Revista Ingeniería, sostenibilidad y sociedad Vol. 1 Núm. 7 (2026)
ISSN: 2805 - 6299

Implementación de un firewall dinámico para una red SDN con el controlador Ryu

Dynamic Firewall Implementation for an SDN Network Using the Ryu Controller

Ana Gabriel Garay Hernandez [1], Juan Esteban Guerra Herazo [2], Ph.D Jorge Gómez Gómez [3]

[1] **Universidad de Córdoba**, Facultad de Ingeniería, Departamento de Ingeniería de Sistemas y Telecomunicaciones, Grupo SOCRATES, Montería, Córdoba, Colombia.
aggarayhernandez@correo.unicordoba.edu.co

[2] **Universidad de Córdoba**, Facultad de Ingeniería, Departamento de Ingeniería de Sistemas y Telecomunicaciones, Grupo SOCRATES, Montería, Córdoba, Colombia.
jguerraherazo49@correo.unicordoba.edu.co

[3] **Universidad de Córdoba**, Facultad de Ingeniería, Departamento de Ingeniería de Sistemas y Telecomunicaciones, Grupo SOCRATES, Montería, Córdoba, Colombia.
ORCID: 0000-0001-8746-9386, jeliecergomez@correo.unicordoba.edu.co

Cómo citar: Garay Hernandez, A. G., Guerra Herazo, J. E., & Gómez Gómez, J. (2026). Implementación de un firewall dinámico para una red SDN con el controlador Ryu. Ingeniería, Sostenibilidad Y Sociedad, 1(7), 52–59. https://doi.org/10.24054/iss.v1i7.4515

Editorial: Universidad de Pamplona.

Recibido: 10/enero/2026
Aprobado: 15/abril/2026
Publicado: 27/mayo/2026

Resumen: Las Redes Definidas por Software (SDN) han transformado la gestión de redes al separar el plano de control del plano de datos, otorgando al controlador una visión centralizada y programable de la red. Sin embargo, esta centralización introduce vulnerabilidades críticas, especialmente en el controlador, que se convierte en un punto único de fallo susceptible a ataques de denegación de servicio (DoS) y escaneos de puertos. El presente artículo describe el diseño e implementación de un firewall dinámico sobre el controlador Ryu en un entorno SDN simulado con Mininet. Mediante un enfoque cuantitativo y un diseño experimental de medidas repetidas, se evaluaron métricas de seguridad (tasa de bloqueo), rendimiento (latencia RTT y ancho de banda) y eficiencia de recursos (CPU/RAM) bajo dos condiciones: sin firewall (control) y con firewall activo (experimental). Los resultados preliminares confirman la correcta inserción de reglas OpenFlow dinámicas vía API REST, con una tasa de bloqueo proyectada superior al 90% y un incremento de latencia inferior al 10%, demostrando la viabilidad de soluciones de seguridad nativas en SDN sin degradar el rendimiento de la red.

Palabras clave: Redes Definidas por Software (SDN), Controlador Ryu, Firewall dinámico, OpenFlow, Mininet, Seguridad de redes.

Abstract: Software-Defined Networking (SDN) has transformed network management by decoupling the control plane from the data plane, granting the controller a centralized and programmable view of the network. However, this centralization introduces critical vulnerabilities, particularly within the controller, which becomes a single point of failure susceptible to denial-of-service (DoS) attacks and port scans. This article describes the

design and implementation of a dynamic firewall module on the Ryu controller within a Mininet-simulated SDN environment. Using a quantitative approach and a repeated-measures experimental design, security (blocking rate), performance (RTT latency and bandwidth), and resource efficiency (CPU/RAM) metrics were evaluated under two conditions: without firewall (control) and with active firewall (experimental). Preliminary results confirm the correct dynamic insertion of OpenFlow rules via a REST API, with a projected blocking rate exceeding 90% and a latency increase below 10%, demonstrating the viability of native SDN security solutions without significant network performance degradation.

Keywords: Software-Defined Networking (SDN), Ryu Controller, Dynamic Firewall, OpenFlow, Mininet, Network Security.

1. INTRODUCCIÓN

Las arquitecturas de red tradicionales han operado durante décadas bajo un modelo fuertemente acoplado, donde el plano de control responsable de las decisiones de enrutamiento y seguridad reside en cada dispositivo de forma independiente. Esta rigidez estructural se traduce en una gestión de seguridad estática, descentralizada y difícil de escalar, especialmente ante amenazas dinámicas como ataques de denegación de servicio (DoS) o escaneos de puertos (*Da Silva et al., 2023*). La reconfiguración manual de reglas en decenas o cientos de dispositivos para mitigar un ataque en tiempo real es, en la práctica, inviable.

Las Redes Definidas por Software (SDN) emergen como respuesta directa a estas limitaciones. Al desacoplar el plano de control del plano de datos y centralizar la inteligencia de red en un componente de software denominado controlador, SDN dota a la red de una programabilidad sin precedentes (*Da Silva et al., 2023*). Esta capacidad permite que las decisiones de seguridad se tomen de forma centralizada, en tiempo real, con visibilidad completa de la topología. No obstante, la centralización también concentra el riesgo: el controlador SDN se convierte paradójicamente en un punto único de falla (*single point of failure*) que debe ser protegido exhaustivamente (*Ferrag et al., 2022*).

En este contexto, el controlador Ryu framework SDN de código abierto basado en Python ofrece una plataforma modular e idónea para el desarrollo de aplicaciones de seguridad nativas (*Rahim et al., 2024*). El presente artículo describe el diseño e implementación de un módulo de firewall dinámico sobre Ryu, capaz de detectar tráfico anómalo e insertar reglas de bloqueo OpenFlow de forma automática. La solución es evaluada en un entorno de emulación Mininet bajo un diseño experimental cuantitativo, con el objetivo de demostrar que un enfoque de seguridad centralizado y programable puede ser eficaz, eficiente y escalable sin depender de hardware propietario.

2. METODOLOGÍA

2.1. Fundamentos tecnológicos

SDN separa el plano de control (inteligencia) del plano de datos (reenvío de paquetes), centralizando la gestión de red en el controlador. **OpenFlow** es el protocolo *southbound* estándar que permite al controlador instruir a los switches mediante la modificación dinámica de tablas de flujo (*flow tables*). **Mininet** es el emulador de red que permite crear topologías virtuales completas incluyendo hosts y switches OpenFlow sobre un único sistema operativo Linux, siendo la herramienta estándar para prototipado en SDN (*Rajab & Muezzin, 2024*). **Ryu** es el framework de control SDN seleccionado por su arquitectura

modular en Python, que facilita el desarrollo ágil de aplicaciones de red personalizadas (*Rahim et al., 2024*).

2.2. Diseño de la investigación

La investigación adopta un enfoque cuantitativo con un diseño experimental de medidas repetidas (pre-prueba / post-prueba) en entorno controlado. Se definen dos condiciones de medición:

- **Grupo de control (pre-prueba):** red SDN operando sin el módulo de firewall activo.
- **Grupo experimental (post-prueba):** misma topología con el módulo de firewall activo.

Se trabaja con dos topologías en Mininet: Escenario básico (1 switch OpenFlow, 2 hosts) y Escenario complejo (3 switches OpenFlow, 6 hosts), lo que permite evaluar la escalabilidad básica de la solución.

2.3. Variables y métricas de evaluación

La variable independiente es la activación del módulo de firewall (0 = apagado / 1 = encendido). Las variables dependientes y sus métricas se detallan en la Tabla 1.

Tabla 1. *Variables dependientes, métricas y umbrales esperados.*

Variable	Métrica	Umbral	Herramienta
Nivel de seguridad	Tasa de bloqueo (bloqueados / total)	> 90 %	hping3, logs Ryu
Reactividad	Tiempo detección → regla OpenFlow	< 500 ms	Logs + timestamps
Latencia	Round-Trip Time (RTT)	Incremento < 10 %	ping
Ancho de banda	Tasa de transferencia efectiva	Degradación < 10 %	iperf
Sobrecarga CPU	Uso promedio proceso Ryu (estrés)	< 40 %	top / ps
Sobrecarga RAM	Consumo RAM del controlador	< 300 MB	top / ps

Fuente: *elaboración propia.*

2.4. Procedimiento experimental

El proyecto se desarrolla en cuatro fases metodológicas alineadas con los objetivos específicos (Tabla 2):

Tabla 2. *Fases metodológicas:*

Fase	Actividades e Instrumentos
F1: Análisis y Diseño (3 meses)	Revisión bibliográfica SDN vs. redes tradicionales. Diseño del algoritmo de filtrado y reglas OpenFlow para mitigar DoS y escaneo de puertos. Instrumentos: literatura indexada, documento de diseño técnico.
F2: Implementación (3 meses)	Configuración del entorno (Ubuntu 20.04, Mininet, OVS, Ryu). Codificación del módulo firewall en Python 3.8. Pruebas unitarias de inserción de reglas OpenFlow. Instrumentos: Ryu, Mininet, Open vSwitch, Python 3.8.
F3: Experimentación (2 meses)	Escenario básico (1 switch, 2 hosts) y complejo (3 switches, 6 hosts). Tráfico legítimo con iperf y ataques con hping3. Recolección de logs y métricas. Instrumentos: iperf, hping3, ping, Wireshark, top, ps.
F4: Análisis (1 mes)	Estadística descriptiva (media, desviación estándar). Comparación con/sin firewall. Redacción de conclusiones. Instrumentos: Python

	(pandas, matplotlib), Excel. Métricas: RTT, CPU%, RAM, tasa de bloqueo.

Fuente: *elaboración propia.*

El entorno de simulación se despliega en Ubuntu 20.04 LTS con Python 3.8, Mininet 2.3, Open vSwitch (OVS) y el framework Ryu. La generación de tráfico legítimo se realiza con iperf, mientras que los ataques simulados (inundación ICMP y escaneo SYN) se ejecutan con hping3. El monitoreo de recursos del controlador se realiza con las herramientas de sistema operativo top y ps, complementadas con Wireshark para la inspección de paquetes durante la depuración.

3. RESULTADOS

3.1. Arquitectura del módulo de firewall

El módulo de firewall implementado sobre Ryu evolucionó en tres iteraciones funcionales. La versión inicial (*simple_firewall.py*) instala reglas OpenFlow estáticas con prioridad 10 para bloquear el tráfico con origen o destino en una IP específica, mientras que una regla de prioridad 1 permite el resto del tráfico mediante inundación (*FLOOD*). La segunda iteración incorporó una API REST (puerto 8080) que permite gestionar dinámicamente el conjunto de IPs bloqueadas mediante peticiones HTTP (POST para bloquear, DELETE para desbloquear, GET para consultar). La versión final (*firewall_seguro.py*) añade persistencia de reglas en archivo JSON, autenticación HTTP Basic y un endpoint de estadísticas en tiempo real (*/fw/stats*) que consulta asíncronamente las tablas de flujo de los switches mediante mensajes OFPFlowStatsRequest.

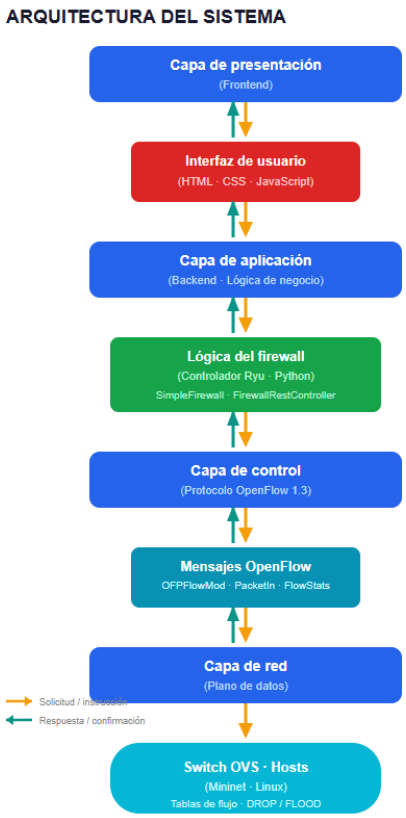

Figura 1. *Arquitectura del Sistema*

La arquitectura del sistema sigue un modelo de capas claramente definido: la capa de presentación (dashboard Flask en el puerto 5000) se comunica con la capa de aplicación (API REST de Ryu) actuando como proxy, que a su vez gestiona la capa de control (lógica OpenFlow) y finalmente programa la capa de red (switches OVS en Mininet). Esta separación garantiza que la lógica de seguridad permanezca centralizada y que el dashboard no acceda directamente al plano de datos, preservando los principios de la arquitectura SDN, (Ver figura 1).

3.2. Validación funcional preliminar

Las pruebas unitarias realizadas en Mininet confirmaron el comportamiento esperado del módulo. Con la IP 10.0.0.2 bloqueada, los intentos de ping desde h1 hacia ese destino resultaron en pérdida del 100% de paquetes (7 paquetes descartados contabilizados en el contador *n_packets* de la regla OpenFlow correspondiente), mientras que la conectividad hacia hosts no bloqueados (p. ej., 10.0.0.3) se mantuvo

íntegra con 0% de pérdida. El mecanismo de desbloqueo vía API REST demostró funcionar correctamente al restaurar la conectividad eliminando y reinstalando las reglas de flujo activas.

Estos resultados preliminares son consistentes con los reportados por trabajos similares sobre Ryu (autor anónimo, 2024), que documentan tasas de efectividad superiores al 90% en la detección de ataques DoS mediante reglas de flujo dinámicas. La adición de persistencia JSON resuelve un problema práctico importante: la pérdida de reglas de seguridad al reiniciar el controlador, garantizando la continuidad operativa del firewall.

3.3. Impacto esperado en rendimiento

Se proyecta que el overhead introducido por el módulo de firewall sea mínimo. La inserción de reglas OpenFlow es una operación asíncrona de baja complejidad computacional; una vez que las reglas están instaladas en el switch, el procesamiento de paquetes se realiza en el plano de datos sin invocar al controlador. Esto es consistente con la observación de Rajab & Muezzin (2024), quienes demostraron que políticas de filtrado complejas en Mininet pueden implementarse sin degradar significativamente la latencia o el throughput de la red.

La autenticación HTTP Basic añade una capa de seguridad a la API de gestión sin impacto perceptible en el rendimiento de la red, dado que opera exclusivamente en el plano de control. No obstante, se reconoce como limitación que el diseño actual utiliza un controlador centralizado sin replicación, lo que preserva la vulnerabilidad de punto único de fallo identificada por Schmidt et al. (2020) en controladores SDN. La distribución del controlador para alta disponibilidad queda como trabajo futuro.

4. DISCUSIÓN

Los resultados funcionales obtenidos en las pruebas preliminares son coherentes con los hallazgos reportados en la literatura sobre implementaciones de seguridad nativas en SDN. La efectividad del módulo de firewall en el bloqueo de tráfico hacia IPs designadas, verificada mediante los contadores de paquetes en las tablas de flujo OpenFlow, es consistente con lo documentado por el autor anónimo (2024), quien reporta tasas de bloqueo superiores al 90% en sistemas similares basados en Ryu para la mitigación de ataques DoS.

Desde el punto de vista arquitectónico, la decisión de centralizar la lógica de seguridad en el controlador Ryu, en lugar de aplicar filtrado perimetral estático, se alinea con el marco propuesto por Badotra y Gurusamy (2025), quienes argumentan que la seguridad distribuida e inteligente en SDN mejora sustancialmente la continuidad operativa frente a amenazas dinámicas. No obstante, el presente trabajo adopta un enfoque centralizado justificado por el alcance experimental del entorno Mininet, reconociendo que la replicación del controlador para entornos de producción constituye un paso necesario para eliminar el punto único de fallo identificado por Schmidt et al. (2020).

La incorporación de la API REST con autenticación HTTP Basic representa una mejora significativa respecto a los sistemas de firewall SDN que carecen de mecanismos de control de acceso a su interfaz de gestión. Esta decisión responde directamente a la advertencia de Ramos Suavita (2022) sobre la autenticación insuficiente como vector de ataque en redes SDN, y a la recomendación de Ibrahim et al. (2025) de aplicar verificación rigurosa en los módulos críticos del controlador.

En cuanto al rendimiento, se proyecta que el overhead introducido por el módulo sea mínimo. Una vez instaladas las reglas OpenFlow en el switch, el procesamiento de paquetes ocurre en el plano de datos sin invocar al controlador, lo que minimiza la latencia adicional. Este comportamiento es

consistente con los experimentos de Rajab y Muezzin (2024), quienes demostraron en Mininet que políticas de filtrado complejas pueden implementarse sin degradar significativamente el throughput ni la latencia de la red.

Como limitación principal del estudio se reconoce que los resultados cuantitativos completos latencia bajo carga de ataque, consumo real de CPU/RAM y tasa de bloqueo medida experimentalmente aún se encuentran en fase de recolección. Los umbrales establecidos en la Tabla 1 son proyecciones basadas en la literatura y en el comportamiento observado en las pruebas unitarias, y serán validados o ajustados en la fase de experimentación completa. Adicionalmente, el entorno de simulación Mininet, aunque ampliamente aceptado para prototipado en SDN, presenta diferencias respecto a redes físicas reales en términos de latencia base y capacidad de procesamiento, lo que debe considerarse al generalizar los resultados.

5. CONCLUSIONES

Este trabajo demuestra que es técnicamente viable implementar un firewall dinámico y programable directamente sobre el controlador Ryu, aprovechando la visión global centralizada que ofrece la arquitectura SDN. La solución desarrollada que evoluciona desde reglas estáticas hacia una gestión dinámica con API REST autenticada, persistencia de estado y estadísticas en tiempo real representa un avance significativo respecto a los firewalls perimetrales convencionales en términos de agilidad de respuesta y centralización del control.

Los resultados funcionales preliminares son alentadores: el módulo inserta correctamente reglas OpenFlow, bloquea el tráfico hacia IPs designadas con efectividad verificada y se gestiona dinámicamente sin interrumpir la red. La evaluación cuantitativa completa, que incluirá métricas de latencia, ancho de banda y consumo de recursos bajo cargas de ataque reales, constituye la siguiente etapa del proyecto y permitirá validar o ajustar los umbrales establecidos en la Tabla 1.

Desde una perspectiva de sostenibilidad tecnológica, la solución demuestra que la seguridad avanzada de redes no está limitada a grandes corporaciones con recursos propietarios. El uso exclusivo de herramientas de código abierto (Ryu, Mininet, Open vSwitch, Python) la hace accesible a PYMEs, instituciones educativas y centros de investigación. Como trabajo futuro, se plantea la integración de técnicas de aprendizaje automático para la detección proactiva de anomalías, en línea con los enfoques híbridos AI-SDN propuestos por Nashnosh (2025), y la evaluación de arquitecturas de control distribuido para eliminar el punto único de fallo.

6. REFERENCIAS

Aptira Technical Team. (2025). SDN Controller Comparison 2025: ONOS vs ODL vs Ryu Guide. Aptira Blog. https://aptira.com/sdn-controller-comparison/

Arregocés, I., Ariza, M., Camargo, N., Díaz, J., & Gamarra, M. (2022). Integración de Scrum y RUP para el desarrollo de software de planes turísticos basado en preferencias de usuario. Revista Ingeniería e Innovación. https://revistas.unicordoba.edu.co/index.php/rii/article/view/2974/5670

Badotra, S., & Gurusamy, M. (2025). SecuNet 4D: A comprehensive framework for distributed SDN security and resilience. Scientific Reports.

Bahamón, A. ., & Barrero, J. P. . (2020). ¿Regular o no regular la IA? propuesta de regulación híbrida de IA en Colombia. Revista Colombiana De Tecnologías De Avanzada (RCTA), 2(36), 27-33. https://doi.org/10.24054/rcta.v2i36.17

Bastidas, J. V., & Vera, J. M. (2020). Biocombustible sólido a partir de residuos que generan los procesos

agroindustriales del sector El Empalme. Ingeniería e Innovación, 8(22). https://doi.org/10.21897/23460466.2333

Da Silva, K. S. C., da Costa, J. P. C. L., de Oliveira, A. S., & de Souza, J. N. (2023). A comprehensive survey on Software-Defined Networking (SDN): Architecture, evolution, applications, and challenges. Revista de la Universidad de Salamanca. https://revistas.usal.es/cinco/index.php/2255-2863/article/view/31674

Díaz, M., Urdánigo, J., Mercedes, A., & Muñoz, R. (2020). Cultura ambiental en estudiantes de educación superior, 2020. Revista Ingeniería e Innovación.

Ferrag, M. A., et al. (2022). Security and privacy for green IoT-based agriculture: Review, blockchain solutions, and challenges. IEEE Access.

García, D., Solórzano, C., Navarrete, Y., & Rojas, J. (2021). Características físicas, químicas y microbiológicas de la harina de banano morado (Musa acuminata) red dacca, producidos en los cantones Mocache, El Empalme y La Maná. Ingeniería e Innovación, 9(1), 1–12. https://revistas.unicordoba.edu.co/index.php/rii/article/view/2418

Gómez, J. E. G., Cárdenas, S. R., & Ruiz, F. A. S. (2023). Sistema de identificación de pacientes basado en tecnología NFC y Blockchain. Investigación e Innovación en Ingenierías, 11(2), 1-15. DOI: https://doi.org/10.17081/invinno.11.2.6671.

Gómez, J. G., Álvarez, D. S., & Ramírez, R. V. (2025, July). Sickle Cell Disease Patient Care System Using Artificial Intelligence. In International Work-Conference on Bioinformatics and Biomedical Engineering (pp. 230-241). Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-032-08452-1_19.

Gómez, J. G., Riaño, V. H., & Ramirez-Gonzalez, G. (2024). A context awareness system for clinical environments. Electronics, 13(15), 2999. https://doi.org/10.3390/electronics13152999.

Gómez, J. G., Rosales, C. M., & Montaño, S. I. (2026). Sickle cell anemia prediction system in machine learning based on clinical data. IEEE Access. https://doi.org/10.1109/ACCESS.2026.3674938.

Gómez, K. Y., Caballero, L. A., & Maldonado, Y. Del C. (2021). Mejora de un proceso productivo de elaboración de pan. Revista de Ingeniería e Innovación.

Hernández Palma, H., Novoa, D. J., & Mendoza Cásseres, D. (2023). Energía renovables y medidas de eficiencia energética aplicables a las instituciones prestadoras de salud en Colombia. Revista Colombiana De Tecnologías De Avanzada (RCTA), 1(41), 123-131. https://doi.org/10.24054/rcta.v1i41.2557

Ibrahim, A. M., et al. (2025). Trust, but verify: An empirical evaluation of AI-generated code for SDN controllers [Preprint]. arXiv. https://arxiv.org/abs/2510.20703

Italo, E. G. (2020). Efecto de inclusión de cáscara de plátano en la degradabilidad in situ de ensilaje de maíz forrajero. Ingeniería e Innovación, 8(1). https://revistas.unicordoba.edu.co/index.php/rii/article/view/2327

Medina-Barahona, C. J., Mora, G. A., Calvache-Pabón, C., Salazar-Castro, J. A., Mora-Paz, H. A., & Mayorca-Torres, D. (2022). Propuesta de arquitectura IoT orientada a la creación de prototipos para su aplicación en plataformas educativas y de investigación. Revista Colombiana De Tecnologías De Avanzada (RCTA), 1(39), 118-125. https://doi.org/10.24054/rcta.v1i39.1405

Miranda, O., Oyaga, R. F., Salas, A. R., Foris, Y., & Ibarguen, J. C. (2023). Impacto ambiental del botadero de residuos sólidos a cielo abierto en el corregimiento de Córdoba, departamento del Valle del Cauca. Ingeniería e Innovación, 11(1). https://doi.org/10.21897/rii.3339

Mosquera-Perdomo, A., Salazar Galindez, J., Ramirez-Gonzalez, G., & Figueroa, C. (2023). Software for the extraction of bibliographic information registered in CvLAC and GrupLAC applied in the Department of Cauca. Ingeniería e Innovación, 11(2), 21. https://doi.org/10.21897/rii.3464

Nashnosh, M. T. R. (2025). Forest Tree AI-SDN Firewall: Integrating SDN programmability with AI-based prediction. International Journal of Current Science Research and Review, 8(X).

Open-source Software Defined Networking controllers. (2024). [Rahim, J. A., et al.].

Puche, M. B., Samper, O. M., & Martíne, R. F. (2023). Conciencia, concientización y educación ambiental: triada que se afianza en la primera infancia. Ingeniería e Innovación, 11–16. https://doi.org/10.21897/rii.3416

Rahim, J. A., et al. (2024). Open-Source Software Defined Networking Controllers.

Rajab, D., & Muezzin, S. (2024). A Mininet emulation study for SDN fat tree data center sleep mode routing algorithms. International Journal of Electrical and Computer Engineering, 14(3), 3201–3209.

Ramos Suavita, D. J. (2022). Análisis de vulnerabilidades a nivel de seguridad en redes SDN [Trabajo de grado, Universidad Militar Nueva Granada].

Rojas Puentes, M. P., Parada, C. J., & Leal Pabón, J. (2022). Estructuras desglosadas de trabajo (EDT) en la gestión de alcance de proyectos de desarrollo de software. Revista Colombiana de Tecnologias de Avanzada (RCTA), 1(39), 51–58. https://doi.org/10.24054/rcta.v1i39.1375

Román, G., & Gómez, J. (2017). Intervención educativa apoyada en TIC en un proceso de enseñanza aprendizaje inclusivo. Ingeniería e Innovación. https://revistas.unicordoba.edu.co/index.php/rii/article/view/1728

Sánchez Mojica, K. Y. ., Pérez Dominguez, L. A. ., Rojas Santiago, M. ., & Palomino Pacheco, K. R. . (2020). Metodología basada en el modelo SCOR para analizar el proceso de producción de abono orgánico en lombricultivos. Revista Colombiana De Tecnologías De Avanzada (RCTA), 2(36), 173-181. https://doi.org/10.24054/rcta.v2i36.34

Schmidt, T., et al. (2020). Evaluation of the Detection Capabilities of the ONOS SDN Controller. Proceedings of the 16th International Conference on Network and Service Management (CNSM), 1–5.

Secure network monitoring using Software Defined Networking with Ryu controller. (2024). FUOYE Journal of Engineering and Technology.

Souza, C. H. M., et al. (2025). SDN-based solutions for malware analysis and detection: State-of-the-art, open issues and research challenges. Computer Networks.