Failed Login Monitoring Documentation

Date: December 09-12-2024

❖ Table of Contents

1. Create Python monitoring script:

#nano monitor_failed_logins.py

```
import re
import subprocess
import logging
from datetime import datetime, timedelta
import os
# Configuration
LOG_FILE = "/var/log/auth.log"  # Path to the log file
THRESHOLD = 5  # Number of failed attempts before blocking
BLOCKED_IPS_FILE = "/var/log/blocked_ips.txt"  # File to keep track of blocked IPs
WHITELIST = ['127.0.0.1']  # IPs to never block
# Setup logging
logging.basicConfig(
    filename='wp_security_monitor.log',
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s'
)
def get_failed_login_ips():
    """
    Parse the log file and return a dictionary of IPs with failed login attempts.
    Only considers attempts within the last hour.
    """
    failed_ips = {}
```

```
  one_hour_ago = datetime.now() - timedelta(hours=1)
try:
    with open(LOG_FILE, "r") as file:
        for line in file:
            try:
                # Extract timestamp and convert to datetime
                timestamp_str = line.split()[0]
                timestamp = datetime.strptime(timestamp_str.split('.')[0],
                                              '%Y-%m-%dT%H:%M:%S')
                # Skip old entries
                if timestamp < one_hour_ago:
                    continue
                # Look for WordPress login failure patterns
                if any(pattern in line.lower() for pattern in [
                    "wp-login.php",
                    "wordpress login",
                    "xmlrpc.php"
                ]):
                    # Extract IP using regex
                    ip_match = re.search(r'from (\d+\.\d+\.\d+\.\d+)', line)
                    if ip_match:
                        ip = ip_match.group(1)
                        if ip not in WHITELIST:
                            failed_ips[ip] = failed_ips.get(ip, 0) + 1
                            logging.debug(f"Failed attempt from IP: {ip}")
            except Exception as e:
                logging.error(f"Error processing line: {line.strip()} - {str(e)}")
                continue
    logging.info(f"Found {len(failed_ips)} IPs with failed attempts")
    return failed_ips
  except FileNotFoundError:
    logging.error(f"Log file not found: {LOG_FILE}")
```

```
        return {}
def block_ip(ip):
    """Block an IP using iptables and record it."""
    if ip in WHITELIST:
        logging.warning(f"Attempted to block whitelisted IP {ip} - skipping")
        return False
 try:
        # Check if IP is already blocked
        check_cmd = f"iptables -L INPUT -v -n | grep {ip}"
        result = subprocess.run(check_cmd, shell=True, capture_output=True, text=True)
        if ip in result.stdout:
            logging.info(f"IP {ip} is already blocked")
            return True
        # Add new block rule
        block_cmd = [
            "iptables",
            "-A", "INPUT",
            "-s", ip,
            "-j", "DROP",
            "-m", "comment",
            "--comment", "Blocked by WP security monitor"
        ]
        subprocess.run(block_cmd, check=True)
        # Record blocked IP
        with open(BLOCKED_IPS_FILE, "a") as file:
            file.write(f"{datetime.now().isoformat()} - {ip}\n")
        logging.info(f"Successfully blocked IP: {ip}")
        return True
    except subprocess.CalledProcessError as e:
        logging.error(f"Error blocking IP {ip}: {str(e)}")
        return False
def load_blocked_ips():
```

```
    """Load the list of already blocked IPs."""
    try:
        with open(BLOCKED_IPS_FILE, "r") as file:
            # Extract IPs from the log lines
            return set(line.split()[-1] for line in file)
    except FileNotFoundError:
        logging.info(f"Blocked IPs file not found, creating new one")
        open(BLOCKED_IPS_FILE, "a").close()  # Create the file
        return set()
def main():
    """Main function to monitor logs and block offending IPs."""
    if os.geteuid() != 0:
        logging.error("This script must be run as root")
        return
    logging.info("Starting WordPress security monitor...")
    # Get current failed attempts
    failed_ips = get_failed_login_ips()
    # Load already blocked IPs
    blocked_ips = load_blocked_ips()
    # Check and block IPs exceeding the threshold
    for ip, count in failed_ips.items():
        if count >= THRESHOLD and ip not in blocked_ips:
            if block_ip(ip):
                logging.info(f"Blocked IP {ip} after {count} failed attempts")
    logging.info("Monitoring complete")
if __name__ == "__main__":
    main()
```

2. **Set up automation:**
    - Change permissions to file

      #chmod +x monitor_failed_logins.py

      #crontab -e

- Add to crontab:

  */5 * * * * /usr/bin/python3 /path/to/monitor_failed_logins.py*

3. Troubleshooting

1. Email Alert Issues:

   - Check mail configuration:

     #sudo postfix status

     #tail -f /var/log/mail.log

2. Monitoring Script Issues:

   - Check script logs:

     #tail -f /var/log/resource_monitor.log

3. Login Monitor Issues:

   - View blocked IPs:

     #sudo iptables -L

     #cat /var/log/blocked_ips.txt

   - Check script execution:

     #tail -f /var/log/authlog