Configure a Vulnerable Environment

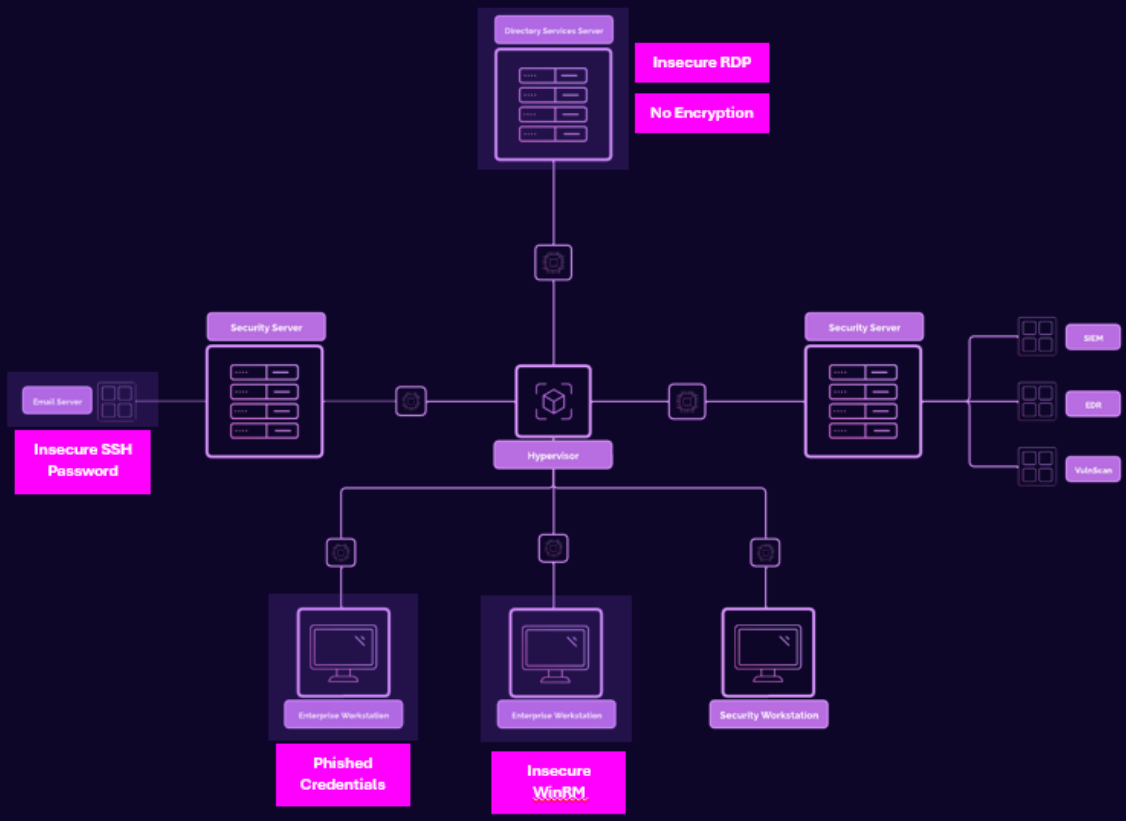
>prøjectsecurity_

Table of Contents

Prerequisites

1. Baseline project-x network has been provisioned and configured.
    - Guides 1 – 9 have been completed.

Network Topology



Vulnerable Environment

Overview

In this guide, we are going to perform configuration changes to make our environment 'vulnerable'.

Depending on the size, scale, and complexity of a business network, attackers will often leverage insecure and default configurations to their advantage. Even though these configurations appear to be obviously insecure, you will still see some of these in

production environments. Often times, this is due to legacy systems, forgotten infrastructure, urgency, or laziness (that one would be me).

❗ These configurations are intended for homelab use only and should not be applied in production environments. Projectsecurity.io assumes no responsibility for any communication or actions taken based on this material.

👉 Please make sure the Setup **Wazuh Section** has been completed in addition to all other guides outlined in the Prerequisites.

Open SSH on [project-x-email-svr]

Update system and install `openssh` if it is not yet installed (should already be installed).

```
sudo apt update
```

```
sudo apt install openssh-server -y
```

Enable the SSH Server and ensure it runs on boot.

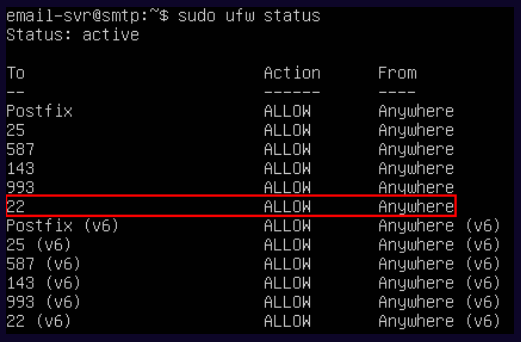
```
sudo systemctl start ssh
```

```
sudo systemctl enable ssh
```

Change UFW rules to allow SSH connections:

```
sudo ufw allow 22
```

```
sudo ufw status
```



Verify SSH is running:

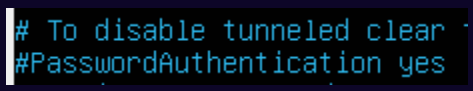
```
sudo systemctl status ssh
```

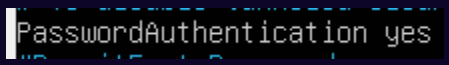
Enable Password Authentication. Open the SSH configuration file:

```
sudo nano /etc/ssh/sshd_config
```

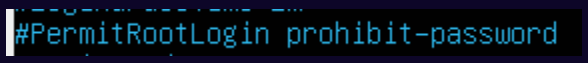
Locate the line for `PasswordAuthentication`. Uncomment if commented.

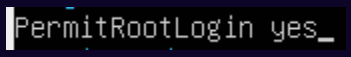
```
sudo nano /etc/ssh/sshd_config
```





Permit root login. Navigate to the `#PermitRootLogin` block. Uncomment and delete `prohibit-password`, change to `yes`.





Restart SSH service:

```
sudo systemctl restart ssh
```

Set root's password (use the password: `november`)

```
sudo passwd root
```

Detection Integration

[project-x-email-svr] does not have the Wazuh agent installed. This is intentional to demonstrate how the absence of detection controls can create a gap in identifying potentially malicious activity.

Open SSH on [project-x-linux-client]

Update system and install `openssh` if it is not yet installed (should already be installed).

```
sudo apt update
```

```
sudo apt install openssh-server -y
```

Enable the SSH Server and ensure it runs on boot.

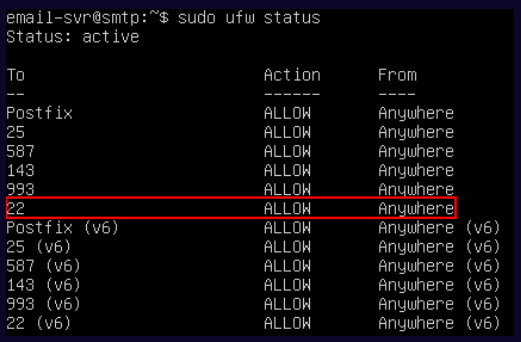
```
sudo systemctl start ssh
```

```
sudo systemctl enable ssh
```

Change UFW rules to allow SSH connections:

```
sudo ufw allow 22
```

```
sudo ufw status
```



Verify SSH is running:

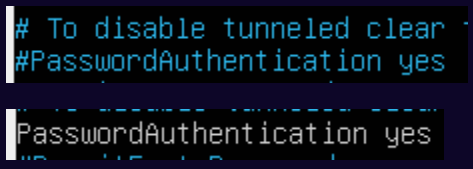
```
sudo systemctl status ssh
```

Enable Password Authentication. Open the SSH configuration file:

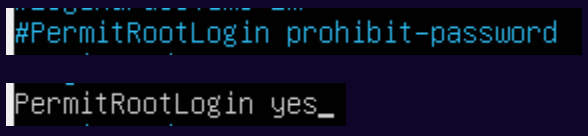
```
sudo nano /etc/ssh/sshd_config
```

Locate the line for `PasswordAuthentication`. Uncomment if commented.

```
sudo nano /etc/ssh/sshd_config
```



Permit root login. Navigate to the `#PermitRootLogin` block. Uncomment and delete `prohibit-password`, change to `yes`.



Restart SSH service:

```
sudo systemctl restart ssh
```

Set root's password (use the password: `november`)

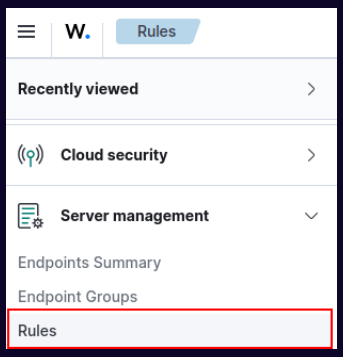
```
sudo passwd root
```

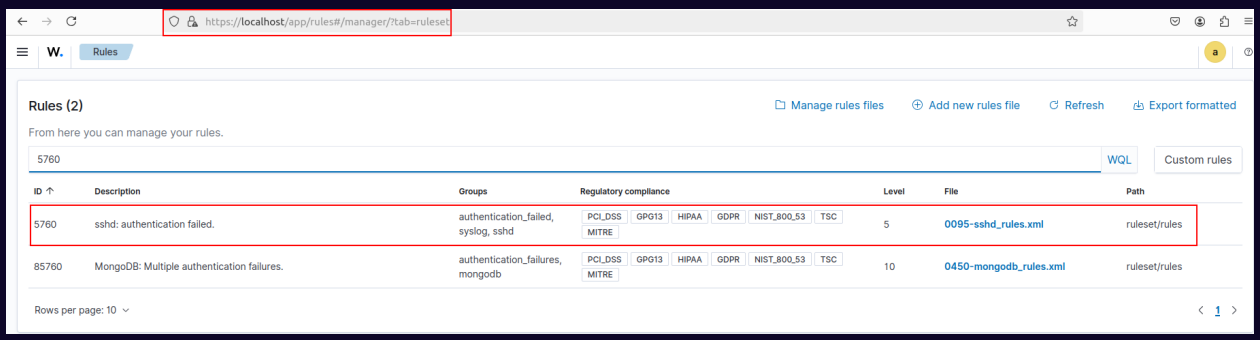
Detection Integration

Wazuh has a built-in rule detection to detect authentication failures from the sshd daemon.

- Wazuh Rule ID: 5760.
- Description: sshd: authentication failed.

Navigate to “Server management” → “Rules”. And look up “5760” to view more detail about this rule.





Here is a sample snapshot of a log generated when the SSH attempt fails.

Go to “Explore” → “Discover” tab. Look up “sshd”.

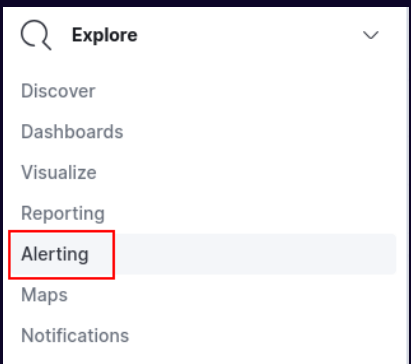
👉 You must have a failed login attempt.

t	full_log	Dec 27 21:21:46 linux-client sshd[37536]: Failed password for root from 10.0.0.100 port 50273 ssh2
t	id	1735334501.575869
t	input.type	log
t	location	journald
t	manager.name	secbox
t	predecoder.hostname	linux-client
t	predecoder.program_name	sshd
t	predecoder.timestamp	Dec 27 21:21:46
t	rule.description	sshd: authentication failed.
#	rule.firedtimes	2
t	rule.gdpr	IV_35.7.d, IV_32.2
t	rule.gpg13	7.1
t	rule.groups	syslog, sshd, authentication_failed
t	rule.hipaa	164.312.b
t	rule.id	5760

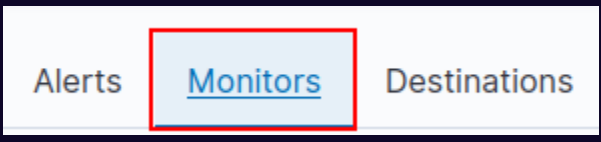
Create Detection Alert

Let's create an alert for Failed SSH attempts. To do this, a Monitor will be set up to analyze logs. Based on certain conditions defined, a Trigger can be setup to open an Alert.

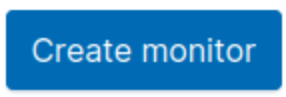
Go to "Explore" → "Alerting".



Select the "Monitors" tab on the top left.

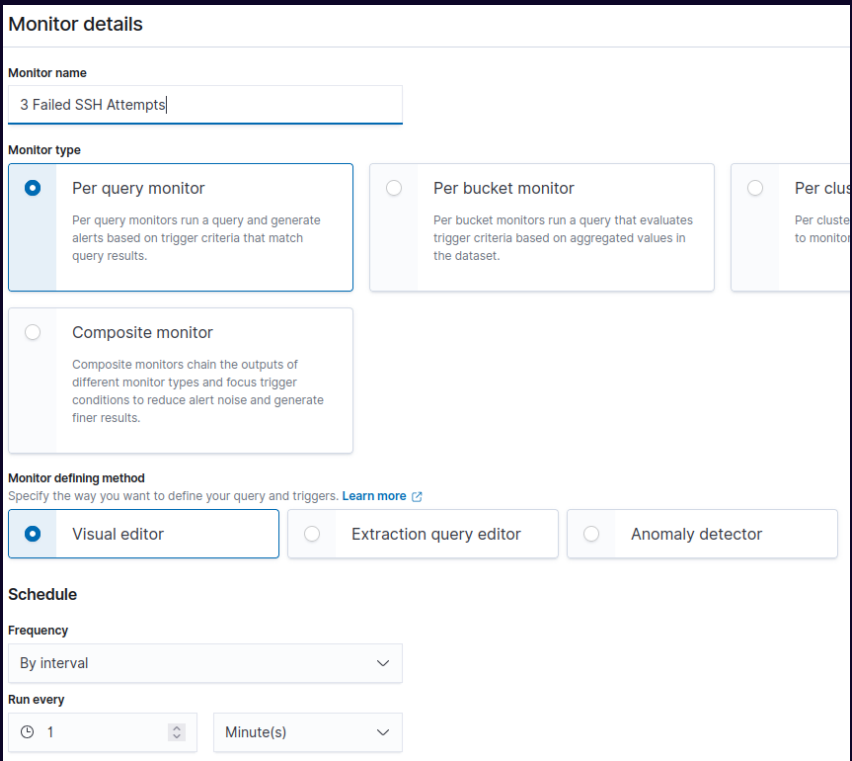


Select "Create monitor".



Here we can create a new monitor.

Title the Monitor “3 Failed SSH Attempts”. Leave everything else default.

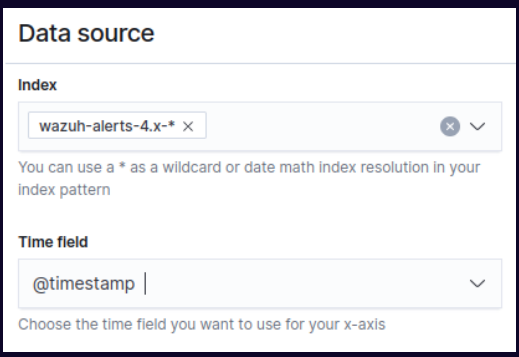


Scroll down to “Data source”. Add the following for the Index, hit the Enter key after typing:

```
wazuh-alerts-4.x-*
```

For “Time Field” select:

```
@timestamp
```



Next, we can add a query to select what logs and log fields we would like to monitor.

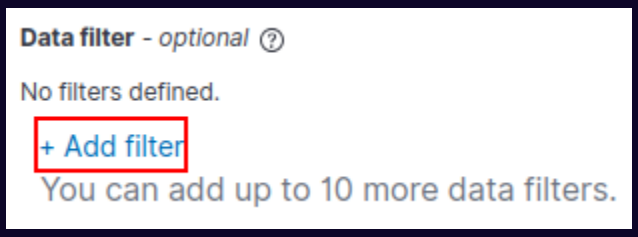
Based on a sample log of a failed ssh attempt, we can construct a query to monitor specific field / value key-pairs.

t	decoder.name	sshd
t	decoder.parent	sshd
t	full_log	Dec 28 13:34:59 linux-client sshd[3319]: Failed password for invalid user sec-user from 10.0.0.10 port 42296 ssh2
t	id	1735414503.22865737
t	input.type	log
t	location	/var/log/auth.log
t	manager.name	secbox
t	predecoder.hostname	linux-client
t	predecoder.program_name	sshd
t	predecoder.timestamp	Dec 28 13:34:59
t	rule.description	sshd: Attempt to login using a non-existent user
#	rule.firedtimes	4
t	rule.gdpr	IV_35.7.d, IV_32.2
t	rule.gpg13	7.1
t	rule.groups	syslog, sshd, authentication_failed, invalid_login

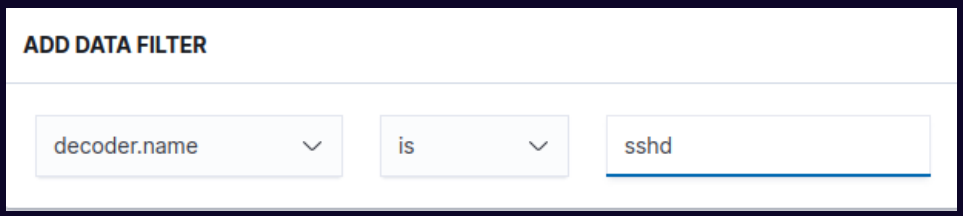
👉 A Failed SSH attempt log.

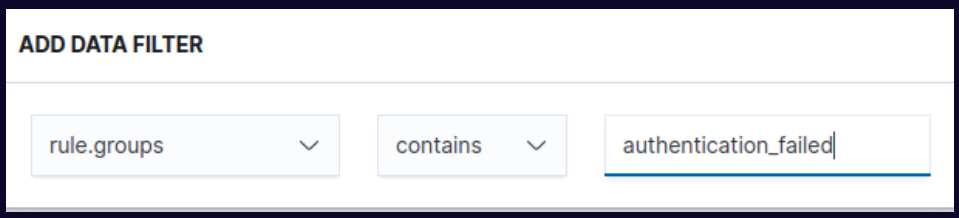
💡 Where would you get sample logs?: We can populate sample logs by emulating the behavior (failing SSH attempts) and having our Wazuh agent send the logs. We can also use open-source rules / log samples to generate logs.

Navigate to the “Data filter” → “+ Add filter”.

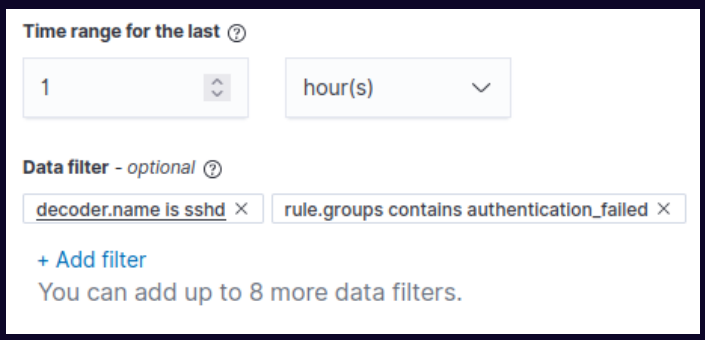


Based on the above sample log, let’s craft a query to select based on the “sshd” process name and the “authentication_failed” rule group.

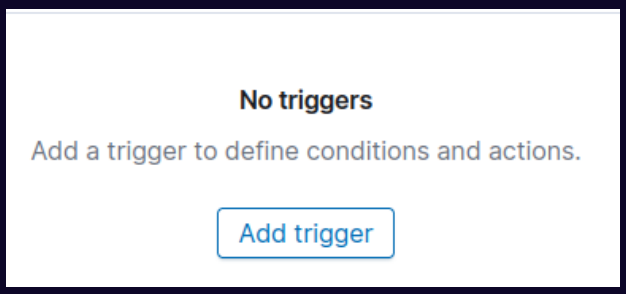




Your “Data filter” tab should now look something like this.

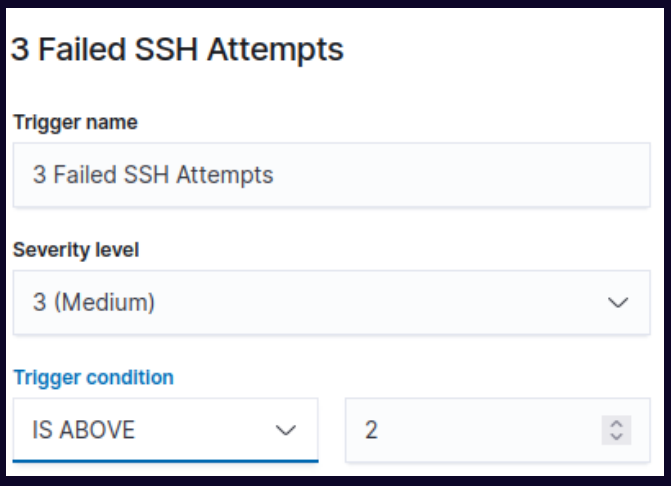


Let’s add a Trigger.

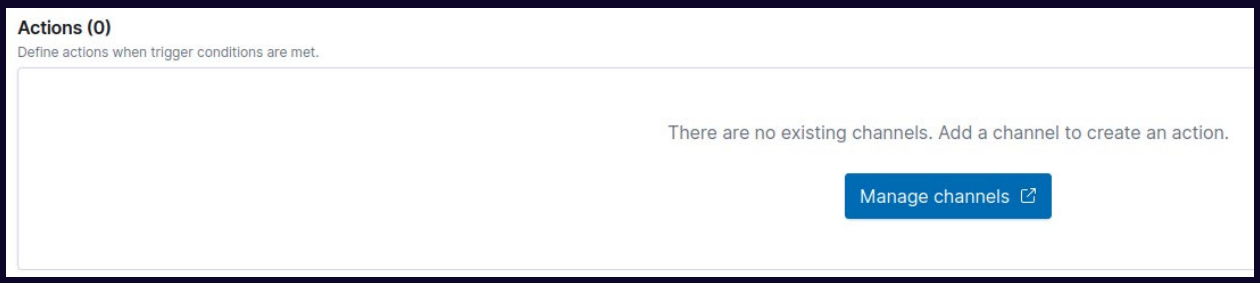


Add the following conditions to the Trigger. We set the “Severity level” to 3 (Medium) and the “Trigger condition” above 2.

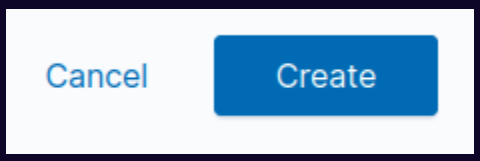
👉 So what we are doing here... Once we get more than 2 logs that have met the following query conditions from above, an alert will be generated.



There's also an "Actions" option. Here we could create an Email, Slack, or Microsoft Teams notification of the alert. We could also launch a playbook that would follow a specific instruction set to analyze, investigate, or isolate a host. We are not going to configure this section. (We will in future modules!)



Scroll to the bottom and Select "Create".



Configure Email Connection from [project-x-email-svr] to [project-x-linux-client]

The current configuration allows [project-x-email-svr] to send email to itself or forward email to other local hosts. If we attempt to send an email from a workstation (ie project-x-linux-client) to the [project-x-email-svr] host, the email will not be able to route.

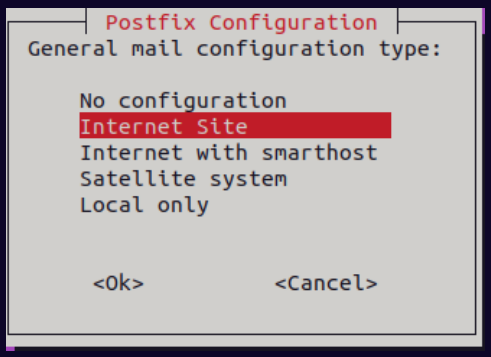
Let's setup and configure postfix on [project-x-linux-client] so we can send emails to the email server, leveraging Postfix again.

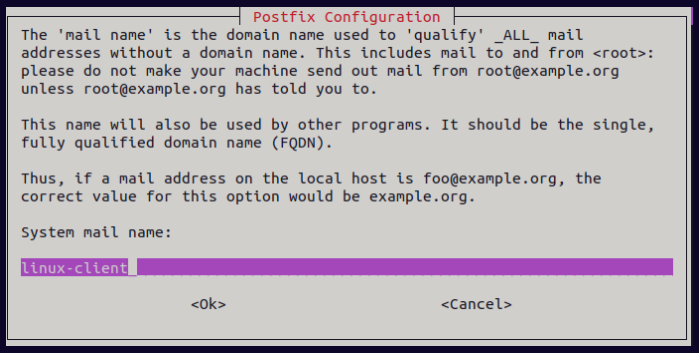
Log into [project-x-linux-client].

Install postfix and the mailutils utility to interact with your email inbox.

```
sudo apt install postfix mailutils -y
```

Choose "Internet Site" → Leave the "System mail name:" the default linux-client.





Navigate to the /postfix/main.cf configuration file:

```
sudo nano /etc/postfix/main.cf
```

Add the following (highlighted):

```
my_domain = corp.project-x-dc.com
```

```
mynetworks = 127.0.0.0/8 10.0.0.0/24 [::ffff:127.0.0.0]/104 [::1]/128
```

```
home_mailbox = Maildir/
```

```
virtual_alias_maps=hash:/etc/postfix/virtual
```

```
smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated defer_unauth_destination
myhostname = linux-client
my_domain = corp.project-x-dc.com
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
mydestination = $myhostname, linux-client, localhost.localdomain, , localhost
relayhost =
mynetworks = 127.0.0.0/8 10.0.0.0/24 [::ffff:127.0.0.0]/104 [::1]/128
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
inet_protocols = all
home_mailbox = Maildir/
virtual_alias_maps=hash:/etc/postfix/virtual
```

Save the file with CTRL + X +Y + Enter.

Next, create the virtual file, then we can begin mapping email accounts to user accounts to Linux system.

```
sudo nano /etc/postfix/virtual
```

Enter any email address to accept:

```
email-svr@smtp.corp.project-x-dc.com janed
```

👉 Here we are routing any email that comes from the email-svr address to janed.

Save and close with CTRL+X, Y, then ENTER.

Apply the mapping to the virtual file:

```
sudo postmap /etc/postfix/virtual
```

```
sudo systemctl restart postfix
```

Clear the screen, create janed's Mailbox directory:

```
mkdir -p ~/Maildir/{cur,new,tmp}
```

```
chmod -R 700 ~/Maildir
```

To interact with mail being delivered, we will use the mail package. mail will look for a variable called MAIL to find mail for your user. Let's ensure the MAIL variable is set regardless of how the account is accessed:

```
echo 'export MAIL=~/Maildir' | sudo tee -a /etc/bash.bashrc |
sudo tee -a /etc/profile.d/mail.sh
```

Supply variable into the current session with:

```
source /etc/profile.d/mail.sh
```

Enable SMTP (postfix config) on UFW:

```
sudo ufw allow postfix
```

```
sudo ufw enable
```

```
sudo ufw reload
```

Restart Postfix:

```
sudo systemctl restart postfix
```

Send mail from [project-x-email-svr] to [project-x-linux-client]:

```
echo “This is a test message.” | mail -s “Hello!” jane@linux-client
```

```
email-svr@smtp:~$ echo "This is a test message." | mail -s "Hello!" jane@linux-client
email-svr@smtp:~$ _
```

```
jane@linux-client:~/Maildir$ mail
"/home/jane/Maildir": 1 message 1 new
>N   1 email-svr          Tue Dec 17 00:37  16/682   Hello!
```

Detection Integration

[project-x-email-svr] does not have the Wazuh agent installed. This is intentional to demonstrate how the absence of detection controls can create a gap in identifying potentially malicious activity.

Enable WinRM on [project-x-win-client]

Log into [project-x-win-client], open a new Administrator Powershell session. Type the following commands to enable WinRM.

```
powershell -ep bypass

Enable-PSRemoting -force

winrm quickconfig -transport:https

Set-Item wsman:\localhost\client\trustedhosts *

net localgroup "Remote Management Users" /add administrator

Restart-Service WinRM
```

Detection Integration

An Event ID does not exist for Enabling Win-RM as a service. However, we can detect Win-RM logins through the Event ID 4624 with a “logonProcessName” of Kerberos as WinRM uses Kerberos.

Once we have enabled Security Windows logs (this should have been done in the **Setup Wazuh** section), we should automatically get Windows Event Logs.

Wazuh has a built-in rule detection to detect successful and unsuccessful authentication attempts into a Windows machine. The Windows Security Event IDs are 4624 (for successful) and 4624 (for unsuccessful).

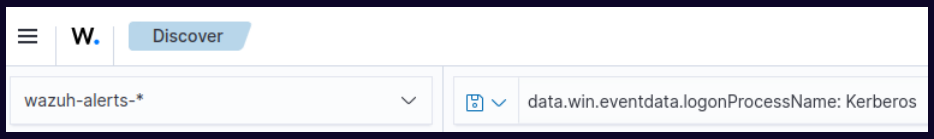
- Wazuh Rule ID: 60106
- Description: User: Windows Logon Success

Navigate to “Server management” → “Rules”. And look up “60106” to view more detail about this rule.

Here is a sample snapshot of a log generated when the WinRM attempt was successful.

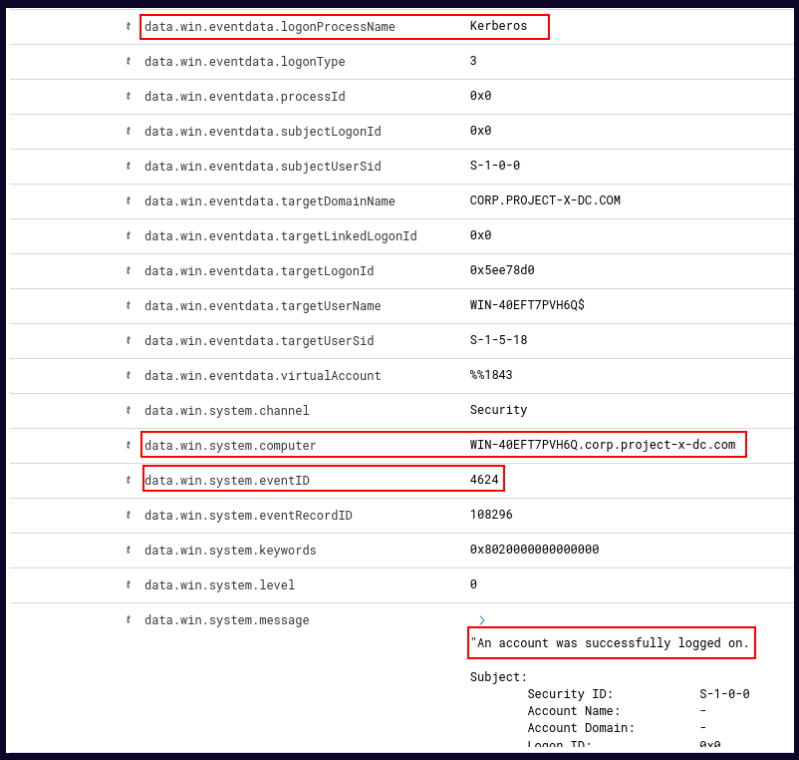
Go to “Explore” → “Discover” tab → Make sure “wazuh-alerts-*” is selected. Look up:

```
data.win.eventdata.logonProcessName: Kerberos
```



👉 You must have a login attempt with WinRM.

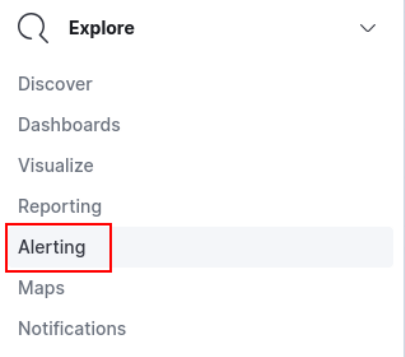
Here we can see the logonProcessName, the computer that was logged into, the EventID and the system message from a successful WinRM logon.



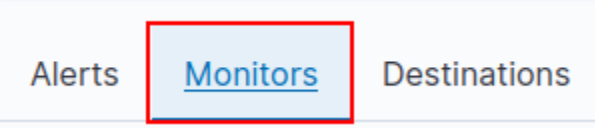
Create Detection Alert

Let’s create an alert for WinRM. We will do something similar to the Failed SSH attempts.

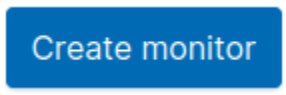
Go to “Explore” → “Alerting”.



Select the “Monitors” tab on the top left.

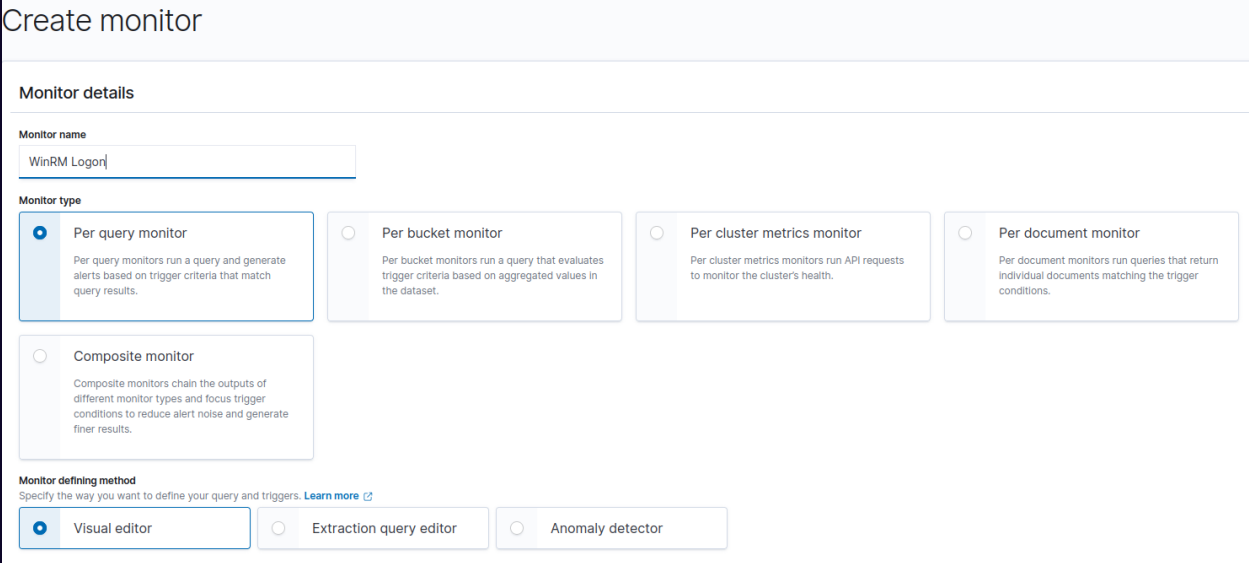


Select “Create monitor”.



Here we can create a new monitor.

Title the Monitor “WinRM Logon”. Leave everything else default.

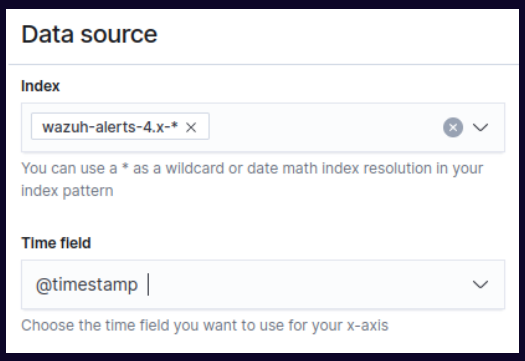


Scroll down to “Data source”. Add the following for the Index, hit the Enter key after typing:

```
wazuh-alerts-4.x-*
```

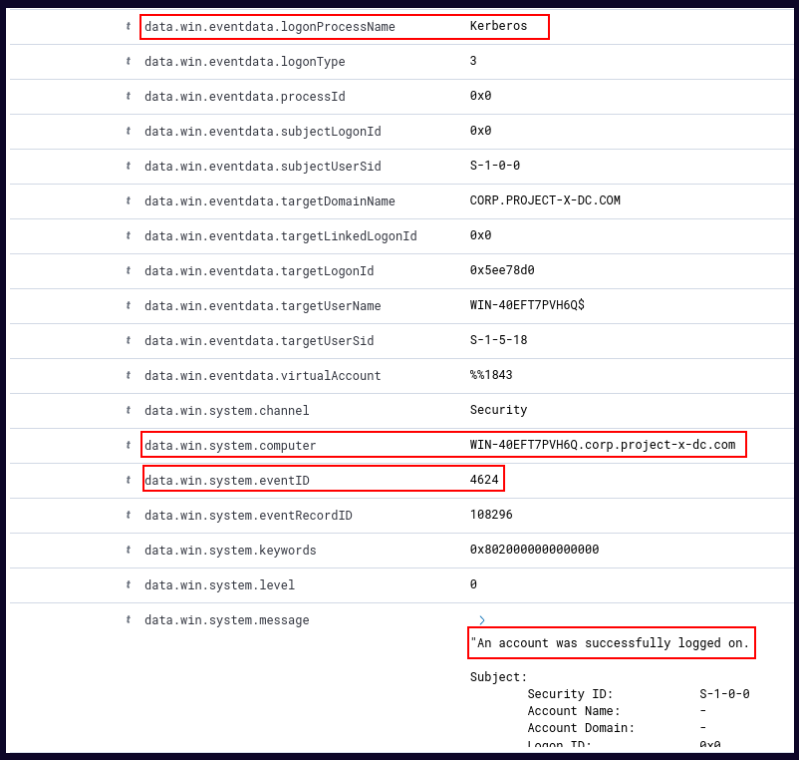
For “Time Field” select:

@timestamp



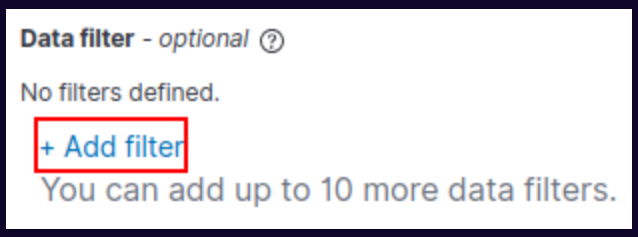
Next, we can add a query to select what logs and log fields we would like to monitor.

Based on a sample log of a WinRM attempt, we can construct a query to monitor specific field / value key-pairs.

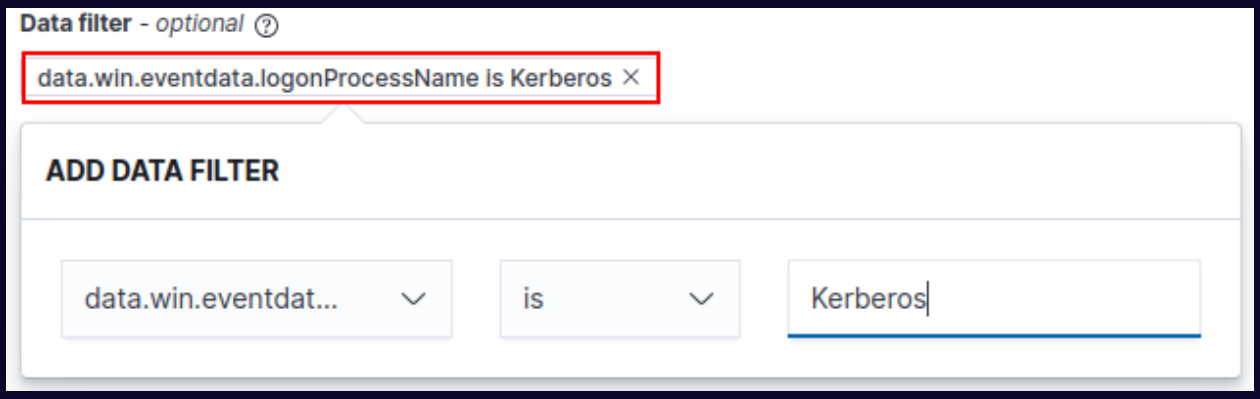


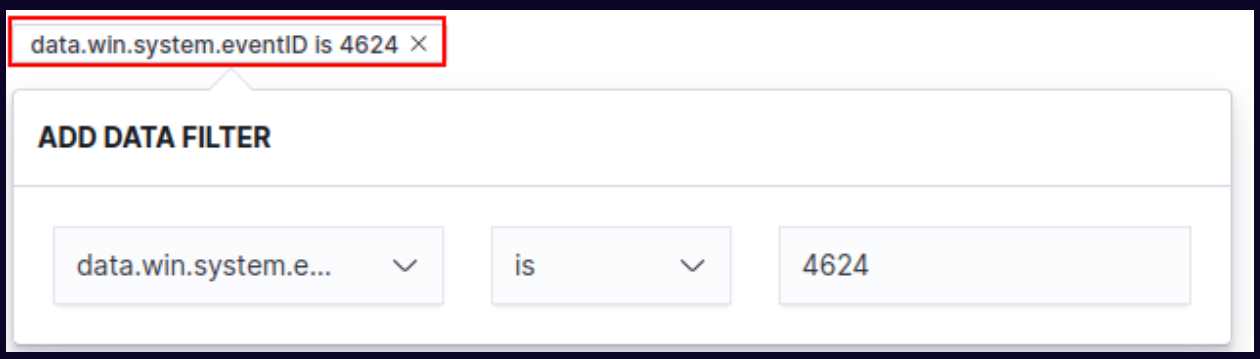
👉 A WinRM Logon attempt log.

Navigate to the “Data filter” → “+ Add filter”.

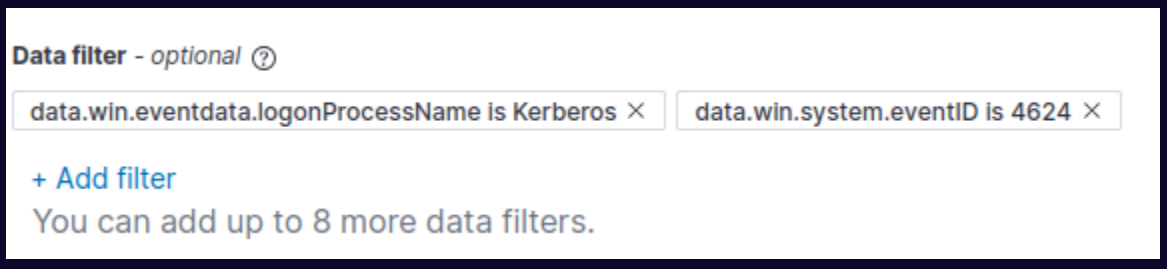


Based on the above sample log, let's craft a query to select based on the "logonProcessName" and the "eventID" fields.

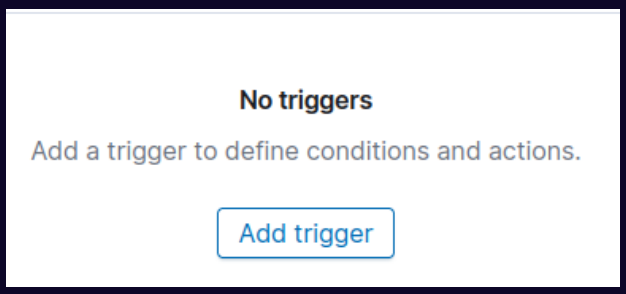




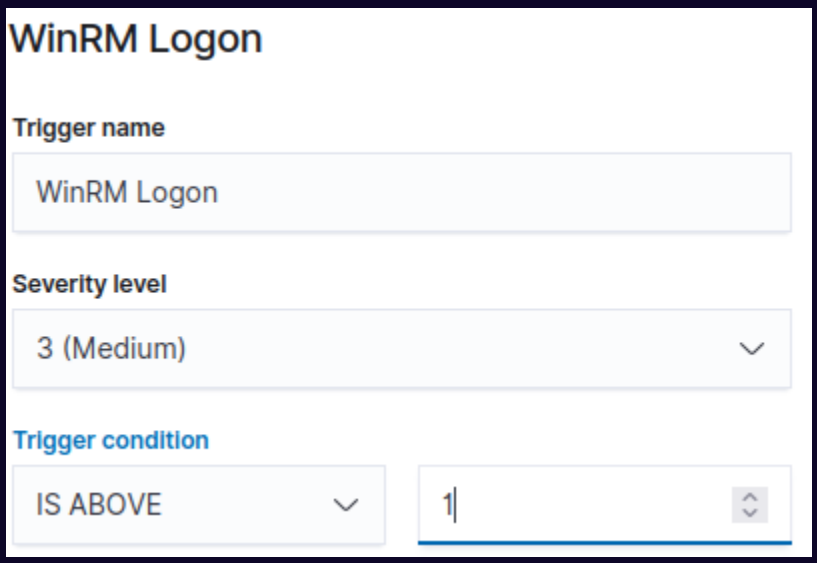
Your "Data filter" tab should now look something like this.



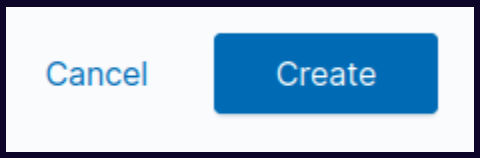
Let's add a Trigger.



Add the following conditions to the Trigger. We set the “Severity level” to 3 (Medium) and the “Trigger condition” above 1.

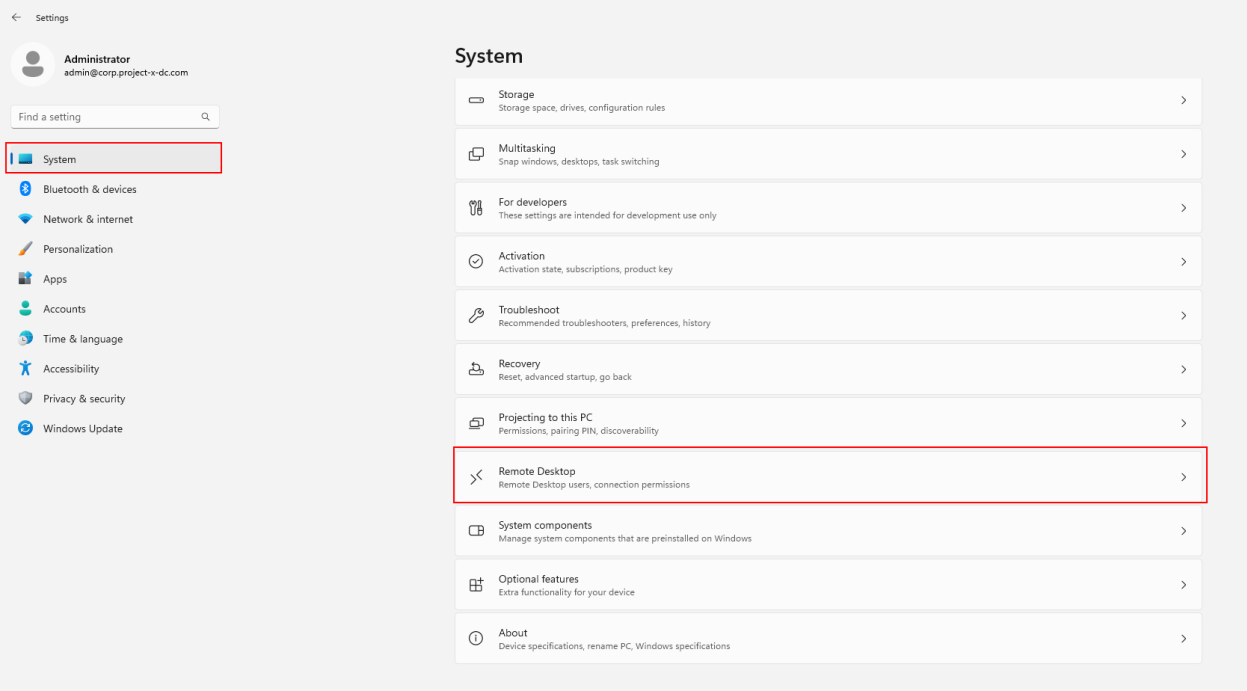


Scroll to the bottom and Select “Create”.

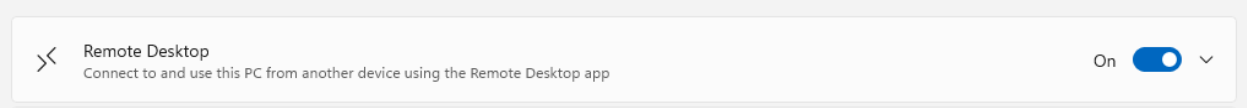


Enable RDP on [project-x-dc]

Go to “Settings” → “System” → “Remote Desktop”.



Toggle Remote Desktop to “On”:



Detection Integration

Wazuh has a built-in rule detection to detect successful and unsuccessful authentication attempts into a Windows machine. The Windows Security Event IDs are 4624 (for successful) and 4624 (for unsuccessful).

- Wazuh Rule ID: 92653
- Description: User: CORP\Administrator logged using Remote Desktop Connection (RDP) from ip:10.0.0.100.

Navigate to “Server management” → “Rules”. And look up “92653” to view more detail about this rule.

Here is a sample snapshot of a log generated when the RDP attempt was successful.

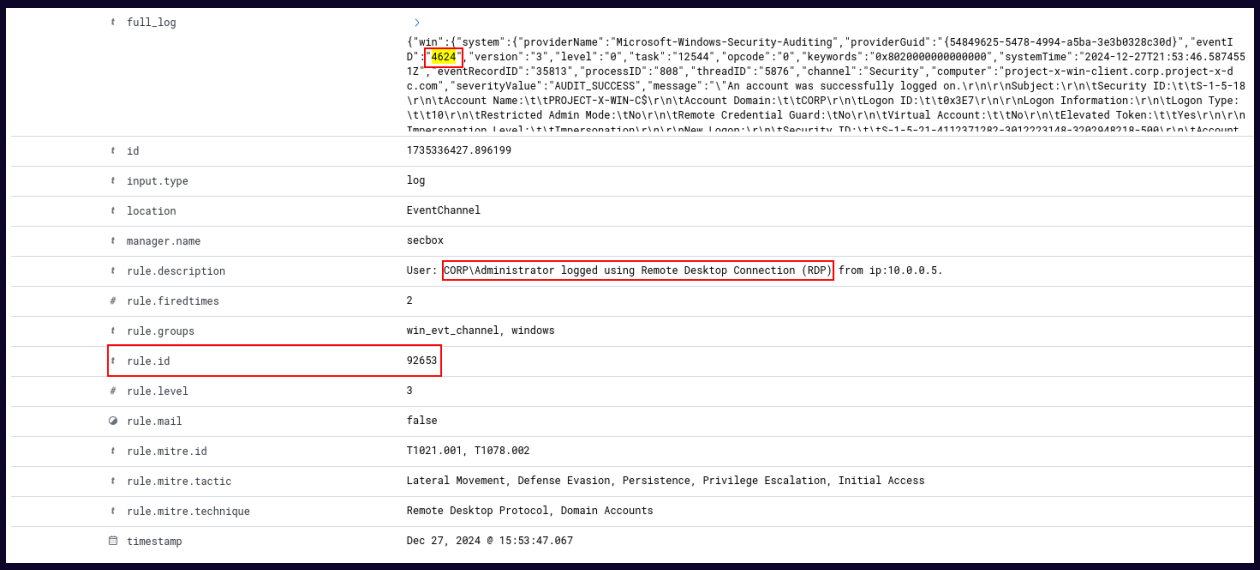
Go to “Explore” → “Discover” tab. Look up “4624” or:

```
data.win.system.eventID: 4624 AND
data.win.eventdata.logonProcessName: User32
```

👉 You must have a login attempt with RDP.

💡 The ***data.win.event.logonProcessName: User32*** denotes what process was used. User32.dll is used as part of the RDP process. How did we know this? We can generate sample logs by logging in over RDP with intentional successful and unsuccessful attempts. From here, the sample logs can be analyzed, and fields + values can be populated. Using your LLM of choice or Google, we can identify why these field values populated.

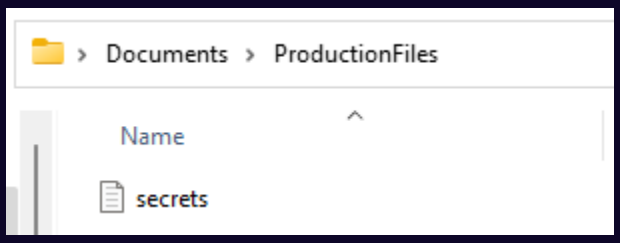
Here we can see in the `rule.description`, that a successful logon attempt has been made.



Setup “Sensitive File” [project-x-dc]

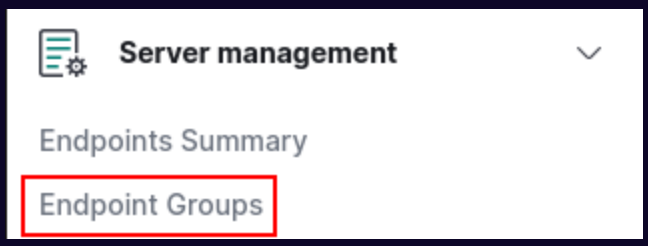
Log into [project-x-dc] go to `C:\Users\Administrator\Documents` → Right-click → New Folder → Name it “`ProductionFiles`”.

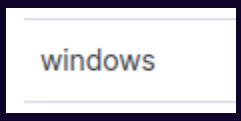
Navigate inside the folder → Right-click → “New” → “Text File” → Name the file “secrets”. Add whatever content you would like. For this example, I added Deeboodah!.



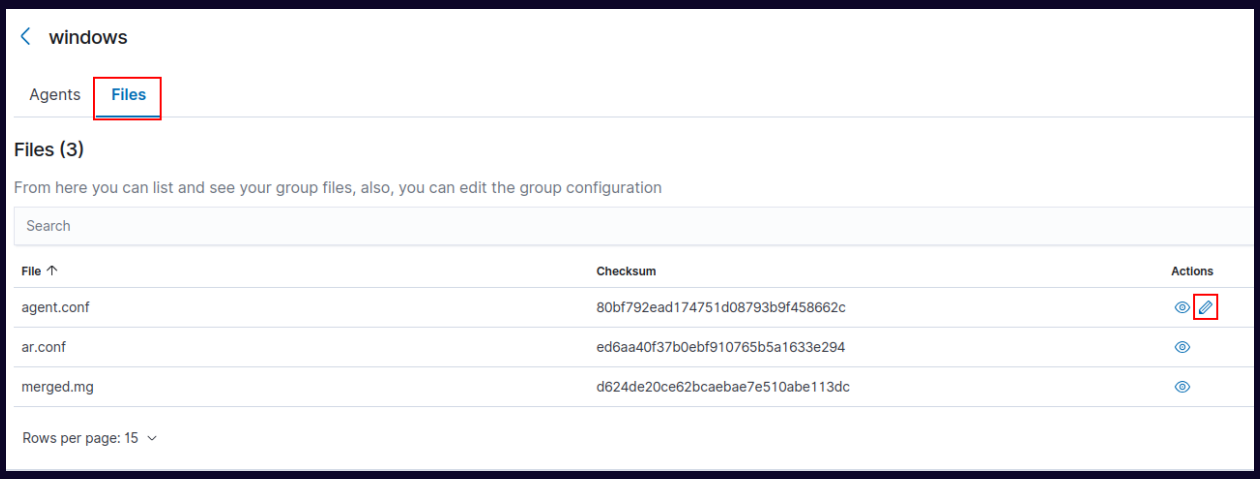
Detection Integration

Navigate to “Server management” → “Endpoint Groups” → Select the “Windows” Agent Group.





Select the “Files” tab → agent.conf → Select the Pencil Icon to edit the file.



Add the following statement at the end of the file.

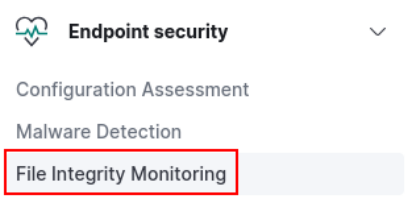
```
<agent_config>
  <!-- Shared agent configuration here -->
  <localfile>
    <location>Security</location>
    <log_format>eventchannel</log_format>
  </localfile>
  <localfile>
    <location>Application</location>
    <log_format>eventchannel</log_format>
  </localfile>
  <localfile>
    <location>Microsoft-Windows-PowerShell/Operational</location>
    <log_format>eventchannel</log_format>
  </localfile>
  <syscheck>
    <directories check_all="yes" report_changes="yes" realtime="yes">C:\Users\Administrator\Documents\ProductionFiles</directories>
    <frequency>60</frequency>
  </syscheck>
</agent_config>
```

```
<syscheck>

  <directories check_all="yes" report_changes="yes"
  realtime="yes">C:\Users\Administrator\Documents\ProductionFiles</
  directories>

  <frequency>60</frequency>
```

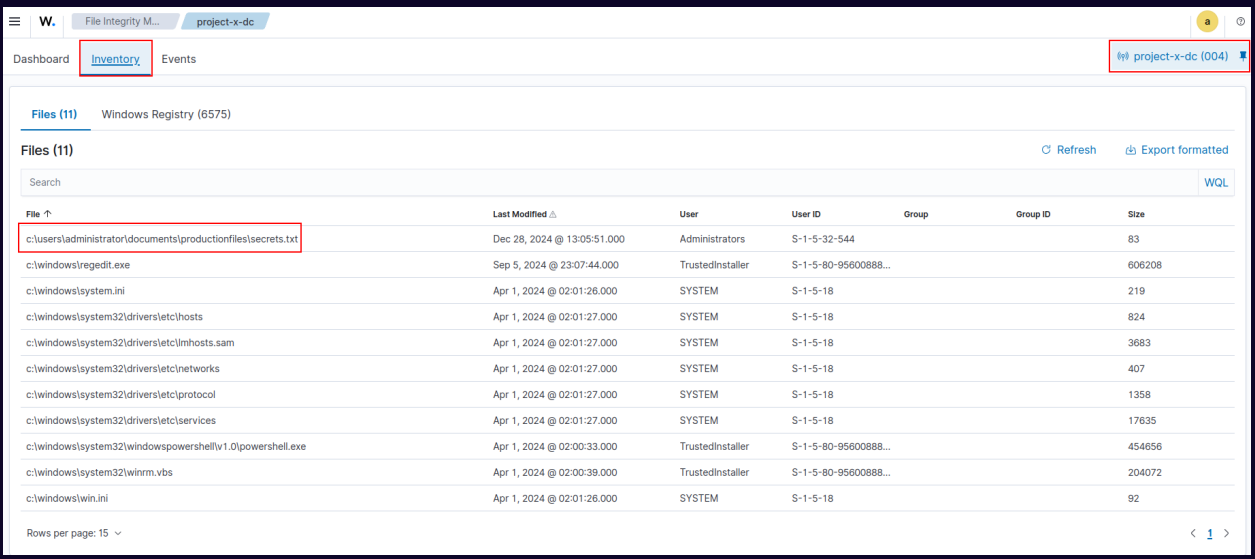
```
</syscheck>
```

💡 `<directories check_all="yes" report_changes="yes" realtime="yes">` defines what directory location to monitor. `check_all` defines the FIM module to scan all files in the specified directory. `report_changes` enables the system to report content changes made to a file. `<frequency>` defines how often the FIM module scans in seconds. The default is every 12 hours.

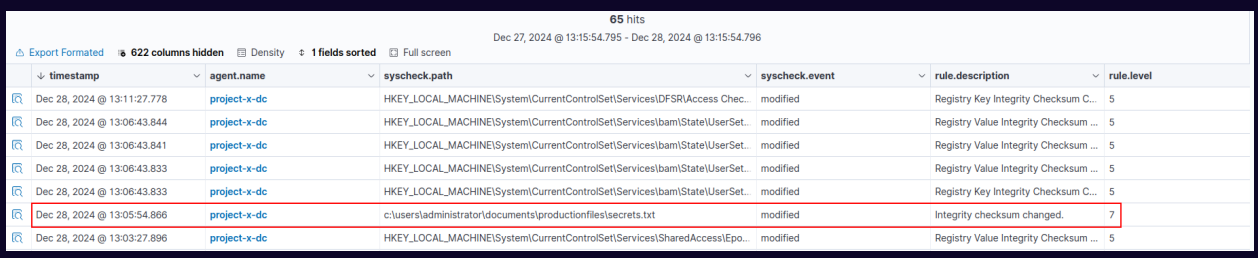
Navigate to the “Endpoint security” → “File Integrity Monitoring”.



Under Inventory there should be a new file path populated. Make sure the [project-x-dc agent] is selected on the right.



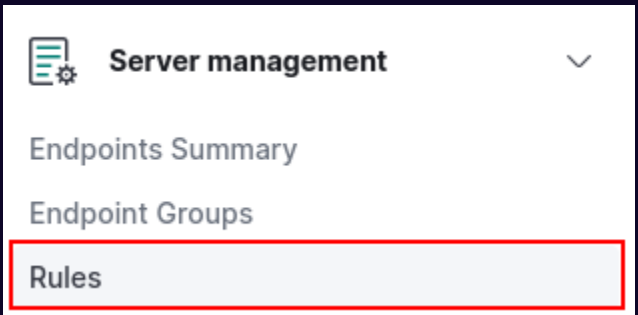
Navigating to the Events tab, if we were to change the content inside the *secrets.txt* file, we would have an event populate showcasing that the file has been modified.



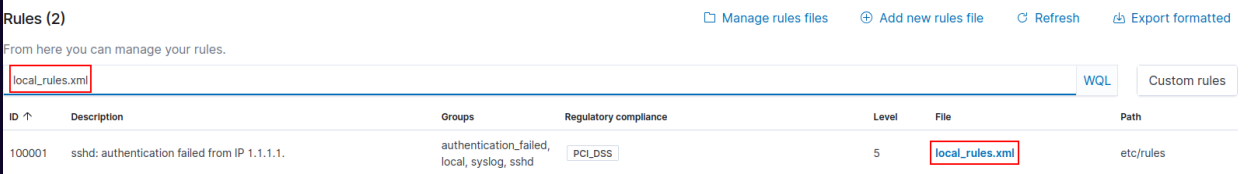
Create Detection Alert

Let's create an alert for accessing the *secrets.txt* file. This time, we are going to use the local_rules.xml file to define a custom rule to monitor for changes occurring.

First go to "Server management" → "Rules".



Search "local_rules.xml" → Click the name under the File tab.



Leave the default rule.

Add the following statement to the bottom of the file.

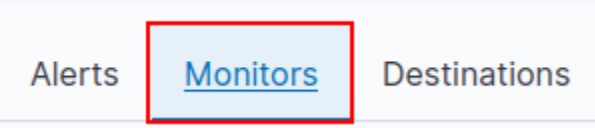
```
<group name="syscheck">
  <rule id="100002" level="10">
    <field name="file">secrets.txt</field>
    <match>modified</match>
    <description>File integrity monitoring alert - access to sensitive.txt file detected</description>
  </rule>
</group>
```

Click "Save" → "Restart".

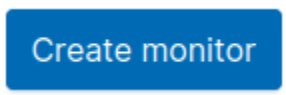
After the console restarts.

Go to “Explore” → “Alerting”.

Select the “Monitors” tab on the top left.

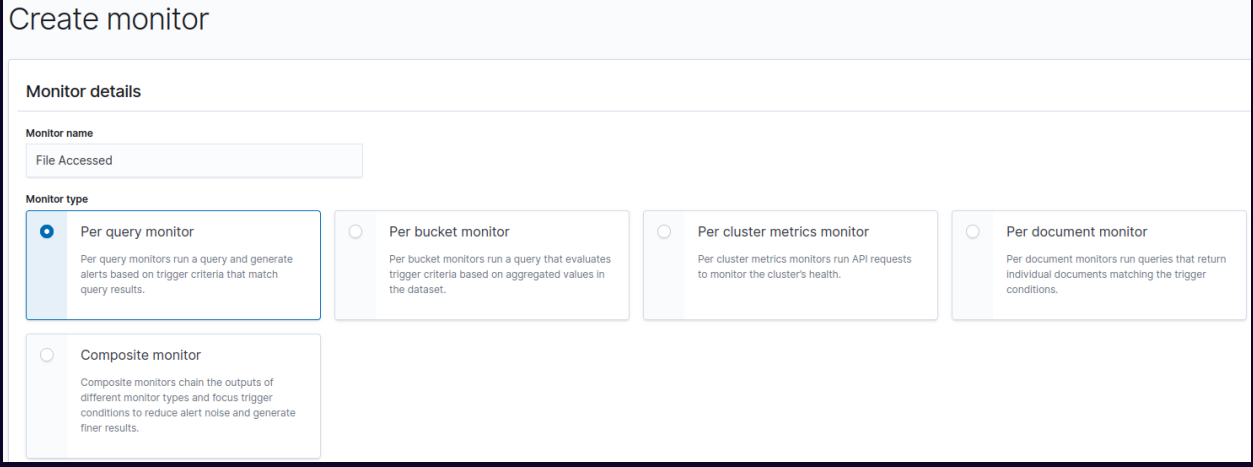


Select “Create monitor”.



Here we can create a new monitor.

Title the Monitor “File Accssed”. Leave everything else default.

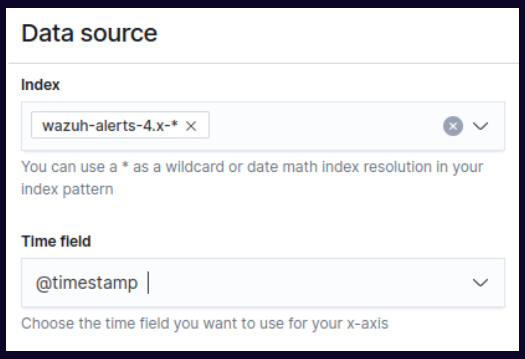


Scroll down to “Data source”. Add the following for the Index, hit the Enter key after typing:

```
wazuh-alerts-4.x-*
```

For “Time Field” select:

```
@timestamp
```



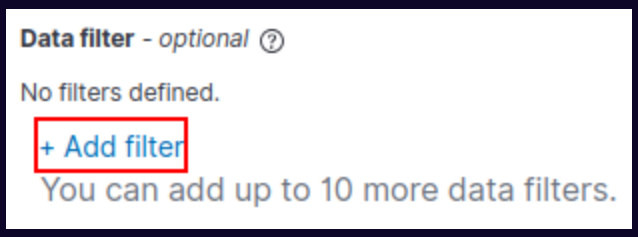
Next, we can add a query to select what logs and log fields we would like to monitor.

Based on a sample log of a modified file attempt, we can construct a query to monitor specific field / value key-pairs.

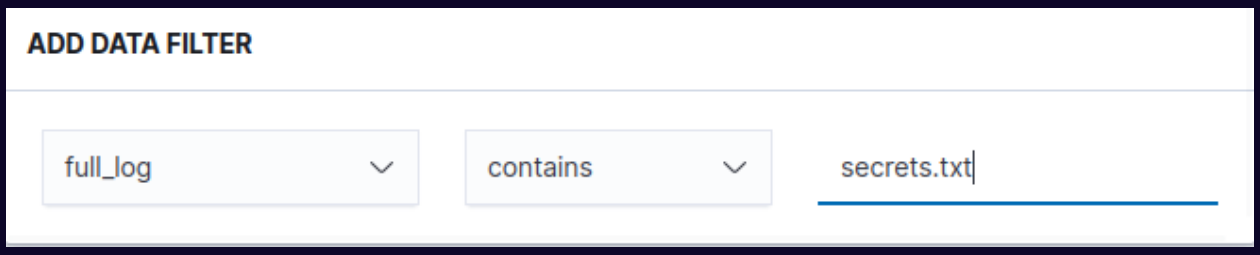
t	full_log	File 'c:\users\administrator\documents\productionfiles\secrets.txt' modified Mode: realtime Changed attributes: size,mtime,md5,sha1,sha256 Size changed from '88' to '83' Old modification time was: '1735602097', now it is '1735603790' Old md5sum was: '95a96ec14d473a6a3f42e20c12305d03' New md5sum is : '443cb90889ad3e9c521fa74cb32e9a5c'
t	id	1735604014.17034950
t	input.type	log
t	location	syscheck
t	manager.name	secbox
t	rule.description	Integrity checksum changed.
#	rule.firedtimes	1
t	rule.gdpr	II_5.1.f
t	rule.gpg13	4.11
t	rule.groups	ossec, syscheck, syscheck_entry_modified, syscheck_file
t	rule.hipaa	164.312.c.1, 164.312.c.2
t	rule.id	550
#	rule.level	7
◔	rule.mail	false
t	rule.mitre.id	T1565.001
t	rule.mitre.tactic	Impact
t	rule.mitre.technique	Stored Data Manipulation

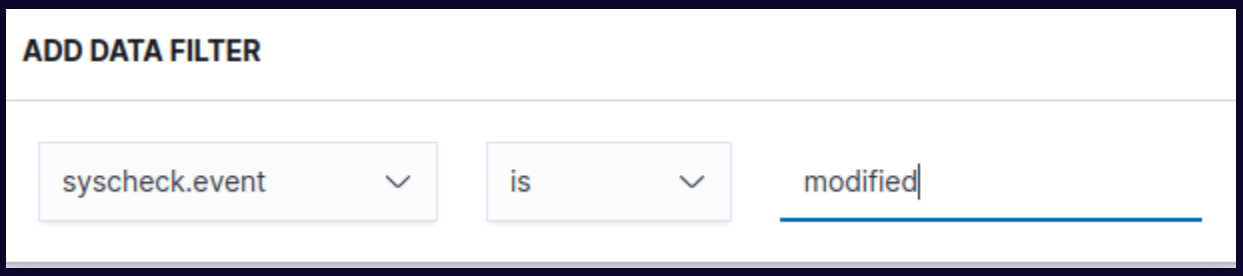
👉 secrets.txt file modified sample log.

Navigate to the “Data filter” → “+ Add filter”.

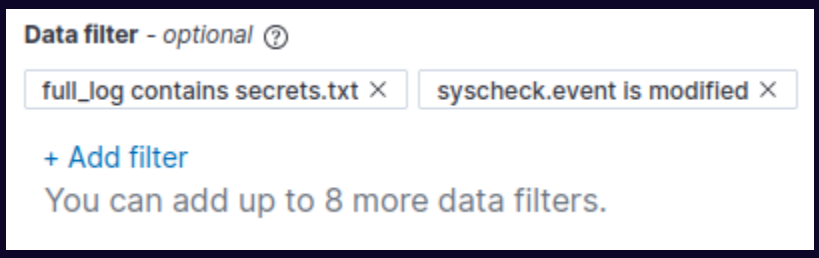


Based on the above sample log, let's craft a query to select based on the "full_log" field containing the secrets.txt file and the "syscheck.event" fields to modified.

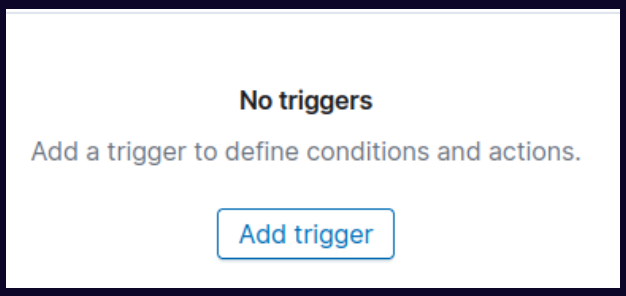




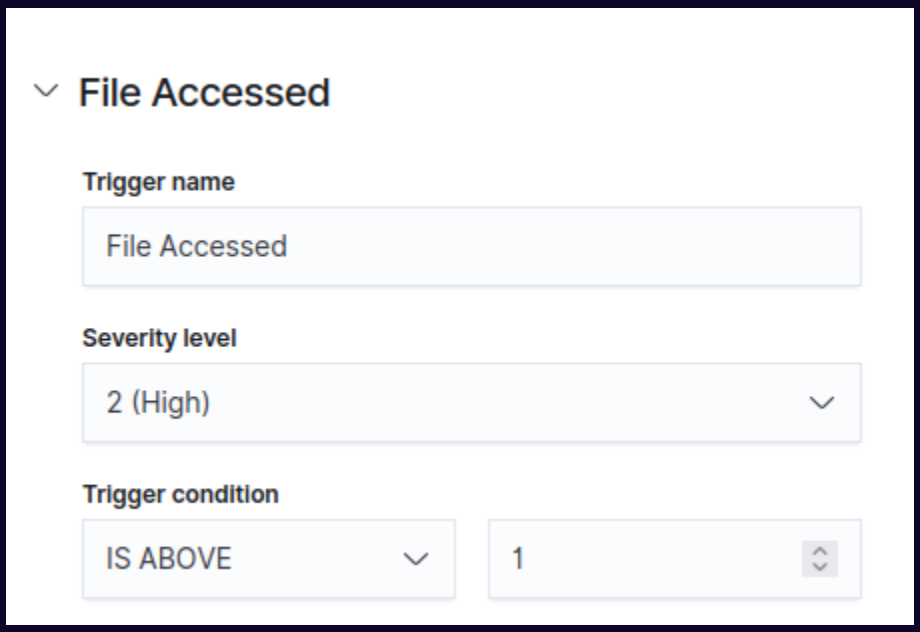
Your "Data filter" tab should now look something like this.



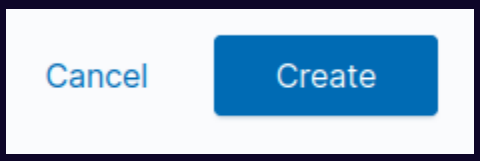
Let's add a Trigger.



Add the following conditions to the Trigger. We set the "Severity level" to 2 (High) and the "Trigger condition" above 1.



Scroll to the bottom and Select “Create”.



Exfiltration to [project-x-attacker]

The scp (Secure Copy) command-line utility allows you to copy files and directories between two systems over the SSH protocol. This tool will be used to exfiltrate the secrets.txt file to our [project-x-attacker] machine.

Enable SSH on Kali Machine:

```
sudo systemctl start ssh.service
```

Create a new file under the Kali Machine, this is where we will copy our *secrets.txt* file to:

```
touch /home/attacker/my_exfil.txt
```

Detection Integration

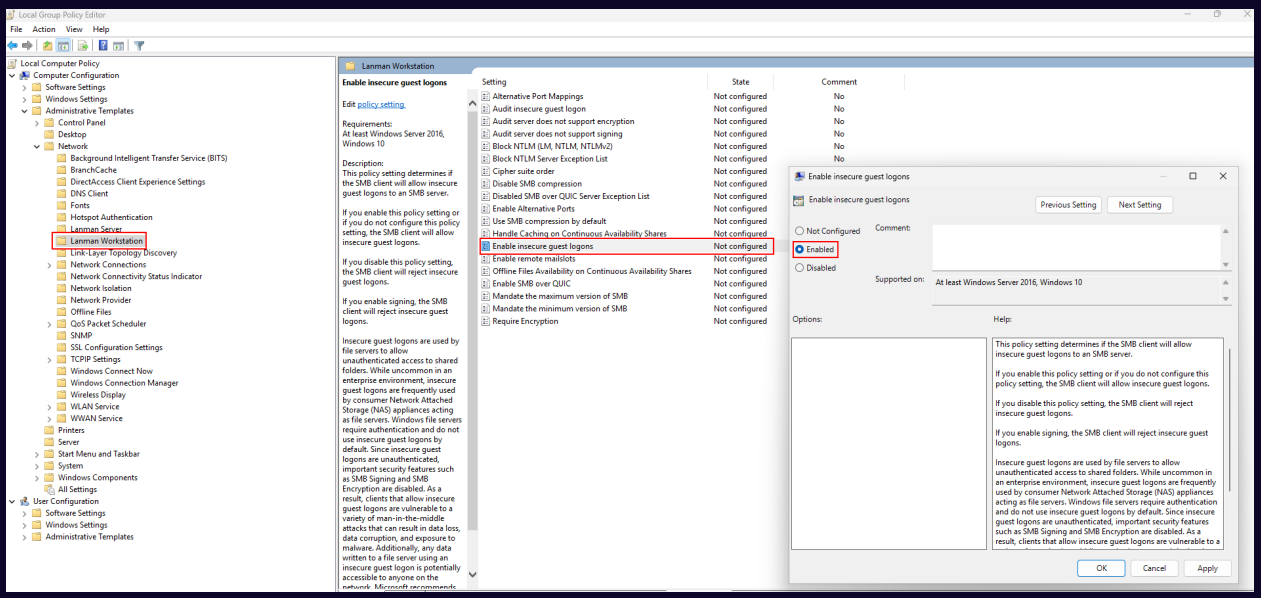
The above detection will be relied upon to detect changes to the *secrets.txt* file.

Next, open a new File Explorer Window → Go to C:\Windows\System32.

Scroll down until you find “gpedit” → Right-click → Run as Administrator.

💡 We can’t look up Local Group Policy Editor and run as Administrator on a client device managed by Active Directory. This is why we must look for the gpedit program in ...\System32.

Go to Computer Configuration → Administrative Template → Network → Lanman Workstation → Double-click on “Enable insecure guest logons” → Select “Enabled”.



Go back to Powershell:

```
Set-ItemProperty -Path
"HKLM:\SYSTEM\CurrentControlSet\Services\LanmanWorkstation\Parame
ters" -Name AllowInsecureGuestAuth -Value 1 -Force
```