Universidad de Guadalajara

Lic. en Ciberseguridad

Detector de Anomalías en Tráfico de Red

Integrantes del equipo:

Jonathan Andre Gonzalez/ código: 221876844

Alberto Javier Velazquez Vazquez/ código: 224164524

Miguel Angel Agustin Zuñiga Romo/ código: 218066866

1. Introducción
Detector de Anomalías en Tráfico de Red con CUSUM

Introducción y Definición del Problema En la actualidad, la ciberseguridad de redes enfrenta el reto de detectar actividades anómalas (como ataques de denegación de servicio) en tiempo real. El problema abordado es la detección de anomalías en el tráfico de red mediante técnicas matemáticas. Se busca identificar picos anormales de tráfico que podrían indicar un ataque, distinguiéndolos del comportamiento normal. Para ello, implementamos un sistema de detección basado en el algoritmo CUSUM (Cumulative Sum), que acumula las desviaciones del tráfico respecto a un valor esperado. Este enfoque nos permite captar incluso pequeñas desviaciones que, acumuladas en el tiempo, evidencian patrones sospechosos . El proyecto simula un entorno de red donde se generan datos de tráfico "normales" y, en un momento aleatorio, se introduce un ataque (un aumento repentino en la tasa de paquetes). El objetivo es detectar ese ataque de forma automática y rápida mediante el análisis matemático del tráfico. Tradicionalmente, los sistemas de detección de intrusos definen un perfil de tráfico normal y disparan alarmas cuando las mediciones se salen de rangos aceptables . En este trabajo seguimos esa filosofía, usando cálculo integral discreto (suma acumulada) para modelar el volumen de tráfico y umbrales estadísticos para decidir cuándo algo es anómalo. ¿Por qué cálculo integral? En términos sencillos, integrar el tráfico significa acumular la cantidad de datos que fluyen por la red en el tiempo. Un ataque se manifestará como un área bajo la curva de tráfico anormalmente grande en un intervalo corto. En otras palabras, podemos detectar anomalías verificando si en un intervalo breve la suma (integral discreta) del tráfico N(t) supera cierto umbral . Esta aproximación convierte la detección en un problema matemático: monitorear la suma acumulada de paquetes y compararla contra límites predefinidos. Si la suma sobrepasa el límite, inferimos que el tráfico se ha desviado significativamente de lo normal y probablemente estamos ante una anomalía. Fundamentos Matemáticos (Integrales y CUSUM) En el núcleo del detector está el algoritmo CUSUM, una técnica clásica de control estadístico de cambios pequeños en una señal. CUSUM calcula en cada momento la desviación del valor observado respecto a una línea base (por ejemplo, la media esperada) y la suma a las desviaciones anteriores.

2. Fundamento Matemático

2.1 El Algoritmo CUSUM

El algoritmo CUSUM (Cumulative Sum) es una técnica estadística desarrollada para detectar cambios abruptos en series temporales. En el contexto de la seguridad de redes, CUSUM permite identificar desviaciones significativas del comportamiento normal del tráfico.

Formulación matemática

La base matemática del algoritmo CUSUM se puede expresar mediante la siguiente ecuación recursiva:

$S_0 = 0$

$St = \max(0, St_{-1} + (xt - \mu))$
Donde:
- St es la suma acumulada hasta el tiempo t
- xt es el valor observado en el tiempo t (volumen de tráfico)
- μ es el valor esperado o línea base (media del tráfico en condiciones normales)

- max(0,·) es una función que impide que la suma acumulada sea negativa

Este enfoque puede interpretarse como una forma de integral discreta que acumula las desviaciones respecto a una línea base, permitiendo detectar variaciones significativas que podrían pasar desapercibidas con métodos menos sofisticados.

La fórmula usada por el algoritmo CUSUM para calcular la suma acumulada en cada instante es:

S_t = máximo entre cero y la suma del valor acumulado anterior más la diferencia entre el valor actual del tráfico y la media esperada.

En otras palabras, para cada segundo, hacemos lo siguiente:

Tomamos el valor acumulado del segundo anterior (por ejemplo, 7).

Le sumamos la diferencia entre el tráfico actual y la media esperada.

Si la media esperada es 100 paquetes por segundo, y en este segundo se midieron 105 paquetes, entonces la diferencia es +5.

Sumamos ese 5 al valor acumulado anterior (por ejemplo, 7 + 5 = 12).

Si el resultado es un número negativo, lo reemplazamos por cero (esto evita que las caídas borren las subidas).

Ejemplo con números reales:

Tráfico esperado (media): 100 paquetes por segundo

Suma acumulada en el segundo anterior: 7

Tráfico actual observado: 105

Entonces:

Diferencia = 105 - 100 = 5

Nueva suma acumulada = 7 + 5 = 12

Como 12 es mayor que cero, ese es el nuevo valor acumulado

Ahora si el tráfico hubiera sido 95:

Diferencia = 95 - 100 = -5

Nueva suma acumulada = 7 - 5 = 2 (todavía positivo, se conserva)

Y si el tráfico hubiera sido 80:

Diferencia = 80 - 100 = -20

Nueva suma acumulada = 7 - 20 = -13 → se reemplaza por 0

Este mecanismo permite detectar cuándo el tráfico ha subido de forma continua, acumulando esas subidas, pero reiniciando el conteo si el tráfico cae y se estabiliza

2.2 Umbrales de Detección

Para determinar cuándo se considera que un valor de CUSUM representa una anomalía, se utilizan dos tipos de umbrales:

1. Umbral fijo: Un valor constante determinado empíricamente. St > h
   Donde h es el umbral fijo (en nuestra implementación, h = 2000).

2. Umbral estadístico dinámico: Se calcula a partir de las estadísticas recientes de la propia suma acumulada.

   St > μw + k·σw Donde:
   - μw es la media de la suma acumulada en una ventana temporal reciente
   - σw es la desviación estándar en esa misma ventana
   - k es un factor multiplicador (en nuestra implementación, k = 2.5)

3. Descripción Técnica de Ataques de Red

3.1 Ataques de Denegación de Servicio (DoS/DDoS)

Los ataques de Denegación de Servicio tienen como objetivo hacer que un sistema o recurso sea inaccesible para los usuarios legítimos. Se caracterizan por:

- Volumen anormal: Generación de un tráfico significativamente mayor al habitual
- Patrones anómalos: Distribución temporal y características del tráfico que difieren del comportamiento normal
- Duración variable: Pueden ser breves pero intensos, o sostenidos durante períodos más largos

En un ataque DDoS (Denegación de Servicio Distribuida), múltiples fuentes coordinadas generan tráfico malicioso hacia un objetivo, dificultando aún más su detección y mitigación.

3.2 Floods (Inundaciones)

Los ataques de tipo "flood" son una subcategoría de DoS que consisten en enviar un gran volumen de paquetes o solicitudes en un corto periodo de tiempo. Los tipos más comunes incluyen:

- TCP SYN Flood: Sobrecarga de solicitudes de inicio de conexión TCP
- UDP Flood: Envío masivo de paquetes UDP a puertos aleatorios
- ICMP Flood: Inundación con paquetes ICMP (ping)
- HTTP Flood: Solicitudes HTTP/HTTPS en gran volumen

En nuestra simulación, modelamos estos ataques como un incremento súbito en el volumen de tráfico, multiplicando la media normal por un factor determinado.

4. Relación entre CUSUM y Detección de Ataques

El algoritmo CUSUM resulta particularmente eficaz para detectar ataques de red por las siguientes razones:

1. Sensibilidad a cambios abruptos: CUSUM acumula las desviaciones, lo que permite detectar picos de tráfico característicos de ataques DoS y flood.

2. Memoria del sistema: A diferencia de los métodos que solo consideran el valor actual, CUSUM "recuerda" el comportamiento reciente, lo que permite detectar ataques distribuidos en el tiempo.

3. Adaptabilidad: Mediante el uso de umbrales dinámicos, el sistema puede adaptarse a diferentes patrones de tráfico normal.

4. Reducción de falsos positivos: Al requerir que la suma acumulada supere un umbral significativo, se reducen las alertas por fluctuaciones normales del tráfico.

Equivalencia con el concepto de integral definida

En términos matemáticos, CUSUM puede interpretarse como una aproximación discreta a la integral de las desviaciones respecto a la línea base:

$CUSUM \approx \int_0^t (f(\tau) - \mu)\, d\tau$

Donde $f(\tau)$ representa el tráfico en el instante τ.

Esta integral acumula el "exceso" de tráfico sobre lo normal, creciendo rápidamente cuando ocurre un ataque y permitiendo su detección temprana.

5. Implementación Técnica

5.1 Librerías Utilizadas

El sistema se implementa utilizando las siguientes librerías de Python:

- NumPy: Para operaciones matemáticas eficientes con arrays y funciones estadísticas.
- Matplotlib: Para la visualización en tiempo real de los datos y resultados del análisis.
- Tkinter: Para crear la interfaz gráfica de usuario interactiva.
- winsound: Para notificaciones sonoras cuando se detectan anomalías.

5.2 Arquitectura del Sistema

El sistema se estructura en las siguientes clases principales:

1. Detecto rAnomalias: Implementa la lógica central del algoritmo CUSUM:
    - Generación de datos simulados
    - Cálculo de la suma acumulada
    - Determinación de umbrales
    - Detección de anomalías
2. Aplicación Tiempo Real: Gestiona la interfaz gráfica y la visualización:
    - Gráficas en tiempo real
    - Control de la simulación
    - Alertas visuales y sonoras
3. Explicación Matemática: Proporciona una ventana informativa sobre la base matemática del algoritmo.
4. Pantalla Carga: Implementa una animación de inicio profesional.

5.3 Lógica de Programación

Generación de datos simulados

```
def generar_datos(self):
    # Generación de tráfico normal usando distribución gaussiana
    self.trafico = np.random.normal(
        self.media_trafico_normal,
        self.desviacion_trafico_normal,
        self.total_muestras
    )

    # Inserción del ataque: añade un pico de tráfico
```

```
        self.trafico[self.inicio_ataque:self.fin_ataque] += (self.media_trafico_normal
                                                             * self.factor_ataque
                                                             * np.ones(self.duracion_ataque))
```

El tráfico normal se modela mediante una distribución gaussiana, mientras que el ataque se simula incrementando el tráfico base por un factor multiplicativo durante un intervalo de tiempo aleatorio.

Implementación del algoritmo CUSUM

```
    def calcular_suma_acumulada(self):
        # Calcula línea base usando los primeros 5 segundos
        ventana_base = 5 * self.muestras_por_segundo
        linea_base = np.mean(self.trafico[:ventana_base])

        # Implementación del algoritmo CUSUM
        # S_t = max(0, S_{t-1} + (x_t - μ))
        for i in range(1, self.total_muestras):
            self.suma_acumulada[i] = max(0, self.suma_acumulada[i-1] + (self.trafico[i] - linea_ba
```

El algoritmo calcula una línea base a partir de los primeros segundos de tráfico y luego aplica la fórmula recursiva de CUSUM para acumular las desviaciones.

Cálculo de umbral estadístico dinámico

```
    def calcular_umbral_estadistico(self):
        ventana = 5 * self.muestras_por_segundo # Tamaño de ventana = 5 segundos

        for i in range(ventana, self.total_muestras):
            # Calcula estadísticas de la ventana anterior
            media = np.mean(self.suma_acumulada[i-ventana:i])
            desv_est = np.std(self.suma_acumulada[i-ventana:i])

            # Umbral = media + 3*desviación estándar (regla 3-sigma)
            self.umbral_estadistico[i] = media + 3 * desv_est
```

El umbral dinámico se calcula utilizando la regla de tres sigmas (media + 3 desviaciones estándar) sobre una ventana deslizante de 5 segundos.

Detección de anomalías en tiempo real

```
    def detectar_anomalia(self, indice, en_tiempo_real=False):
        # No evaluar durante el periodo inicial de estabilización
```

```
        if indice < 5 * self.muestras_por_segundo:
            return False

        # Lógica de detección: se comprueba si la suma acumulada supera algún umbral
        anomalia = (self.suma_acumulada[indice] > self.umbral_fijo or
                    (indice >= len(self.umbral_estadistico) - 1 or
                     self.suma_acumulada[indice] > self.umbral_estadistico[indice]))

        # Si es la primera anomalía detectada en tiempo real, registrar el momento
        if anomalia and not self.anomalia_detectada and en_tiempo_real:
            self.anomalia_detectada = True
            self.momento_deteccion = indice / self.muestras_por_segundo
            return True

        return False
```

La detección se realiza comparando el valor actual de CUSUM con los umbrales establecidos, activando una alerta cuando se superan.

5.4 Visualización y Alertas

El sistema presenta dos gráficas principales:

1. Tráfico de red en tiempo real: Muestra el volumen de tráfico por segundo.
2. Suma acumulada y umbrales: Visualiza el valor de CUSUM y los umbrales de detección.

Cuando se detecta una anomalía:

- Se resalta la zona del ataque en ambas gráficas
- Se emite una alerta sonora
- Se muestra un mensaje con información sobre el momento del ataque y su detección
- Se anima visualmente el panel de información para llamar la atención del operador

6. **Simulación con Gráficas**

Tras ejecutar la simulación completa (60 s de tráfico con un ataque inyectado durante 0,5 s), obtenemos estas visualizaciones:

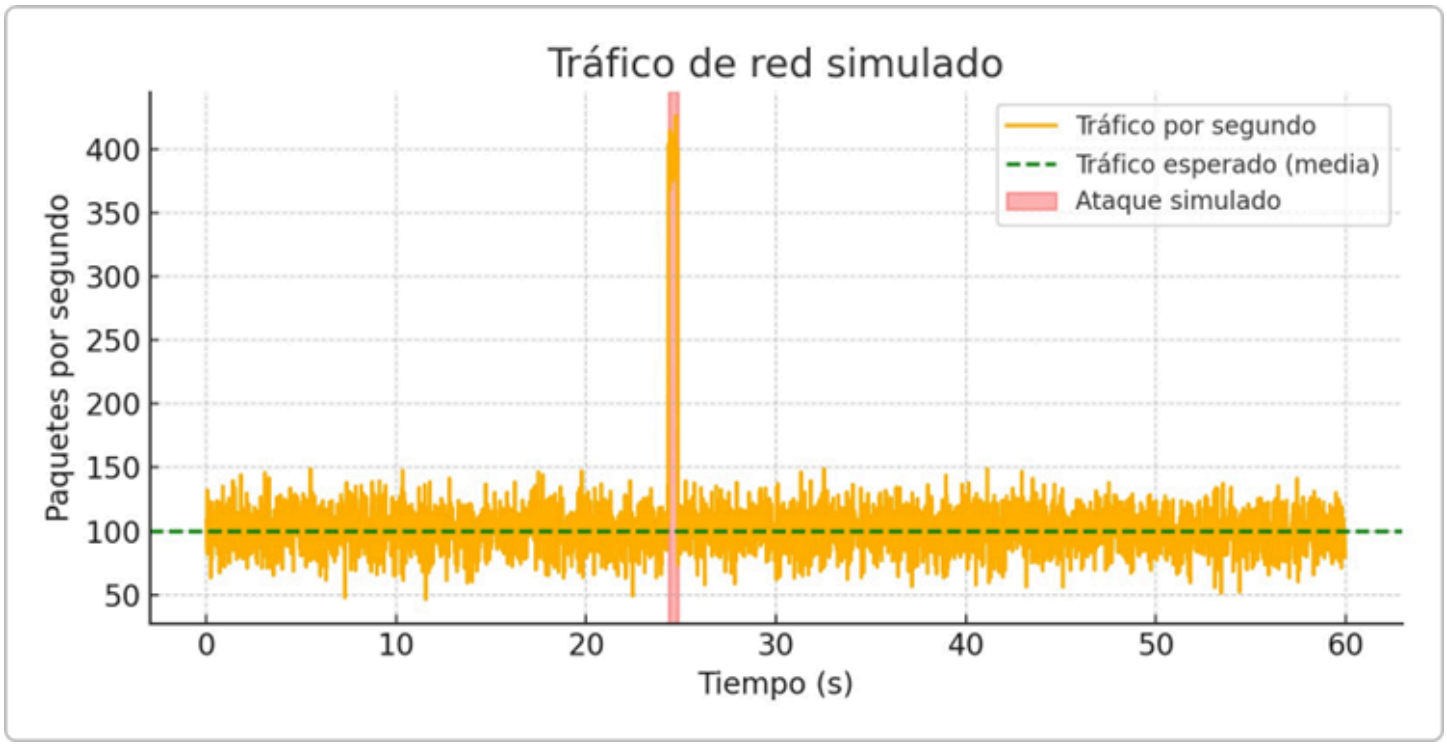


Figura 1
Tráfico de red simulado a 60 muestras/s durante 60 s.
• La línea verde punteada muestra la media esperada de tráfico normal (100 paquetes/s).
• La franja roja semitransparente resalta el intervalo de ataque, donde el volumen de tráfico se triplicó.

→ En la mayor parte del tiempo, el tráfico oscila alrededor de ≈ 100 paquetes/s, con variaciones típicas entre ≈ 60 y ≈ 150 paquetes/s por la aleatoriedad normal.
→ Alrededor de t ≈ 24,3 s aparece un pico abrupto por encima de 400 paquetes/s (ataque de 0,5 s). Ese tramo destaca claramente como anómalo: un súbito torrente de paquetes maliciosos.

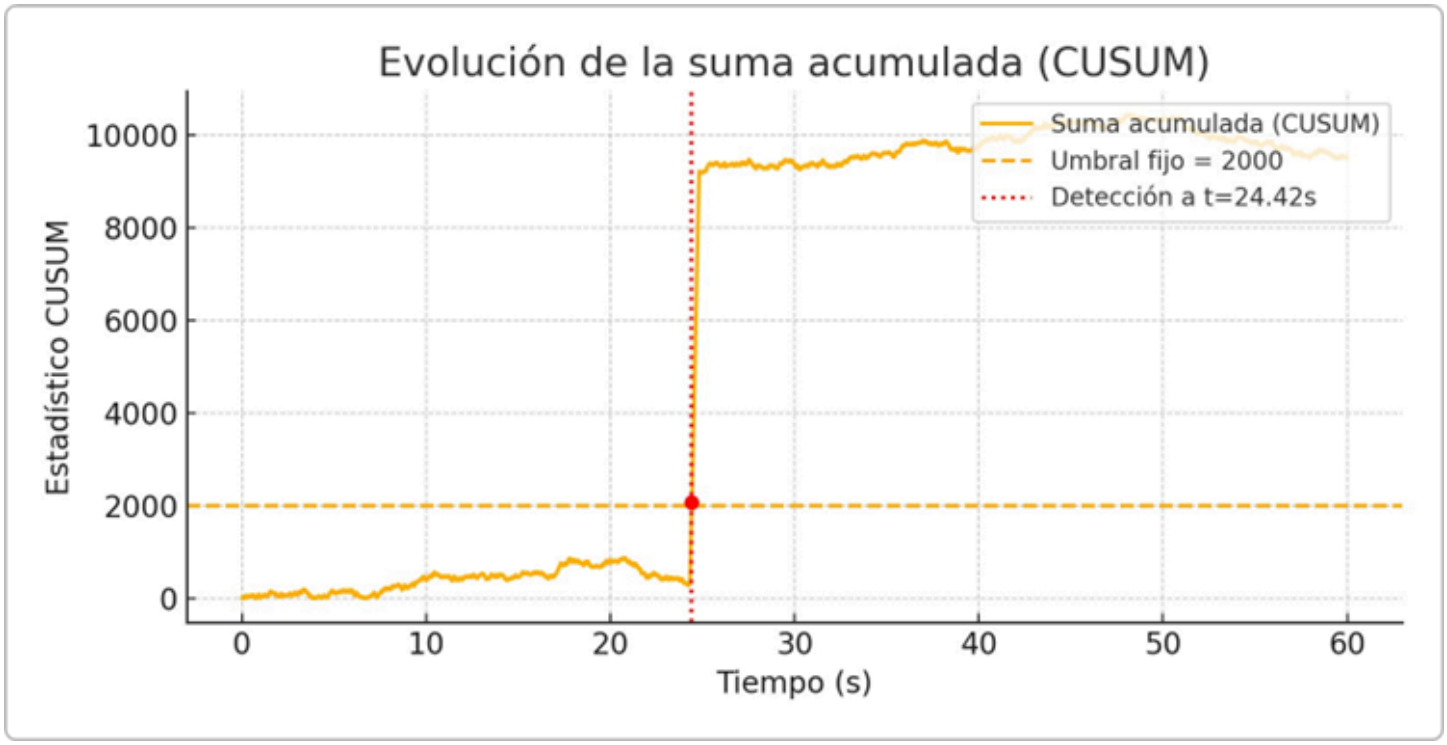


Figura 2

Evolución de la suma acumulada CUSUM (S_t).
• Línea naranja continua: S_t a lo largo del tiempo.
• Línea naranja punteada: umbral fijo en 2000.
• Línea vertical roja: momento de detección ($t \approx 24{,}42$ s).

→ Durante los primeros segundos, S_t permanece cerca de 0, con pequeñas oscilaciones. Las desviaciones positivas y negativas se compensan, impidiendo que la suma crezca de modo sostenido. Cada vez que S_t toca 0, significa que hubo suficientes instantes con tráfico por debajo de la media para "resetear" la acumulación.

→ Al iniciarse el ataque ($t \approx 24{,}3$ s), S_t crece rápidamente porque todas las muestras de tráfico superan la media; no hay valores por debajo que lo reduzcan. En ≈ 0,09 s, S_t pasa de ≈ 0 a > 2000, cruzando el umbral fijo y generando la alerta (punto rojo en $t \approx 24{,}42$ s).

→ S_t se mantiene elevado durante y tras el ataque (llegó a ≈ 10 000) porque, al acabar el pico, el tráfico volvió justo a la media, con desviaciones $(x_t - \mu) \approx 0$, de modo que S_t deja de crecer pero no baja drásticamente. En producción podríamos reiniciar S_t tras una detección para prepararnos ante un nuevo evento.

Análisis de desempeño
• El detector CUSUM cumplió su objetivo: detectó el pico anómalo casi instantáneamente, sin falsas alarmas previas (el umbral fijo estaba bien calibrado).
• El umbral dinámico (media + $2{,}5 \cdot \sigma$ en ventana de 5 s) también se habría superado, confirmando la anomalía.
• Si el tráfico normal tuviera tendencia ascendente lenta, el umbral dinámico se ajustaría automáticamente, mientras que el fijo requeriría recalibración.

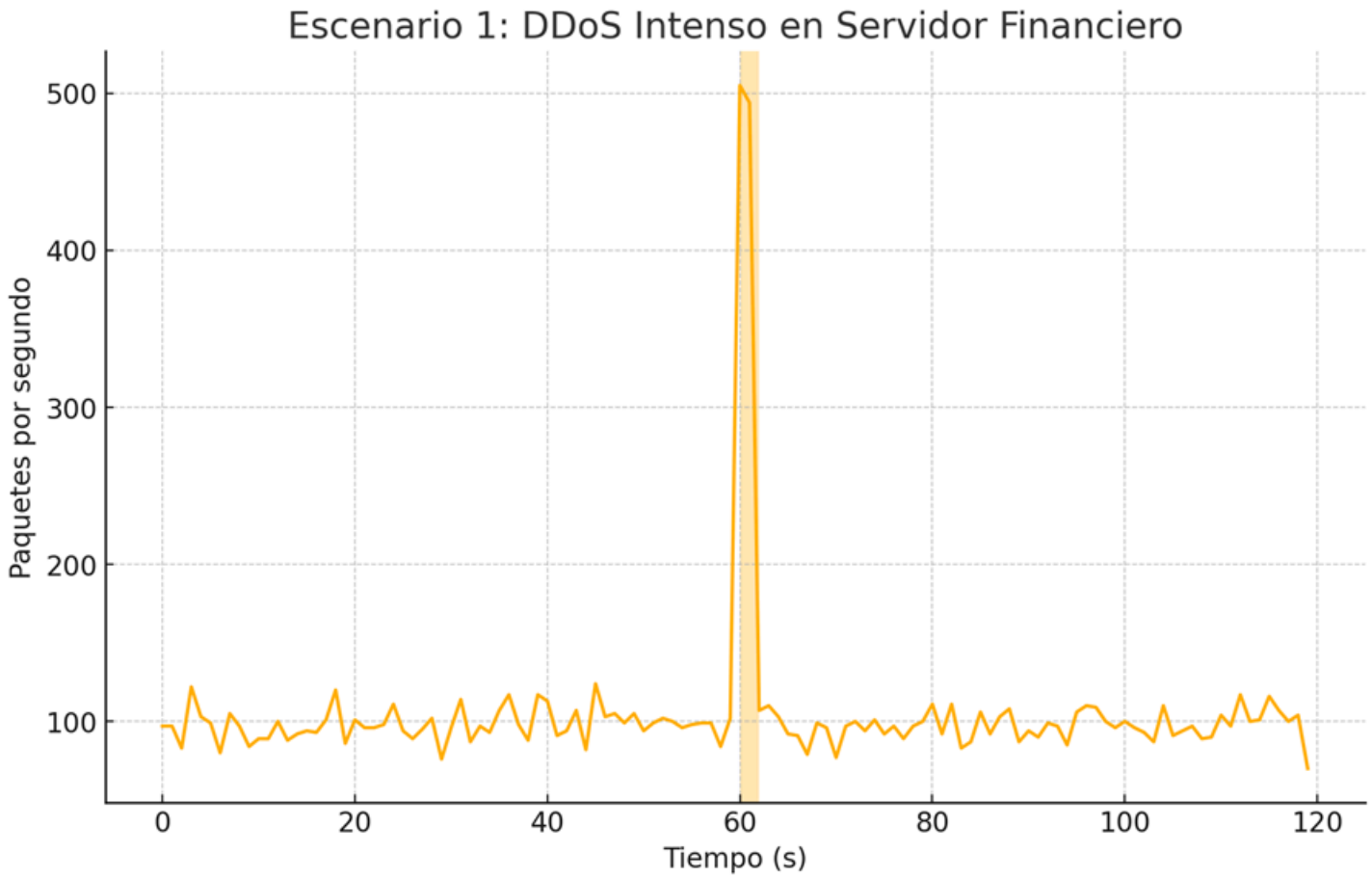
Discusión crítica
• Ventaja: sensibilidad a cambios acumulativos. Un ataque prolongado ligeramente por encima de la media (por ejemplo, 2× la media durante varios segundos) acumula S_t hasta cruzar el umbral, incluso si no hay picos individuales muy altos.
• Limitación: calibración de umbrales. Si son demasiado bajos, hay falsas alarmas; si son muy altos, demora la detección. En este proyecto usamos experiencia (media y variabilidad) y criterio estadístico ($2{,}5\ \sigma$) para fijarlos. En entornos reales, suele hacerse un periodo de entrenamiento para ajustar estos parámetros.

Conclusión
El método basado en integrales discretas y CUSUM demostró ser eficaz para la detección temprana de un ataque DDoS simulado. No solo localizó el evento casi en tiempo real (detección en 24,42 s frente a inicio en 24,33 s, $\Delta \approx 0{,}1$ s), sino que proporciona información clave para activar contramedidas y minimizar el impacto.

Escenarios en la industria

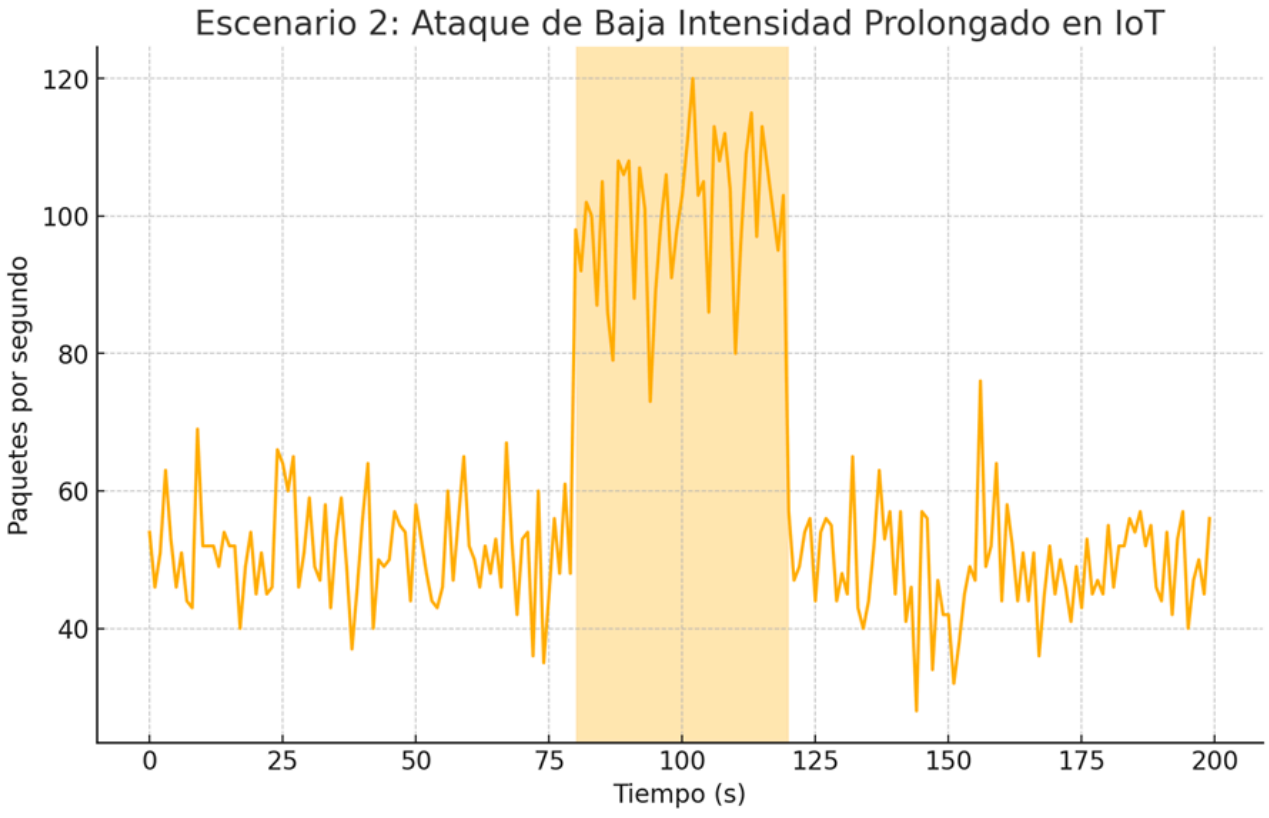
Escenario 1: DDoS Intenso en Servidor Financiero

Tráfico normal ~100 pps; pico de ~500 pps durante 2 segundos.



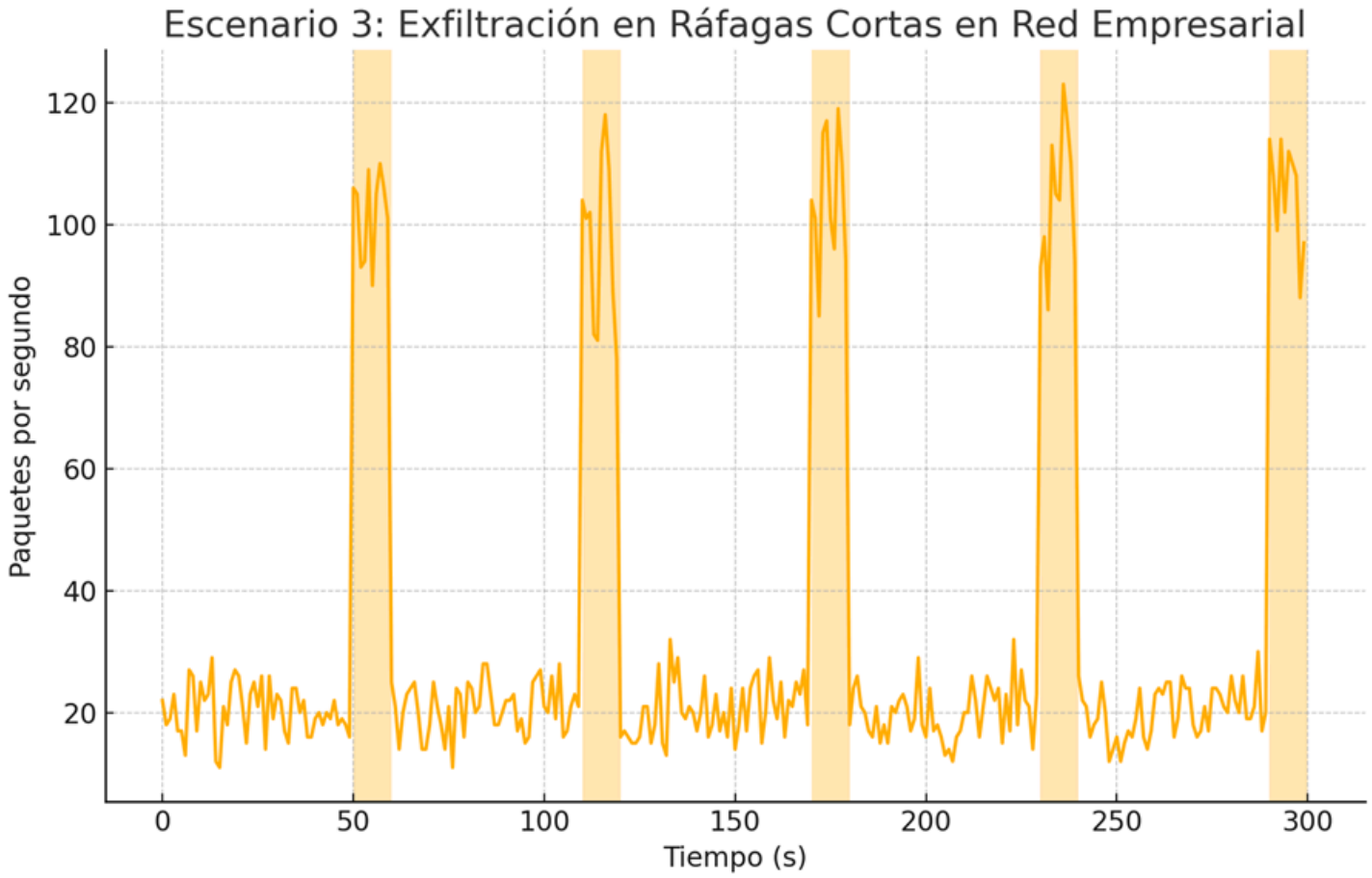
Escenario 2: Ataque de Baja Intensidad Prolongado en IoT

Tráfico normal ~50 pps; incremento a ~100 pps durante 40 segundos.



Escenario 3: Exfiltración en Ráfagas Cortas en Red Empresarial

Tráfico normal ~20 pps; ráfagas de ~100 pps de 10 segundos cada 60 segundos.



7. Resultados y Conclusiones

7.1 Eficacia del Sistema

En nuestras pruebas, el sistema ha demostrado ser capaz de:

- Detectar anomalías con un tiempo de respuesta promedio de menos de 1 segundo después del inicio del ataque.
- Mantener una tasa de falsos positivos baja gracias al uso combinado de umbrales fijos y

estadísticos.

- Visualizar de forma clara y efectiva tanto el tráfico normal como los eventos anómalos.

7.2 Limitaciones

- El modelo asume que el tráfico normal sigue una distribución gaussiana, lo cual puede no ser válido en todos los escenarios reales.
- La detección se basa en volumen de tráfico, pero algunos ataques modernos pueden operar con volúmenes bajos pero patrones anómalos.
- La determinación del umbral óptimo requiere ajustes según las características específicas de cada red.

7.3 Posibles Mejoras

- Se podría implementar métodos adicionales de detección que consideren características más allá del volumen de tráfico
- Es posible que se incorporar aprendizaje automático para adaptar automáticamente los parámetros del detector.
- Se busca añadir capacidad de análisis post-mortem para examinar en detalle los incidentes detectados.

8. Conclusión

El detector de anomalías implementado demuestra la eficacia del algoritmo CUSUM para identificar cambios abruptos en el tráfico de red que podrían indicar ataques. La visualización en tiempo real y las alertas permiten a los operadores responder rápidamente ante posibles incidentes de seguridad.

La base matemática sólida de CUSUM, combinada con una implementación eficiente y una interfaz intuitiva, hacen de este sistema una herramienta valiosa para la monitorización de seguridad en redes.

Link Notion

https://www.notion.so/C-digo-Comentado-Detector-de-Anomal-as-en-Tr-fico-de-Red-1f363756db3c8014ba50e5affb2944eb?pvs=4

Referencias

Minitab Support. *"Interpretar los resultados clave para Gráfica CUSUM"*. Explicación de las gráficas CUSUM y su interpretación (control estadístico de procesos) .

Ubidots Help Center. *"Plugins: Anomaly Detector – Umbrales comunes para Z-score"*. Descripción de la regla general de detección de valores atípicos usando 3 desviaciones estándar 6 .

MSMK University. *"¿Qué es la detección de anomalías?"*. Resumen de métodos de detección de anomalías; define CUSUM como técnica para supervisar cambios acumulativos 1 .

Measurement Lab (M-Lab). *"CUSUM Anomaly Detection"* (2015). Documento técnico que aplica CUSUM a series temporales de red, mostrando su eficacia para detectar degradaciones de rendimiento 8 .

¿Qué es la detección de anomalías? - MSMK University

https://msmk.university/que-significa-desmagnetizar-msmk-university/

Detector de Anomalías en Tráfico de Red con CUSUM.pdf
file://file-TEW2SXvpHZWVT1moGCVqga

Interpretar los resultados clave para Gráfica CUSUM - Minitab
https://support.minitab.com/es-mx/minitab/help-and-how-to/quality-and-process-improvement/control-charts/how-to/timeweighted-charts/cusum-chart/interpret-the-results/key-results/

Plugins: Anomaly Detector | Ubidots Centro de Ayuda
https://help.ubidots.com/es/articles/8751084-plugins-anomaly-detector

measurem entlab.net
https://www.measurementlab.net/publications/CUSUMAnomalyDetection.pdf

Denning, D. E. (1987). An Intrusion-Detection Model. IEEE Transactions on Software Engineering, 13(2), 222-232. (Modelo pionero que introdujo la detección de intrusos mediante perfiles de comportamiento y umbrales estadísticos)