Syrian Private University
Faculty of Engineering
Computer systems and
network security engineering

الجامعة السورية الخاصة
كلية الهندسة
هندسة أمن النظم والشبكات الحاسوبية

مشروع أعد لنيل شهادة الهندسة في تخصص أمن النظم والشبكات الحاسوبية (تخرج ١)

Design of IDS for detecting SQL injection in a SIEM system

تصميم نظام كشف تسلل لرصد هجمات SQL ضمن بيئة SIEM

إعداد :

أحمد عبدالرحمن خلوف

طلال محمد الحلو

إشراف :

د . وسيم جنيدي

٢٠٢٥

الملخص :

يهدف هذا المشروع إلى تصميم وتنفيذ نظام كشف تسلل (IDS) متخصص لرصد هجمات حقن قواعد البيانات (SQL Injection)ضمن بيئة تحليل أحداث أمنية مركزية(SIEM) ، من خلال دمج الكشف الشبكي القائم على التواقيع مع التحليل والترابط المركزي للأحداث. يعتمد الحل المقترح على استخدام Suricata كنظام كشف تسلل شبكي لمراقبة حركة HTTP وتحليلها واكتشاف أنماط هجماتSQL Injection ، ثم تمرير التنبيهات الناتجة بصيغة JSON إلى منصة Wazuh التي تقوم بدورها بتجميع السجلات، وفك ترميزها، وتصنيفها، وربطها ضمن سياق أمني موحد.

تم بناء بيئة اختبار مخبرية واقعية باستخدام تطبيق ويب ضعيف أمنيًا (DVWA) يعمل فوق خادم Apache وقاعدة بياناتMySQL ، وذلك لتوليد سيناريوهات إدخال تمثل هجمات SQL Injection بمختلف أنواعها، بما في ذلك In-band وBlind/Inferential، مع استثناء هجمات .Out-of-Band اعتمد المشروع على تصميم طبقي واضح يبدأ من توليد الحركة، مرورًا بالكشف الشبكي، ثم التجميع، فالتحليل والتصنيف، وصولًا إلى التخزين والعرض عبر لوحة تحكم.Wazuh Dashboard

أظهرت نتائج الاختبارات نجاح التكامل من طرف إلى طرف، حيث تم رصد الهجمات، وتصنيفها بدقة، وعرضها ضمن منصة SIEM مع الاحتفاظ بالسياق التشغيلي الكامل لكل حدث. يثبت هذا العمل أن دمج IDS متخصص مع SIEM يساهم في تحسين دقة الكشف، تقليل الضوضاء التحليلية، وتسريع الاستجابة للحوادث مقارنة بالحلول المنعزلة، مما يجعله قابلًا للتطبيق في البيئات المؤسسية الواقعية .

Abstract

This project aims to design and implement a specialized Intrusion Detection System (IDS) for detecting SQL Injection attacks within a centralized Security Information and Event Management (SIEM) environment. The proposed solution combines signature-based network detection with centralized event analysis and correlation to enhance detection accuracy and incident visibility.

The system leverages Suricata as a Network-based IDS to monitor HTTP traffic, inspect requests, and detect SQL Injection patterns. Detected alerts are generated in JSON format and forwarded to the Wazuh SIEM platform, where events are collected, decoded, classified, correlated, and stored for analysis and visualization. A realistic laboratory testbed was constructed using the Damn Vulnerable Web

Application (DVWA) deployed on an Apache web server with a MySQL backend database, enabling the generation of realistic SQL Injection attack scenarios.

The proposed architecture follows a layered design, including traffic generation, network detection, log collection, analytics and classification, and storage and presentation layers. Experimental results demonstrate successful end-to-end integration, accurate detection of multiple SQL Injection categories (In-band and Blind/Inferential), and effective visualization of alerts within the SIEM dashboard while preserving full operational context.

The findings confirm that integrating a specialized IDS with a SIEM platform significantly improves detection precision, reduces false positives, and enhances incident response capabilities compared to standalone detection solutions, making the approach suitable for real-world enterprise security environments.

فهرس المحتويات :

المحتويات

فهرس الاشكال:

فهرس الجداول :

قائمة المصطلحات :

المصطلح	التعريف
SQL Injection (SQLi)	أسلوب هجوم يستغل ضعف التحقق من مدخلات المستخدم لإدخال أوامر SQL خبيثة ضمن استعلامات قاعدة البيانات.
Intrusion Detection System (IDS)	نظام أمني يقوم بمراقبة وتحليل الأنشطة لاكتشاف السلوكيات غير المصرح بها أو الهجمات المحتملة.
Security Information and Event Management (SIEM)	منصة مركزية لجمع، تحليل، ربط، وتخزين الأحداث الأمنية من مصادر متعددة.
Network-based IDS (NIDS)	نوع من أنظمة IDS يعمل على مراقبة حركة المرور الشبكية لاكتشاف الهجمات.
Host-based IDS (HIDS)	نظام IDS يعمل على مستوى الجهاز المضيف لمراقبة السجلات وسلوك النظام.
Suricata	نظام كشف تسلل شبكي مفتوح المصدر يعتمد على التواقيع لتحليل حركة الشبكة.
Wazuh	منصة SIEM مفتوحة المصدر تُستخدم لجمع السجلات، تحليلها، وتصنيف التنبيهات الأمنية.
Alert	إشعار أمني يتم توليده عند اكتشاف نشاط مريب أو نمط هجوم.
Signature-based Detection	آلية كشف تعتمد على مطابقة الأنشطة مع تواقيع معروفة للهجمات.
Anomaly-based Detection	أسلوب كشف يعتمد على اكتشاف الانحراف عن السلوك الطبيعي للنظام.
In-band SQL Injection	نوع من هجمات SQLi يحصل فيه المهاجم على نتائج مباشرة من استجابة الخادم.
Blind SQL Injection	هجوم SQLi يعتمد على الاستجابات المنطقية أو الزمنية دون عرض البيانات مباشرة.

Time-based SQL Injection	نوع من Blind SQLi يعتمد على تأخير الاستجابة الزمنية لاستخلاص المعلومات.
Boolean-based SQL Injection	أسلوب Blind SQLi يعتمد على شروط منطقية صحيحة أو خاطئة.
Piggybacked Query	هجوم SQLi يتم فيه تنفيذ أكثر من استعلام في طلب واحد.
Union-based SQL Injection	نوع من In-band SQLi يستخدم UNION لدمج نتائج استعلامات خبيثة.
Error-based SQL Injection	هجوم يعتمد على رسائل الخطأ الصادرة من قاعدة البيانات لاستخلاص المعلومات.
Out-of-Band (OOB) SQL Injection	هجوم SQLi يعتمد على قنوات خارجية مثل DNS أو HTTP callbacks.
Log	سجل يحتوي على أحداث وعمليات النظام أو التطبيق.
JSON EVE	صيغة سجل تعتمدها Suricata لتخزين الأحداث والتنبيهات.
Correlation	عملية ربط أحداث متعددة من مصادر مختلفة لاستخلاص سياق أمني موحد.
False Positives	تنبيهات تشير إلى هجوم غير حقيقي.
Dashboard	واجهة رسومية لعرض وتحليل الأحداث الأمنية.
Attack Lifecycle	تسلسل مراحل الهجوم من الاستطلاع إلى الاستغلال وما بعده.
Layered Architecture	تصميم يعتمد على تقسيم النظام إلى طبقات مستقلة ذات وظائف محددة.
SOC (Security Operations Center)	مركز عمليات أمنية مسؤول عن مراقبة وإدارة الحوادث الأمنية.

١.١ المقدمة :

في عصر التحوّل الرقمي المتسارع، أصبحت الأنظمة الحاسوبية تُشكّل العمود الفقري لعمل المؤسسات في مختلف القطاعات. ومع تزايد الاعتماد على الخدمات الإلكترونية والبنى التحتية الرقمية، ارتفعت أهمية حماية هذه الأنظمة من التهديدات السيبرانية. إن اختصاص «أمن النظم» يُعنى بضمان استمرارية عمل الأنظمة الحاسوبية، حماية البيانات من الوصول غير المصرّح به، وتأمين الموارد الحسّاسة ضد الهجمات التي قد تؤدّي إلى تعطيل الخدمة أو خسارة المعلومات. في هذا الإطار، تُعدّ أنظمة الكشف عن التسلّل(IDS) ، ونظم تحليل الأحداث الأمنية (SIEM) من الأدوات الحيوية التي تسهم في تعزيز منظومة الدفاع الأمني للمؤسسات.

من بين أنواع الهجمات السيبرانية التي تستهدف قواعد البيانات وتطبيقات الويب، يبرز هجوم (SQL Injection) كأحد الأخطر والأكثر شيوعًا. يتم هذا الهجوم حين يُدخِل المهاجم تعليمات SQL خبيثة في مدخلات المستخدم أو في المتغيرات التي تُركّب في استعلامات قاعدة البيانات، ما يسمح له بتجاوز الحماية، استخراج بيانات حسّاسة، تعديلها أو حذفها. تُشير الأدلة إلى أن ضعف التحقق من المدخلات (input validation)، والاستخدام غير المُحكم لاستعلامات ديناميكية، وعدم تفعيل مبدأ الأقلّ امتيازًا (least privilege) يزيدان من قابلية التطبيق للتعرض لهذا النوع من الهجمات وبالتالي، فإن التصدي لهجوم حقن SQL يمثّل ضرورة أمنية ملحّة لحماية البيانات وضمان استقرار الأنظمة. من بين أنواع الهجمات السيبرانية التي تستهدف قواعد البيانات وتطبيقات الويب، يبرز هجوم (SQL Injection) كأحد الأخطر والأكثر شيوعًا. يتم هذا الهجوم حين يُدخِل المهاجم تعليمات SQL خبيثة في مدخلات المستخدم أو في المتغيرات التي تُركّب في استعلامات قاعدة البيانات، ما يسمح له بتجاوز الحماية، استخراج بيانات حسّاسة، تعديلها أو حذفها. وبالتالي، فإن التصدي لهجوم حقن SQL يمثّل ضرورة أمنية ملحّة لحماية البيانات وضمان استقرار الأنظمة.

على الرغم من توافر أدوات تقنية متعدّدة لمراقبة الشبكات وتحليل الأحداث، إلا أن العديد من المؤسسات لا تزال تعاني من صعوبة في ربط تنبيهات نشاط الشبكة المشتبه به مع الأحداث المخزنة في قواعد البيانات والتطبيقات .بعبارة أخرى: رغم وجود نظام SIEM قادر على جمع وتحليل السجلات، فإن الكشف استباقيًا لهجمات الحقن ليس دائمًا مهيّأً بشكل كافٍ. كما أن تحليل النشاط في الزمن الحقيقي واتخاذ إجراءات تلقائية ما تزال تشكّل تحديًا في البيئات الحقيقية. من هنا تنشأ الحاجة إلى تصميم وحدة IDS مخصصة لاكتشاف حقن SQL داخل بيئة SIEM وتُفعّل الربط بين تنبيهات قاعدة البيانات ونظام تحليل الأحداث لضمان توقيت استجابة أسرع ودقة أعلى في الكشف.

١.٢ الهدف من البحث :

يهدف هذا البحث إلى تصميم وتنفيذ نظام كشف تسلّل (IDS) مخصّصاً لاكتشاف (SQL Injection) ضمن بيئة تحليل أحداث أمنية (SIEM) في الزمن الحقيقي، بحيث يعزّز قدرة المؤسسة على اكتشاف محاولات التسلّل والتلاعب بقاعدة البيانات، وربط التنبيهات المستخلصة من الشبكة والتطبيقات بقواعد بيانات المؤسسة لتحليل مركزي وتحقيق استجابة أوتوماتيكية.

من جهة، يُعالج البحث الفجوة التي تفيد بأنه رغم وجود منصّات SIEM وأنظمة IDS منفصلة، إلا أن الربط بينها لا يزال محدوداً في كثير من الحالات، مما يقلّل من قدرة الرصد الاستباقي للتهديدات. ومن جهة أخرى، يستند البحث إلى التطوّرات الحديثة في الكشف مثل استخدام تحليل أنماط حركة البيانات الصادرة

(outbound traffic) لاكتشاف هجمات حقن SQL بدقة عالية، كما في دراسة قدمت نسبة دقة بلغت أكثر من ٩٨٪ باستخدام هذه الطريقة. [1]

كما يُراعي البحث التطوّرات في أساليب الهجوم المعقّدة والمتغيّرة بسرعة، والتي تتجاوز الأساليب التقليدية القائمة على قواعد ثابتة فقط، ويُقترح دمج تقنيات مثل التعلم الآلي أو تحليل الأنماط المتقدمة لزيادة فعاليّة الكشف وتقليل التنبيهات الكاذبة.

بالتالي، النتائج المنتظَرة لهذا المشروع تشمل: تحسين دقة كشف هجمات حقنSQL ، تقليل زمن الاستجابة للحوادث، وتوفير بيئة اختبار قابلة للتطبيق العملي في سياقات المؤسسات.

١.٣ التطبيقات العملية :

يمكن لهذا النظام أن يُطبّق عملياً في عدة بيئات ومجالات تشغّل بنية تحتية تعتمد على قواعد بيانات وتطبيقات ويب، مثل المؤسسات المالية، ومراكز البيانات، وشركات التجارة الإلكترونية، أو حتى في المراكز الأمنية (SOC)
على سبيل المثال:

في مؤسسة تستخدم قاعدة بيانات خلفية لتطبيق الويب، يمكن لـ IDS أن يراقب الاستعلامات الواردة والصادرة، ويرسل التنبيهات إلى منصة SIEM عند رصد نمط يشير إلى حقن.SQL

يمكن أيضاً دمج النظام مباشرة مع لوحة عرض (Dashboard) في SIEM تعرض مؤشرات مثل: عدد محاولات الحقن خلال آخر ٢٤ ساعة، عناوين الـ IPالمكرّرة، أو الحقول التي تم استهدافها.

إضافة إلى ذلك، يمكن تنفيذ استجابة تلقائية، مثل: حظر عنوان IP المهاجم، تعطيل حساب المستخدم المشبوه، أو جلب تقرير تلقائي لمشرف الأمن.

أيضاً،

هذا النظام قابل للتوسعة ليعمل في بيئة متعددة الأجهزة أو متعددة الشبكات(Distributed Deployment) ، حيث تجمع بيانات IDS متعددة في منصة SIEM واحدة لتحليل مركزي، مما يعزّز قدرة المؤسسة على التعامل مع هجمات موزّعة أو تنسيقية. دراسة حديثة تشير إلى إمكانية استخدام حركة البيانات الصادرة (outbound traffic)كعنصر رئيسي في الكشف، ما يجعل التطبيق العملي أكثر فعالية حتى في بيئات الشركات.

١.٤ التحديات :

تنفيذ مثل هذا النظام يواجه عدة تحديات تقنية ومنهجية، ومنها:

١. التنوّع والتطوّر في أنماط هجمات :SQL Injection هجمات الحقن لا تقتصر على نمط محدد، بل تتضمّن أنواعاً مثلIn-Band ، Blind, Time-Based، أو حتى حقن عبر بروتوكولات غير مباشرة، ما يجعل بناء قاعدة توقيع (signature) واحدة فعالة غير كافٍ.

٢. الدمج والتكامل مع منصة :SIEM ليس كافياً أن يُكتشف الهجوم داخل IDS فقط، بل يجب أن يُنقل التنبيه إلى منصةSIEM ، تُحلّل ارتباطه مع أحداث أخرى (مثل تسجيل دخول فاشل، تغييرات قاعدة البيانات، حركة مشبوهة) وتتخذ إجراءاً. هذا يتطلّب تنسيقاً بين الأدوات، وضبط بروتوكولات الإرسال، وتوحيد تنسيق السجلات.(logs)

٣. تقليل التنبيهات الكاذبة وتحسين صيانة النظام :قواعد الكشف تحتاج إلى تحديث مستمر لمتابعة الهجمات الجديدة، كما أن التنبيهات الكاذبة إن ارتفعت بفعل قدرات النظر غير الدقيقة، ستقود إلى تجاهل التنبيهات الحقيقية. دراسة عن تصميم التوقيعات بـ IDSأظهرت أن ضعف التوقيعات يُضعف كفاءة النظام بشدة.

الفصل الأول : الدراسات النظرية

٢.١ المقدمة :

في العصر الرقمي الحديث أصبحت الأنظمة المعلوماتية تمثّل العمود الفقري لمختلف المؤسسات الحكومية والخاصّة. وتعتمد جميع عملياتها اليومية – من المعاملات المالية إلى إدارة البيانات الحسّاسة – على بنية تحتية مترابطة من الخوادم والشبكات وقواعد البيانات. ومع ازدياد هذا الاعتماد، تصاعدت في المقابل التهديدات السيبرانية التي تستهدف استغلال الثغرات التقنية والبشرية لتحقيق مكاسب غير مشروعة أو تعطيل الخدمات الحيوية.

هنا يبرز مفهوم أمن النظم والمعلومات (Information and System Security) بوصفه علمًا يهدف إلى حماية أصول المعلومات والأنظمة التي تُعالجها من أي خطر يمكن أن يؤثر على سريّتها أو سلامتها أو توافرها. ويعرّف المعهد القومي للمعايير والتقنية (NIST) أمن المعلومات بأنه: "حماية نظم المعلومات وبياناتها من الوصول أو الاستخدام أو الإفصاح أو التعديل أو التعطيل أو التدمير غير المصرّح به."

يرتكز أمن المعلومات على ما يُعرف بـ مثلث الـ(Confidentiality – Integrity – Availability) CIA ، وهو النموذج الذي يشكّل الأساس لجميع السياسات الأمنية المعتمدة في المؤسسات:

١. السرية :(Confidentiality) ضمان عدم وصول البيانات إلّا للمستخدمين المصرّح لهم بذلك.
٢. السلامة :(Integrity) التأكد من عدم تعديل البيانات أو إتلافها أثناء التخزين أو النقل.
٣. التوافر :(Availability) التأكد من بقاء الأنظمة والبيانات متاحة عند الحاجة إليها دون تعطيل.

ولتحقيق هذه الأهداف تعتمد المؤسسات على منهجية الدفاع المتعدد الطبقات(Defense in Depth) ، والتي تقوم على بناء طبقات حماية متتابعة تشمل جدران الحماية(Firewalls) ، أنظمة كشف التسلل(IDS/IPS) ، أنظمة إدارة الأحداث الأمنية(SIEM) ، وأنظمة التحكم في الوصول .(Access Control) والهدف من ذلك هو ألا يشكّل اختراق طبقة واحدة انهيارًا للنظام بالكامل.

من جانب آخر، تتنوّع التهديدات الأمنية ما بين برمجيات خبيثة(Malware) ، وهجمات الحرمان من الخدمة (DoS / DDoS)، ومحاولات التصيّد(Phishing) ، وحقن الأكواد في التطبيقات (Code Injection) بما فيها هجمات حقن SQL التي تُعد من أخطرها على الإطلاق. وتشير تقارير (IBM X-Force Threat Intelligence 2024)إلى أنّ أكثر من ٦٠ ٪ من الثغرات المسجّلة ترتبط بسوء تأمين تطبيقات الويب أو ضعف آليات التحقق من إدخال المستخدمين.

تُظهر هذه المعطيات أن أمن النظم لم يعد خيارًا إضافيًا بل أصبح ضرورة استراتيجية لضمان استمرارية الأعمال وثقة المستخدمين. ولهذا طوّرت المنظمات الدولية مثل ISOو NIST و OWASP أطرًا ومعايير إجرائية تهدف إلى إنشاء سياسات أمنية متكاملة، تشمل تحديد المخاطر(Risk Assessment) ، وتطبيق الضوابط التقنية(Security Controls) ، والتعامل مع الحوادث الأمنية.(Incident Response)

ضمن هذا السياق يأتي مشروعنا ، الذي يستند إلى أحد أهم مفاهيم الحماية وهو الكشف المبكر عن التسلّل (Intrusion Detection)، وذلك من خلال تصميم نظام IDS قادر على اكتشاف محاولات حقن SQL وتحليلها بالتكامل مع منصة SIEM لتقديم رؤية شاملة للأنشطة الأمنية داخل البيئة المستهدفة.

٢.٢ مفاهيم الأمن السيبراني(امن المعلومات) والبنى الدفاعية

يُعدّ الأمن السيبراني(Cybersecurity) أو أمن المعلومات بصيغة أكبر الإطار الأشمل الذي يندرج ضمنه أمن النظم والمعلومات، ويُعنى بحماية الأنظمة الرقمية من الهجمات التي تستهدف سرقة المعلومات أو تعطيل العمليات أو إتلاف البنية التحتية التقنية. يُعرّفه NIST بأنه" :مجموعة من الأنشطة والضوابط المصمَّمة لحماية نظم المعلومات من التهديدات الرقمية وضمان استمرارية العمليات ."ويشمل هذا المجال تحليل المخاطر، والاكتشاف المبكر للثغرات، والاستجابة الفورية للحوادث، والتعافي من آثار الهجوم.

تقوم البنية الأمنية في المؤسسات عادةً على مبدأ يُعرف باسم الأمن الطبقي (Layered Security) أو ما يُسمّى Defense in Depth، وهو مفهوم أساسي في تصميم الأنظمة الحديثة.
هذا المبدأ يقوم على فكرة أن النظام الآمن يجب أن يحتوي على عدة طبقات من الحماية تعمل بشكل متكامل، بحيث يُعيق كل مستوى محاولات الاختراق قبل أن تصل إلى البيانات الحسّاسة.
من هذه الطبقات:

- الطبقة المحيطية :(Perimeter Security) وتشمل جدران الحماية (Firewalls) وأنظمة كشف ومنع التسلل.(IDS/IPS)
- طبقة التطبيقات :(Application Security) وتشمل تأمين كود التطبيق ضد الثغرات مثل–Cross site scripting (XSS) وSQL Injection.
- طبقة البيانات :(Data Security) وتشمل التشفير (Encryption) ، وإدارة الوصول (Access Management).
- طبقة المراقبة والتحليل :(Monitoring & Analytics) وتشمل أنظمة إدارة الأحداث الأمنية (SIEM)التي تقوم بدمج وتحليل البيانات الأمنية من جميع الطبقات السابقة.

يُضاف إلى ذلك مبدأ Zero Trust Architecture الذي يعتمد على "عدم الثقة الافتراضية"، أي عدم افتراض أن أي كيان داخل الشبكة آمن تلقائيًا، بل يجب التحقق من هوية وصلاحيات كل مستخدم أو تطبيق في كل خطوة.
وقد أصبحت هذه الفلسفة من ركائز الأمن المؤسسي الحديثة وفقًا لتوصيات.NIST SP 800–207

من جهة أخرى، تُصنَّف الهجمات السيبرانية عادةً ضمن ثلاث فئات رئيسية وفق الهدف من الهجوم:

هجمات سرية Confidentiality Attacks مثل سرقة كلمات المرور أو البيانات الحساسة.

هجمات تكامل Integrity Attacks مثل تعديل البيانات أو التلاعب في قواعد البيانات.

هجمات التوافر Availability Attacks مثل الحرمان من الخدمة.(DoS/DDoS)

ويُعتبر حقن SQL من أخطر أنواع هجمات التكامل، لأنه يتيح للمهاجم تغيير منطق استعلامات قاعدة البيانات عبر إدخال تعليمات SQL غير متوقعة، ما يؤدي إلى تسريب أو تعديل أو حذف بيانات أساسية.
لذلك فإن فهم طبيعة هذه الهجمات يعد أساسياً لتصميم آليات الكشف المناسبة.

أما من حيث مبادئ التصميم الآمن للأنظمة(Secure System Design Principles) ، فهناك مجموعة من القواعد التي وضعتها منظمة OWASPو Saltzer & Schroeder، وهي تُشكّل الإطار النظري لأي نظام آمن. من أهمها:

- مبدأ الأقل امتيازًا :(Least Privilege) إعطاء المستخدم أو العملية أقل قدر ممكن من الصلاحيات اللازمة لأداء المهام.
- الدفاع بالعمق :(Defense in Depth) عدم الاعتماد على وسيلة حماية واحدة.
- الفصل بين الواجبات :(Separation of Duties) توزيع المهام الأمنية بين مكونات مختلفة لتقليل الأخطاء البشرية.
- الفشل الآمن :(Fail-Safe Defaults) في حال فشل النظام يجب أن يبقى في وضع آمن.
- قابلية المراجعة :(Accountability) ضرورة تسجيل الأحداث والأنشطة لمتابعة أي محاولة خرق.

تؤكد هذه المبادئ على أن الأمن ليس مجرد تركيب أدوات مثل IDS أوSIEM ، بل هو منهجية متكاملة تبدأ من مرحلة التصميم البرمجي حتى التشغيل والمراقبة.
فالنظام الذي يفتقر لتصميم آمن سيكون عرضة للاختراق حتى لو استخدم أحدث الأدوات.
ولهذا يركّز مشروعنا على الجمع بين الكشف التقني عبر IDS والتحليل الذكي عبر SIEM في إطار متكامل، مما يجسد عمليًا مبدأ "الدفاع المتعدد الطبقات."

إن فهم هذه الأسس النظرية يمنح الباحث القدرة على تفسير سلوك الأنظمة الأمنية وتحليل التنبيهات الناتجة بدقة أكبر. كما يساعد على ربط المكونات التقنية — مثل snort أو — Wazuh ضمن فلسفة أمنية متكاملة، وهو ما سيُبنى عليه التصميم العملي في الفصول التالية.

٢.٣ هجوم حقن SQL (SQL injection):

٢.٣.١ تعريف ومبدأ العمل

حقن SQL (SQL Injection) هو أسلوب هجوم يستغل ضعف التحقق من مدخلات المستخدم في تطبيقات الويب التي تقوم بتكوين استعلامات (Queries) إلى قاعدة بيانات بطريقة ديناميكية. بدلاً من أن تُعامل المدخلات كبيانات فقط، يتم دمجها في جملة SQL بما يغيّر منطق الاستعلام، مما يمكن المهاجم من تنفيذ أوامر غير مرغوبة على قاعدة البيانات، مثل استخراج بيانات حساسة أو تعديلها أو حذفها أو تجاوز آليات المصادقة.

آلية الهجوم الأساسية تمر عبر ثلاث مراحل عملية ضمن دورة الهجوم:(Attack Lifecycle)

١. الاستطلاع :(Reconnaissance) المهاجم يحدد نقاط الإدخال) نماذج، وسائطGET/POST ، رؤوسHTTP ، (cookiesالتي تُكوّن استعلامات لقاعدة البيانات.

٢. الاستغلال :(Exploitation) إدخال سلسلة حقن (payload) تغير من بنية الاستعلام، مثلاً تحويل WHERE clauseإلى دائمًا صحيح.

٣. الاستخراج/التحكم :(Exfiltration/Control) استغلال النتائج لإظهار بيانات، تنفيذ أوامر تخريبية، أو الحصول على وصول أكبر داخل النظام.

٢.٣.٢ تصنيف أنماط حقن SQL :

تختلف أساليب حقن SQL تبعًا لطريقة حصول المهاجم على استجابة من الخادم، ومن أشهر الأنواع:

- In-Band SQLi (Union-based / Error-based)

  المهاجم يحصل على بيانات مباشرة من استجابة الخادم.

  - Union-based يستخدم UNION SELECT لدمج نتيجة استعلام خبيث مع استجابة التطبيق.

- Blind SQLi (Boolean-based / Time-based)

  المهاجم لا يرى البيانات مباشرة؛ يعتمد على استجابات منطقية أو زمنية.

  - Boolean-based (content-based) يحقن شرطًا يؤدي إلى اختلاف الاستجابة (صفحة مختلفة أو وجود عنصر) للدلالة على قيمة بايت أو بت.
  - Time-basedيحقن استعلامًا يُجبر قاعدة البيانات على الانتظار مدة محددة (SLEEP())، وبناءً على استجابة الوقت يستنتج قيمة.

- Out-of-Band (OOB) SQLi

يحدث عندما لا تُعاد النتائج مباشرة عبر القناة الحالية؛ يُستخدم قناة خارجية (مثل DNS أو HTTP callback) لجلب بيانات من الخادم. أقل شيوعًا لكنه فعال في بيئات محددة.

٢.٣.٣ آثار الهجوم والمخاطر الأمنية

هجمات حقن SQL قد تؤدي إلى نتائج خطيرة مؤسسيًا وتقنيًا، منها:

- انتهاك سرية البيانات :(Data Breach) تسريب قواعد بيانات كاملة تشمل معلومات شخصية وحساسة.
- فقدان تكامل البيانات :تعديل أو حذف سجلات حرجة يؤدي إلى فقدان موثوقية النظام.
- تجاوز المصادقة :(Authentication Bypass) الدخول بحسابات إدارية دون معرفة كلمات المرور.
- الإضرار بالتوافر :استغلال أوامر ثقيلة تسبب بطء أو تعطل قاعدة البيانات.(Denial of Service)
- نفاذ حدود النظام (Privilege Escalation) والتنقل الجانبي :في حالات متقدمة يمكن للمهاجم تنفيذ أوامر تؤدي إلى اختراق طبقات أخرى داخل الشبكة.

٢.٣.٤ صعوبة الاكتشاف والدوافع لوجود IDS متخصص

حقن SQL يصعب اكتشافه أحيانًا للأسباب التالية:

- تداخل مع الاستخدام الشرعي :بعض المدخلات الشرعية قد تشترك بسمات نحوية تشبه هجمات (مثلاً نصوص البحث).
- هجمات Blended و :Slow-Rateالهجمات البطيئة (time-based blind) لا تولّد كمية كبيرة من الأحداث في وقت قصير، لذا يمكنها التسلّل دون إثارة إنذارات عابرة.
- تنويع القنوات :الهجوم قد يتم عبر REST APIs ، رؤوسHTTP ، أو حتى بيانات مخزنة داخل ملفاتJSON ، ما يزيد من احتمالات التخفي.
- التعلم من قواعد التوقيع التقليدية :التوقيعات الثابتة قد تتجاوزها صيغ حقن جديدة أو تحويلات تشفير/ترميز .(encoding/obfuscation)

لهذا تُصبح الحاجة إلى نظام كشف تسلّل مخصّص (IDS) قادر على رصد أنماط SQLi أمراً محورياً—سواء عبر قواعد توقيع متقدمة تطابق أدق أشكالPayloads ، أو عبر الكشف بالسلوك (anomaly detection) الذي يراقب أنماط الاستعلامات إلى قاعدة البيانات ويكشف الانحرافات.

٢.٣.٥ مؤشرات الكشف (Indicators of Compromise) مفيدة لـIDS

عند تصميم IDS لاكتشافSQL injection ، من المهم مراقبة مؤشرات محددة مثل:

وجود كلمات مفتاحية في مُدخلات المستخدم.(UNION, SELECT, --, /*, SLEEP, OR 1=1)

زيادة مفاجئة في استجابات الأخطاء من قاعدة البيانات.

استعلامات طويلة غير اعتيادية أو استدعاءات متكرّرة للـinformation_schema.

تغيّر في زمن الاستجابة) دلالات على.(Time-based payloads

نشاطات خارجية مثل طلبات لإرسال استجابات عبر استدعاءات خارجية.(OOB)

٢.٤ أنظمة كشف ومنع التسلل(IDS/IPS) :

٢.٤.١ المفهوم العام

تُعدّ أنظمة كشف التسلل (Intrusion Detection Systems – IDS) من الركائز الأساسية في البنية الدفاعية لأي مؤسسة رقمية.
يُقصد بـ "التسلل" أي نشاط غير مصرح به يسعى إلى انتهاك سياسات الأمان أو استغلال الثغرات التقنية.
وظيفة الـ IDS هي مراقبة حركة الشبكة والأنشطة الداخلية وتحليلها لاكتشاف أي سلوك يُحتمل أن يكون خبيثًا، وإرسال تنبيه إلى فرق الأمان عند حدوثه.

تُكمل هذه الأنظمة عمل جدران الحماية (Firewalls) ، فبينما يتحكم الجدار الناري في حركة المرور المسموح بها أو المحظورة، يقوم IDS بمراقبة ما يحدث داخل هذا المرور لاكتشاف أي نشاط ضار قد يتجاوز السياسات المعلنة.

عند تطور الحاجة من الكشف إلى الاستجابة التلقائية ظهر نوع آخر يُعرف باسم نظام منع التسلل (Intrusion Prevention System – IPS)، والذي لا يكتفي بإصدار تنبيه بل يقوم أيضًا باتخاذ إجراء فوري مثل حظر الاتصال أو إسقاط الحزمة المشبوهة أو عزل المضيف.

٢.٤.٢ الأنواع الأساسية لأنظمة الكشف

تنقسم أنظمة IDS حسب موقع المراقبة إلى نوعين رئيسيين:

١. أنظمة كشف التسلل الشبكية:(Network-based IDS – NIDS)

تُنصّب عادة على نقاط استراتيجية في الشبكة وتقوم بمراقبة حركة المرور المارة عبر الشبكة في الزمن الحقيقي.

أمثلةSnort :، Suricata،

تمتاز NIDS بقدرتها على مراقبة نطاق واسع لكنها محدودة في تحليل الأنشطة الداخلية على الأجهزة المضيفة.

٢. أنظمة كشف التسلل المستندة إلى المضيف:(Host-based IDS – HIDS)

تُثبّت على الأجهزة نفسها لمراقبة سلوكيات النظام مثل ملفات السجل(Logs) ، عمليات التشغيل، سلامة الملفات.(File Integrity)

أمثلةOSSEC :، .Wazuh

تمتاز بقدرتها على اكتشاف التهديدات الداخلية أو تغييرات النظام، لكنها تتطلب موارد أكبر.

٢.٤.٣ آليات الكشف المستخدمة فيIDS

تعمل أنظمة IDS وفق منهجين رئيسيين للكشف:

١. الكشف بالتوقيعات:(Signature-based Detection)

يعتمد على قاعدة بيانات تحتوي على أنماط (Patterns) أو تواقيع للهجمات المعروفة (مثل سلاسل نصية، تعليمات معينة، أو سلوك محدد).

عندما تتطابق حركة مرور أو طلب معين مع أحد هذه التواقيع، يتم إطلاق تنبيه.

- الميزة: دقة عالية في اكتشاف الهجمات المعروفة.
- العيب: غير فعّالة أمام هجمات جديدة أو معدّلة.(Zero-Day Attacks)

٢. الكشف بالشذوذ:(Anomaly-based Detection)

يقوم النظام أولًا بتعلّم النمط الطبيعي لسلوك الشبكة أو التطبيق(Baseline Behavior) ، ثم يُصدر تنبيهًا عند ملاحظة أي انحراف كبير عن هذا النمط.

يمكن أن يعتمد على خوارزميات إحصائية أو تعلم آلي.

- الميزة: إمكانية كشف هجمات غير معروفة مسبقًا.
- العيب: احتمال مرتفع للإنذارات الكاذبة.(False Positives)

وفي بعض الأنظمة الحديثة يُدمج المنهجان ضمن ما يُعرف بـ Hybrid IDS لتحقيق توازن بين الدقة والتكيّف مع التهديدات الجديدة.

٢.٤.٤ المكونات الرئيسية لنظامIDS

يتكوّن أي نظام IDS فعال من مجموعة من الوحدات الأساسية:

١. وحدة الالتقاط :(Data Collection) تلتقط حركة المرور أو سجلات النظام من الشبكة أو المضيف.

٢. وحدة التحليل :(Analysis Engine) تُفسر البيانات باستخدام قواعد أو خوارزميات للكشف.

٣. وحدة قاعدة المعارف :(Knowledge Base) تخزّن تواقيع الهجمات أو ملفات الأنماط.

٤. وحدة التنبيه والتقارير :(Alerting & Reporting) تُرسل تنبيهات إلى مدير الأمان أو إلى نظام SIEM.

٥. واجهة الإدارة :(Management Console) تُستخدم لتكوين النظام وإدارة القواعد ومراجعة التنبيهات.

٢.٤.٥ مؤشرات أداء النظام والتحديات

أداء أنظمة IDS يقاس بعدة معايير:

- معدل الكشف الحقيقي.(True Positive Rate)
- معدل الإنذارات الكاذبة.(False Positive Rate)
- الزمن اللازم للكشف.(Detection Latency)
- معدل المعالجة.(Throughput)

التحديات الرئيسية تشمل:

- صعوبة مواكبة أنماط الهجمات الجديدة.
- استهلاك الموارد الحاسوبية عند تحليل كمّ كبير من البيانات.
- التعامل مع التشفير (Encrypted Traffic) الذي يُعيق التحليل العميق للحزم (Deep Packet Inspection).
- الحاجة إلى تكامل مع أنظمة SIEM لتجميع وتحليل الأحداث من مصادر متعددة بشكل مركزي.

٢.٥ نظام إدارة معلومات وأحداث الأمن(SIEM)

٢.٥.١ المفهوم العام

يُعتبر نظام إدارة معلومات وأحداث الأمن (SIEM) أحد أهم أدوات التحليل والمراقبة في مراكز العمليات الأمنية (Security Operations Centers – SOC).

ظهر مفهوم SIEM نتيجة الحاجة إلى دمج وظائف نظامين سابقين:

- SIM (Security Information Management): يهتم بتخزين وتحليل البيانات الأمنية التاريخية على المدى الطويل.
- SEM (Security Event Management): يركز على تحليل الأحداث الفورية وإطلاق التنبيهات في الزمن الحقيقي.

ومن هنا جاء نظام SIEM كحل متكامل يجمع بين تحليل البيانات التاريخية والتحليل الفوري، ليمنح فرق الأمن رؤية مركزية شاملة لجميع الأنشطة الأمنية في بيئة المؤسسة.

يعمل نظام SIEM على جمع وتحليل وربط الأحداث الأمنية الصادرة من مختلف الأجهزة والتطبيقات مثل الجدران النارية، أنظمة كشف التسلل IDS ، الخوادم، قواعد البيانات، والتطبيقات السحابية.

من خلال ذلك، يمكن لـ SIEM تحديد الأنماط المريبة والتصرفات غير المعتادة وربطها ببعضها لتوليد إنذار دقيق يدل على حادث أمني فعلي.

٢.٥.٢ مكونات نظام SIEM

يتكوّن أي نظام SIEM فعّال من مجموعة من المكونات المترابطة، يمكن تلخيصها فيما يلي:

١. جامع البيانات:(Data Collectors / Agents)

مسؤول عن جمع السجلات (Logs) والأحداث من مصادر متعددة وإرسالها إلى النظام المركزي للتحليل.

تُستخدم بروتوكولات مثل Syslog ، Winlogbeat، Filebeat، و API Integrations.

٢. قاعدة بيانات الأحداث:(Event Repository)

تخزّن جميع السجلات الواردة بطريقة منظمة لتسهيل التحليل والاسترجاع.

يجب أن تكون هذه القاعدة قابلة للتوسع (Scalable) وتوفر فهرسة سريعة.(Indexing)

٣. محرك التحليل:(Correlation & Analytics Engine)

القلب الحقيقي للنظام، حيث يتم تحليل الأحداث باستخدام قواعد ارتباط (Correlation Rules) أو خوارزميات تعلم آلي لتحديد العلاقات بين الأحداث من مصادر مختلفة.

مثلًا: محاولة تسجيل دخول فاشلة تليها استعلامات قاعدة بيانات غير طبيعية قد تشير إلى هجوم SQL Injection.

٤. وحدة التنبيه والاستجابة:(Alerting & Response Module)

تُصدر تنبيهات فورية عند اكتشاف تهديد، ويمكن أن تُنفذ إجراءات تلقائية) مثل حظر عنوانIP ، تعطيل مستخدم، إرسال بريد فوري إلى الفريق الأمني.(

٥. لوحات المراقبة والتقارير :(Dashboards & Reporting)

تتيح عرض الأحداث في شكل رسومي وتحليلي لفهم الاتجاهات الأمنية، وتقديم تقارير دورية للإدارة.

٦. نظام إدارة المستخدمين والصلاحيات:

لضمان أن الوصول إلى بيانات SIEM يتم فقط من قبل الموظفين المصرّح لهم، وفق مبدأ الأقل امتيازًا.(Least Privilege)

٢.٥.٣ آلية عملSIEM

تمر دورة العمل داخل نظام SIEM بعدة مراحل متتابعة كما هو موضّح أدناه:

١. جمع البيانات:(Data Collection)

استلام السجلات من الأجهزة المختلفة عبر الوكلاء (Agents) أو بروتوكولات النقل.

٢. تطبيع البيانات:(Normalization)

تحويل السجلات من تنسيقات مختلفة إلى نموذج موحد (Common Event Format – CEF) ليسهل تحليلها.

٣. التحليل والترابط:(Correlation & Analysis)

تطبيق قواعد منطقية لتحديد تسلسل أحداث غير طبيعي أو مريب.

مثال: "إذا تم تسجيل دخول غير ناجح أكثر من ٥ مرات في دقيقة، تلاه تنفيذ استعلام SQL غريب ← إصدار تنبيه."

٤. التخزين والفهرسة:(Storage & Indexing)

حفظ البيانات وتحليلها لاحقًا في التحقيقات.(Forensics)

٥. التنبيه والاستجابة:(Alerting & Incident Response)

إرسال تنبيهات عبر البريد الإلكتروني، أو إلى أنظمة الاستجابة التلقائية.(SOAR)

٢.٥.٤ فوائد استخدامSIEM

١. الرؤية المركزية للأمن:(Centralized Visibility)

يمنح الإدارة الأمنية رؤية موحدة لجميع الأحداث داخل الشبكة.

٢. تحسين سرعة الاستجابة:(Incident Response)

بفضل التنبيهات الآلية ودمج الأحداث المتعددة في سياق واحد، يمكن للفريق الأمني الاستجابة بسرعة للحوادث.

٣. الكشف عن التهديدات المركّبة:(Complex Threats)

من خلال correlation rules يمكن كشف الهجمات متعددة المراحل مثل SQL Injection متبوعة بسرقة بيانات.

٤. تحليل الاتجاهات:(Trend Analysis)

يُساعد في اكتشاف أنماط تكرار الهجمات لتطوير سياسات دفاعية جديدة.

٥. الامتثال للمعايير:(Compliance)

أنظمة SIEM تساعد المؤسسات على الامتثال لمعايير مثل ISO 27001 و PCI DSS و GDPR من خلال التوثيق الدقيق للأحداث.

٢.٦ التكامل بين أنظمة IDS وSIEM

٢.٦.١ المفهوم العام للتكامل

يُعد التكامل بين أنظمة كشف التسلل (IDS) وأنظمة إدارة معلومات وأحداث الأمن (SIEM) أحد أهم ممارسات الأمن السيبراني الحديثة، إذ يهدف إلى الانتقال من الكشف المنعزل إلى المراقبة والتحليل الشامل متعدد الطبقات.

بينما يعمل نظام IDS على اكتشاف الأنشطة المشبوهة في الشبكة أو المضيف بشكل فوري، فإن SIEM يُمثل الطبقة التحليلية التي تجمع وتربط هذه التنبيهات مع أحداث أخرى من مصادر مختلفة.

بذلك يتحوّل الحدث من مجرد "إنذار محلي" إلى "حادثة أمنية موثقة" يمكن تتبعها وتحليلها ضمن سياقها الكامل.

في بيئة متكاملة، يصبح IDS بمثابة مستشعر أمني(Sensor) ، بينما يُعتبر SIEM العقل التحليلي المركزي (Analytical Brain).

هذا التكامل يمكّن المؤسسة من الانتقال من الكشف السلبي (Passive Detection) إلى الاستجابة الاستباقية (Proactive Response).

٢.٦.٢ آلية التكامل الفني بين IDS وSIEM

تتم عملية الدمج عبر سلسلة من الخطوات التقنية المدروسة، تختلف حسب الأدوات المستخدمة ولكنها تتبع غالبًا نفس المفهوم العام:

١. توليد الحدث:(Event Generation)

يقوم IDS بتحليل حركة المرور ورصد نشاط مريب) مثل محاولة(SQL Injection ، ثم يُنشئ سجلًا تفصيليًا (Alert Log) يحتوي على بيانات الحدث:

- عنوان الـ IP المصدر.
- المنفذ المستهدف.
- نوع التوقيع الذي تم مطابقته.(Signature ID)
- التاريخ والوقت.
- الرسالة التوضيحية للهجوم.

٢. نقل الحدث إلى:(Event Forwarding) SIEM

يتم إرسال هذه السجلات إلى خادم SIEM باستخدام بروتوكولات قياسية مثل Syslogأو JSON API.

في الأنظمة الحديثة مثل Suricata + Wazuh، يتم ذلك آليًا عبر وحدة تكامل مخصصة تتيح إرسال التنبيهات لحظة وقوعها.

٣. تطبيع البيانات:(Normalization)

عند وصول الحدث إلىSIEM ، يتم تحويله إلى تنسيق موحّد(Unified Event Schema) ، ما يسمح بدمج الأحداث القادمة من مصادر مختلفة) مثل IDS و Firewallو (Database Logsفي قاعدة بيانات واحدة.

٤. التحليل والترابط:(Correlation Analysis)

يُطبّق SIEM قواعد ترابط (Correlation Rules) للبحث عن علاقات منطقية بين الأحداث.

مثال عملي:

- حدث من IDS يشير إلى محاولة SQL Injection من IP محدد.
- حدث من Web Server يُظهر استعلامات غير اعتيادية في نفس الدقيقة من نفس الـIP.
- النتيجة SIEM :يُصدر تنبيهًا مركّبًا يشير إلى "هجوم مؤكد."

٥. الاستجابة والتنبيه:(Alerting & Response)

يتم إرسال تنبيه موحّد للفريق الأمني، ويمكن تفعيل استجابة تلقائية مثل حظر العنوان المهاجم عبر الجدار الناري (Firewall Integration) أو تعطيل الجلسة النشطة.

٢.٦.٣ الفوائد العملية للتكامل بين IDS وSIEM

١. تحسين دقة الكشف وتقليل الإنذارات الكاذبة:(False Positives)

عندما يدمج SIEM تنبيهات IDS مع مصادر أخرى، يمكنه التحقق من مدى مصداقيتها قبل إطلاق إنذار حقيقي.

مثلاً، إذا كشف IDS محاولة SQL Injection لكن لم تُسجَّل أي استعلامات غريبة في قاعدة البيانات، قد يعتبر SIEM الحدث غير مؤكد ويقلل الإزعاج للفريق الأمني.

٢. رؤية موحدة للأحداث:(Unified Visibility)

في بيئة متعددة الأنظمة، قد يصعب على المهندسين تتبع مصدر الهجوم. التكامل يسمح بعرض جميع الأحداث ذات الصلة في لوحة واحدة، مما يسهّل عملية التحليل والتحقيق.

٣. التحليل الزمني والتسلسلي:(Temporal Correlation)

يمكن لـ SIEM أن يربط بين سلسلة من الأحداث على مدار الوقت) مثل محاولات تسجيل دخول متكررة ثم حقن (SQL لتحديد هجمات متقدمة.(Advanced Persistent Threats)

٤. الاستجابة التلقائية:(Automated Incident Response)

بعض منصات) SIEM مثل Wazuh أو (Splunk SOAR يمكنها تنفيذ إجراءات تلقائية فور استقبال تنبيه منIDS ، مثل إرسال أوامر إلى الجدار الناري لحظر المهاجم.

٥. دعم الامتثال والتدقيق:(Compliance & Auditing)

الدمج بين النظامين يسهل عملية إعداد تقارير امتثال لمعايير أمنية مثل ISO 27001وPCI-DSSو GDPRعبر توثيق كل الأحداث بشكل مركزي ومنظم.

٢.٦.٤ التحديات المرتبطة بعملية التكامل

رغم الفوائد الكبيرة، إلا أن الدمج بين IDS و SIEMيواجه بعض العقبات الفنية والتنظيمية، مثل:

- اختلاف تنسيقات السجلات :(Log Formats) تحتاج المؤسسات إلى آليات تطبيع فعّالة لتوحيد البيانات من مصادر متعددة.

- الكمّ الهائل من البيانات :(Data Volume) كل من IDS و SIEMيولّد آلاف الأحداث يوميًا، مما يتطلب خوادم قوية وسعة تخزين كبيرة.
- ضبط القواعد :(Rule Tuning) أي خطأ في قواعد الترابط قد يؤدي إلى تنبيهات زائدة أو فقدان أحداث مهمة.
- الزمن الحقيقي :(Real-time Processing) التكامل الفعّال يتطلب معالجة سريعة دون تأخير في نقل وتحليل الأحداث.
- التوافق البرمجي :قد تختلف واجهات الربط (APIs) بين الأنظمة المختلفة، مما يتطلب كتابة أدوات وسيطة (Middleware) أحيانًا.

٢.٧ الخلاصة :

استعرض هذا الفصل الأسس النظرية اللازمة لفهم وتصميم نظام كشف تسلل مخصّص لاكتشاف هجمات SQL Injection وربطه بمنصة SIEM للتحليل المركزي. بدأنا بتأكيد أهمية أمن النظم والمعلومات كمدخل استراتيجي لحماية الأصول الرقمية، ومراجعة نموذج المثلث الأساسي (CIA) ومبادئ التصميم الآمن كالحدّ من الامتيازات والدفاع المتعدد الطبقات و .Zero Trust ثم انتقلنا إلى تحليل طبيعة هجمات تطبيقات الويب، مع تفصيل آليات وطرق تنفيذ حقن SQL (Union-based, Error-based, Blind, Time-based و(OOB، مع إبراز صعوبة اكتشاف بعض الأنماط البطيئة أو المشفّرة والحاجة إلى مؤشرات كشف متخصّصة.

بعد ذلك عرضنا بشكل مفصّل وظائف وأدوار أنظمة كشف ومنع التسلل(IDS/IPS) ، وشرحنا فروق NIDS مقابلHIDS ، ومنهجيات الكشف (Signature vs Anomaly) ومزايا كل منهما وقيودهما. بيّنا أيضاً مكوّنات IDS الأساسية (التقاط البيانات، محرك التحليل، قاعدة التواقيع، وحدة التنبيه) ومعايير قياس الأداء مثل معدل الكشف والزمن اللازم للتحقق ومؤشرات الإنذارات الكاذبة. هذا العلم يوفر الأساس التقني لكتابة قواعد كشف فعّالة، وفهم متطلبات الأداء والموارد لنشر IDS في بيئة حقيقية.

ثم عرض الفصل دور نظم SIEM كمنصة مركزية تجمع سجلات البنية التحتية، وتقوم بتطبيعها وربطها وتحليلها عبر قواعد ارتباط وخوارزميات ذكية، وتدعم الاستجابة الآلية والإجراءات التصحيحية. بيّنا كيف يعالج SIEM مشكلات التشتت في السجلات ويخفض الإنذارات الكاذبة عبرcorrelation ، كما يدعم الامتثال والتدقيق وتحليل الاتجاهات. وربطنا بينهما في قسم مخصّص تناول آليات التكامل) توليد الأحداث، نقلها عبر بروتوكولات مثلSyslog/JSON ، التطبيع، الترابط، والاستجابة التلقائية(، مع توضيح فوائد التكامل وأبرز التحديات التقنية مثل حجم البيانات، تطبيع الصيغ، وضبط قواعد الترابط.

أجريت مراجعة مستهدفة للدراسات السابقة التي تناولت كشف SQL Injection ودمج IDS معSIEM ، وخلص الفصل إلى استنتاجين عمليين: الأول أن الاعتماد على توقيعات بسيطة وحدها غير كافٍ لمجابهة تطوّر

أساليب الحقن، والثاني أن الدمج الفعّال مع SIEM يعزز الدقّة ويقلّل الضوضاء التحليلية، خصوصًا عند إكماله بتقنيات تحليل سلوكي أو تعلم آلي. تلك النتائج تدعم الفرضية المركزية للمشروع: أن تصميم IDS مخصّص لاكتشاف أنماط SQLi وربطه بشكل صحيح بمنصة SIEM سيؤدي إلى تحسين ملحوظ في معدلات الكشف وسرعة الاستجابة مقارنة بالحلول المنعزلة.

ختامًا، يؤسس هذا الفصل لمرحلة التصميم والتنفيذ التي ستركز عليها الفصول التالية. سنتناول في الفصل الثاني متطلبات النظام ، بنية الاختبار (testbed) المقترحة، واختيار الأدوات(مثل Suricata /Snort كـ IDS و Wazuh/ELKكسيرفيسSIEM) مع تبرير تقني لاختيار كل مكوّن. بعد ذلك سننتقل إلى فصل يشرح كتابة قواعد الكشف، آليات تطبيع السجلات، آليات الربط (log forwarding) إلىSIEM ، واختبارات الأداء والحالات التجريبية لهجمات SQLi المتنوعة.

الفصل الثاني : بيئة العمل

٣.١ المقدمة :

هدف هذا الفصل إلى توثيق خطوات تجهيز بيئة العمل التي تم الاعتماد عليها لتنفيذ مشروع كشف محاولات حقن قواعد البيانات (SQL Injection) ضمن بيئة مخبرية، مع ضمان أن تكون المكوّنات مترابطة وقابلة لإنتاج سجلات (Logs) قابلة للتحليل من طرف نظام إدارة أحداث الأمن. ويُعدّ إعداد البيئة خطوة تأسيسية لأن دقة النتائج اللاحقة تعتمد مباشرة على سلامة التنصيب، وضبط الإعدادات، وتوافق المخرجات بين الأدوات.
يركّز هذا الفصل حصراً على: عرض الأدوات المستخدمة، مبررات اختيارها، آلية تنصيبها، ثم طريقة ربطها معاً بحيث يتم توليد أحداث HTTP وتنبيهات IDS وتجميعها وتحليلها مركزياً.

٣.٢ الأدوات المستخمة :

اعتمدت بيئة العمل في هذا المشروع على منظومة أدوات متكاملة تعمل كسلسلة مترابطة تبدأ من توليد حركة HTTP داخل بيئة اختبار، مروراً بالكشف الشبكي عن أنماط SQLi ، وانتهاءً بالتجميع والتحليل المركزي للأحداث. وفيما يلي شرح الأدوات المستخدمة مع تضمين مكوّنات Wazuh الأساسية التي تشكّل البنية التشغيلية للمنصة:

٣.٢.١ Wazuh

تم استخدام Wazuh بوصفه منصة SIEM مفتوحة المصدر لالتقاط السجلات وتحليلها وتوليد التنبيهات وفق قواعد مُعرّفة. ويقوم Wazuh ضمن المشروع على عدة مكوّنات أساسية، لكل منها دور محدد في دورة حياة الحدث:(Log/Event Pipeline)

- Wazuh Manager

هو المكوّن المركزي المسؤول عن استقبال البيانات القادمة من الوكلاء ومصادر السجلات، ثم إجراء عمليات التحليل والتطبيع والتصنيف وتطبيق قواعد الكشف. داخل الـ Manager تعمل وظائف محورية مثل:

- فكّ وترميز الرسائل وتفسيرها (Decoding/Parsing) اعتماداً على الـ.Decoders
- تطبيق قواعد الكشف (Rules Engine) لتوليد تنبيه أو رفع مستوى الخطورة عند تحقق شروط معينة.
- إدارة الوكلاء (Agent Management) من حيث التسجيل (Registration) والتحقق من الاتصال وتحديث الإعدادات.

- Wazuh agent

هو عميل مراقبة يتم تثبيته على النظام الذي تنتج فيه السجلات (مثل جهاز تشغيل Suricata أو الخادم الذي يُقرأ منه .(eve.jsonمهمته:

- مراقبة ملفات السجلات المحددة في الإعدادات.
- إرسال الأحداث بشكل آمن ومنتظم إلى.Wazuh Manager
- تمكين جمع معلومات إضافية على مستوى المضيف عند الحاجة.(Host-based Visibility)

- Wazuh Indexer

يمثل طبقة التخزين والفهرسة (Indexing Layer) التي تحفظ الأحداث بعد معالجتها، وتتيح البحث السريع والاستعلام عنها. وجود الـ Indexer ضروري لتمكين:

- حفظ كميات كبيرة من الأحداث بكفاءة.
- تنفيذ عمليات بحث وتحليل لاحقة بشكل سريع ومرن.

- Wazuh Dashboard

واجهة العرض والتحكم التي تُستخدم لمتابعة التنبيهات والأحداث والرسوم البيانية والتحليلات وتتيح :

- مراقبة تدفق السجلات والتنبيهات بشكل بصري.
- استعراض نتائج القواعد والبحث في الأحداث المؤرشفة داخل الـ.Indexer
- تسهيل التحقق من نجاح التكامل بين الأدوات عبر تتبّع وصول تنبيهات Suricata وتصنيفها.

- Decoder/Rules

رغم أنها ليست "أدوات" مستقلة، إلا أنها مكوّنات تشغيلية داخل Wazuh لا يمكن إهمالها في وصف المنظومة:

- Decoders مسؤولة عن استخراج الحقول من السجلات الخام مثل JSON الخاص بـSuricata وتحويلها إلى حقول قابلة للاستخدام في القواعد.
- Rules تمثّل منطق الكشف الذي يحدد متى يتم اعتبار الحدث مؤشراً لهجوم أو سلوك غير طبيعي، وكيف يتم تقييم شدته.

بهذا التكوين، يعمل Wazuh كنظام متكامل يبدأ باستقبال سجل قادم من مصدر خارجي(Suricata) ، ثم يفكّكه ويفهمه(Decoders) ، وبعد ذلك يطبّق منطق الكشف(Rules) ، ويخزّن النتائج(Indexer) ، ويعرضها للمراقبة.(Dashboard)

Suricata ٣.٢.٢

تم اعتماد suricata في هذا المشروع بوصفها نظام كشف تسلل شبكي

(Network Intrusion Detection System – NIDS)، أي أنها تعمل على المراقبة والتحليل وإطلاق التنبيهات دون تنفيذ إجراءات منع تلقائية. وظيفة Suricata هنا تتمحور حول رصد حركة المرور الشبكية المرتبطة بتطبيق الويب داخل بيئة الاختبار، وتحليلها لاكتشاف أنماط هجمات SQL Injection، ثم تسجيل نتائج الكشف ضمن سجل أحداث قابل للاستهلاك منWazuh

- آلية عمل Suricata كـ IDS في سياق المشروع
- تعتمد Suricata كـ IDS على مبدأ أساسي وهو فحص حركة المرور (Traffic Inspection) ومقارنتها بمجموعة من التواقيع .(Signatures/Rules) وعند تحقق شروط توقيع معيّن، تقوم Suricata بإنشاء حدث تنبيه (Alert Event) بدل منع الاتصال أو إسقاط الحزم.

ضمن هذا المشروع، تمر العملية عملياً بالمراحل التالية:

- التقاط حركة المرور(Traffic Capture)

تقوم Suricata بمراقبة واجهة الشبكة داخل بيئة العمل لالتقاط تدفق الاتصالات المتجهة نحو خادم الويب الذي يستضيف DVWA هذا الالتقاط يتيح فحص طلبات HTTP الفعلية الناتجة عن تفاعل المستخدم مع التطبيق، بما في ذلك الطلبات التي قد تحتوي مدخلات خبيثة. تحليل بروتوكول HTTP واستخراج الأجزاء ذات الدلالة الأمنية

- مطابقة حركة المرور مع قواعد الكشف(Rule Matching)

تطبّق Suricata القواعد المعرّفة للكشف عن SQLi عبر مطابقة أنماط نصية (مثل كلمات دلالية أو سلوك) وعند تطابق الطلب مع قاعدة معينة، يتم إنتاج تنبيه يحمل هوية القاعدة ومعلوماتها.

- توليد سجل تنبيه بصيغة JSON داخلeve.json

الناتج التشغيلي الأكثر أهمية في المشروع هو أن Suricata تسجّل التنبيهات ضمن ملفeve.json

حيث يظهر الحدث عادة كنوع "event_type:"alert ويتضمن معلومات تشخيصية مفيدة لمرحلة التجميع والتحليل فيWazuh ، مثل:

signature_id رقم التوقيع الذي تم تفعيله (يمثل القاعدة التي اكتشفت النمط).

signature وصف القاعدة (رسالة التنبيه).

src_ip / dest_ip / ports / proto بيانات المصدر والوجهة التي تساعد على تتبع سياق الهجوم.

هذا التسجيل بصيغة JSON هو ما يجعل التكامل مع Wazuh عملياً، لأن Wazuh Agent يستطيع قراءة هذا الملف وتمريره إلى Wazuh Manager لمعالجته.

٣.٢.٣ DVWA

تم استخدام DVWA كتطبيق ويب تدريبي لتوليد سيناريوهات SQLi ضمن سياق حقيقي. يتيح ذلك:

- إنشاء طلبات HTTP تحتوي مؤشرات SQLi في URI أو RAW-URI أو جسم الطلب (Request Body).
- اختبار قدرة Suricata على الالتقاط، ثم قدرة Wazuh على الاستقبال والتحليل وإظهار التنبيه.

٣.٢.٤ قاعدة البيانات(MySQL)

تعمل قاعدة البيانات كخلفية تشغيلية لـDVWA ، ما يجعل سيناريوهات SQLi مرتبطة بمعمارية Web App حقيقية. وجود قاعدة البيانات ضروري لأن:

- التطبيق يعتمد على استعلامات SQL فعلية.
- إدخالات SQLi تتفاعل ضمن سياق قريب من بيئات الإنتاج من حيث المنطق العام للتطبيق.

٣.٢.٥ خادم الويب وبيئة التشغيل

تم تشغيل DVWA على Apache مع PHP ضمن نظام لينكس، وذلك من أجل:

- استقبال وتنفيذ طلبات HTTP/GET/POST الخاصة بالتطبيق.
- توفير بيئة تشغيل متكاملة تضمن أن التفاعلات داخل DVWA تنتج حركة مرور قابلة للرصد والتحليل بواسطة.Suricata

٣.٢.٦ بيئة افتراضية(Virtualization)

تم تنفيذ البيئة ضمن آلة افتراضية لدعم:

- العزل الأمني أثناء تشغيل تطبيق ضعيف أمنياً وإجراء اختبارات هجومية.
- التحكم بالشبكات والواجهات (Interfaces) اللازمة لالتقاط الحركة بواسطة Suricata

- سهولة إعادة التجربة وتكرار نفس سيناريوهات الاختبار.

٣.٢.٧ سبب اختيار الأدوات عن منافسيها

المحاور	Wazuh	Splunk ES	IBM QRadar
نموذج النشر	محلي/سحابي	محلي/سحابي	محلي/سحابي
جمع البيانات	Agents + قراءة ملفات Logs وإرسالها للمدير (مثل JSON) للتحليل	Forwarders، غني من ناحية مصادر البيانات	تجميع أحداث (Events)وتدفّقات (Flows)من مصادر متعددة
منهجية الكشف	قواعد + فك ترميز Decoders/ +تنبيهات؛ مناسب لربط سجلات أدوات خارجية) مثل IDS)	تركيز قوي على محتوى SOC التجاري و-**Risk Based Alerting** لتقليل التنبيهات الكاذبة	Correlationعلى مستوى الأحداث/التدفّقات لإنتاجOffenses
التخزين والفهرسة والعرض	Indexerللتخزين والفهرسة + Dashboardللعرض	قدرات بحث/تحليلات قوية (Splunk)مع لوحات متقدمة	قدرات إدارة حوادث "Offenses"وواجهات تحقيق
الترخيص والتكلفة	مفتوح المصدر (مع خيارات خدمة سحابة)	تجاري وكلفته ترتفع مع حجم البيانات	تجاري

جدول (١) : الفروقات بين أدوات SIEM

المحور	Suricata	Snort
طبيعة الأداة	IDS/IPSقائم على قواعد(Rules)	IDS/IPSقائم على قواعد(Rules)
أسلوب الكشف	Rule-based signature detectionقواعد تحدد أنماط نشاط خبيث وتولّدAlerts	Rule-based signature detectionقواعد تحدد أنماط نشاط خبيث وتولّدAlerts
الأداء والمعمارية	**Multi-threaded** وقابلية تفتيش عالية الأداء؛ مناسب لرفعThroughput	يعتبر جيد مثل Suricata كأداء لكن لا يدعم **Multi-threaded**
فهم البروتوكول	**Automatic Protocol Detection** وتطبيق منطق الكشف/التسجيل حتى على منافذ غير قياسية	يعتمد على تحليل/مسبقات معالجة وقواعد
المخرجات والتكامل	يوحد alerts/metadata فيJSON ؛ ممتاز للتكامل معSIEM	يدعم alerts/logs اصعب في التكامل مع wazuh

جدول (٢) : الفروقات بين أدوات IDS

٣.٣ تنصيب الأدوات

توضح هذه الفقرة خطوات تنصيب الأدوات الأساسية التي بُنيت عليها بيئة المشروع، بدءاً من منصة Wazuh ثم Suricata، وبعدها إعداد بيئة الاختبار المتمثلة بـ DVWA مع Apache يركّز هذا القسم على جانب "التنصيب والتجهيز الأولي"

٣.٣.١ تنصيب Wazuh

هناك طريقتين لتنصيب أداة Wazuh وهما (Installation guide-Quickstart)

تم اختيار طريقة Installation guide لانها تناسب طبيعة المشروع

أولا تنصيب Wazuh indexer :

يعتمد Wazuh في مكوّناته المركزية على اتصال مشفّر بين طبقات المنظومة، لذلك يبدأ دليل التنصيب الرسمي بمرحلة **إنشاء شهادات TLS** ثم الانتقال لتنصيب الـ Indexer وضبطه.

أ) المرحلة الأولى: إنشاء الشهادات(Certificate creation)

١. **تنزيل أداة إنشاء الشهادات** وملف الإعداد config.yml

٢. **تعديل config.yml** لإدخال أسماء العقد (Indexer/Server/Dashboard) وعناوين IP/Hostnames الخاصة بها، بحيث تُستخدم نفس القيم لاحقاً أثناء ضبط كل مكوّن .

٣. **توليد الشهادات** عبر تشغيل سكربت الشهادات، ثم ضغط المخرجات ضمن ملف واحد wazuh-certificates.tar ونقله إلى جميع العقد .

```
# curl -sO https://packages.wazuh.com/4.14/wazuh-certs-tool.sh
# curl -sO https://packages.wazuh.com/4.14/config.yml
```

```
nodes:
  # Wazuh indexer nodes
  indexer:
    - name: node-1
      ip: "<indexer-node-ip>"
    #- name: node-2
    #  ip: "<indexer-node-ip>"
    #- name: node-3
    #  ip: "<indexer-node-ip>"

  # Wazuh server nodes
  # If there is more than one Wazuh server
  # node, each one must have a node_type
  server:
    - name: wazuh-1
      ip: "<wazuh-manager-ip>"
```

الشكل (١) :أداة تنزيل الشهادات واعداد الملف confi.yaml

ب) المرحلة الثانية: تنصيب عقد الـ Indexer وضبطها

بعد تجهيز الشهادات، يتم تنصيب الحزم المطلوبة ثم إضافة مستودع Wazuh وتثبيت حزمة wazuh-indexer. بعدها يُضبط ملف إعدادات الـ Indexer (opensearch.yml) لتحديد عنوان الشبكة

نقاط الضبط الأساسية كما في الدليل:

- network.host لعنوان العقدة (ويُفضّل مطابقته لما تم وضعه في config.yml)
- node.name لاسم العقدة مثل كما عُرّف في config.yml
- إعدادات الكلاستر مثل cluster.initial_master_nodes و discovery.seed_hosts عند استخدام أكثر من عقدة .

ج) المرحلة الثالثة: نشر الشهادات وتهيئة cluster وتشغيل الخدمة

يتم استخراج الشهادات الخاصة بعقدة الـ Indexer إلى المسار المخصص للشهادات ثم تعيين الأذونات والملكية كما في الدليل، وبعدها تشغيل خدمة الـ Indexer وأخيراً تُجرى خطوة **Cluster initialization** عبر تشغيل سكربت التهيئة الأمنية indexer-security-init.sh لتحميل الشهادات وتفعيل الإعدادات الأمنية .

ثانيا تنصيب Wazuh server :

يعرّف الدليل الرسمي Wazuh Server باعتباره مكوّناً مركزياً يضم **Wazuh Manager** لتحليل بيانات الوكلاء وإنتاج التنبيهات، إضافةً إلى **Filebeat** لنقل التنبيهات والأحداث المؤرشفة إلىWazuh Indexer يتم تنصيب هذه الطبقة عبر : إضافة المستودع، تثبيت wazuh-manager، تثبيت Filebeat وضبطه، نشر الشهادات، ثم تشغيل الخدمات .

أ) إضافة مستودع Wazuh

تبدأ العملية بإضافة مستودع Wazuh أو تخطيها إذا كانت نفس العقدة تحتوي مسبقاً على المستودع نتيجة تنصيب مكوّن سابق

ب) تثبيت Wazuh Manager

يتم تثبيت الحزمة wazuh-manager باستخدام مدير الحزم المناسب للنظام .

ج) تثبيت Filebeat ثم تنزيل ملف إعداد جاهز وضبطه

يوجّه الدليل إلى تثبيت Filebeat ثم تنزيل ملف إعداد مُسبق منWazuh ، وبعدها تعديل قيمة hosts لتشير إلى عقد/عقدة الـ Indexer المعتمدة

```
# Wazuh - Filebeat configuration file
output.elasticsearch:
hosts: ["10.0.0.1:9200"]
protocol: https
username: ${username}
password: ${password}
```

الشكل (٢) : تعديل قيمة host لتيشر الى indexer

د) نشر الشهادات وربط Filebeat/Manager بالـIndexer

يتم نقل ملف wazuh-certificates.tar إلى عقدة السيرفر ثم نشر الشهادات وفق اسم العقدة المحدد في config.yml

بالإضافة إلى ذلك، يوضح الدليل خطوة حفظ بيانات اعتماد الـ Indexer في **Wazuh manager** باستخدام wazuh-keystore وتُذكر إمكانية تخطيها إذا لم تكن بعض القدرات مطلوبة

هـ) تشغيل الخدمات والتحقق

بعد اكتمال الخطوات السابقة، يتم تشغيل wazuh-manager ثم تشغيل filebeat والتحقق من حالتهما .

ثالثا تنصيب Wazuh Dashboard :

تم تنصيب Wazuh Dashboard كواجهة ويب لعرض وتنقيب تنبيهات السيرفر والأحداث المؤرشفة. يوضح الدليل مراحل: تثبيت الاعتماديات، إضافة المستودع، تثبيت الحزمة، ضبط ملف opensearch_dashboards.yml خاصة (server.host وopensearch.hosts)، نشر الشهادات، تشغيل الخدمة، ثم ضبط ملف wazuh.yml لربط الـ Dashboard بـ Wazuh Server

أ) تثبيت الحزمة وضبط الاتصال بالـIndexer

يتم تثبيت wazuh-dashboard ثم ضبط:

- server.host للسماح بالاتصال بالـ server
- opensearch.hosts لتحديد عنوان/عناوين الـ Indexer التي سيتصل بها الـ Dashboard

```
server.host: 0.0.0.0
server.port: 443
opensearch.hosts: https://localhost:9200
opensearch.ssl.verificationMode: certificate
```

الشكل (٣) : تحديد server host & openserch.host

ج) نشر الشهادات ثم تشغيل الخدمة وربط الـ Dashboard بالـServer

يشترط الدليل وجود wazuh-certificates.tar لنشر شهادات الـ Dashboard في /etc/wazuh-dashboard/certs ثم تشغيل الخدمة. بعد ذلك يتم تعديل الملف: /usr/share/wazuh-dashboard/data/wazuh/config/wazuh.yml لتحديد عنوان Wazuh Server عادةً عبر منفذ API

كما يوضح الدليل طريقة الدخول الأولى إلى الواجهة عبر HTTPS باستخدام بيانات الاعتماد الافتراضية في سيناريو الـ step-by-step

رابعا تنصيب Wazuh agent :

يوضح الدليل الرسمي أن تنصيب وكيل Wazuh على Linux يمكن تنفيذه باستخدام **Deployment variables** لتسهيل عمليتي التثبيت والتسجيل والتهيئة الأولية، حيث يتم تحديد عنوان الـ Manager عبر متغير مثل WAZUH_MANAGER أثناء التثبيت، ثم تشغيل خدمة الوكيل .

خطوات التنصيب وفق الدليل

١. تنفيذ أمر التثبيت باستخدام مدير الحزم، مع تعيين قيمة WAZUH_MANAGER بعنوان مدير Wazuh (IP/Hostname).

٢. تفعيل خدمة wazuh-agent وتشغيلها .

٣.٣.٢ تنصيب Suricata

اعتمد المشروع على تنصيب Suricata بوصفها **نظام كشف اختراق شبكي (IDS)** لإجراء فحص لحركة المرور وإنتاج تنبيهات يتم تسجيلها بصيغة JSON داخل ملف eve.json، بحيث تكون هذه المخرجات قابلة للتجميع والمعالجة لاحقاً ضمن منصة .Wazuh تم تنفيذ خطوات التنصيب والإعداد وفق الدليل الرسمي الذي يوضح آلية إعداد Suricata على **Ubuntu 22.04**

(أ) تثبيت Suricata على جهاز المراقبة(Endpoint)

يبدأ الدليل بإضافة مستودع Suricata المستقر ثم تحديث قائمة الحزم وتنصيبSuricata

(ب) تنزيل وتفعيل مجموعة القواعد(Ruleset)

يوضح الدليل تنزيل حزمة قواعد **Emerging Threats** الخاصة بـ Suricata ثم استخراجها وإنشاء مسار قواعد مخصص ونقل ملفات القواعد إليه.

(ج) تهيئة ملف Suricata الرئيسي suricata.yaml

بعد تثبيت الأداة والقواعد، يتم ضبط المتغيرات الأساسية داخل الملف /etc/suricata/suricata.yaml بما يضمن:

- تعريف نطاق الشبكة الداخلية (HOME_NET) بشكل صحيح.
- تحديد مسار القواعد الافتراضي.(default-rule-path)
- تحميل ملفات القواعد.(rule-files)
- تفعيل الإحصاءات العامة (اختياري ضمن الدليل).

- تحديد واجهة الشبكة المراد مراقبتها ضمن af-packet مثل enp0s3

```
HOME_NET: "<UBUNTU_IP>"
EXTERNAL_NET: "any"

default-rule-path: /etc/suricata/rules
rule-files:
  - "*.rules"

stats:
  enabled: yes

af-packet:
  - interface: enp0s3
```

الشكل (٤) :تهيئة IDS

(د) إعادة تشغيل Suricata لتطبيق الإعدادات

بعد حفظ التعديلات في suricata.yaml، يتم إعادة تشغيل خدمة Suricata لتطبيق الإعدادات الجديدة

(هـ) تمكين Wazuh Agent من قراءة سجلSuricata eve.json

لكي تصل تنبيهات Suricata إلى منصةWazuh ، يوضح الدليل إضافة مقطع داخل ملف إعداد وكيل Wazuh:

/var/ossec/etc/ossec.conf

بحيث يقرأ الوكيل ملف سجل:Suricata

/var/log/suricata/eve.json

وبصيغة .json

```
<ossec_config>
  <localfile>
    <log_format>json</log_format>
    <location>/var/log/suricata/eve.json</location>
  </localfile>
</ossec_config>
```

الشكل (٥) : إضافة localfile في ملف ossec

(و) إعادة تشغيل Wazuh Agent لتطبيق إعداد جمع السجلات

بعد تعديل إعدادات الوكيل، يتم إعادة تشغيل خدمة Wazuh Agent لتفعيل مراقبة ملف eve.json

٣.٣.٣ تنصيب DVWA & MYSQL & APACHE

في هذا الجزء تم تجهيز بيئة تطبيق ويب ضعيف أمنياً (DVWA) ليعمل فوق خادم ويب (Apache) مع ربطه بقاعدة بيانات(MySQL) ، بهدف توفير بيئة اختبار تولّد طلبات HTTP واقعية يمكن استخدامها لاحقاً في سيناريوهات الكشف والتحليل. تم تنفيذ الخطوات على نظام Linux

(أ) تنصيب Apache و(Web Server Stack) PHP

تم تنصيب **Apache** كخادم ويب و **PHP** كبيئة تنفيذ لتطبيق DVWA بعد التثبيت تم تفعيل الخدمة والتأكد من أنها تعمل وتستجيب لطلبات HTTP ، لأن ذلك شرط أساسي قبل نشر ملفات التطبيق.

(ب) تنصيب وتجهيز قاعدة بيانات

تم تنصيب **MySQL** كـ Database backend لتطبيق DVWA ، ثم تم إنشاء **Database** ومستخدم مخصص للتطبيق وتحديد صلاحياته، وبعدها تم الاعتماد على هذه البيانات في إعدادات DVWA (Database name / username / password / host / port)وجود MySQL بشكل جاهز ومهيأ يضمن أن DVWA سيعمل بخصائصه المعتمدة على SQL بشكل صحيح.

(ج) نشر DVWA على Apache وربطه بقاعدة البيانات

بعد تجهيز Apache وMySQL، تم نشر ملفات **DVWA** ضمن مسار الويب الخاص بـ Apache ليصبح التطبيق قابلاً للوصول عبر المتصفح. ثم تم ضبط ملفات الإعداد الخاصة بـ DVWA لإدخال بيانات الاتصال بـ MySQL بعد ذلك تم التحقق من جاهزية التطبيق عبر صفحة **Setup Check** والتأكد من:

- توفر وحدات PHP المطلوبة مثل mysqli pdo_mysql
- صلاحيات الكتابة للمجلدات التي يتطلبها التطبيق مثل config وuploads
- نجاح الاتصال بقاعدة البيانات.

٣.٤ ربط الأدوات

بعد إتمام تنصيب جميع الأدوات، تم تنفيذ عملية الربط بينها وفق سلسلة عمل متكاملة تضمن انتقال البيانات من نقطة توليدها وحتى ظهورها كتنبيه قابل للتحليل داخل منصة Wazuh تبدأ السلسلة من طبقة التطبيق، حيث تم تشغيل DVWA على خادم الويب Apache وربطه بقاعدة البيانات MySQL بحيث يعمل التطبيق بصورة طبيعية ويولّد طلبات HTTP فعلية

تمثل هذه الخطوة أساس بيئة الاختبار لأنها تضمن أن التفاعلات داخل DVWA تنتج حركة مرور واقعية مرتبطة باستعلامات SQL ، ما يسمح لاحقاً برصد مؤشرات SQL Injection في أماكن مختلفة من الطلب مثل URI و RAW-URI و HTTP Request Body

نقطة التسليم الأولى (Handover Point) في المنظومة كانت هنا :حركة HTTP الناتجة عن DVWA/Apache هي المدخل الذي تعتمد عليه طبقة الكشف الشبكي. بعد ذلك تم إعداد Suricata للعمل بوصفها IDS فقط ضمن المختبر، وتم ضبطها لالتقاط حركة المرور عبر واجهة الشبكة المعتمدة en0s3 ثم تحليل بروتوكول HTTP ومطابقة المحتوى مع قواعد الكشف المفعّلة.

عند تحقق المطابقة تقوم Suricata بتوليد حدث تنبيه وتسجيله ضمن ملف السجل القياسي /var/log/suricata/eve.json بصيغة JSON؛ ويحتوي هذا الحدث على عناصر أساسية تم الاعتماد عليها في المشروع، أهمها نوع الحدث event_type:"alert" ومعرّف التوقيع signature_id ووصف التوقيع signature، إضافة إلى معلومات المصدر والوجهة (IP/Ports) التي تحفظ سياق التنبيه.

نقطة التسليم الثانية كانت :ملف eve.json باعتباره واجهة التكامل الرسمية بين Suricata ومنصة Wazuh ولإيصال هذه التنبيهات إلى منصة التحليل المركزية، تم ربط Suricata مع Wazuh عبر Wazuh Agent من خلال تفعيل جمع السجلات من ملف eve.json مباشرة، بحيث يقوم الوكيل بمراقبة الملف وقراءة كل حدث جديد وإرساله إلى Wazuh Manager بشكل مستمر.

بعد وصول الأحداث إلى جهة الإدارة يقوم Wazuh Manager بمعالجتها عبر آليات التحليل (Decoding/Parsing) ثم تطبيق قواعد Wazuh التي تم إعدادها لتحديد أن الحدث صادر عن Suricata وأنه يرتبط بتواقيع SQLi المعتمدة، وبعد ذلك تُرسل النتائج إلى Wazuh Indexer لغرض التخزين والفهرسة.

نقطة التسليم الثالثة كانت : نقل الأحداث المُعالجة من Wazuh Manager إلى Wazuh Indexer لضمان حفظها وإتاحتها للبحث والتحليل. أخيراً، يتم عرض البيانات والتحقق منها عبر Wazuh Dashboard باستخدام البحث داخل الواجهة للتأكد من وصول أحداث Suricata وتصنيفها بالشكل الصحيح، وبذلك تكتمل السلسلة التشغيلية من طرف إلى طرف (DVWA/Apache/MySQL) :يولّد حركةHTTP ، و (Suricata) يكتشف ويسجل التنبيه، و (Wazuh Agent)يجمع السجل، و (Wazuh Manager)يحلل ويصنّف، ثم (Indexer/Dashboard)يخزّن ويعرض النتائج للتحليل.

أهم التعليمات/الإعدادات التي حققت الربط بين المكوّنات

- تحديد واجهة الالتقاط في Suricata: ضبط الواجهة ضمن قسم af-packet في ملف suricata.yaml إلى interface: en0s3 لضمان التقاط حركة HTTP المتجهة إلى خادم DVWA؛ أي خطأ في اسم الواجهة يؤدي إلى عدم التقاط الحركة وبالتالي عدم توليد أي تنبيهات.
- تفعيل سجل Suricata بصيغة EVE JSON وتحديد موقعه: الاعتماد على الملف /var/log/suricata/eve.json كمصدر أساسي للتنبيهات الشبكية، لأنه يسجل أحداث alert بالحقول اللازمة للتحليل داخل Wazuh.
- تعريف جمع السجلات في Wazuh Agent: إضافة كتلة localfile ضمن ملف إعداد الوكيل /var/ossec/etc/ossec.conf مع:
  - log_format = json لضمان تفسير السجل كبنية JSON قابلة للاستخراج القاعدي.
  - Location = /var/log/suricata/eve.json لتحديد ملف السجلات الذي تتم مراقبته.
- إعادة تشغيل الخدمات بعد أي تعديل: إعادة تشغيل Suricata بعد تعديل suricata.yaml، وإعادة تشغيل wazuh-agent بعد تعديل ossec.conf للتأكد من تطبيق الإعدادات فعلياً على مستوى التشغيل وليس فقط حفظها في الملفات.
- التحقق من وصول الأحداث عبر Wazuh Dashboard: إجراء تحقق عملي عبر واجهة Dashboard باستخدام البحث/الاستعلام للتأكد من ظهور أحداث Suricata ، وهو ما يثبت نجاح الربط بين طبقة الكشف الشبكي ومنصة التجميع والتحليل المركزية.

الفصل الثالث : الحل المقترح

٣.١ مقدمة الفصل

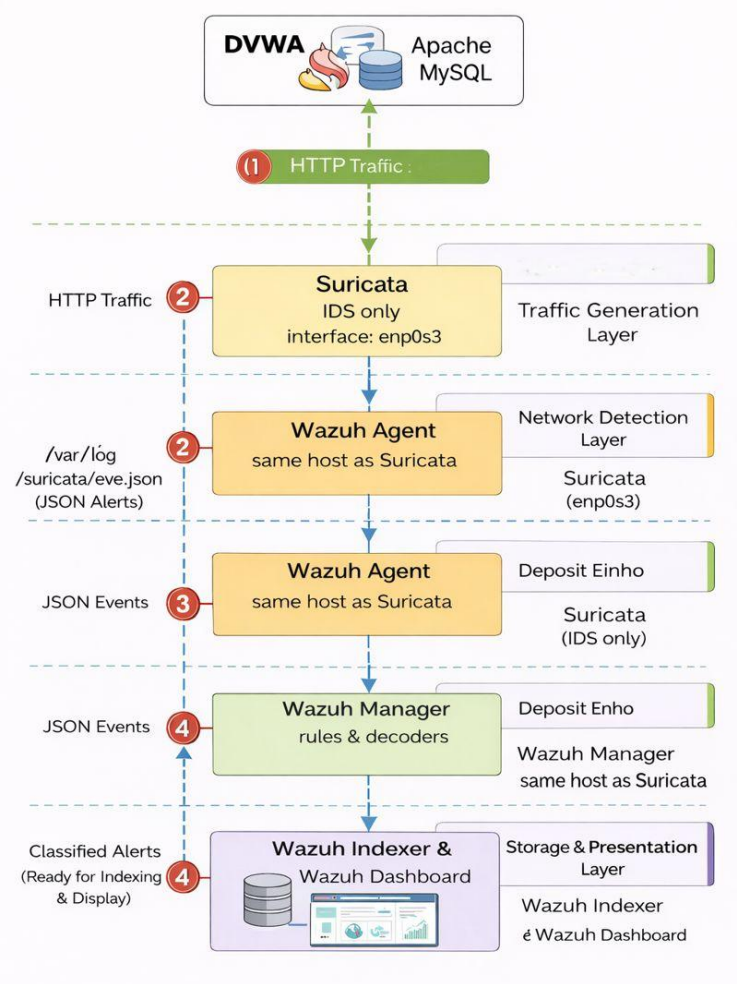
يقدّم هذا الفصل الحل المقترح لكشف هجمات SQL Injection ضمن بيئة ويب مخبرية عبر تصميم طبقي (Layered Architecture) يدمج بين طبقة توليد الحركة(DVWA/Apache/MySQL) ، وطبقة كشف شبكي تعمل كـ IDS (Suricata) لإنتاج تنبيهات توقيعية، ثم طبقة تجميع وتحليل مركزية (Wazuh) لتحويل هذه التنبيهات إلى أحداث مصنّفة وقابلة للبحث والعرض. يركّز الحل على نوعين رئيسيين من هجمات SQLi وفق التصنيف المعتمد في المشروع:

١. Blind/Inferential SQL Injection مثل (Time Delay و Boolean Exploitation و Heavy Query)
٢. In-band SQL Injection مثل (Error-based و UNIONو Tautologyو Comments و Piggybacked)

ولا يتضمن هذا الحل هجمات Out-of-Band SQL Injection لأنها خارج نطاق المشروع الحالي.

يتكون الحل من خمس طبقات مترابطة، لكل طبقة مسؤولية محددة ومدخلات ومخرجات واضحة:

١. Traffic Generation Layer توليد طلبات HTTP واقعية عبر DVWA على Apache وربطها بـMySQL
٢. Network Detection Layer مراقبة حركة HTTP واكتشاف أنماط SQLi باستخدام Suricata
٣. Log Collection Layer جمع سجل تنبيهات Suricata (eve.json) عبر Wazuh Agent الموجود على نفس جهاز Suricata
٤. Analytics & Classification Layer تحليل أحداث Suricata داخل Wazuh Manager وتصنيفها وفق نوع SQLi وموضعه في الطلب.(URI/RAW-URI/BODY)
٥. Storage & Presentation Layer فهرسة الأحداث داخل Wazuh Indexer وعرضها والتحقق منها عبر Wazuh Dashboard



الشكل (٦) : طبقات الحل

٣.٢ طبقة توليد الحركة Traffic Generation Layer

المكونات DVWA + Apache + MySQL :

الهدف :إنتاج حركة HTTP تمثل سيناريوهات إدخال واقعية يمكن أن تحتوي مؤشرات SQLi ضمن أجزاء الطلب المختلفة.

تعتمد هذه الطبقة على تشغيل DVWA كـ Web application يعمل عبر Apache ويستخدم MySQL كـ Database backend تكمن أهمية وجود قاعدة البيانات في أن مدخلات DVWA تمر بمنطق يعتمد على SQL، وبالتالي تصبح هجمات SQLi ضمن المدخلات "ذات أثر" داخل مسار التطبيق، ما يجعل الحركة المتولدة أقرب لسلوك تطبيقات الويب الواقعية. ضمن المشروع، بحيث يمكن إعادة توليد نفس الأنماط خلال الاختبارات والتأكد من قدرة طبقة الكشف على التقاطها.

على مستوى سطح الهجوم (Attack Surface) ، يمكن أن تظهر مؤشرات SQLi ضمن:

- URI / Query Parameters في طلبات GET
- RAW-URI في حال استخدام Encoding
- HTTP Request Body في طلباتPOST

Handover Point 1:

الخرج الأساسي من هذه الطبقة هو HTTP Traffic المتجه إلىApache ، وهو المدخل المباشر لطبقة Suricata التي تقوم بالالتقاط والتحليل.

٣.٣ طبقة الكشف الشبكي Network Detection Layer

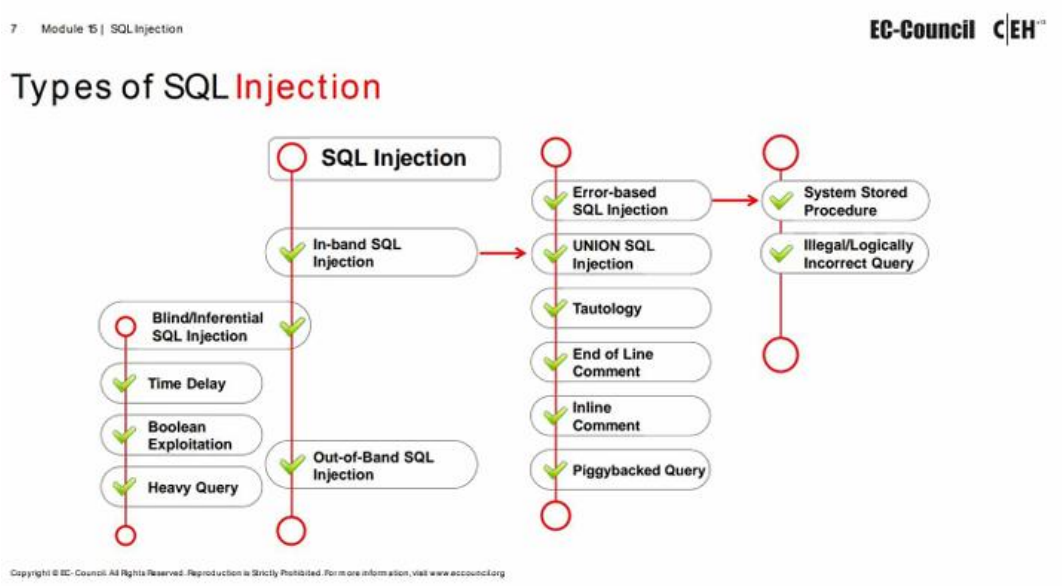
الهدف : كشف أنماط SQLi على مستوى الشبكة عبر قواعد توقيعية، وتوليد Alerts بصيغة JSON قابلة للتجميع.

في هذه الطبقة تعمل Suricata كـ IDS فقط، أي أن دورها يقتصر على المراقبة والكشف والتسجيل دون منع الحركة. تم ضبط الالتقاط على الواجهة en0s3 لضمان رصد حركة HTTP التي تولّدها طبقة DVWA/Apache. بعد الالتقاط، تقوم Suricata بتحليل HTTP واستخراج عناصر قابلة للتفتيش مثل URI و RAW-URI وBody، ثم تطبيق قواعد SQLi المخصّصة التي تم إعدادها في المشروع.

٣.٣.١ منطق قواعد SQLi ضمن Suricata في المشروع

اعتمد المشروع على قواعد SQLi مخصّصة، وتم تنظيمها بحيث تحقق هدفين:

١. التغطية حسب التصنيف المعتمد:



الشكل (٧) : تصنيف sqli

ولأغراض الاتساق والتحليل اللاحق داخلWazuh ، تم ترقيم هذه القواعد ضمن نطاق Signature IDs (SID) = 2001xxx بحيث يصبح تمييز تنبيهات SQLi وتمريرها إلى مرحلة التصنيف أسهل وأكثر دقة.

٣.٣.٢ مخرجات Suricata ونقطة التسليم

عند تحقق قاعدة معينة، تقوم Suricata بإنشاء Alert وتسجيله في الملف /var/log/suricata/eve.json: بصيغة JSON

Handover Point 2:

ملف eve.json يمثل قناة التسليم الرسمية بين Suricata وطبقة التجميع فيWazuh

٣.٤ طبقة التجميع Log Collection Layer

المكونات Wazuh Agent : على نفس جهاز Suricata

الهدف : تحويل سجل Suricata المحلي إلى تدفق أحداث مُرسل إلى Wazuh Manager بصيغة قابلة للمعالجة.

تؤدي هذه الطبقة دوراً حاسماً لأنها تمثل نقطة الانتقال من "تنبيه محلي على جهاز IDS إلى حدث مركزي داخل SIEM ، يوجد Wazuh Agent على نفس الجهاز الذي تعمل عليه Suricata ، وهذا يمنح ميزتين عمليتين: لا توجد حاجة لنقل ملفات عبر الشبكة أو مشاركة Storage ويقل احتمال فقد السجلات لأن القراءة تتم مباشرة من الملف المحلي.

تم تفعيل جمع السجلات عبر إضافة تعريف قراءة ملف eve.json ضمن إعدادات الوكيل، مع ضبط :

log_format = json لضمان التعامل مع الأحداث كبنية حقول، لا كنص خام

location = /var/log/suricata/eve.json لتحديد مصدر السجل بشكل صريح.

Handover Point 3:

لحظة وصول الحدث من Wazuh Agent إلى Wazuh Manager هي الحد الفاصل بين طبقة التجميع وطبقة التحليل والتصنيف.

٣.٥ طبقة التحليل والتصنيف Analytics & Classification Layer

المكونات Wazuh Manager (Rules/Decoding) :

الهدف :تحويل Alerts القادمة من Suricata إلى تنبيهات مصنّفة ضمن سياق SQLi الخاص بالمشروع.

تُعد هذه الطبقة "القيمة المضافة" للحل المقترح، لأن Suricata بمفردها تنتج Alert توقيعي، لكن Wazuh يضيف بعداً تحليلياً وإدارياً عبر :

- استخراج الحقول من JSON وربطها بمصدرها(Suricata)
- تطبيق قواعد محلية للتعرّف على Alert كحدث SQLi ضمن نطاق SIDs الخاص بالمشروع (2001xxx).
- تصنيف التنبيه وفق نوع SQLi وموضعه في الطلب (URI/RAW-URI/BODY) عندما يكون ذلك ممكناً عبر معلومات التوقيع أو الـSID
- رفع مستوى الخطورة (Level) وإضافة مجموعات (Groups/Tags) تجعل البحث داخل Dashboardأكثر تنظيماً وتمنح التقرير مخرجات واضحة.

عملياً، يعتمد التصنيف في Wazuh على حقيقتين أساسيتين من تصميم المشروع:

١. أحداث Suricata تصل إلى Wazuh كـJSON ، وبالتالي يمكن استخراج signature_id و signature بدقة.

٢. قواعد Suricata مميّزة بـ SID range 2001xxx، ما يتيح لـ Wazuh تعريف قواعد "شاملة" تلتقط SQLi دون الاعتماد على نصوص قد تتغير، ثم قواعد أدق لكل نوع/جزء عند الحاجة.

Handover Point 4:

بعد تطبيق قواعد التحليل والتصنيف، تُرسل الأحداث إلى Wazuh Indexer لتخزينها وفهرستها، وتصبح جاهزة للعرض داخلDashboard

مثال على قاعدة من WAZUH :

```
<!-- Piggybacked/Stacked -->
<rule id="910017" level="12">
  <if_sid>910001</if_sid>
  <field name="alert.signature_id">^20017[0-9][0-9]$</field>
  <description>SQLi IN-BAND - Piggybacked/stacked statement indicators</description>
  <group>sqli,inband,piggybacked</group>
</rule>

<!-- Correlation: multiple SQLi alerts from same src_ip -->
<rule id="910050" level="15" frequency="5" timeframe="120">
  <if_matched_sid>910001</if_matched_sid>
  <same_field>src_ip</same_field>
  <description>SQL Injection attack campaign detected (multiple attempts)</description>
  <group>sqli,correlation,critical</group>
```

الشكل (٨) : مثال على ربط Wazuh

SQLi Category	Sub-type (as in project)	Inspection Target
Blind/Inferential SQL Injection	Time Delay	http.uri / http.uri.raw / http.request_body
Blind/Inferential SQL Injection	Boolean Exploitation	http.uri / http.uri.raw / http.request_body
Blind/Inferential SQL Injection	Heavy Query	http.uri / http.uri.raw / http.request_body
In-band SQL Injection	Error-based SQL Injection	http.uri / http.uri.raw / http.request_body
In-band SQL Injection	UNION SQL Injection	http.uri / http.uri.raw / http.request_body
In-band SQL Injection	Tautology	http.uri / http.uri.raw / http.request_body
In-band SQL Injection	End of Line Comment	http.uri / http.uri.raw / http.request_body
In-band SQL Injection	Inline Comment	http.uri / http.uri.raw / http.request_body
In-band SQL Injection	Piggybacked Query	http.uri / http.uri.raw / http.request_body
In-band SQL Injection	System Stored Procedure	http.uri / http.uri.raw / http.request_body
In-band SQL Injection	Illegal/Logically Incorrect Query	http.uri / http.uri.raw / http.request_body

جدول (٣) : تقسيم قواعد SUricata

٣.٦ طبقة التخزين والعرض Storage & Presentation Layer

المكونات : Wazuh Indexer + Wazuh Dashboard

الهدف :حفظ الأحداث المصنّفة وإتاحة البحث والتحقق والتوثيق.

في هذه الطبقة يتم تخزين الأحداث داخل Indexer وفهرستها، ما يتيح إجراء عمليات بحث مبنية على الحقول (Field-based search) مثل البحث بـ signature_id أو مجموعات القواعد. ثم تُستخدم Wazuh Dashboard للتحقق النهائي من أن: تنبيهات Suricata وصلت فعلاً.التصنيف الخاص بـ SQLi تم تطبيقه.يمكن استخراج أدلة التقرير (Screenshots/Queries) بسهولة.

هذه الطبقة ليست للعرض فقط، بل هي جزء من آلية ضمان الجودة، لأن نجاح البحث وإظهار الأحداث يثبت عملياً صحة الربط من طرف إلى طرف.

٣.٧ ملخص نقاط التسليم Handover Points في الحل المقترح

لإظهار التصميم بصورة علمية داخل التقرير، يُلخّص الحل المقترح نقاط التسليم كما يلي:

١. DVWA/Apache → HTTP Traffic مدخل Suricata

٢. Suricata → /var/log/suricata/eve.json JSON Alerts مدخل Wazuh Agent

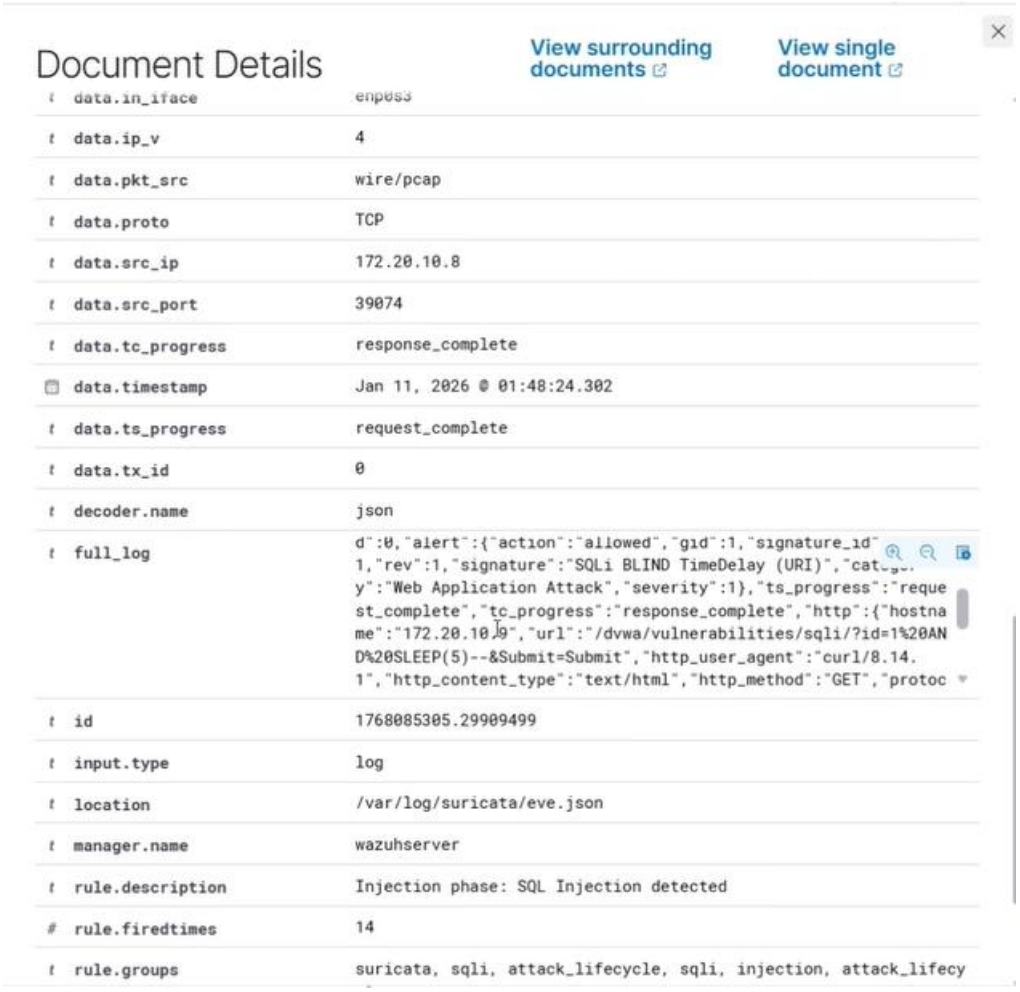
٣. Wa zuh Agent → Wazuh Manager بدء التحليل والتصنيف

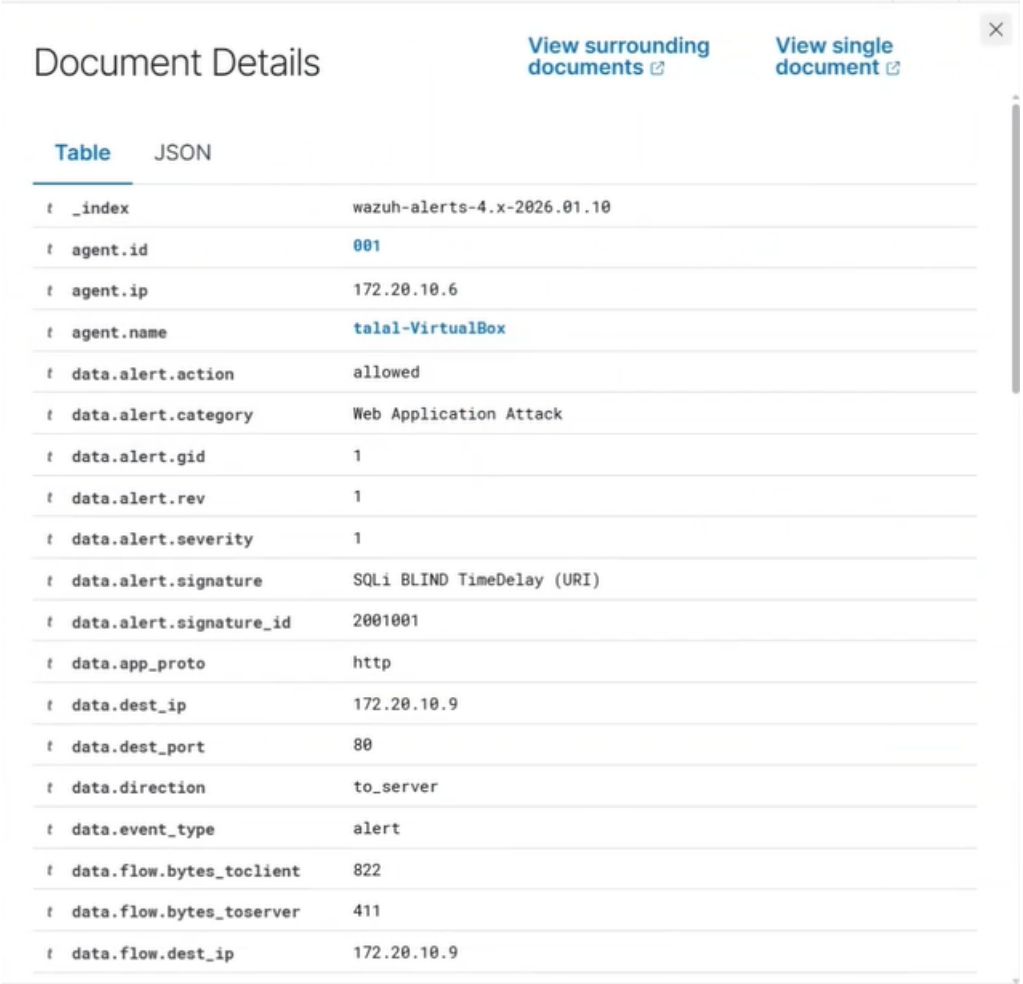
٤. Wazuh Manager → Wazuh Indexer → Wazuh Dashboard تخزين/عرض/بحث

٣.٨ النتائج

تم توثيق نتائج الحل المقترح من خلال التحقق من التنبيهات المعروضة في **Wazuh Dashboard** بعد تشغيل سيناريوهات الاختبار على DVWA اعتمدت عملية التحقق على استخدام عوامل تصفية مباشرة مرتبطة بمخرجات Suricata داخل الأحداث، بحيث تم حصر النتائج أولاً ضمن أحداث التنبيه فقط عبر الحقل event_type: "alert" لضمان أن المعروض يمثل تنبيهات IDS وليس سجلات تشغيلية أخرى. بعد ذلك تم التركيز على تمييز تنبيهات SQLi الخاصة بالمشروع باستخدام الحقل alert.signature_id، حيث جرى فلترة الأحداث ضمن نطاق المعرّفات المخصّصة لقواعد SQLi (2001xxx) ، الأمر الذي أكد أن التنبيهات المعروضة ناتجة عن القواعد التي تم تطويرها واعتمادها داخل Suricata. ولإظهار تنوع الهجمات المغطاة

ضمن الحل، تم استخدام الحقل alert.signature لعرض رسائل القواعد، مما أتاح التحقق من ظهور أكثر من نمط SQLi ضمن نفس مسار المعالجة، بما يشمل أنماط (**Blind/Inferential** مثل Time Delay و Boolean Exploitationو (Heavy Queryوأنماط) **In-band** مثل UNION و Error-basedو Tautologyو Comments و(Piggybacked Query، مع استثناء Out-of-Band لأنه خارج نطاق العمل. كما تم فتح تفاصيل أحداث محددة داخل Dashboard للتحقق من اكتمال السياق التشغيلي للتنبيه، حيث ظهر ضمن الحدث الواحد كل من alert.signature_id و alert.signature بالإضافة إلى معلومات المصدر والوجهة والطابع الزمني، وهو ما يثبت أن التنبيه لم يصل كنص خام بل كحدث منظم يمكن تتبعه وتحليله. بناءً على ذلك، تؤكد النتائج نجاح التكامل من طرف إلى طرف: توليد حركة HTTP عبر DVWA/Apache، اكتشافها شبكياً بواسطة Suricata على الواجهة enp0s3 وتسجيلها في eve.json، ثم جمعها بواسطة Wazuh Agent وإرسالها إلى Wazuh Manager لتطبيق التصنيف، وصولاً إلى فهرستها وعرضها داخل Dashboard مع إمكانية البحث والتنقيب وفق alert.signature_id و alert.signature بصورة تدعم توثيق المشروع وإثبات فعاليته.





الشكل (٩) : مثال واحد على ناتج عن كشف هجوم

بالإضافة إلى المثال الموثّق في **Wazuh Dashboard** (كما في الصور المرفقة) والذي يبيّن حدث كشف من نوع **SQLi BLIND TimeDelay (URI)** مع ظهور الحقول الدالة مثل data.alert.signature و data.alert.signature_id وسياق الشبكة (data.src_ip ، data.dest_ip، data.dest_port) ومصدر السجل (location: /var/log/suricata/eve.json) ومجموعات التصنيف(rule.groups) ، فقد أظهرت النتائج أن **باقي أنماط SQL Injection المعتمدة ضمن المشروع** تم كشفها وتسجيلها وعرضها **بنفس البنية والأسلوب** ضمن لوحة التحكم. حيث تم رصد تنبيهات تمثل بقية فئات (**Blind/Inferential SQLi** و **In-band SQLi**) ظهرت كأحداث alert ضمن Dashboard مع قيم مميزة في data.alert.signature و data.alert.signature_id ضمن نطاق المعرّفات المخصّص لقواعد المشروع(2001xxx) ، وبنفس سياق المصدر/الوجهة والطابع الزمني. وتم الاكتفاء بإدراج **مثال واحد ممثل** في التقرير لتوثيق آلية العرض والتحقق

دون إغراق الفصل بعدد كبير من اللقطات المتشابهة، وذلك لتجنب ازدحام المحتوى البصري مع بقاء إمكانية الرجوع إلى جميع الأدلة التفصيلية متاحة داخل Dashboard عند الحاجة عبر الفلاتر المعتمدة على data.alert.signature و data.alert.signature_id

٤.١ الخاتمة

قدّم هذا المشروع حلاً عملياً ومتكاملاً لكشف هجمات **SQL Injection** ضمن بيئة ويب مخبرية، من خلال تصميم طبقي يدمج بين **Suricata** كـ **IDS** لرصد أنماط الهجوم على مستوى الشبكة، و **Wazuh** كمنصة تجميع وتحليل مركزية لإدارة السجلات وتصنيف التنبيهات وعرضها. تم بناء البيئة التجريبية بالاعتماد على **DVWA** فوق **Apache** مع قاعدة بيانات **MySQL** لتوليد حركة **HTTP** واقعية تمثل سيناريوهات إدخال قابلة للاحتواء على مؤشرات SQLi في مواضع متعددة من الطلب مثل **URI** و **RAW-URI** و **HTTP Request Body**. بعد ضبط Suricata على واجهة الالتقاط enp0s3 وتفعيل مخرجات **EVE JSON** في الملف eve.json، تم تمكين **Wazuh Agent** على نفس جهاز Suricata من قراءة السجلات بصيغة json وإرسالها إلى **Wazuh Manager**، حيث تم تطبيق قواعد محلية تستفيد من نطاق **signature_id** **(2001xxx)** لتمييز تنبيهات SQLi وفصلها عن غيرها وتصنيفها ضمن مجموعات ومستويات ملائمة. أثبتت النتائج ظهور التنبيهات بشكل منظم داخل **Wazuh Dashboard** مع احتفاظها بسياقها التشغيلي (المصدر/الوجهة، المنفذ، البروتوكول، والطابع الزمني)، مما يؤكد نجاح التكامل من طرف إلى طرف وفاعلية مسار التحليل والتوثيق. وضمن نطاق المشروع تم التركيز على فئتي **Blind/Inferential** و **In-band SQLi**، مع استثناء **Out-of-Band** كخيار خارج نطاق التنفيذ الحالي.

٤.٢ الآفاق المستقبلية

على الرغم من أن الحل المقترح أثبت قابلية الكشف والتجميع والتصنيف المركزي بشكل ناجح، إلا أن هناك مسارات تطوير مستقبلية يمكن أن ترفع من دقة الكشف وجودة النتائج وقيمة الحل التشغيلية، أهمها:

١. **توسيع نطاق التغطية إلى Out-of-Band SQLi**

إضافة سيناريوهات OOB ضمن بيئة اختبار مناسبة، وتطوير قواعد كشف أو مؤشرات سلوكية تدعم هذا النوع، بما يوسّع نطاق الحل ليشمل تصنيفات SQLi كاملة.

٢. **تحسين منهجية تقليل False Positives عبر Correlation أعمق**

حالياً يتم الاعتماد على منطق التوقيعات (signature-based). يمكن تطوير Correlation داخل Wazuh(أو دمج مصادر سجلات إضافية لاحقاً) لتمييز المحاولات التي نجحت فعلياً عن محاولات

الاستطلاع أو الضجيج، وإضافة شروط سياقية مرتبطة بالحالة (مثل تكرار الهجوم أو تسلسل الأحداث).

٣. **تعزيز التصنيف ليشمل "مرحلة الهجوم(Attack Lifecycle)** "

البناء على مفهوم chain داخل قواعد Wazuh لتمييز مراحل مثل probing → exploitation → post-exploitation indicators ضمن حدود ما تسمح به البيانات المتاحة attempt :

٤. **مراجعة القواعد من منظور الأداء والتطويع**

إجراء اختبار منهجي لتأثير القواعد على الأداء، وتطوير تحسينات مثل تقليل تعابير مطابقة مكلفة أو إضافة شروط "context" أكثر دقة لتقليل المطابقات غير الضرورية.

مراجع

[١] C. G. C. J. Y. P. X. L. Houlong Fu، SDSIOT: An SQL Injection Attack Detection and Stage Identification Method Based on Outbound Traffic، MDPI (Published in the journal Electronics, Volume 12, Issue 11)، ٢٠٢٣.

[٢] EC-Council، "Types of SQL Injection"، Available: https://www.eccouncil.org. [تاريخ الوصول ٢٠٢٤].

[٣] I. Wazuh، "Wazuh Documentation"، Available: https://documentation.wazuh.com. [تاريخ الوصول ٢٠٢٦].

[٤] O. I. S. Foundation، "Suricata User Guide"، Available: https://suricata.io. [تاريخ الوصول ٢٠٢٦].