Case Study: Intrusion Detection & Prevention System (IDPS) for SQL Injection Attacks

Background

SQL Injection (SQLi) remains one of the most critical security threats to web applications (OWASP Top 10). Our Intrusion Detection & Prevention System (IDPS) was deployed in a demo environment to monitor and block SQLi attempts in real-time. This case study presents insights derived from analyzing malicious logs captured by the system.

Timestamp (UTC)	IP Address	Endpoint	Payload	Action	Reasons
2025-08-25T14:10:00Z	192.168.1.25	/	' OR 1=1 --	Blocked	Matched SQLi pattern, contains oper
2025-08-25T14:11:30Z	192.168.1.25	/	UNION SELECT * FROM users	Blocked	UNION SELECT pattern
2025-08-25T14:15:10Z	10.0.0.50	/	DROP TABLE users;	Blocked	DROP TABLE keyword detected
2025-08-25T14:20:42Z	172.16.0.77	/	admin' --	Blocked	Excessive quotes, suspicious termina

Captured Malicious Queries (Sample Logs)

Below are selected entries recorded in the IDPS security logs. Each row shows when an attack attempt was made, the originating IP, the malicious payload, and the reason it was flagged.

Attack Patterns Identified

1. Authentication Bypass Attempts: Payloads like `' OR 1=1 --` indicate attackers trying to bypass login or filters.
2. Data Extraction Attempts: `UNION SELECT * FROM users` shows attempts to extract sensitive user data.
3. Destructive Queries: `DROP TABLE users;` highlights an intent to damage or wipe critical data.
4. Reconnaissance Probes: Inputs like `admin' --` suggest probing for system response.

System Response

• All malicious queries were blocked before reaching the database.
• Logs were stored with IP, payload, and reasons, enabling forensic analysis.

• Admin dashboard allowed real-time monitoring of suspicious activity.

Lessons Learned

• Attackers often begin with simple SQLi payloads (OR 1=1, UNION SELECT) to test vulnerabilities.
• Layered defense combining parameterized queries with IDPS detection provides robust protection.
• Logging is critical, not only for blocking attacks but also for post-incident analysis and improving defenses.

Recommendations

• Expand detection to include time-based SQLi (e.g., SLEEP attacks).
• Add alerting mechanisms (email/SMS) when repeated attacks occur from the same IP.
• Integrate with SIEM tools for enterprise-scale monitoring.
• Use threat intelligence feeds to blacklist malicious IPs.

Outcome

The IDPS successfully identified and prevented SQL injection attempts. It not only protected the database but also generated actionable insights for strengthening future security practices.