Advanced International Journal for Research (AIJFR)
E-ISSN: 3048-7641 ● Website: www.aijfr.com ● Email: editor@aijfr.com

Detection of Web Attacks Using Signature-Based Network Intrusion Detection System

Aditya Yadav[1], Dhruv Pratap Singh[2], Yash Borai[3], Dr. Shipra Saraswat[4]

[1,2,3]Student, [4]Professor
[1,2,3,4]Amity school of engineering and technology

Abstract

Web apps spreading fast have made networks more visible to digital dangers along with harmful intrusions. Instead of secure walls, weak spots appear through actions like slipping code into databases, sneaking scripts across sites, flooding systems until they freeze, or guessing passwords nonstop. Watching data move becomes key when spotting risks early - this is where tools called intrusion detectors step in. Rather than guess what might happen, these systems often rely on stored blueprints of past attacks to raise alarms. Matching live traffic against familiar danger signs works well - if the pattern fits, the alert sounds. A pattern-matching tool scans data flowing across networks, checking each piece against stored records of recognized cyberattacks to spot harmful actions. If something lines up, warnings pop up so tech staff know trouble might be near. This document looks into how such tools are built, what they do, and how they run when catching online threats. It weighs their strong points, weak spots, and real-world results spotting different kinds of digital dangers. Results show these systems work well on familiar risks, boosting protection if fresh threat details get added often.

Index Terms- Network Security, Signature-Based Intrusion Detection, Web Application Attacks, Deep Packet Inspection, Threat Detection, Cyber Attack Detection, Pattern Matching Algorithms, HTTP/HTTPS Traffic Analysis, Security Monitoring Systems.

1. Introduction

Fast advances in internet tools plus how much people now rely on websites changed the way we talk, work, and share information. E-commerce sites, digital banking, along with cloud platforms form part of everyday routines today. Yet that heavy reliance makes them attractive spots for hackers to strike. Criminals hunt flaws in these systems using methods like SQL Injection, Cross-Site Scripting (XSS), injecting commands, or moving through folders without permission - these moves often cause stolen data, lost money, broken systems. Because of this, protecting apps online and their networks matters more than ever before.

When dangers pop up, systems that catch intrusions help keep an eye on digital activity across networks. These tools scan bits of information moving around online spaces - especially ones called NIDS zero in on those details. Instead of guessing, they rely heavily on blueprints of past break-ins. Think of them checking each message against a library built from old alerts. Spotting familiar risks becomes quick work because matches stand out clearly. Fewer mistakes happen when the threat has shown up before. Accuracy

stays strong under normal conditions. That reliability helps teams act fast without second-guessing too much.

A single step into network defense begins here - using Python to build a signature-driven NIDS that spots typical web threats as they happen. Capturing live traffic, the tool pulls out key parts of HTTP requests, then checks them against known attack patterns through precise matching routines. Though quick to set up and works well for familiar dangers, it cannot catch brand-new exploits hiding in plain sight. Still, this version teaches core ideas about spotting intrusions, opening doors later for smarter upgrades like behavioral analysis or adaptive models down the road.

1. Signature Based NIDs

One way to spot trouble on a network? A signature-based NIDS keeps watch over data moving through systems. Instead of guessing threats, it checks each packet against a list of familiar danger signs. Think of those signs as digital fingerprints left behind by past attacks - specific code bits, odd character chains, or repeated risky actions. It does not learn new tricks; it knows only what has already been seen. Whenever live traffic lines up with one of these markers, alarms go off. Someone gets notified. That match might mean an old threat is trying again. The whole process runs fast, silent, automatic. Recognition happens in moments, not minutes. Patterns stay fixed unless someone updates them. No guesswork involved, just comparison. Alerts pop when similarities appear. This method misses unknown tactics. Yet for repeat offenders, it works. Protection depends entirely on how current the list remains. Old records mean blind spots grow.

A signature-based NIDS runs by moving through distinct phases. Right away, it pulls live traffic from the network via packet capture methods. Following that, raw data gets cleaned - important bits like protocol headers and message contents pulled out, mostly focusing on things like web requests. Next up comes scanning: pieces of data meet defined templates, matched using rules or regex patterns built to spot known threats. Each step flows into the next without pause, keeping detection steady under changing conditions. When the packet data matches a signature from the database, it flags the attack kind - like SQL injection or XSS - and sends an alert. Quick detection of familiar threats happens smoothly, using little processing power along the way.

Spot on with familiar threats, signature-driven NIDS rarely raises a false alarm - beats many alternative techniques hands down. Built without heavy complexity, fits neatly into modest setups just as well as sprawling corporate networks. Yet blind spots exist. Anything fresh, anything unseen before, slips through quiet - no match found, no alert triggered. Now here's a twist - attackers tweak their moves just enough to slip past alarms, using tricks to hide what they’re doing. Because of this sneaky shift, keeping the system’s memory fresh with new threat details matters more than it seems at first glance.

Out in the real world, tools like Snort and Suricata - built around known threat patterns - are common choices for spotting cyber risks. Because new dangers pop up often, specialists constantly refresh their rule databases to stay effective. For this study, though, a pared-down version of such a system was coded in Python, focused only on recognizing set attack fingerprints within incoming HTTP data. By checking

traffic against these fixed markers, it shows basic but working detection when catching frequent website-focused attacks as they happen across live networks.

2. Components of Signature-Based NIDs

A Signature-Based Network Intrusion Detection System (NIDS) is composed of several essential components that work together to monitor, analyze, and detect malicious activities in network traffic. Each component plays a specific role in ensuring accurate and efficient detection of known web attacks. Basically it consists of following major components as shown in figure 1:

1. Packet Capture Module
2. Preprocessing Module
3. Signature Database
4. Detection Engine (Pattern Matching Engine)
5. Alert Generation Module
6. Logging and Reporting Module

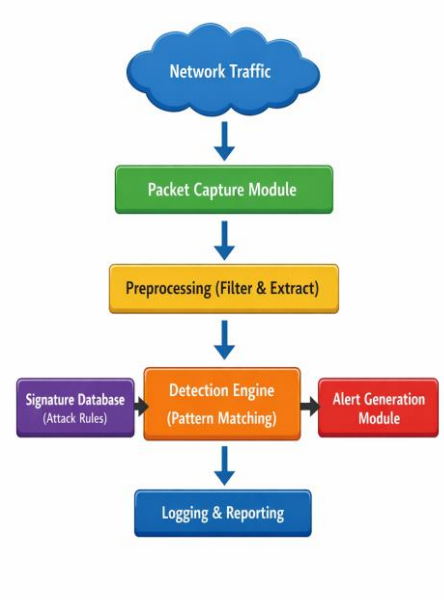

Figure 1: Component of Signature Based NIDs

1.Packet Capture Module:
From the start, this piece handles live network traffic within the NIDS setup. Flowing data gets pulled straight off the wire by tapping into active streams. Instead of guessing, it records every bit using methods like packet sniffing. Programs often lean on tools including Scapy or libpcap to make that happen. What comes out includes full details - headers along with actual content - for deeper inspection later

2. Preprocessing Module:
After capture, packets move into the preprocessing stage. From there, key details emerge - things like where it came from, where it's headed, what kind of protocol it uses, along with its content, particularly if it involves web requests. Decoding might happen here, turning encoded parts back into usable form.

Normalization adjusts formats so everything lines up consistently across samples. Some data gets removed too, especially noise that doesn't help spotting threats faster. Efficiency improves when only meaningful streams remain. Each step quietly shapes raw flow into clearer signals.

3. Signature Database:
Starting off, the signature database holds many ready-made attack templates or guidelines. Because these reflect familiar dangers - like SQL Injection, for example, or XSS and moving through folders illegally - it matters quite a bit what's inside. Maintaining it regularly makes a big difference, since the NIDS can only work as well as the data allows.

4. **Detection Engine (Pattern Matching Engine):**
Right at the center sits the main piece of the setup. When data packets come through, the scanner checks them by lining up what it sees with records saved earlier. Instead of just searching blindly, it uses smart methods like coded rules or text patterns to spot similarities. Finding one of these matches often means something suspicious is happening. That alert signals that an unwanted visitor might be trying to get in.

5. Alert Generation Module:
Something goes wrong, then a signal pops up through the system to mark trouble. The message might show what kind of breach happened, where it came from, where it aimed, along with data carried inside. Messages appear on screens, settle into logs, even reach admin hands when next steps are needed.

6. Logging and Reporting Module:
Keeps track of known threats along with what the system does every day. Because logs help review past events, they support audits while pointing out ways to boost efficiency. Summaries show patterns in attacks, also revealing how well defenses hold up over time.

3. Signature-Based NIDs Topology

Where the sensors sit shapes what a signature-based intrusion detector can catch. Traffic must flow past its line of sight - only then does it see every move. Hidden paths mean missed signs. Position matters because gaps hide threats. Watching the right lanes lets it spot familiar attacks fast. Miss one junction and blind spots grow. Full view comes from smart placement across key links.

Usually, the NIDS sits close to key spots like routers or gateways so it sees all traffic moving in and out as shown in figure 2. Watching without touching - this setup just takes notes on data flow instead of changing anything.

Working Flow:

- Through the network infrastructure, data moves steadily. Information travels across connections without pause. Along cables and wireless paths, signals pass continuously. From device to device, communication shifts quietly. Across systems, messages find their way persistently
- A copy of the traffic is sent to the NIDS

- The NIDS analyzes packets using signature matching
- Motion triggers a warning when something harmful shows up

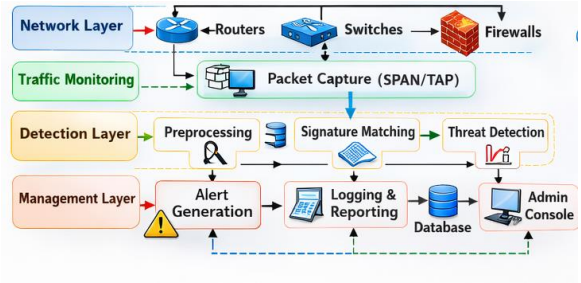

Figure 2 : Signature-Based NIDs Topology

One look at Figure 2 shows how a basic passive Signature-Based NIDS connects into a network. Traffic arrives from the internet, moves past protective gear like firewalls, then passes through routers and switches on its way inward. Instead of altering flow, this design quietly pulls aside duplicates via tools like SPAN ports or taps. That mirrored data travels straight to the detection unit for analysis.

One part grabs copied data flowing across cables. Another checks what inside by matching against known bad patterns. A third holds those rules that help spot trouble. When something harmful shows up, warnings go out automatically. These signals reach a central monitoring screen without delay. It works quietly beside the network instead of within it. No slowdown happens because nothing gets blocked on the fly. This setup stays passive yet spots threats just the same.

4. Tools And Language Used In Signature Based NIDs System

1.Snort

Most folks know Snort as a free tool that spots intrusions by checking network data using prewritten signs of danger. Instead of guessing, it sniffs live traffic, pulling apart each piece to see if anything fits those preset clues. Its logic lives in custom rules - crafted so experts can pinpoint nasty behavior like odd web queries or rare port moves. A trigger happens once something lines up; then, warnings pop, records save, or harmful bits vanish if set up to block threats outright.

Inside Snort, key pieces work together - the packet decoder unpacks data, preprocessors adjust it for analysis, the detection engine checks patterns, while alerts get logged when matches occur. What makes the detection engine stand out is how it compares live network flow against known threat fingerprints. Though built to handle many protocols, its real strength lies in letting people tweak detection rules based on unique needs. Big companies often choose it because it runs steadily, uses resources well, and benefits from active user contributions. Staying sharp means frequently refreshing those rules so new dangers don’t slip through unnoticed.

2.Python

Sometimes folks choose Python because it handles many tasks well when making tailored NIDS that rely on signatures. In schools and labs, you often see it pop up - its clear structure plus wide tool collection help explain why. Building compact detection tools becomes doable once someone adds code for grabbing packets, studying flow patterns, matching known signs of trouble.

With tools like Scapy, capturing and altering packets becomes possible. At the same time, the re module handles searching for patterns linked to known threats. Real-time tracking of data across networks shows up when Python connects with sockets. Flexibility stands out in Python, letting analysts build unique methods that target certain online dangers - say, SQL injection or cross-site scripting. Custom logic fits neatly into these designs because the language adapts without forcing rigid structures.

Another thing. Python works well when building quick test versions of programs, also linking up easily with tools used in training smart algorithms - a path toward mixing traditional alert methods with smarter threat spotting. Even if network monitors built in Python fall short next to speed-focused software such as Snort, they still offer solid ground for trying new ideas, testing defenses, shaping personal approaches to digital safety.

5. Implementation Details

One way to start is by setting up Snort with specific rules that spot known threats. Following that path, a homemade detection tool takes shape using Python code. A diagram reveals how these pieces fit - one side tunes Snort, the other builds logic from scratch. Instead of relying only on tools, crafting new scripts adds control. While Snort watches traffic through predefined patterns, the Python part handles tasks in its own way. This blend splits effort but keeps goals aligned. Each segment runs parallel, yet connects at key points. Through such structure, both systems contribute without overlap.

Starting off, Snort gets set up with special rules meant to catch certain web threats - like SQL injection. A standard rule uses pieces such as what happens when matched (say, alert), the transfer method (often TCP), where it's coming from, where it's going, along with key data signs. Take one case: spotting an attack using something like ' OR 1=1 inside website requests. With that line placed into Snort’s setup file, live traffic begins getting scanned nonstop. Matches pop up instantly as warnings each time a pattern fits. How well this runs shows the power of checking inputs against known threat markers.

Later on, a small-scale NIDS gets built with Python, most often inside Jupyter Notebooks. Instead of relying on heavy tools, it uses Scapy to grab live packets off the wire. For spotting suspicious patterns, the re library steps in, handling text searches through regex rules. Once a packet flows in, its contents get pulled out and scanned line by line. If anything triggers a known signature, an alert shows up right away - plain text, no delay. Though basic in design, it adapts easily to different test scenarios. Mainly, people turn to it when learning or trying new detection ideas.

What stands out is how Snort alongside Python handles familiar web threats with clear results. Though Snort brings power and room to grow in live settings, scripting in Python opens doors to tweak and test freely. Side by side, these tools create space to learn while assembling detection methods based on signatures.

6. Conclusion and Future Work

This study introduced a way to spot typical online threats like SQL Injection, XSS, and path manipulation through a Signature-Based NIDS built with Snort and Python. Through comparing live data with preset patterns, results showed strong precision in recognizing familiar dangers alongside minimal incorrect alerts. Instead of complex setups, the solution leaned on Snort for instant analysis that works outside lab settings, yet kept adaptability via code written in Python. Still, since detection depends entirely on prewritten rules, new or unseen exploits easily slip past unnoticed.

One path ahead involves boosting the system with anomaly detection plus smart pattern recognition to catch new kinds of threats. A live tracking interface could take shape next, feeding fresh threat signatures without manual steps while tuning speed for fast-moving data flows. Running the setup within cloud platforms might happen later, layering different detection styles to handle growth and stress better. This kind of shift helps fit today's tangled, evolving networks more naturally. Built-in adaptability becomes key when facing unknown risks down the line.

REFERENCES

1. M. Schrötter et al., "A Comparison of Neural-Network-Based Intrusion Detection against Signature-Based Detection in IoT Networks," Information, vol. 15, no. 3, 2024.
2. M. Baklizi et al., "Web Attack Intrusion Detection System Using Machine Learning Techniques," International Journal of Online and Biomedical Engineering (iJOE), vol. 20, no. 3, 2024.
3. U. Ahmed et al., "Signature-Based Intrusion Detection Using Machine Learning and Deep Learning Approaches Empowered with Fuzzy Clustering," Scientific Reports, 2025.
4. L. Diana et al., "Overview on Intrusion Detection Systems for Computer Network Security," Computers (MDPI), vol. 14, no. 3, 2025.
5. Y. Reddy and G. ShankarLingam, "Artificial Intelligence in Intrusion Detection Systems: Trends and Future Directions," International Journal of Intelligent Systems, 2024.
6. G. Gebremariama et al., "Design of Advanced Intrusion Detection Systems Based on Hybrid Machine Learning Techniques," Journal of Information and Communication Technology, 2023.
7. A. Hozouri et al., "A Comprehensive Survey on Intrusion Detection Systems with Advances in Machine Learning and Cybersecurity Challenges," Springer Discover AI, 2025.
8. R. Singh et al., "Hybridized Bio-Inspired Intrusion Detection System for Internet of Things," Frontiers in Big Data, 2023.
9. A. Pinto et al., "Survey on Intrusion Detection Systems Based on Machine Learning Techniques for Critical Infrastructure," Sensors (MDPI), vol. 23, 2023.
10. "A Comprehensive Review of AI-Based Intrusion Detection Systems," Measurement: Sensors, 2023.