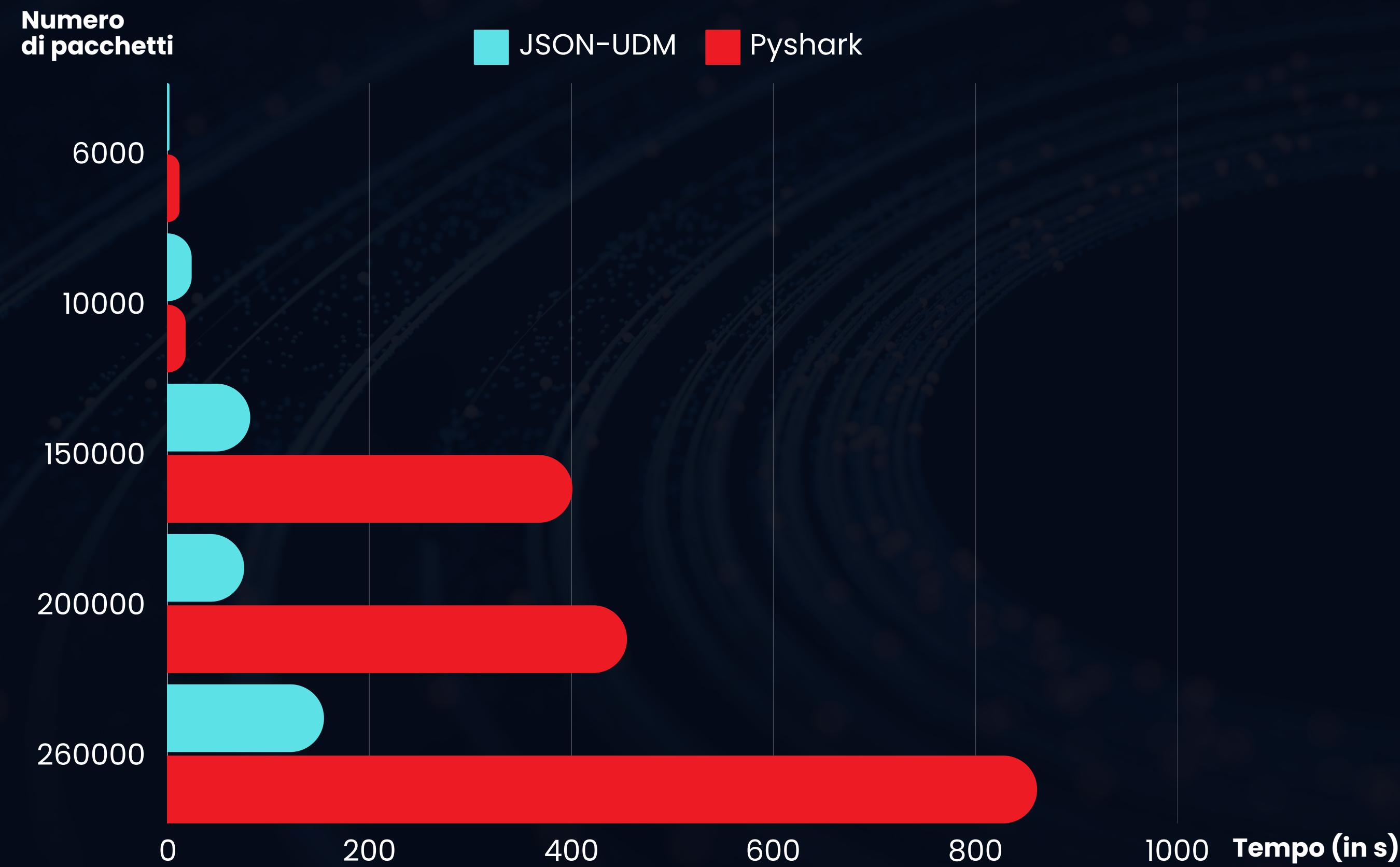
Automatizza l'autenticazione delle richieste HTTP utilizzando l'oggetto **http\_session** per allegare le credenziali.

Contiene l'**URL** completo dell'**API**.  
Esegue **batchCreate** per inviare eventi in un'unica richiesta

**Body** della request: contiene il customer ID e gli eventi in formato **JSON**

# TEST

## Confronto prestazioni



# Test velocità parser JSON

**Pacchetti:** 263.649

**Eventi:** 260.621

**Tempo** totale di esecuzione: 2m 46s

## Specs

**CPU:** Intel i7-12700K

**RAM:** 2x16GB DDR4

**GPU:** NVIDIA RTX 3060

**VRAM:** 12GB

**NVMe:** SN850X 1TB

Tempo per convertire il **.pcap** in **.json**  
con tshark

Output della conversione in  
**JSON-UDM**:  
creati 181 file tutti  $\leq$  1 MB

**Tempo** per strutturare i file in  
JSON-UDM e per dividere gli  
**eventi** in file di dimensione  $\leq$  1 MB

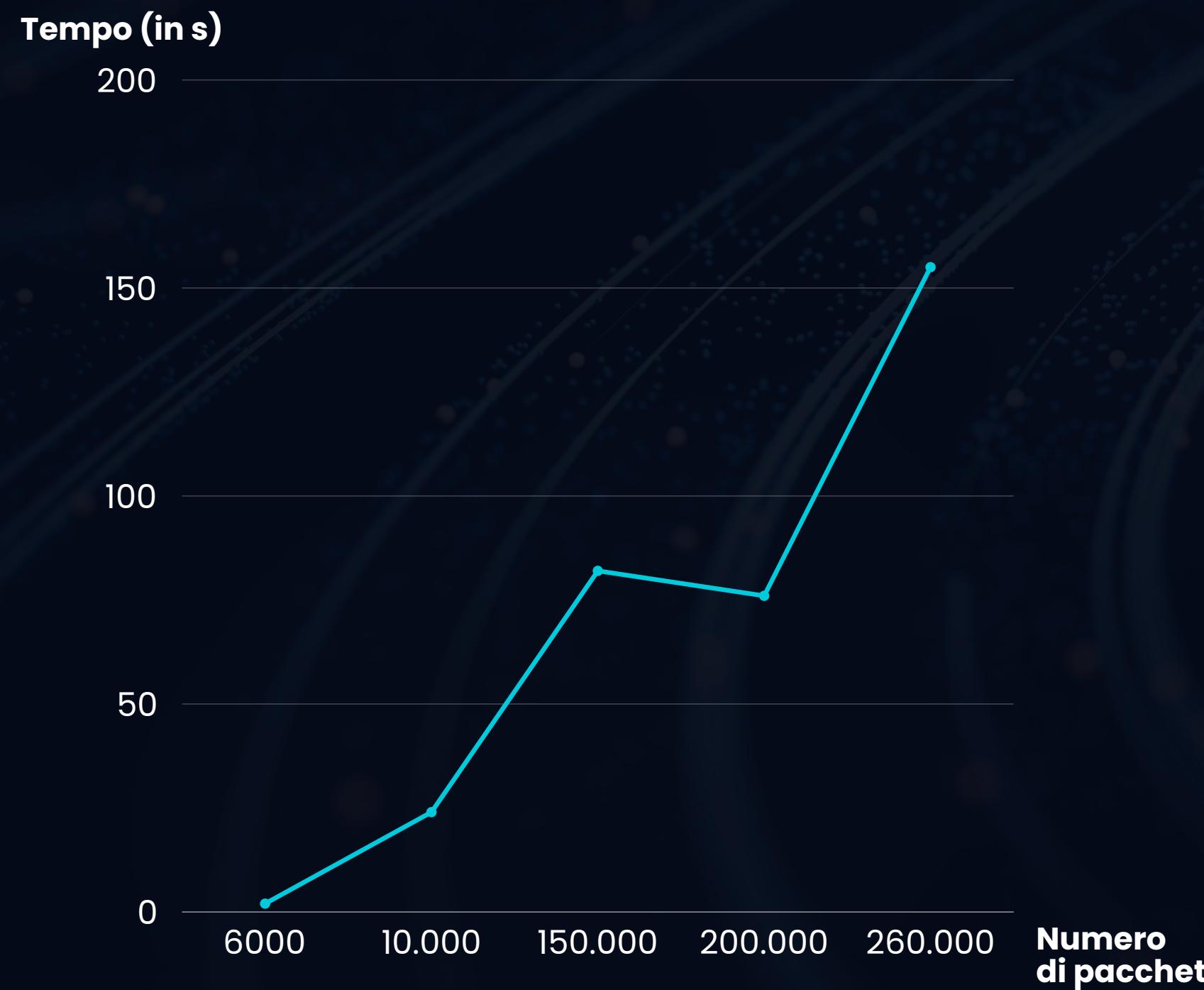


```
fefe@Thousand-Sunny: ~/wir  $ time tshark -r capture2.pcap -T json > capture.json ; time python3 json2udm.py capture.json output.json

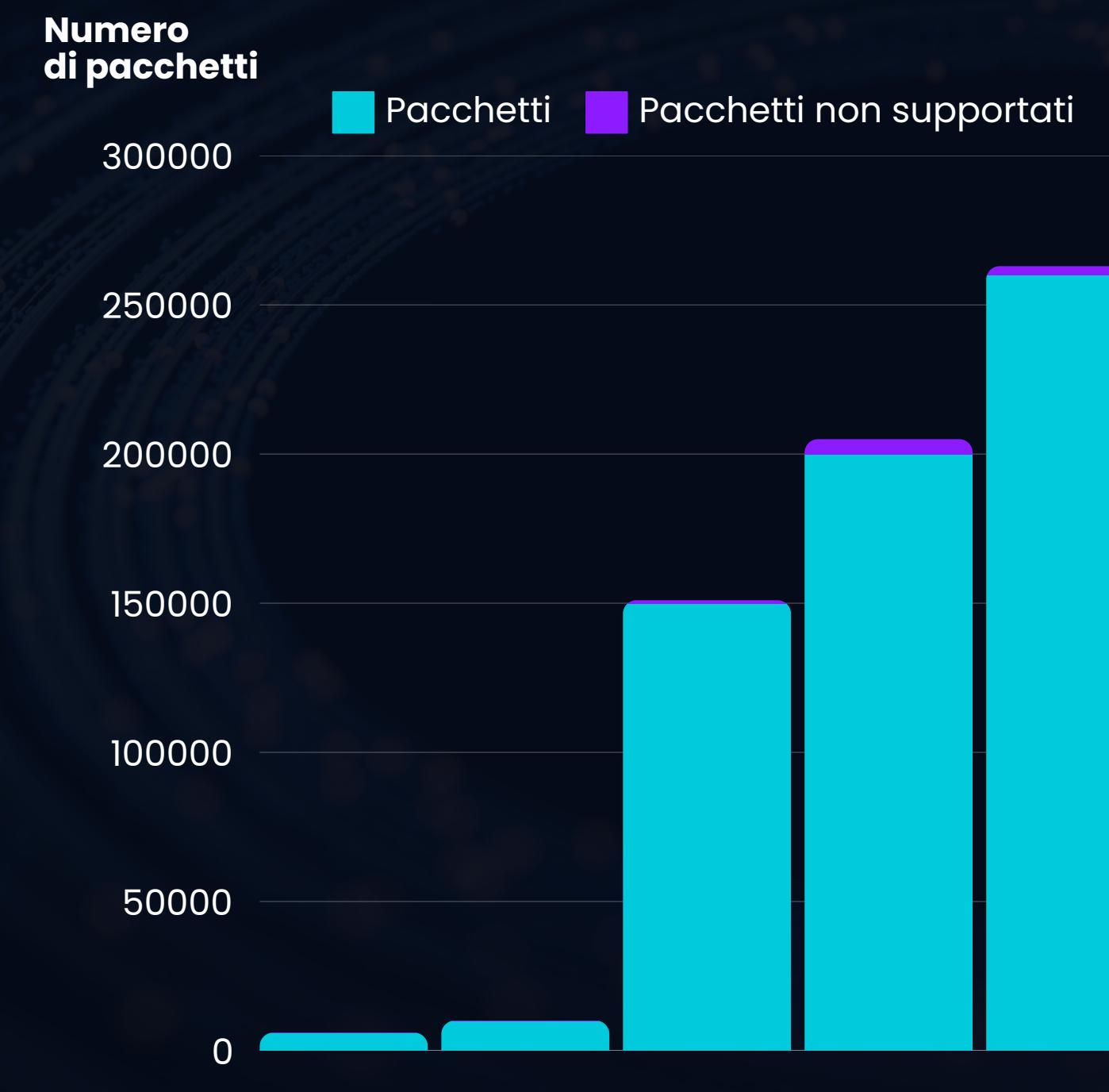
real    1m52.363s
user    0m47.717s
sys     0m2.569s
2025-02-06 12:34:41,604 - INFO - Saved 260621 events.
2025-02-06 12:34:53,506 - INFO - Creati 181 file.

real    0m34.509s
user    0m20.349s
sys     0m4.606s
```

## Confronto **velocità** esecuzione per numero di pacchetti



## Valutazione dei pacchetti non **supportati**



## Test parsing

Sono riportati degli output ottenuti dal parser JSON → JSON-UDM

```
Window
"dns": {
    "query": {
        "name": [
            "b7:b1:4c:0d:4f:01"
        ],
        "ttl": [
            "3481",
            "280",
            "373"
        ],
        "type": [
            "28"
        ]
    }
}
```

```
Window
"mdns": {
    "query": {
        "name": [
            "b7:b1:4c:0d:4f:01"
        ],
        "type": [
            "255"
        ]
    }
}
```

```
Window
"event": {
    "type": "NETWORK_CONNECTION",
    "vendor_name": "Wireshark",
    "product_name": "Wireshark PacketCapture",
    "event_timestamp": "2025-01-27T13:30:39.466000+00:00"
},
"network": {
    "transport_protocol": "UDP",
    "ip": {
        "source": "192.168.178.1",
        "destination": "192.168.178.30",
        "ttl": "64"
    },
    "eth": {
        "source_mac": "b7:b1:4c:0d:4f:01",
        "destination_mac": "b7:b1:4c:0d:4f:01"
    },
    "udp": {
        "source_port": "53",
        "destination_port": "54457"
    }
},
```

Indirizzi

Protocollo UDP

Esempi di output DNS e MDNS

## Ulteriori esempi

```
Window
"tcp": {
    "source_port": "50718",
    "destination_port": "443",
    "flags": "0x0018"
},
"tls": {
    "record_version": "0x0301",
    "handshake": {
        "version": "0x0303"
    }
},
"frame": {
    "timestamp": "2025-01-27T11:07:15.463000+00:00",
    "length": "583",
    "protocols": "eth:ethertype:ip:tcp:tls"
}
```

Protocollo **TCP**

Protocollo **TLS**

Dettagli aggiuntivi

```
Window
"arp": {
    "source_mac": "00:0C:29:00:00:00",
    "source_ip4": "192.168.178.67",
    "destination_mac": "00:00:00:00:00:00",
    "destination_ip4": "192.168.178.1"
},
"frame": {
    "timestamp": "2025-01-27T15:14:29.464000+00:00",
    "length": "42",
    "protocols": "eth:ethertype:arp"
}
```

Protocollo **ARP**

```
Window
"icmp": {
    "type": "8",
    "code": "0"
},
"tls": {},
"frame": {
    "timestamp": "2025-01-27T11:05:55.435000+00:00",
    "length": "98",
    "protocols": "eth:ethertype:ip:icmp:data"
}
```

Protocollo **ICMP**

# DEPLOYMENT DOCKER

```
Window
FROM python:3.9-slim

RUN apt-get update && \
    apt-get install -y tshark inotify-tools && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

WORKDIR /app
COPY json2udm.py /app/json2udm.py
#COPY send_to_chronicle/ingestion_comm.py /app/ingestion_comm.py
#COPY send_to_chronicle/chronicle-api.conf /app/chronicle-api.conf
COPY entrypoint.sh /app/entrypoint.sh

RUN chmod +x /app/entrypoint.sh

ENTRYPOINT ["/app/entrypoint.sh"]
```

## ■ Configurazione automatizzata

Docker costruisce il container iniettando le **dipendenze** e impostando un ambiente di esecuzione semplice ed **intuitivo**

**Installazione** degli strumenti necessari

**Copia** degli script

■ Adattamento al **filesystem** locale

```
Window
version: '3.7'
services:
  wireshark2chronicle:
    build: .
    network_mode: "host"
    cap_add:
      - NET_ADMIN
      - NET_RAW
    volumes:
      - ./sniff:/app/input
      - ./processed:/app/trash
      - ./chronicle:/app/output
    environment:
      - ROTATE=-b filesize:1024
      - LIMITS=-c 20000
```

Cartella di **Input**

Cartella di **output**

**Limiti** per test

## ■ Avvio dello sniffing

Il container, dopo aver determinato automaticamente l'**interfaccia** di rete dell'host, inizia la **cattura** dei pacchetti...

Verifica l'esistenza delle **directory**,  
se necessario procede alla  
creazione

Termina lo script in caso di **errore**

Avvio di **tshark** in background

Mantiene attivo lo script finchè lo è tshark

```
Window
for DIR in "$INPUT_DIR" "$TRASH_DIR" "$MID_DIR" "$OUTPUT_DIR"; do
    if [ ! -d "$DIR" ]; then
        echo "Directory $DIR not found. Creating..."
        mkdir -p "$DIR"
    fi
done

trap 'echo "Terminating tshark due to script exit"; kill $TSHARK_PID' EXIT

[...]

echo "Starting tshark..."
tshark $INTERFACE $ROTATE $LIMITS -w $INPUT_DIR/capture.pcap &
TSHARK_PID=$!

[...]

wait $TSHARK_PID
```

## ■ Ad ogni pacchetto pronto...

iNotify **monitors** gli eventi di "close\_write" all'interno della cartella INPUT e ne **lancia** la conversione

Funzione di post-processing →

```
inotifywait -m -e close_write --format "%w%f" "$INPUT_DIR" | while read -r NEW_FILE; do
    echo "File completed: $NEW_FILE"
    # Small delay to ensure file is fully written
    sleep 1
    process_file "$NEW_FILE"
done
```

## ■ Esempio di esecuzione di **iNotify**

```
[fillo@spike)~[~/Scrivania/test]
$ inotifywait -m -e close_write --format "%w%f" .
Setting up watches.
Watches established.
./capture_00001_20250128165917.pcap
./capture_00002_20250128165920.pcap
./capture_00003_20250128165922.pcap
./capture2_00001_20250128165951.pcap
./capture2_00002_20250128165953.pcap
./capture2_00003_20250128165956.pcap
]
```

## ■ La funzione bash **process\_file()**

Si occupa di tutto il **workflow** che il pacchetto attraversa una volta completato

**Step 1:** pcap → json

```
Window
process_file() {
    local FILE=$(basename "$1" .pcap).json"
    if tshark -r "$1" -T json > "$MID_DIR/$FILE"; then
        mv "$1" "$TRASH_DIR/"
    else
        echo "Error converting file $1"
        return 1
    fi

    if python3 /app/json2udm.py "$MID_DIR/$FILE" "$OUTPUT_DIR/$FILE"; then
        echo "Processing successful, removing file: $MID_DIR/$FILE"
        rm "$MID_DIR/$FILE"
    else
        echo "Error processing file: $MID_DIR/$FILE. Keeping the original file."
    fi
}

if python3 /app/ingestion_comm.py "$OUTPUT_DIR/$FILE"; then
    echo "Results successfully sent to Google Chronicle"
fi
}
```

**Step 2:** json → UDM

**Step 3:** invio dei risultati  
a Chronicle

■ Rimozione file processati in caso di successo

## ■ La funzione bash **recover\_pending\_files()**

Questa funzione opera in modo analogo alla precedente, ma sui file già presenti nella cartella

In questo modo si **recuperano** i file **interrotti** dalle esecuzioni precedenti...

```
Window
if ls "$INPUT_DIR"/*.pcap "$MID_DIR"/*.json 2> /dev/null | grep -q .; then
    recover_pending_files
fi
```

```
Window
recover_pending_files() {
    echo "Recovering pending files before starting new sniffing session..."

    for PENDING in "$INPUT_DIR"/*.pcap; do
        [ -e "$PENDING" ] && tshark -r "$PENDING" -T json > "$MID_DIR/${basename "$PENDING"} .json" && mv "$PENDING" "$TRASH_DIR/"
    done

    for PENDING in "$MID_DIR"/*.json; do
        if [ -e "$PENDING" ]; then
            python3 /app/json2udm.py "$PENDING" "$OUTPUT_DIR/${basename "$PENDING"}" && rm "$PENDING"
            python3 /app/ingestion_comm.py "$OUTPUT_DIR/${basename "$PENDING"}" && echo "Results successfully sent to Google Chronicle"
        fi
    done
}
```

... o se ne processano di **aggiunti manualmente!**



```
(fillo@spike) - [~/Scrivania/test]
$ docker compose up
WARN[0000] /home/fillo/Scrivania/test/compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to
avoid potential confusion
[+] Building 29.8s (12/12) FINISHED
  => [wireshark2chronicle internal] load build definition from Dockerfile
  => => transferring dockerfile: 474B
  => [wireshark2chronicle internal] load metadata for docker.io/library/python:3.9-slim
  => [wireshark2chronicle internal] load .dockerignore
  => => transferring context: 2B
  => [wireshark2chronicle 1/6] FROM docker.io/library/python:3.9-slim@sha256:f9364cd6e0c146966f8f23fc4fd85d53f2e604bdde74
  => => resolve docker.io/library/python:3.9-slim@sha256:f9364cd6e0c146966f8f23fc4fd85d53f2e604bdde74e3c06565194dc4a02f85
  => => sha256:f9364cd6e0c146966f8f23fc4fd85d53f2e604bdde74e3c06565194dc4a02f85 10.41kB / 10.41kB
  => => sha256:e278b827219b44d04b4be79af72fb3c4458dbe15d96440d8694bb3fd9d0bbb5c 1.75kB / 1.75kB
  => => sha256:096343841dd9d6129087a72170c0d3cd51e64a0c00934029a8927625637f51d1 5.28kB / 5.28kB
  => => sha256:c29f5b76f736a8b555fd191c48d6581bb918bcd605a7cbcc76205dd6acff3260 28.21MB / 28.21MB
  => => sha256:9315c4821e723a7fb42220024e56534ecd042bd9ebb91aa3487c0a90cbbe9710 3.51MB / 3.51MB
  => => sha256:7e8ac65e25aaa3b7a0cef09164fd2c6879c88b27bc9d50b5be34166a11479656 14.93MB / 14.93MB
  => => sha256:b26995e9f45f14d5d4a45d1d0c50396482c1a6c1fcbbf5e6105cac49ef9afbba 248B / 248B
  => => extracting sha256:c29f5b76f736a8b555fd191c48d6581bb918bcd605a7cbcc76205dd6acff3260
  => => extracting sha256:9315c4821e723a7fb42220024e56534ecd042bd9ebb91aa3487c0a90cbbe9710
  => => extracting sha256:7e8ac65e25aaa3b7a0cef09164fd2c6879c88b27bc9d50b5be34166a11479656
  => => extracting sha256:b26995e9f45f14d5d4a45d1d0c50396482c1a6c1fcbbf5e6105cac49ef9afbba
  => [wireshark2chronicle internal] load build context
  => => transferring context: 15.55kB
  => [wireshark2chronicle 2/6] RUN apt-get update &&      apt-get install -y tshark inotify-tools &&      apt-get clean &&
  => [wireshark2chronicle 3/6] WORKDIR /app
  => [wireshark2chronicle 4/6] COPY json2udm.py /app/json2udm.py
  => [wireshark2chronicle 5/6] COPY entrypoint.sh /app/entrypoint.sh
  => [wireshark2chronicle 6/6] RUN chmod +x /app/entrypoint.sh
  => [wireshark2chronicle] exporting to image
  => => exporting layers
  => => writing image sha256:4af8f8500ec3ab62e3b06ec168e7535712d1f17b92b2f86995290ca1befa083a
  => => naming to docker.io/library/test-wireshark2chronicle
  => [wireshark2chronicle] resolving provenance for metadata file
[+] Running 2/2
✓ wireshark2chronicle          Built
✓ Container test-wireshark2chronicle-1 Created
Attaching to wireshark2chronicle-1
wireshark2chronicle-1 | Directory /app/jsonized not found. Creating...
wireshark2chronicle-1 | Starting tshark...
wireshark2chronicle-1 | Starting tshark on interface -i enp3s0 with args: -b filesize:1024 -c 20000
wireshark2chronicle-1 | Setting up watches.
wireshark2chronicle-1 | Watches established.
wireshark2chronicle-1 | Running as user "root" and group "root". This could be dangerous.
wireshark2chronicle-1 | Capturing on 'enp3s0'
wireshark2chronicle-1 |   ** (tshark:10) 19:37:19.044108 [Main MESSAGE] -- Capture started.
wireshark2chronicle-1 |   ** (tshark:10) 19:37:19.044152 [Main MESSAGE] -- File: "/app/input/capture_00001_20250208193719.pcap"
wireshark2chronicle-1 | File completed: /app/input/capture_00001_20250208193719.pcap
wireshark2chronicle-1 |   ** (tshark:10) 19:37:22.097135 [Main MESSAGE] -- File: "/app/input/capture_00002_20250208193722.pcap"
wireshark2chronicle-1 | Running as user "root" and group "root". This could be dangerous.
wireshark2chronicle-1 | 2025-02-08 19:37:23,832 - ERROR - Unexpected error processing packet: string indices must be integers
wireshark2chronicle-1 | 2025-02-08 19:37:23,841 - INFO - Saved 1073 events.
wireshark2chronicle-1 | 2025-02-08 19:37:23,964 - INFO - Creati 1 file.
wireshark2chronicle-1 | Processing successful, removing file: /app/jsonized/capture_00001_20250208193719.json
wireshark2chronicle-1 | File completed: /app/input/capture_00002_20250208193722.pcap
wireshark2chronicle-1 |   ** (tshark:10) 19:37:24.912922 [Main MESSAGE] -- File: "/app/input/capture_00003_20250208193724.pcap"
wireshark2chronicle-1 | Running as user "root" and group "root". This could be dangerous.
wireshark2chronicle-1 | 2025-02-08 19:37:26,482 - ERROR - Unexpected error processing packet: string indices must be integers
wireshark2chronicle-1 | 2025-02-08 19:37:26,491 - INFO - Saved 1061 events.
wireshark2chronicle-1 | 2025-02-08 19:37:26,588 - INFO - Creati 1 file.
wireshark2chronicle-1 | Processing successful, removing file: /app/jsonized/capture_00002_20250208193722.json
```

# SVILUPPI FUTURI

## MIGLIORAMENTO PARSING

Aggiunta di nuovi  
**protocolli** e campi

Maggiore flessibilità del parser  
per integrare strutture peculiari  
di alcuni **pacchetti** applicativi

## ACCESSO A CHRONICLE

Otttenere **API-KEY** Chronicle  
con account Enterprise

## POTENZIAMENTO DEPLOYMENT

Riduzione **carico** CPU e  
aumento resistenza a **fault**

Introduzione di un **Message  
broker** per velocizzare la  
pipeline

Introduzione **DB**  
per gestione **log**

A dark blue background featuring a faint, glowing circuit board pattern with light blue lines and small white dots. Two dark hexagonal shapes, resembling integrated circuit components, are positioned on the left and right sides of the text.

**GRAZIE PER  
L'ATTENZIONE**