Mobile and Forensics
Vol. 7, No. 1, March 2025, pp. 55-68
ISSN: 2714-6685 (online), 2656-6257 (print)
DOI: http://dx.doi.org/10.12928/mf.v1i1.9365

A SECURITY DEVELOPMENT LIFE CYCLE (SDLC)-BASED APPROACH FOR DESIGNING INTRUSION DETECTION AND PREVENTION SYSTEMS TO COUNTER SQL INJECTION ATTACKS AT MAN 2 MAGETAN

[1,*]Muhammad Naufal Hafizh, [2]Nuril Anwar, [3]Ahmad Azhari
[1,2,3]Universitas Ahmad Dahlan, Yogyakarta, Indonesia
[1]muhammad1900018347@webmail.uad.ac.id, [2]nuril.anwar@tif.uad.ac.id, [3]ahmad.azhari@tif.uad.ac.id
*correspondence email

Abstract

Information security is a critical aspect of ensuring the validity, integrity, and availability of data while protecting users' access to services. Inadequate security measures can expose systems to various threats, potentially compromising their functionality. One such threat is SQL Injection, a common attack vector targeting web applications. MAN 2 Magetan, an Islamic high school located in Purwosari, Magetan Regency, East Java, Indonesia, operates an online admission system on its website. However, this website contains input fields that are not properly validated, creating a vulnerability to SQL Injection attacks. This study aims to design and implement an Intrusion Detection and Prevention System (IDPS) to mitigate SQL Injection attacks using the Security Development Life Cycle (SDLC) methodology. The SDLC process for the system development consists of five stages: Analysis, Design, Implementation, Enforcement, and Enhancement. A hybrid system combining Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) was utilized to create an effective solution. The results of the research demonstrate that the developed IDPS successfully detects and prevents SQL Injection attacks, ensuring the security and integrity of the online admission system. The integration of IDS and IPS within the SDLC framework has proven to be an effective approach to enhancing web application security at MAN 2 Magetan.

.

Keywords: Intrusion Detection System, Intrusion Prevention System, SQL Injection, Security Development Life Cycle, Web Application Security

INTRODUCTION

The progress of industry 4.0 requires all aspects to be more advanced and developed, especially communication and information technology. The development of communication and information technology makes information security very important[1], [2], especially in networks connected to the internet. In this day and age, computer network connections are no longer a factual matter, almost every service and media uses a computer network. When connected to the internet, the network we use is not completely secure, there are negative impacts on the development of the internet which cause new problems regarding information security threats[3], [4]. In poor security conditions, various threats can arise that have the potential to damage the system. Of the many threats is SQL-Injection.

SQL-injection is an attack that exploits gaps in the database by entering SQL commands so that the perpetrator can send the command to the database[5], [6]. SQL-Injection is a favorite technique for hacking because the way it works is quite simple, namely by misusing security gaps in the database[7]. In addition, the impact of this attack is quite large, there are many losses due to damage to the database[8]–[10].

Article History: *Received November 17, 2023; Revised March 19, 2025; Accepted March 20, 2025*

Injection was ranked first, but in 2021 it dropped to third place, although it dropped, injection is still on the list[11]. This clearly proves that this attack is easy to do and many systems are not proper in securing the system and handling this attack[12]–[15]. There are many ways to secure the system from these threats, including limiting input boxes and turning off error handling. MAN 2 Magetan is an educational unit with MA level in Purwosari, Magetan District, Magetan Regency, East Java. In carrying out its activities, MAN 2 MAGETAN is under the auspices of the Ministry of Religion. School. This school implements online new student admissions (PPDB) on their website. Their PPDB website has a number of unvalidated forms, making it vulnerable to sql-injection attacks on the website. One alternative to overcome this threat is to utilize a prevention system or what is called an Intrusion Prevention System (IPS) so that the threat can be overcome before it enters deeper into the system. Intrusion Prevention System (IPS) is a combined system between Intrusion Detection System (IDS) and IPTables. IDS is able to detect attacks based on the rules created while IPTables blocks attacks according to the command.

METHODS

The method used in this study is Experiment. The experimental method itself is a research method that aims to explain the causal relationship between one variable and another[16]. This research was conducted sequentially starting from the formulation of the problem, observation, literature study and system development. This research was conducted by implementing the Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) based on the website interface with the Security Development Life Cycle (SDLC) development model[17]–[20]. In the development of SDLC there are five stages, namely,

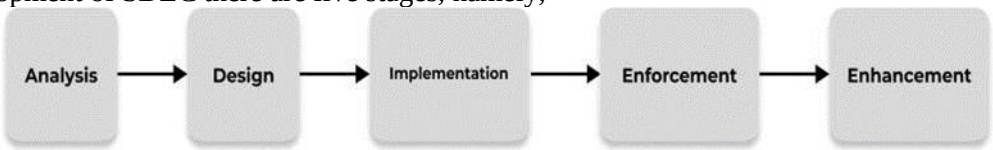

Fig 1. System Development Flow

Analysis

The analysis stage is useful for analyzing the specifications of the system to be built such as the hardware needed and the software needed to build IDPS.

Design

The design stage is to create a system design as a representation of a real system.

Impelementation

The implementation stage is to create a design on a design that is realized by installing and configuring the IDS system components, namely snort, barnyard2, and website monitoring as IPS.

Enforcement

The stage which is the implementation or implementation through operating activities and observing the system that has been built also checks whether IDPS is running properly and correctly and testing the system whether it can handle the threat of sql-injection properly.

Enhancement

This last stage is carried out repair activities on the system that has been built.

RESULT AND DISCUSSIONS

In overcoming the possibility of SQL-Inection attacks I created a detection and prevention system using the Security System Development Life Cycle (SSDLC) method. Snort as a detection engine, barnyard2 as a bridge between snort log and mysql and website monitoring used to monitor attacks and prevent attacks.

Analysis

The SDLC development model begins at the analysis stage. At this stage, security requirements are determined based on the needs of the system and its users. This involves understanding the

desired security features, identifying potential risks and threats, and determining the necessary security controls. This study was conducted using a VPS Ubuntu Server from IDCloudhost with the aim of not disrupting the process on the original website.

The PPDB Man 2 Magetan website is actively used in the new student registration process. After observing the website, it turned out that the website had a vulnerability in the admin login form. The admin login form was not validated so that when tested using single quotes ('), the website showed a mysql database error.

This information also became the basis for analyzing the software and hardware requirements needed in this study.

Design

At the design stage, the details of the specifications in the system become the system design built in this study. The author divides the system design into several stages as follows,

A. IDPS System Topology Design

This design aims to see how Snort works in dealing with an incoming attack on the website and is the basis for designing the system that will be built at the implementation stage.

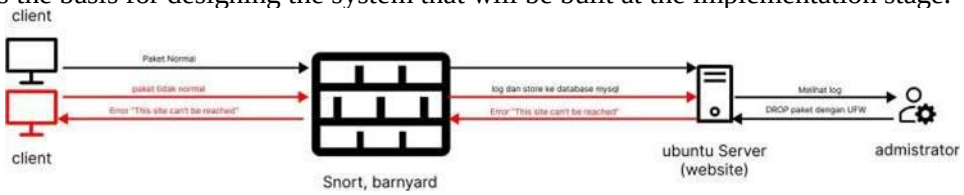

Fig 2. *IDPS System Topology*

Figure 2 shows the topology of the system that will be created in the implementation process, from the flow if the hacker accesses the website and performs a SQL-Injection attack then the package will be filtered first by snort, if the package matches the snort rules then it will be logged, the log from snort will be entered into the mysql database by barnyard2 then displayed to the monitoring website in a form that is easier for the user to understand. Conversely if the package does not match all snort rules then the package will be allowed to access the website.

B. IDPS System Design

The IDPS system design is useful as a collector of information from every attack recorded by snort. So that the system can work and function properly and run as expected,

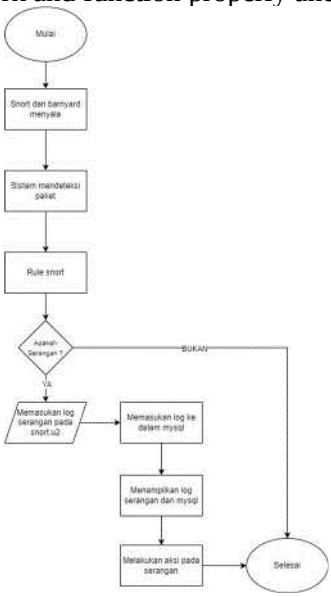

Fig 3. *System Workflow*

The system will be designed to be on a VPS server where the website is run, snort will monitor every client that accesses the website. Activities will be matched with existing rules to

determine whether the data is an attack or not. Data that is considered an attack will be stored in the snort log then processed by barnyard2 and entered into a mysql database which will then be displayed on the IDPS monitoring website, on the website the attack can be blocked or left alone.

C. IDPS Monitoring Website Design

The design of the monitoring website aims to facilitate the work on the monitoring website display and as a reference in the implementation stage. Here are some system designs that have been made,

Fig 4. *Login Page*

The login page will later function as user authentication to enter the dashboard of the Intrusion Prevention System and also as a security measure so that not just anyone can enter the dashboard to prevent unwanted things from happening.

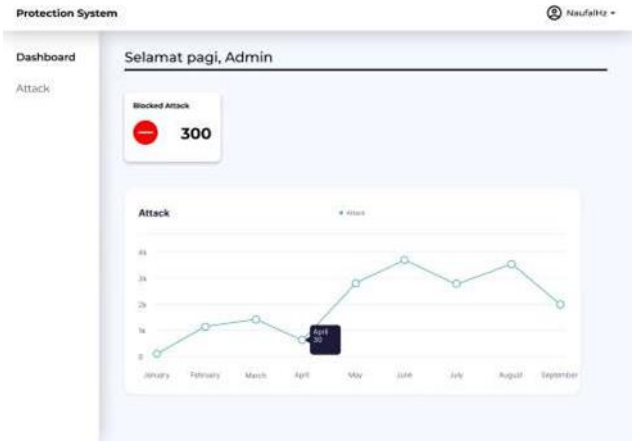

Fig 5. *Main Dashboard*

The login page will later function as user authentication to enter the dashboard of the Intrusion Prevention System and also as a security measure so that not just anyone can enter the dashboard to prevent unwanted things from happening.

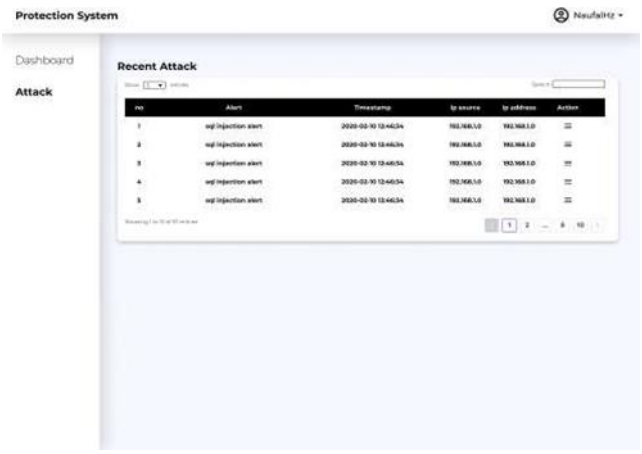

Fig 6. *Dashboard Attack*

Still on the dashboard but in a different menu. This attack dashboard is used to view all attack logs captured by the system then the administrator in carrying out prevention by blocking the IP address that is considered invalid.

Impelementation

The next phase is the implementation or application of the details of the system design in the real environment. The details of the design will be used as a guide or instruction so that the system built is relevant to the system that has been designed. The implementation stage consists of installation and configuration.

A. Supporting Component Installation

Installation of these supporting components is necessary to support the system that will be built based on the analysis and design that was made late.

Table 1. Supporting Components

Command	Function
sudo apt install apache2	Used to run the webserver
sudo apt-get install php7.4-cli php7.4-json php7.4-common php7.4-mysql php7.4-zip php7.4-gd php7.4-mbstring php7.4-curl php7.4-xml php7.4-bcmath	Installation of PHP along with the module version 7.4 to run the snort web monitoring
sudo apt install mysql-server	Installation of MYSQL as a database to accommodate
sudo apt install -y libpcap-dev libpcre3-dev libdumbnet-dev bison flex zlib1g-dev liblzma-dev openssl libssl-dev	every alert from snort
sudo apt-get install -y libmysqlclient-dev mysql-client autoconf libtool	Dependencies required for snort to run properly.

B. Snort Installation and Configuration

Snort used in this study is snort version 2.9.15.1 GRE (Build 15125). In the installation process, snort requires some initial configurations such as the network interface and address range on the system. On VPS, the network interface is ens3 and the IP address range used is 10.83.232.200/24.

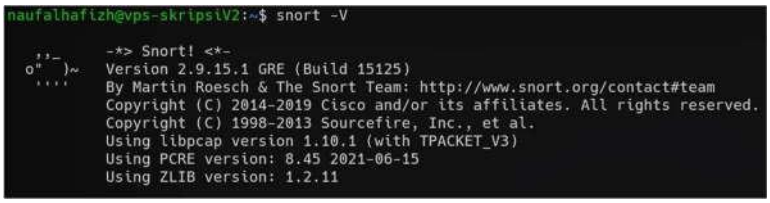

Fig 7. *Snort Version*

Figure 7 shows the snort version used in this study in detail. Starting from the version used, Library for Capabilities or a library that provides a programming interface (API) to manage capabilities in program code. Perl Compatible Regular Expressions (PCRE) which is used for processing and matching text patterns (regular expressions) in the C programming language and several other programming languages. ZLIB which is used for reducing data size for storing and transferring data over a network.

After installing snort, the next step is to configure snort. At this stage, several configurations are carried out such as adjusting the rules used and validating the configuration in snort.conf. The rules used in snort are local rules, other rules such as community rules and default snort rules that are not needed must be turned off first in snort.conf. In addition to configuring snort.conf, what needs to be done is to validate the snort configuration. This configuration validation aims to ensure that when snort is run, no errors occur.

C. Baryard2 Installation and Configuration

The installation process of barnyard2 is slightly different because it is not available in the Ubuntu repository. The process is by doing a git clone on the github repository, then doing a little configuration such as configuring mysql and mysql-libraries according to the architecture version on the VPS. The last is to compile so that barnyard2 is installed on the system.

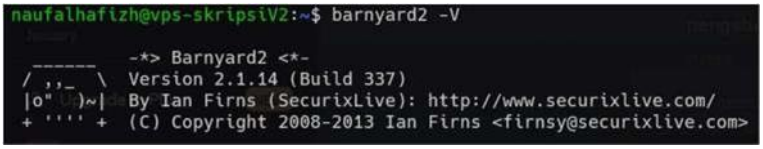

Fig 8. *Barnyard2 Version*

Figure 8 shows the version of barnyard used and the copyright of barnyard2. In addition to showing the version and copyright, this indicates that barnyard2 has been successfully installed on the system and is ready to be configured.

The next process is to configure barnyard so that it can read alerts on snort and store on the mysql database. Before being able to configure, the barnyard2.conf file in the installation directory must be moved to the snort directory so that it is easy and does not cause errors. The next process creates a log directory from baryard2 at "/var/log/barnyard2" and creates a barnyard.waldo file in the "/var/log/snort/barnyard2.waldo" directory, this barnyard.waldo file is used to help Barnyard2 to continue processing the log from the last point if the processing stops or fails, in short the barnyard.waldo file prevents the repetition of processing the same log file every time the process is restarted. Before entering the configuration file, first create a database by importing the barnyard2 database schema into the database that was created.

Fig 9. *Barnyard2 Configuration*

As seen in Figure 9 above is the core configuration of barnyard2, the configuration shows several files such as references, classifications, genes and sids of all attack alerts captured by snort. The configuration by default should not be changed unless there are files placed in different directories.

Fig 10. *Barnyard2 Database Configuration*

In Figure 10, namely configuring barnyard2 and the mysql database to communicate with each other so that the alerts captured by snort can then be entered into the mysql database by barnyard2. Barnyard2 also supports other databases besides mysql such as postgresql, odbc, mssql and oracle, making it easier to implement on other systems.

D. Website Monitoring Creation

The monitoring website is made so that users or administrators can read the attack logs from snort more easily. This website is made using the PHP codeigniter 3 framework and the dashboard template from sneat-1.0.0. Using codeigniter 3 because the website administrator also uses the same framework, this aims to make it easier for writers and website managers to perform maintenance on the website. The website and snort databases are also distinguished with the aim of distinguishing between the PPDB and snort databases.

The creation of the website begins by modifying the dashboard template from sneat by distinguishing between the header, menu and footer then configuring the database, creating a controller as the brain of the website and creating a model as a query manager from the database.

The configuration of the codeigniter 3 database is done in the application/config/database.php file by implementing a multiple database connection system, one as the original database from the website and the second database used by snort to create an interface or display based on the existing design process. At this stage, using the sneat-1.0.0 dashboard template by breaking down several parts into headers, menus and footers. The author also creates a display of the main dashboard and alert/attack, later this alert/attack display functions to see attacks that enter the system and see details of the attack. The last stage is to create a controller as the brain of the dashboard. This controller functions as an intermediary between the model and the view, the controller receives input from the user through the View, processes the input, and instructs the Model or View according to the actions given. The controller in codeigniter 3 is created at /application/controllers/PreventionSystem.php.

Enforcement

The next stage is Enforcement or implementation or implementation, at this stage all components that have been created in the implementation process are run and tested so that it can be assessed whether the component can prevent attacks. Components such as snort and barnyard2 are tested using basic rules to see if they can work properly, if the component works properly then it will be tested using SQL-Injection attacks. Below are some of the testing activities,

A. Snort Sensor Functionality Testing

Snort testing ensures that snort is working properly in detecting threats accurately. This test uses a simple local.rule to detect ICMP message requests or pings from Windows 11.

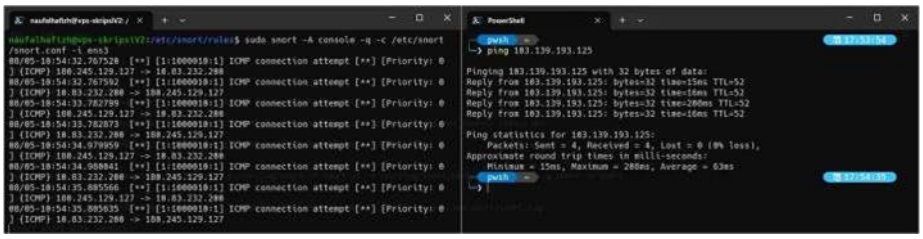

Fig 11. *Snort Testing*

As seen in Figure 11, snort successfully detected pings from Windows 11 devices. It can also be seen that snort also records the date and time of the attack, sid, message, priority, layer protocol, source address and destination address. All of these alerts will be stored in unified2 format in /var/log/snort with the name snort.u2. This format will later be used by barnyard2 and translated into a MySQL database.

B. Barnyard2 Functionality Testing

Testing on barnyard2 aims to ensure that all implementation and configuration processes run well so that the snort log can be entered into the database. The barnyard2 testing process itself is carried out by running snort and barnyard2 simultaneously, this is done because when snort detects an attack, barnyard2 can immediately read the log from snort and then enter it into the mysql database and will be queried on web monitoring in real time.

Fig 12. Barnyard2 Testing

Figure 12 shows that barnyard2 has successfully entered data from the snort log into the mysql database. The data is stored in the event table as master data. Data such as the alert name will be stored in the signature table, the IP address will be stored in the iphdr table and the protocol will be stored in the tcphdr, icmphdr and udphdr tables.

C. Website Monitoring Functionality Testing

Website monitoring testing aims to test whether all functions are running as expected. Testing is done by testing each function on each page starting from login then the main page to the alert page.

Testing on website monitoring uses blackbox testing, namely software quality testing that focuses on software functionality. This test aims to find functions that are not running properly.

D. System Testing in Handling SQL-Injection Attacks

The final test is to test all components with different rules from all previous tests. The rules applied are SQL-Injection prevention rules and snort will run side by side with UFW. UFW acts as a prevention or IPS later snort will not only function as an intrusion detection system (IDS) that only provides warnings or reports about suspicious activity, by running with UFW it can become an intrusion prevention system (Intrusion Prevention System/IPS) that has the ability to take direct action to stop traffic identified as an attack or security threat.

Before carrying out the slq-injection attack scenario on the system, the author adjusts the rules to detect several sql-injection threats. In table 2 below are the sql injection rules or signatures

Table 2. Rules for Sql-Injection

Rule	Explanation
alert tcp any any -> any 80 (msg: "Error Based SQL Injection Detected"; content: "%27"; classtype:web-application-attack; sid:100000111; rev:1;)	To detect input ' on forms and urls on TCP protocol and port 80
alert tcp any any -> any 80 (msg: "Error Based SQL Injection Detected"; content: "22"; classtype:web- application-attack; sid:100000112; rev:1;)	To detect input on forms and urls on TCP protocol and port 80
alert tcp any any -> any 80 (msg: "AND SQL Injection Detected"; content: "AND"; nocase; classtype:web- application-attack; sid:100000113; rev:1;)	To detect input and, both uppercase and lowercase on forms and urls on TCP protocol and port 80

In this test scenario the author turns on snort and barnyard2 simultaneously, this is done so that when snort detects an attack immediately barnyard2 reads the log from snort and enters it into the mysql database and can then be viewed on the monitoring website. Snort and barnyard2 are activated simultaneously for approximately 20 minutes and perform various sql-injection attack scenarios.

```
Packet I/O Totals:
   Received:        11708
   Analyzed:        11698 ( 99.915%)
    Dropped:            0 (  0.000%)
   Filtered:            0 (  0.000%)
Outstanding:           10 (  0.085%)
   Injected:            0
```

Fig 13. *Snort Received Packets*

The results obtained after running the test scenario for 20 minutes can be seen in Figure 13. It can be seen that snort received 11708 packets and snort analyzed 99.915% of the packets received. There are 10 packets around 0.085% which shows how many packets are buffered waiting for processing. Packet buffering is used to overcome problems such as network latency, data transmission speed mismatch, or differences in data production and consumption rates.

```
Breakdown by protocol (includes rebuilt packets):
      Eth:        11720 (100.000%)
     VLAN:            0 (  0.000%)
      IP4:        11405 ( 97.312%)
     Frag:            0 (  0.000%)
     ICMP:          232 (  1.980%)
      UDP:          618 (  5.273%)
      TCP:         9954 ( 84.932%)
```

Fig 14. *Protocol Received Packets*

In Figure 14, we can see the details of the packet analysis performed by snort, where snort records threats on the network. It can be seen that snort records the ICMP protocol as many as 232 packets, UDP as many as 618 packets and 9954 packets on TCP with a percentage of 84.932%. Sql-injection is not directly related to the TCP network protocol, but sql-injection can occur in applications that use transport protocol types such as TCP, this is why the TCP protocol receives high packets.

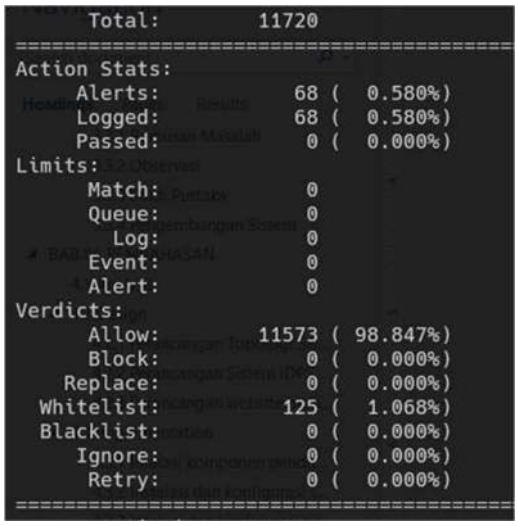

Fig 15. *Total Log Snort*

Figure 15 shows the total packets received by snort are 11720 and also snort sends alerts and logs 68 times with a percentage of 0.580%. This means that snort runs well in recognizing the signature rule given and can filter and analyze data packets. 11573 packets with a percentage of 98.847% were captured by snort and allowed to pass, this means that the packets passed the validation of the snort rule and were considered not a threat. A package of 125 with a percentage of 1,068% is on the whitelist, a package that is included in the whitelist has 2 meanings, namely the package can be implicitly trusted so that the package can be directly permitted by snort without going through the checking process or the package is an attacker so that snort immediately gives a warning, but this does not affect the course of the rule check on snort, snort only informs whether there is a trusted package or not, but if the package matches the snort rule, an alert will still be given and entered into the database.

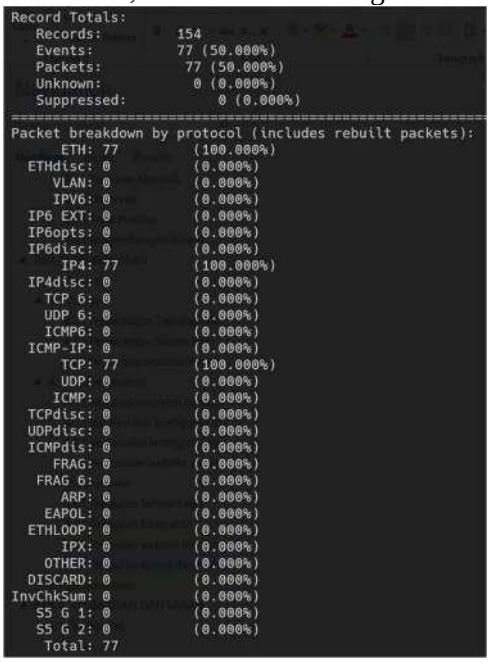

Fig 16. *Record Barnyard2*

Figure 16 shows that barnyard2 also successfully analyzed the snort.u2 log of 77 records, this is different from what snort captured because barnyard2 uses waldo to see when the last

record of the snort log was. However, if you calculate the data entered by barnyard2 into mysql, it will match the snort log, which is 68 data. All records from barnyard2 show TCP has a percentage of 100,000%, this matches the snort rule which only detects TCP.

The monitoring website can also display data well, namely 282 data, this includes snort trial data before the sql-injection rule is applied, where 68 data are the latest data from snort and barnyard2. The monitoring website can display all sql-injection signatures along with time, destination IP and source IP. If the drop button is pressed, the website will automatically be inaccessible to the attacker. The attacker's IP that has been dropped will not be accessible again and will display the message This site can't be reached 103.139.193.125 took too long to respond.

Table 3 Sql-Injection Attacks Detected by the System

Sql-Injection Attack Type	Number of Attacks	Percentage
Error Based SQL Injection	24	35.29%
AND SQL Injection	2	2.94%
OR SQL Injection	14	20.59%
Form Based SQL Injection	23	33.82%
ORDER BY SQL Injection	3	4.41%
UNION SELECT SQL Injection	2	2.94%
Total	68	

In Table 3, 68 sql-injection attacks were successfully detected by the system. From the table above, it can be seen that the system can detect sql-injection attack patterns based on the signature rule that has been created. The attack was carried out on the admin login form which has a sql-injection vulnerability. Error base sql-injection has the highest number with a percentage of 35.29% this occurs because each sql-injection attack uses single quote notation (‘) and quotation marks (“) such as 'or' 1'='1, The pattern matches snort and is then considered an attack by the system as Error Based SQL Injection and OR SQL Injection, if done on a form it will produce a Form Based SQL Injection alert. In general, the system that was built managed to recognize sql-injection well even though some attacks such as AND SQL Injection and UNION SELECT SQL Injection have a low percentage. AND SQL Injection and UNION SELECT SQL Injection are rarely used in form-based attacks, both attacks.

In long-term handling it would be better if the form that has the SQL-Injection bug is fixed by adding validation to the form. This prevention is a repressive action to reduce SQL-Injection attacks. If analogized, when the path (bug) is closed, the hacker will no longer have access to enter so that the attack will be reduced drastically.

With the completion of this testing phase, the system is expected to be able to detect and prevent SQL-injection attacks, it is expected that with the system that has been developed, the PPDB Man 2 Magetan website can be protected from SQL-Injection attacks. Of course, there are changes before and after the IDPS system is installed, here are the changes before and after after the IDPS is installed,

Before Using IDPS

1. Websites are vulnerable to various types of attacks such as SQL-Injection hacking
2. There is no complete visibility into network activity, making it difficult to detect threats quickly.
3. Responding to attacks must be done manually by entering commands directly into the shell, which can be time-consuming and increases the risk of further damage.

After Using IDPS

1. IDPS enables early detection of attacks by monitoring network traffic and identifying suspicious behavior.
2. IDPS systems not only detect attacks, but can also prevent attacks by stopping malicious traffic.

3. IDPS systems provide in-depth reports and analysis of detected attacks, helping security teams respond more efficiently.
4. Using IDPS effectively can improve the security level of a website and reduce the risk of a successful attack.

Enhancement

The final phase in the development of the Security System Development Life Cycle (SSDLC) model is enhancement. Development of the system certainly does not produce a perfect system when it is first created, it takes a lot of trials to produce a system that is truly proper. The final phase of this system development aims to improve and enhance the system that has been built to make it even more proper.

This phase includes improving the system that has been built. The enhancement phase goes through a series of improvement processes for a number of purposes as follows,

1. Fixing some errors that occur in the system implementation.
2. Adding functionality to specific components or new features to complement the shortcomings of the previous system.
3. Adapting previously built systems with newer technologies to address a number of new problem developments.

The system that was built has gone through a long development process to produce a proper system, in this development it is not far from perfect. This stage is carried out to improve and ensure the system works well as expected. This phase includes improvements at the Implementation and Enforcement stages.

CONCLUSIONS

The implemented system was successfully built and developed well using the Security System Development Life Cycle (SSDLC) method through 5 stages, namely Analysis, Design, Implementation, Enforment and Enhancement. The entire system, from the IDS sensor to the website, works as it should, where the system can detect and prevent SQL-injection attacks effectively. The working mechanism between snort, barnyard2 and website monitoring was successfully implemented well. Snort can detect the network, barnyard2 can process logs from snort and enter them into the MySQL database well, website monitoring works well in preventing attacks and displaying logs from snort. Prevention can be done by utilizing the Uncomplicated Firewall (UFW) which is a more concise version of iptable and uses the shell_exec() function from the PHP programming language to execute commands from UFW. The result is that incoming attacks can be handled well.

REFERENCES

[1] M. Kiss, G. Breda, and L. Muha, 'Information security aspects of Industry 4.0', *Procedia Manuf.*, vol. 32, pp. 848–855, 2019, doi: 10.1016/j.promfg.2019.02.293.

[2] H. Stewart and J. Jürjens, 'Information security management and the human aspect in organizations', *Inf. Comput. Secur.*, vol. 25, no. 5, pp. 494–534, Nov. 2017, doi: 10.1108/ICS-07-2016-0054.

[3] N. Miloslavskaya and A. Tolstoy, 'Internet of Things: information security challenges and solutions', *Cluster Comput.*, vol. 22, no. 1, pp. 103–119, Mar. 2019, doi: 10.1007/s10586-018-2823-6.

[4] T. Pereira, L. Barreto, and A. Amaral, 'Network and information security challenges within Industry 4.0 paradigm', *Procedia Manuf.*, vol. 13, pp. 1253–1260, 2017, doi: 10.1016/j.promfg.2017.09.047.

[5] K. Ahmad and M. Karim, 'A Method to Prevent SQL Injection Attack using an Improved Parameterized Stored Procedure', *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no.

6, 2021, doi: 10.14569/IJACSA.2021.0120636.
[6] A. Paul, V. Sharma, and O. Olukoya, 'SQL injection attack: Detection, prioritization & prevention', *J. Inf. Secur. Appl.*, vol. 85, p. 103871, Sep. 2024, doi: 10.1016/j.jisa.2024.103871.
[7] A. D. Yudistiawan and N. Anwar, 'Website Application Security Value Analysis Using Crawling Method Against SQL Injection Attacks', *Mob. Forensics*, vol. 6, no. 1, pp. 39–50, Aug. 2024, doi: 10.12928/mf.v6i1.8198.
[8] P. Liu and M. Yu, 'Damage assessment and repair in attack resilient distributed database systems', *Comput. Stand. Interfaces*, vol. 33, no. 1, pp. 96–107, Jan. 2011, doi: 10.1016/j.csi.2010.03.009.
[9] F. Cohen, 'Simulating cyber attacks, defences, and consequences', *Comput. Secur.*, vol. 18, no. 6, pp. 479–518, Jan. 1999, doi: 10.1016/S0167-4048(99)80115-1.
[10] S. Chen, Z. He, C. Sun, J. Yang, and X. Huang, 'Universal Adversarial Attack on Attention and the Resulting Dataset DAmageNet', *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1–1, 2020, doi: 10.1109/TPAMI.2020.3033291.
[11] A. G. Kakisim, 'A deep learning approach based on multi-view consensus for SQL injection detection', *Int. J. Inf. Secur.*, vol. 23, no. 2, pp. 1541–1556, Apr. 2024, doi: 10.1007/s10207-023-00791-y.
[12] A. Humayed, J. Lin, F. Li, and B. Luo, 'Cyber-Physical Systems Security—A Survey', *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1802–1831, Dec. 2017, doi: 10.1109/JIOT.2017.2703172.
[13] I. Corona, G. Giacinto, and F. Roli, 'Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues', *Inf. Sci. (Ny).*, vol. 239, pp. 201–225, Aug. 2013, doi: 10.1016/j.ins.2013.03.022.
[14] H. Fawzi, P. Tabuada, and S. Diggavi, 'Secure Estimation and Control for Cyber-Physical Systems Under Adversarial Attacks', *IEEE Trans. Automat. Contr.*, vol. 59, no. 6, pp. 1454–1467, Jun. 2014, doi: 10.1109/TAC.2014.2303233.
[15] N. Dayal, P. Maity, S. Srivastava, and R. Khondoker, 'Research Trends in Security and DDoS in SDN', *Secur. Commun. Networks*, vol. 9, no. 18, pp. 6386–6411, Dec. 2016, doi: 10.1002/sec.1759.
[16] S. M. Ross and G. R. Morrison, *Experimental research methods Handbook of research on educational communications and technology*. Britania Raya: Routledge, 2013.
[17] N. Agarwal and S. Z. Hussain, 'A Closer Look at Intrusion Detection System for Web Applications', *Secur. Commun. Networks*, vol. 2018, pp. 1–27, Aug. 2018, doi: 10.1155/2018/9601357.
[18] R. A. Khan, S. U. Khan, H. U. Khan, and M. Ilyas, 'Systematic Literature Review on Security Risks and its Practices in Secure Software Development', *IEEE Access*, vol. 10, pp. 5456–5481, 2022, doi: 10.1109/ACCESS.2022.3140181.
[19] Safana Hyder Abbas, Wedad Abdul Khuder Naser, and Amal Abbas Kadhim, 'Subject review: Intrusion Detection System (IDS) and Intrusion Prevention System (IPS)', *Glob. J. Eng. Technol. Adv.*, vol. 14, no. 2, pp. 155–158, Feb. 2023, doi: 10.30574/gjeta.2023.14.2.0031.
[20] T. Firdyanto and R. Rushendra, 'Implementation of Intrusion Detection

System with Rule-Based Method on Website', *J. Comput. Networks, Archit. High Perform. Comput.*, vol. 6, no. 3 SE-, pp. 1245–1252, Jul. 2024, doi: 10.47709/cnahpc.v6i3.4256.

AUTHORS BIBLIOGRAPHY

Muhammad Naufal Hafizh is a bachelor's degree student in the Informatics Engineering Department of Ahmad Dahlan University. His research interests are centered on Cyber Security.
Email : muhammad1900018347@webmail.uad.ac.id

Nuril Anwar is a lecturer at the Department of Informatics Engineering, Ahmad Dahlan University. His research interests are centered on Computer Networks & Security, Digital Forensics.
Email : nuril.anwar@tif.uad.ac.id

Ahmad Azhari is a Assistant Professorat the Departmentof Informatics, Faculty of Industrial Technology, Universitas Ahmad Dahlan, Indonesia. He completed his undergraduate degree in Informatics at Universitas Islam Indonesia. He then obtained a Master's degree in Electrical Engineering and Information Technology from Universitas Gajah Mada, Indonesia. His research focuses on Pattern Recognition and Machine Learning.
Email : ahmad.azhari@tif.uad.ac.id