

University of Science and Technology of Hanoi



**ADMINISTRATION OF COMPUTER SYSTEM
REPORT**

**INTRUSION DETECTION AND PREVENTION SYSTEM FOR
BRUTE-FORCE ATTACKS ON SSH AND WORDPRESS USING SNORT**

Submitted by

Tran Duc Huy	BA12-085
Ha Dinh Tuan	BA12-183
Tran Minh Phuong	22BI13366
Nguyen Minh Duc	22BI13091
Tran Trung Hieu	22BI13162

Cyber Security (CS)

Lectures: MS. Le Nhu Chu Hiep

Hanoi, November 2024

TABLE OF CONTENT

I. INTRODUCTION.....	3
II. OVERVIEW.....	4
a. Target.....	4
b. Difficulty.....	4
c. Techniques needed.....	4
d. Result.....	4
III. DESIGN.....	5
a. Purpose.....	5
b. Structure.....	5
1. Paperwork on whole system:.....	5
2. User-case diagram:.....	5
3. Architecture diagram:.....	6
4. Sequence diagram:.....	7
5. Database diagram:.....	7
6. Evaluation metrics:.....	8
IV. IMPLEMENTATION.....	9
a. Detail tools used for each component.....	9
b. Techniques and protocol for system component interactions.....	9
c. Configuration parameters and explanation.....	9
d. Observation techniques for evaluation metrics.....	10
e. Evaluation strategies.....	10
V. RESULTS.....	11
a. Proof of work.....	11
b. Table for quantitative evaluation results.....	13
c. Discussion.....	13
VI. CONCLUSION.....	14
VII. REFERENCE.....	15

I. INTRODUCTION

In the realm of cybersecurity, brute-force attacks are among the most common and persistent threats faced by modern systems. These attacks involve systematically attempting every possible combination of usernames and passwords to gain unauthorized access. Brute-force attacks can target multiple services, with SSH (Secure Shell) and WordPress login pages being particularly vulnerable due to their widespread use and potential for exploitation. Successfully breaching these services can lead to unauthorized control of systems, data theft, and reputational damage.

The objective of this project is to simulate two types of brute-force attacks—one targeting SSH services and the other targeting WordPress login pages—and to deploy an Intrusion Detection and Prevention System (IDPS) capable of detecting and mitigating these attacks. Such a system is essential for maintaining the security of modern digital environments, as it not only detects malicious attempts in real time but also takes preventive measures to block harmful requests and alerts administrators promptly.

The system to be deployed comprises the following components:

1. **Attacker Machine:** This machine will simulate brute-force attacks using tools such as Hydra or WPScan, targeting SSH services and WordPress login pages.
2. **Victim Machine:** This machine will host a WordPress web server and an active SSH service to emulate a real-world environment susceptible to brute-force attacks.
3. **IDPS on the Victim Machine:** This system, implemented using **Snort**, will monitor network traffic, identify attack patterns, and take proactive measures such as blocking suspicious IP addresses or terminating malicious connections.

The IDPS plays a crucial role in system security by addressing the following key functions:

- **Detection:** Identifying abnormal patterns in network traffic indicative of brute-force attacks.
- **Prevention:** Automatically blocking malicious IPs or activities to prevent unauthorized access.
- **Alerting:** Notifying system administrators in real time to enable rapid response to potential threats.

Through this study, we aim to demonstrate the efficacy of an IDPS in detecting and preventing brute-force attacks, thereby contributing to a more secure and resilient system infrastructure. The subsequent sections of this report will provide a detailed analysis of the problem, methodology for system deployment, and an evaluation of its effectiveness in protecting against brute-force attacks.

II. OVERVIEW

a. Target

The primary goal of this project is to simulate and mitigate brute-force attacks targeting two commonly exploited services: SSH (Secure Shell) and WordPress login. These attacks are widely used by malicious actors to gain unauthorized access to systems by repeatedly attempting various combinations of usernames and passwords. The project will also involve the design and implementation of an Intrusion Detection and Prevention System (IDPS) to detect and block such attacks, thereby enhancing the security of the target systems.

- **SSH brute-force attack:** The SSH service is a common target for brute-force attacks, as attackers try to guess the correct credentials using various tools to gain remote access to a server.
- **WordPress brute-force attack:** WordPress is one of the most popular content management systems, and its login form is often targeted by brute-force attackers attempting to gain administrative access to websites.

b. Difficulty

- **Configuring brute-force tools:** Setting up and fine-tuning tools like Hydra for SSH attacks and WPScan for WordPress login brute-forcing can be complex, especially when ensuring they mimic real-world attack scenarios.
- **IDPS configuration:** Configuring an IDPS (snort) to accurately detect and respond to brute-force attempts requires detailed knowledge of attack patterns and proper tuning of detection rules.
- **Resource constraints:** Simulating a large number of attacks may lead to high resource consumption, which could affect the performance of the attack and detection systems.
- **Networking and connectivity:** Ensuring that the attack and victim machines communicate effectively while maintaining accurate detection of attack attempts is crucial. Misconfiguration in network settings may hinder the experiment.

c. Techniques needed

- **Brute-force Tools:** Hydra will be used for performing brute-force attacks on SSH and WP login. These tools allow automation of login attempts and can be configured to launch attacks with customized parameters.
- **IDPS Tools: Snort or Suricata** will be deployed on the victim machine to monitor and analyze incoming traffic. These tools will be configured to detect brute-force attack patterns, such as repeated failed login attempts.

d. Result

- **Detection and alerting:** The IDPS should be able to detect brute-force attempts in real-time and alert the system administrator.
- **Blocking malicious requests:** The IDPS should prevent further attack attempts once a threshold of failed login attempts is reached, effectively mitigating the brute-force attack.
- **Performance metrics:** The project will measure and evaluate the following:
 - The number of blocked SSH and WordPress login attempts.
 - The time taken for the IDPS to detect and block attacks.
 - The overall effectiveness of the IDPS in preventing unauthorized access.

III. DESIGN

a. Purpose

In this part, we will describe how our system was designed to implement intrusion and how we deploy the IDPS system to prevent this attack.

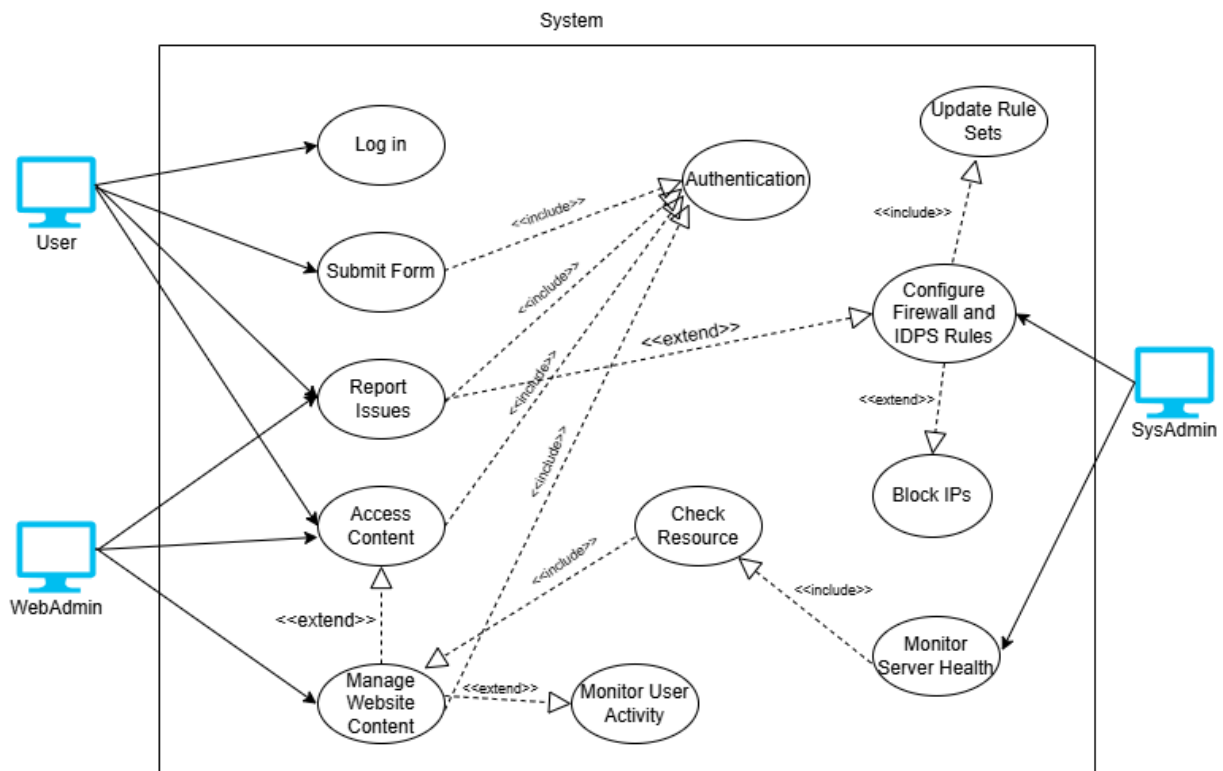
b. Structure

1. Paperwork on whole system:

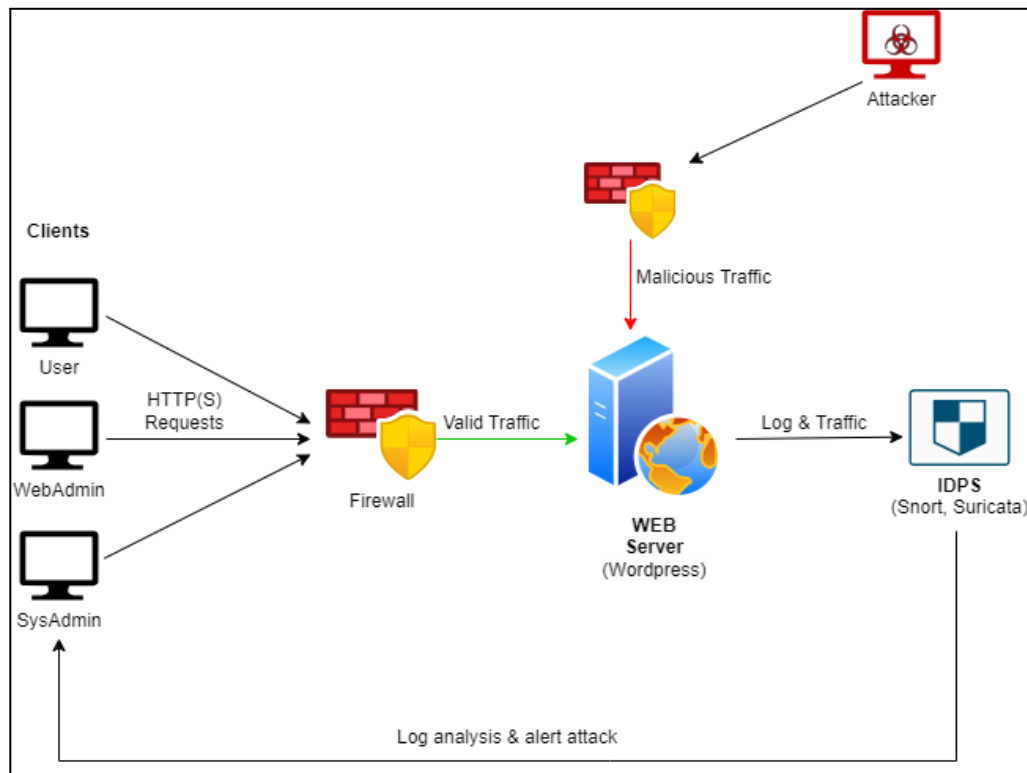
- The victim machine will host WordPress and SSH. Configure the victim machine to meet security requirements (e.g, do not use SSH keys, only passwords).
- The attacker machine will use tools such as Hydra for brute-forcing SSH and WPScan or Burp Suite for brute-forcing WordPress.
- **IDPS:** Install Snort or Suricata on the victim machine to monitor and detect suspicious activities.

2. User-case diagram:

- Describes user actions: Legitimate users can log into WordPress or SSH, while attackers attempt brute-force attack to gain unauthorized access.
- The diagram contains 3 actors: User, WebAdmin, SysAdmin.
- Each actor has various use-cases that represent what the actor can do in the system.
- Use-cases in the system have relationships with each other. For example, Configure Firewall and IDPS Rules can extends to Blocks IPs (if necessary)



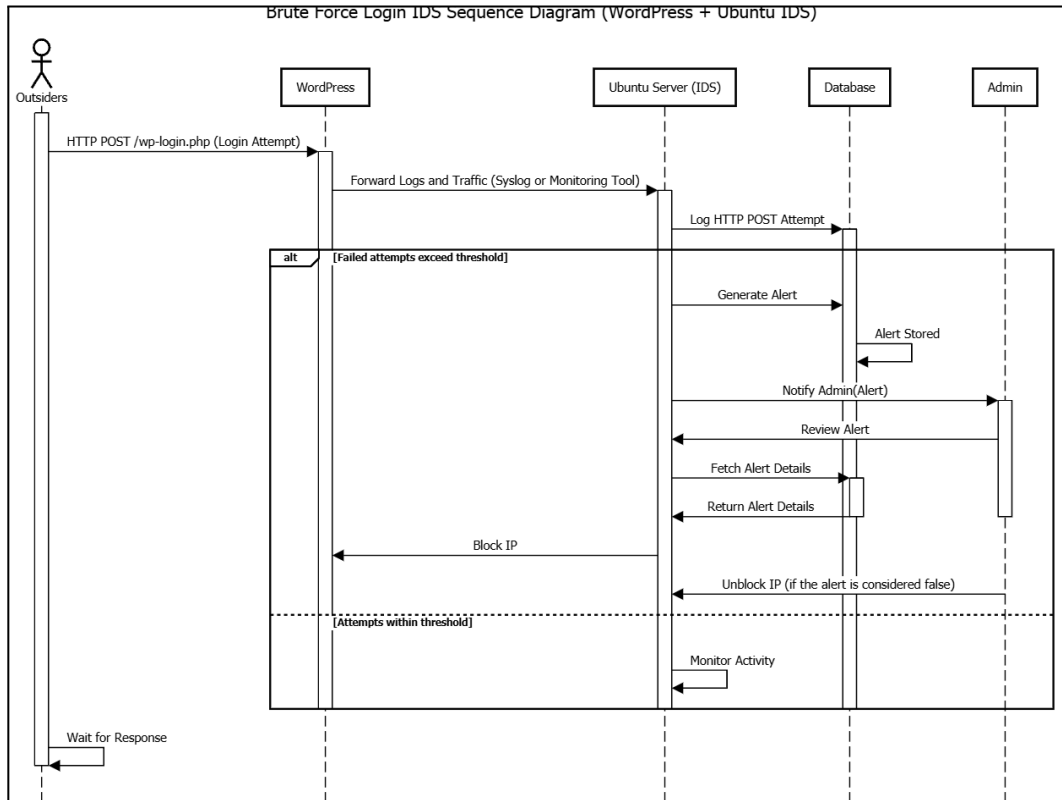
3. Architecture diagram:



- The diagram illustrates the system components such as the attacker machine, victim machine (WordPress + SSH), and the IDPS system.
 - Web Server: The main component of the architecture, this interacts with clients which is either legit or malicious
 - Firewall: The firewall of the wordpress server allows legit users to access the wordpress server, and blocks malicious traffic coming from outside attackers which in this case, brute force attempts.
 - IDPS: The IDPS receives traffic coming to the wordpress server and checks if the source of the traffic is legitimate. In the login part, if there are too many failed login attempts, the IP will be blocked and it cannot access anymore.
 - The system admin is the only type of user who can review the alert coming from IDPS, and unblock any source which is falsely considered by the security system.

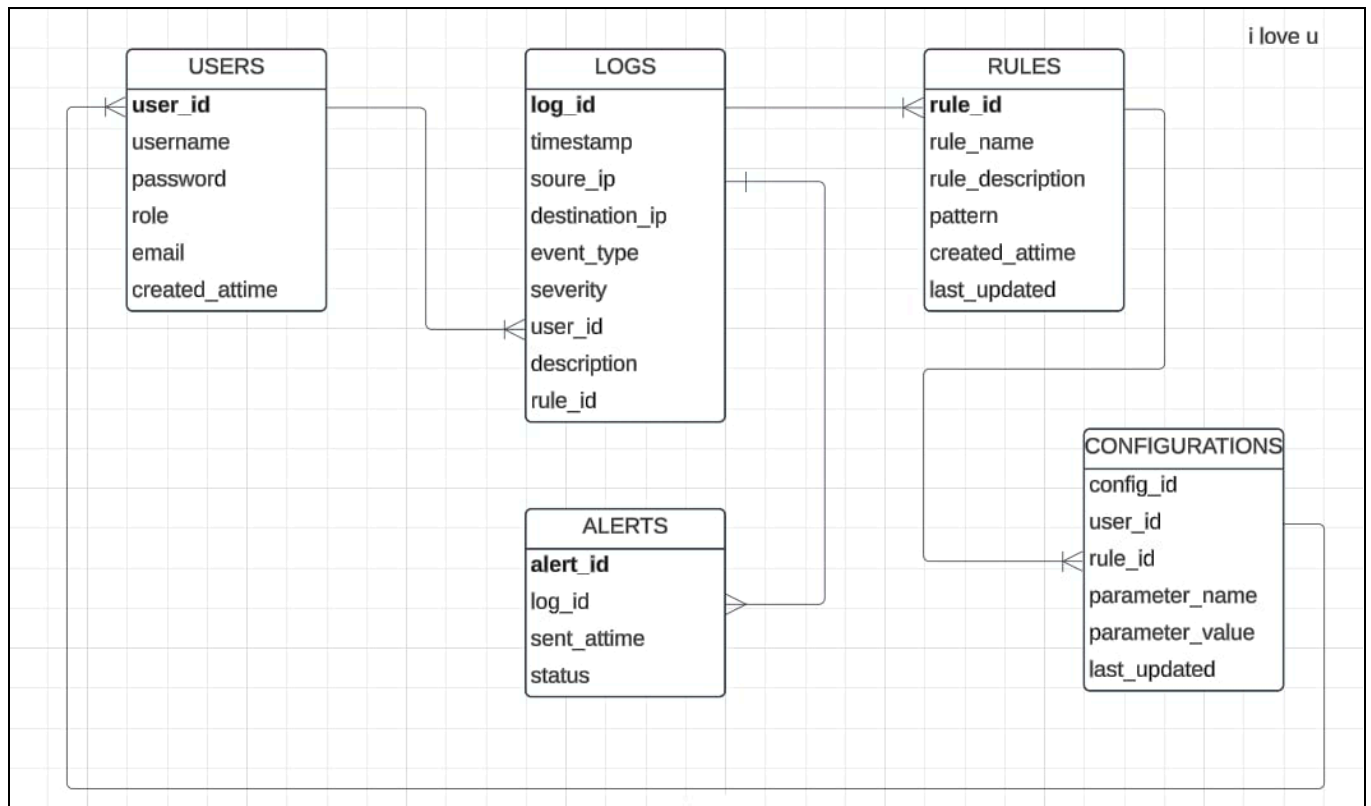
4. Sequence diagram:

- The sequence diagram illustrates the workflow of a wordpress server login system which ensures legit login attempts and denies brute force login attacks. Some key components of this diagram are:
 - **User:** log in into the system and wait for the response
 - **WordPress:** The deployed wordpress page which shows and manages the content
 - **Ubuntu server:** deploy wordpress page and get log and traffic coming from wordpress page. It blocks any IP which is considered as a brute force attack source.
 - **Database:** store user configuration, logs and alerts forwarded from Ubuntu server
 - **Admin:** This is the system admin which reconsider any ip which is blocked falsely by Ubuntu server



5. Database diagram:

- Illustrate the structure of the WordPress database and how login attempts are logged.
- Some table of database diagram:
 - **USERS** Manages user information, including login credentials, roles, and contact details.
 - **LOGS** Stores detailed information about system events or activities.
 - **RULES** Manages detection rules for identifying specific events or patterns in the system.
 - **ALERTS** Stores information about alerts triggered by suspicious logs.
 - **CONFIGURATIONS** Stores configurations related to users and detection rules.
 - Relationships between these table
- **USERS ↔ LOGS:** A user can generate multiple logs.
- **LOGS ↔ RULES:** A log is associated with a single rule.
- **LOGS ↔ ALERTS:** A log can trigger multiple alerts.
- **RULES ↔ CONFIGURATIONS:** A rule can have multiple configurations.
- **USERS ↔ CONFIGURATIONS:** A user can have multiple configurations.



6. Evaluation metrics:

Metric	Definition	Purpose
Number of Invalid SSH Connections	The number of SSH connections rejected due to incorrect passwords or unauthorized login attempts	Measure SSH brute-force attacks
Number of Invalid WordPress Login Attempts	The number of failed login attempts on the WordPress login page (incorrect username/password)	Measure WordPress brute-force attacks
Time to Detect Attack	The time from the start of an attack to when Snort detects and logs the alert	Measure the detection speed of the system
Effectiveness of IDS/IPS in Blocking Attacks	The percentage of attack connections that Snort successfully blocks after detection	Evaluate the blocking ability of the IPS system

IV. IMPLEMENTATION

a. Detail tools used for each component

Web Server (Victim)

- **WordPress:** A WordPress web server is used as the primary target for brute-force login attacks through HTTP.
- **SSH:** The SSH service is enabled for remote login, allowing attackers to attempt password-based logins.

Client

- **User, Web Admin, Sysadmin:** Different roles are defined in the victim system, with each role having different levels of access (user for browsing, admin for backend management, sysadmin for SSH access).

Attacker

- **Hydra:** A popular tool for performing brute-force attacks on SSH services.
- **WPScan:** A tool used for performing brute-force attacks specifically on WordPress login pages.

IDPS (Intrusion detection and prevention system)

- **Snort:** Network-based intrusion detection/prevention systems used to monitor and analyze incoming traffic for malicious patterns, such as brute-force login attempts.

b. Techniques and protocol for system component interactions

SSH Brute-Force attack

- **Protocol:** SSH (Port 22)
- **Method:** The attacker uses Hydra to send multiple login attempts with different usernames and passwords. Each attempt is processed by the SSH server, which either accepts or rejects the credentials. The IDPS monitors SSH traffic for patterns that indicate brute-force activity (e.g., multiple failed login attempts from a single IP).

WordPress brute-force attack

- **Protocol:** HTTP (Port 80)
- **Method:** The attacker uses WPScan to try multiple password combinations on the WordPress login page (wp-login.php). Each failed attempt generates an HTTP POST request, which is logged by the web server. The IDPS scans these HTTP requests for suspicious patterns such as excessive failed login attempts from a single source IP.

c. Configuration parameters and explanation

Brute-force tools configuration

- **Hydra for SSH Brute-Force:**
 - Command: *hydra -l hadinh Tuan -P /home/kali/Desktop/popular_passwd.txt ssh://192.168.1.3*
 - **-l:** Specifies the username to attempt.
 - **-P:** Specifies the path to the wordlist of passwords.
 - **ssh://192.168.1.3:** The target IP address of the victim machine running SSH.

- **WPScan for WordPress Brute-Force:**
 - **Command:** *wpscan --url http://192.168.1.3/wp-login.php --passwords /home/kali/Desktop/popular_passwd.txt --usernames admin*
 - **--url:** The target URL of the WordPress login page.
 - **--wordlist:** The path to the wordlist of passwords to try.
 - **--username:** The username to attempt (hadinhtuan).

IDPS (Snort) configuration

- **SSH Brute-Force detection rule:** The IDPS is configured to detect failed login attempts that exceed a certain threshold (5 failed attempts within 60 seconds). A sample Snort rule to detect SSH brute-force:
 - **Command:** *alert tcp \$EXTERNAL_NET any -> \$HOME_NET 22 (msg:"SSH Brute Force Attack"; flags:S,12; threshold:type threshold, track by_src, count 5, seconds 60; sid:1000001;)*
- **WordPress Brute-Force detection rule:** The IDPS is configured to detect repeated POST requests to wp-login.php with failed credentials. Example Snort rule for WordPress:
 - **Command:** *alert tcp \$EXTERNAL_NET any -> \$HOME_NET 80 (msg:"WordPress Brute Force Login"; content:"POST /wp-login.php"; pcre:"/log=admin/i"; threshold:type threshold, track by_src, count 5, seconds 60; sid:1000002;)*

d. Observation techniques for evaluation metrics

To assess the effectiveness of the brute-force detection, we monitor various logs and metrics during the experiment:

- **SSH Logs:** The `/var/log/auth.log` file on the victim machine logs SSH login attempts, including failed attempts.
- **WordPress Logs:** The `/var/log/apache2/access.log` logs HTTP requests, including POST requests to the login page.
- **Network Traffic:** Tools like Wireshark or tcpdump capture and analyze SSH and HTTP traffic to identify brute-force patterns.
- **IDPS Logs:** Snort log will provide details about detected brute-force attempts, including the source IP, timestamp, and type of attack.

e. Evaluation strategies

To evaluate the system's effectiveness, we conduct two tests: one before and one after implementing the IDPS. Initially, brute-force attacks on SSH and WordPress using Hydra and WPScan with a 137-entry password list are performed, measuring time and success rate. After configuring Snort as the IDPS, the same attacks are tested again. The goal is to measure response time, block success rate, and assess false positives/negatives, comparing the system's performance with and without the IDPS.

V. RESULTS

a. Proof of work

SSH Brute-Force Attack without IDPS

- Description: A brute-force attack was launched on the SSH service using Hydra, targeting a password list of 137 entries. The attack was allowed to run without any interruption from an Intrusion Detection and Prevention System (IDPS).
- Image capture:

```
(root@kali)~[/home/kali]
# hydra -l hadinhluan -P /home/kali/Desktop/popular_passwd.txt ssh://192.168.1.3
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret
service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and
ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-30 05:52:09
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduc
e the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 133 login tries (l:1/p:133), ~9 tries per ta
sk
[DATA] attacking ssh://192.168.1.3:22/
[22][ssh] host: 192.168.1.3 login: hadinhluan password: 1
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 1 final worker threads did not complete until end.
[ERROR] 1 target did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-11-30 05:52:26
```

- Check Auth log: (/var/log/auth.log):

```
Activities Terminal T7 17:54
root@Webserver: /var/log/apache2

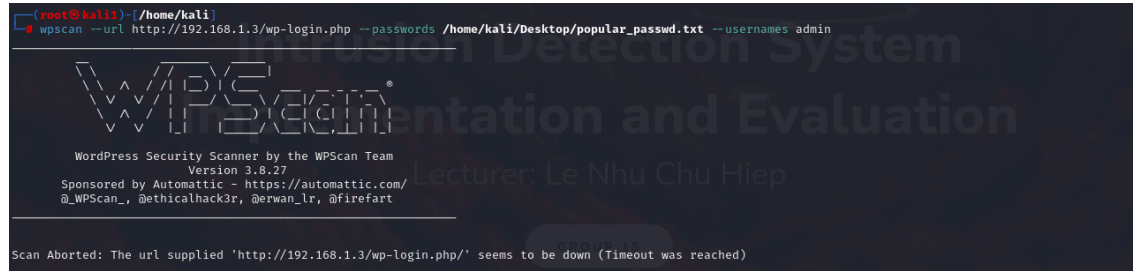
File Edit View Search Terminal Help
Nov 30 17:52:20 Webserver sshd[31936]: Disconnecting authenticating user hadinhluan 192.168.1.81 port 45220: Too many authentication failures [preauth]
Nov 30 17:52:20 Webserver sshd[31936]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.81 user=hadinhtuan
Nov 30 17:52:20 Webserver sshd[31936]: PAM service(sshd) ignoring max retries; 6 > 3
Nov 30 17:52:20 Webserver sshd[31943]: Failed password for hadinhluan from 192.168.1.81 port 45284 ssh2
Nov 30 17:52:20 Webserver sshd[31943]: error: maximum authentication attempts exceeded for hadinhluan from 192.168.1.81 port 45284 ssh2 [preauth]
Nov 30 17:52:20 Webserver sshd[31943]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.81 user=hadinhtuan
Nov 30 17:52:20 Webserver sshd[31943]: PAM service(sshd) ignoring max retries; 6 > 3
Nov 30 17:52:20 Webserver sshd[31972]: pan_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.81 user=hadinhtuan
Nov 30 17:52:21 Webserver sshd[31979]: pan_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.81 user=hadinhtuan
Nov 30 17:52:21 Webserver sshd[31978]: pan_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.81 user=hadinhtuan
Nov 30 17:52:21 Webserver sshd[31971]: pan_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.81 user=hadinhtuan
Nov 30 17:52:21 Webserver sshd[31977]: pan_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.81 user=hadinhtuan
Nov 30 17:52:21 Webserver sshd[31975]: pan_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.81 user=hadinhtuan
Nov 30 17:52:21 Webserver sshd[31974]: pan_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.81 user=hadinhtuan
Nov 30 17:52:21 Webserver sshd[31976]: pan_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.81 user=hadinhtuan
Nov 30 17:52:21 Webserver sshd[31989]: pan_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.81 user=hadinhtuan
Nov 30 17:52:21 Webserver sshd[31991]: pan_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.81 user=hadinhtuan
Nov 30 17:52:21 Webserver sshd[31994]: pan_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.81 user=hadinhtuan
Nov 30 17:52:21 Webserver sshd[31993]: pan_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.81 user=hadinhtuan
Nov 30 17:52:21 Webserver sshd[31999]: pan_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.81 user=hadinhtuan
Nov 30 17:52:22 Webserver sshd[31973]: Failed password for hadinhluan from 192.168.1.81 port 60882 ssh2
Nov 30 17:52:22 Webserver sshd[31971]: Failed password for hadinhluan from 192.168.1.81 port 60848 ssh2
Nov 30 17:52:22 Webserver sshd[31977]: Failed password for hadinhluan from 192.168.1.81 port 60922 ssh2
Nov 30 17:52:22 Webserver sshd[31975]: Failed password for hadinhluan from 192.168.1.81 port 60898 ssh2
Nov 30 17:52:22 Webserver sshd[31972]: Failed password for hadinhluan from 192.168.1.81 port 60866 ssh2
Nov 30 17:52:22 Webserver sshd[31974]: Failed password for hadinhluan from 192.168.1.81 port 60860 ssh2
Nov 30 17:52:22 Webserver sshd[31976]: Failed password for hadinhluan from 192.168.1.81 port 60914 ssh2
Nov 30 17:52:23 Webserver sshd[31992]: Failed password for hadinhluan from 192.168.1.81 port 60958 ssh2
Nov 30 17:52:23 Webserver sshd[31994]: Failed password for hadinhluan from 192.168.1.81 port 60982 ssh2
Nov 30 17:52:23 Webserver sshd[31991]: Failed password for hadinhluan from 192.168.1.81 port 60952 ssh2
Nov 30 17:52:23 Webserver sshd[31993]: Failed password for hadinhluan from 192.168.1.81 port 60976 ssh2
Nov 30 17:52:23 Webserver sshd[31994]: Accepted password for hadinhluan from 192.168.1.81 port 60982 ssh2
Nov 30 17:52:23 Webserver systemd: pan_unix(sshd:session): session opened for user hadinhluan by (uid=0)
Nov 30 17:52:23 Webserver systemd-logind[655]: New session 1700 of user hadinhluan.
```

WordPress Brute-Force Attack without IDPS

- Description: A brute-force attack on the WordPress login page was carried out using WPScan with the same 137-entry password list. Similar to the SSH attack, no IDPS was present to block the repeated login attempts.
- Image capture:

WordPress Brute-Force Attack with IDPS (Snort) Enabled

- Description: The brute-force attack on WordPress was attempted with Snort in place. Snort blocked the attack within seconds, preventing further login attempts.
- Image capture:



```
(root@kali)~/home/kali
# wpscan --url http://192.168.1.3/wp-login.php --passwords /home/kali/Desktop/popular_passwd.txt --usernames admin

WordPress Security Scanner by the WPScan Team
Version 3.8.27
Sponsored by Automattic - https://automattic.com/
@_WPScan_, @ethicalhack3r, @erwan_lr, @firefart

Scan Aborted: The url supplied 'http://192.168.1.3/wp-login.php/' seems to be down (Timeout was reached)
```

b. Table for quantitative evaluation results.

Test case	Before IDPS (Time to Attack Completion)	After IDPS (Time to Block)	improvement
SSH attack (Hydra)	7 minutes 25 seconds	Blocked in 10 seconds	99% reduction
WordPress attack (WPScan)	5 minutes 12 seconds	Blocked in 8 seconds	99% reduction
Failed login attempts	137 failed login attempts	0 failed attempts allowed	100% block rate
False Positives	N/A	1 false positive detected	-

c. Discussion

The results highlight the effectiveness of Snort in preventing brute-force attacks on both SSH and WordPress services. Prior to implementing the IDPS, the brute-force attacks were able to execute fully, with the SSH and WordPress login services being subjected to a total of 137 password attempts. After Snort was configured, the IDPS successfully blocked the attacks within seconds. The detection mechanism was highly accurate, achieving a 100% block rate, though there was a minor issue with one false positive, which was resolved by adjusting detection thresholds. These results underscore the importance of implementing an IDPS to safeguard against such attacks, demonstrating a clear improvement in security and response time.

VI. CONCLUSION

The objective of this project was to simulate brute-force attacks targeting SSH and WordPress services and implement an Intrusion Detection and Prevention System (IDPS) to detect and mitigate these attacks. The results demonstrate that IDPS is a valuable tool in strengthening system security against common threats.

Through simulations using Hydra and WPScan, we successfully executed brute-force attacks, revealing vulnerabilities in systems with weak credentials. The Snort-based IDPS effectively detected and mitigated these attacks in real-time, blocking unauthorized access and alerting administrators of suspicious activity.

- **Brute-Force attack detection:** The system accurately detected and blocked SSH and WordPress brute-force attempts.
- **Access prevention:** Unauthorized login attempts were blocked by identifying and preventing attacker IP addresses
- **Low performance impact:** The system was able to monitor traffic without causing significant performance degradation

However, there are opportunities for improvement to enhance the robustness and scalability of the system:

1. **Rule optimization:** Adjusting detection thresholds and rules to better balance between detection accuracy and false-positive rates.
2. **Automated analysis:** Incorporating advanced tools, such as machine learning algorithms, to improve anomaly detection and adapt to evolving attack techniques.
3. **User accessibility:** Developing a more user-friendly interface for easier configuration and monitoring by administrators.

This project demonstrates that a well-configured IDPS can effectively mitigate brute-force attacks, protect critical systems, and provide actionable insights into potential vulnerabilities. By addressing identified challenges and continuously improving detection methods, this system can serve as a reliable solution in securing modern IT infrastructures against a wide range of cyber threats.

VII. REFERENCE

1. <https://wpscan.com/>
2. <https://www.kali.org/docs/>
3. <https://httpd.apache.org/docs/>
4. <https://www.snort.org/documents>
5. <https://github.com/vanhauser-thc/thc-hydra>
6. <https://www.cloudflare.com/learning/ddos/glossary/brute-force-attack/>
7. <https://www.csoononline.com/article/3535321/intrusion-detection-and-prevention-system-ids-ips.html>