

## Matthew Veroutis Journal #3

### Dates:

#### 2024-11-25:

I wanted to start the project as soon as possible, so I decided to work on my script on Monday. It was honestly a lot harder and took a lot longer than I originally expected. At first, I had to decide what my script would do. In the proposal and as a team we originally said that we would make a script that would warn the user if the usage of a component like CPU goes too high it would warn the users but when I was looking through the Checkmk logs I could not find the one which has all the usages of CPU/RAM. So when I was looking through all the logs I found one which was called "security.log" which stores login attempts (Failed or Successful). So then I thought of making a bash script that would look in the log to warn the user of failed login attempts and show the IP of the computer that had the failed login attempt. So I went to ChatGPT to make a script that would satisfy these criteria but I had to do a lot of refining because the script was either way too complicated or didn't work at all. After some refining and trial and error, I was able to come up with a functional and understandable script that looks like this:

```
#!/bin/bash

#Path to the security log
SECURITY_LOG="/opt/omd/sites/free_test/var/log/security.log"

#Get the current time and time one minute ago
CURRENT_TIME=$(date +"%Y-%m-%d %H:%M")
PREVIOUS_TIME=$(date -d "1 minute ago" +"%Y-%m-%d %H:%M")

#Check the log for entries with "authentication failed" in the last minute
sudo grep -a -E "$PREVIOUS_TIME|$CURRENT_TIME" grep -E
"$PREVIOUS_TIME|$CURRENT_TIME" "$SECURITY_LOG" | grep
"summary": "authentication failed" | while read -r line; do
# Extract the remote IP of the failed login
REMOTE_IP=$(echo "$line" | grep -oP "remote_ip: \"\K[^\"]+")
zenity --warning --text="Warning: Failed login attempt from IP:
$REMOTE_IP"
done
```

Firstly before you run this script you have to install zenity: which allows you to send warning messages with a custom message to the user. There are also a few other steps down below that you must follow

To install zenity you do:

## **Apt Install zenity**

Then you put this text into a script file by using any text editor such as nano:

## **Nano securityBash.sh**

Then you have to make the file executable by doing:

## **chmod +x securityBash.sh**

After you can finally run the file by doing:

## **./securityBash.sh**

(if you want to see the pop up make sure you try to log into the website with a bad username or password within a minute of running the command since this script checks for failed attempts done in the last minute)

I Initially wanted to do both the scheduling and the scripting in one day but the scripting part took me so long and drained so much energy that it was to do it the next day so I would have a fresh mind.

## **2024-11-26**

Unfortunately, the scheduling of the script was harder than making the script since I ran into many problems that I had to find workarounds. I decided to use crontab since it's easier to use than systemctl. So firstly to make it a cronjob that will execute every minute, you have to do the following:

First:

crontab -e

Second:

When the crontab file opens add the line:

\* \* \* \* \* /home/osboxes/securityBash.sh

(make sure you put the absolute file path to the script)

So now onto the first issue which was that one of the commands needs to run in sudo, so it would prompt for the password but if it's automated this does not work so:

I edited the sudo file by doing:

First:

```
sudo visudo
```

Then I add the line:

```
Osboxes ALL=(ALL) NOPASSWD: ALL
```

So osboxes can run any command in sudo without the need to enter the password.

After this, I also did more research which led me to learn that you also must use the full path names of the command since crontab doesn't have full access to the \$PATH variable so I had to change the script to look like the following:

```
#!/bin/bash
```

```
#Absolute paths for commands
```

```
DATE_CMD="/bin/date"
```

```
GREP_CMD="/bin/grep"
```

```
ZENITY_CMD="/usr/bin/zenity"
```

```
SUDO_CMD="/usr/bin/sudo"
```

```
#Path to the security log
```

```
SECURITY_LOG="/opt/omd/sites/free_test/var/log/security.log"
```

```
#Get the current time and time one minute ago
```

```
CURRENT_TIME=$(($DATE_CMD +"%Y-%m-%d %H:%M")
```

```
PREVIOUS_TIME=$(($DATE_CMD -d "1 minute ago" +"%Y-%m-%d  
%H:%M")
```

```
#Check the log for entries with "authentication failed" in the last minute
```

```
$SUDO_CMD $GREP_CMD -a -E
```

```
"$PREVIOUS_TIME|$CURRENT_TIME" "$SECURITY_LOG" |
```

```
$GREP_CMD "'summary': 'authentication failed'" | while read -r line; do
```

```
# Extract the remote IP of the failed login
```

```
REMOTE_IP=$(echo "$line" | $GREP_CMD -oP "'remote_ip': '\K[^"]+')
```

```
Display a warning using Zenity
```

```
$ZENITY_CMD --warning --text="Warning: Failed login attempt from IP:
$REMOTE_IP"
```

done

So I had to change it so all the commands use the full path instead of their short names.

But even after changing this it still didn't work so I had a theory that it was something to do with Zenity not being able to display so to just test out Zenity I added this line to the crontab file:

```
* * * * * /usr/bin/zenity --info --text="This is a test"
```

This should have popped up a message but it did not do i asked ChatGPT what the issue could be and it said that




It seems like you're trying to display a Zenity message every minute via `cron`, but it's not working as expected. There are a few potential reasons why it might not be showing up.

### Key Issues to Consider:

1. **Cron's Environment:** Cron runs with a minimal environment, which means it does not have access to the same display settings as a regular user session. Zenity relies on the X11 display to show GUI dialogs, and cron jobs don't have access to this display by default.
2. **Accessing the Display in Cron:** You need to explicitly tell cron to use the graphical display (`$DISPLAY`), which is used by GUI applications like Zenity.

### How to Fix This:

To fix this, you need to tell cron where to find the display for GUI applications. You can do this by setting the `DISPLAY` environment variable in  ur cron job.

So I changed it to the following so I specify the display and I output any errors to a file (Unfortunately there is still more errors):

```
* * * * * DISPLAY=:0 /usr/bin/zenity --info --text="This is a test" >>
/home/osboxes/zenity.log 2>&1
```

After it gave me the following error:

```
Authorization required, but no authorization protocol specified (zenity:108016):
Gtk-WARNING **: 19:27:02.196: cannot open display: :0
```

After some more research, I found out you need to allow to user to access the display server so I did the following:

```
export DISPLAY=:0  
xhost +SI:localuser:$(whoami)
```

So this allows the user access to the display.

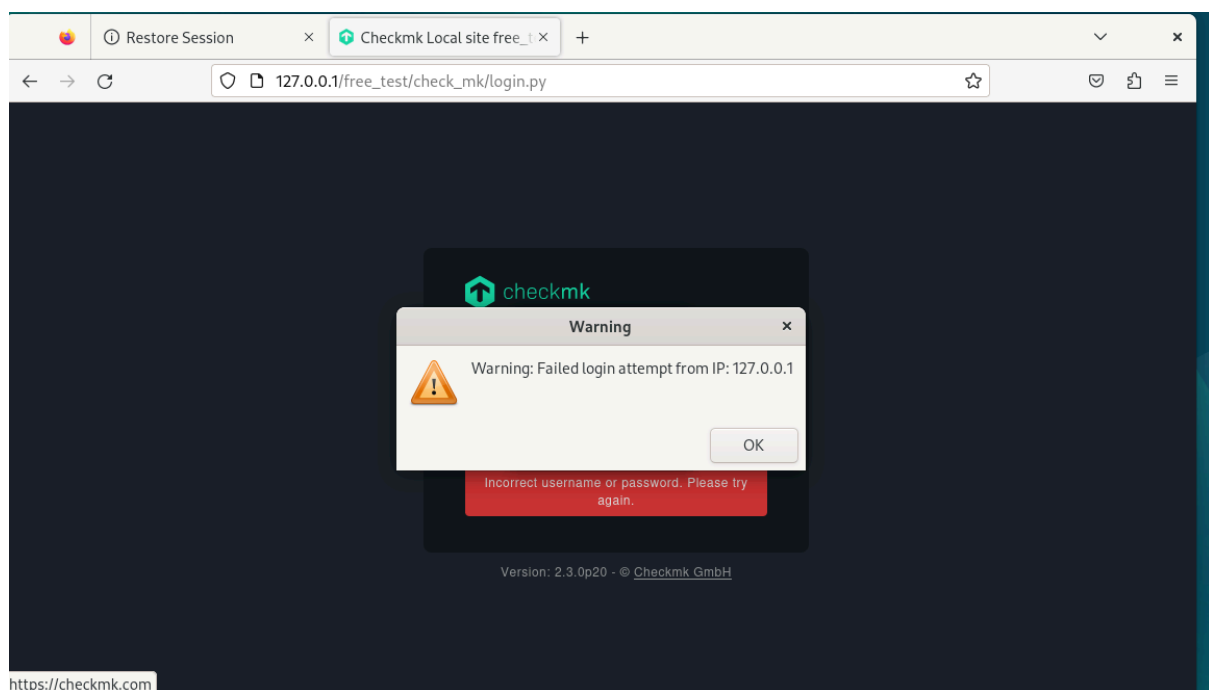
Lucky after fixing so many problems the test worked and the pop-up message would pop up every minute so I changed the initial cronjob to also use "DISPLAY=:0"

```
* * * * * DISPLAY=:0 /home/osboxes/securityBash.sh
```

This also worked beautifully after all the adjustments, so when I would make a failed attempt the script would automatically run and put a prompt on the screen to warn the user of the failed login attempt.

## 2024-11-28

Unfortunately, when I closed the system and tried to run it again it was not popping up and with further research, I realized that the access to the display gets reset every time the VM restarts. So for it to work, I would have to run `export DISPLAY=:0 && xhost +SI:localuser:$(whoami)`, at first I ran this in crontab but no matter what I did I could not get it to work. So I waited until class to ask what I could do, and Ms. Tassia told me that I could put that line in the `.bashrc`. Luckily after I did this after every restart it would work beautifully. So now the script is fully functional look at the following windows for the popup after a failed attempt:



This popped up after the failed attempt, I also, unfortunately, learned that this script may not be useful in the real world because servers should not have a graphical user interface. However I feel like this was still beneficial since I learned many interesting tools, and I proved to myself that I could keep finding solutions for the countless errors I have faced.

Here are the instructions to implement a script like this on your machine:

**1. First download zenity by doing:**

Sudo apt install zenity

**2. Make a file named securityBash.sh by doing:**

nano securityBash.sh

**3. Then make it executable by doing**

chmod +x securityBash.sh

**4. Make a script file that contains:**

```
#!/bin/bash
```

```
#Absolute paths for commands
```

```
DATE_CMD="/bin/date"
```

```
GREP_CMD="/bin/grep"
```

```
ZENITY_CMD="/usr/bin/zenity"
```

```
SUDO_CMD="/usr/bin/sudo"
```

```
#Path to the security log
```

```
SECURITY_LOG="/opt/omd/sites/free_test/var/log/security.log"
```

```
#Get the current time and time one minute ago
```

```
CURRENT_TIME=$(($DATE_CMD +"%Y-%m-%d %H:%M")
```

```
PREVIOUS_TIME=$(($DATE_CMD -d "1 minute ago" +"%Y-%m-%d %H:%M")
```

```
#Check the log for entries with "authentication failed" in the last minute
```

```
$SUDO_CMD $GREP_CMD -a -E "$PREVIOUS_TIME|$CURRENT_TIME"
```

```
"$SECURITY_LOG" | $GREP_CMD "summary": "authentication failed" | while read  
-r line; do
```

```
# Extract the remote IP of the failed login
```

```
REMOTE_IP=$(echo "$line" | $GREP_CMD -oP "remote_ip": "\K[^\"]+")
```

```
Display a warning using Zenity
```

```
$ZENITY_CMD --warning --text="Warning: Failed login attempt from IP:
$REMOTE_IP"
```

done

\*note paths to commands may differ so it is better to verify that they are correct,  
replace free\_test with the actual name of your website.

**5. You also have type:**

```
sudo visudo
```

**6. Then type the following while in the visudo :**

```
Osboxes ALL=(ALL) NOPASSWD: ALL
```

**7. Then go to bashrc variable by doing:**

```
Sudo .bashrc
```

**8. type bashrc inside that file:**

```
export DISPLAY=:0 && /usr/bin/xhost +SI:localuser:osboxes
```

\*replace osboxes with your own user)

**9. Finally open crontab file by doing:**

```
crontab -e
```

**10. Put the following in that file for it to schedule to run every minute:**

```
* * * * * DISPLAY=:0 /home/osboxes/securityBash.sh
```

Link to the GitHub: <https://github.com/4lex16/UnixProject.git>