



## Configure a Vulnerable Environment

>pr0jectsecurity\_

# Table of Contents

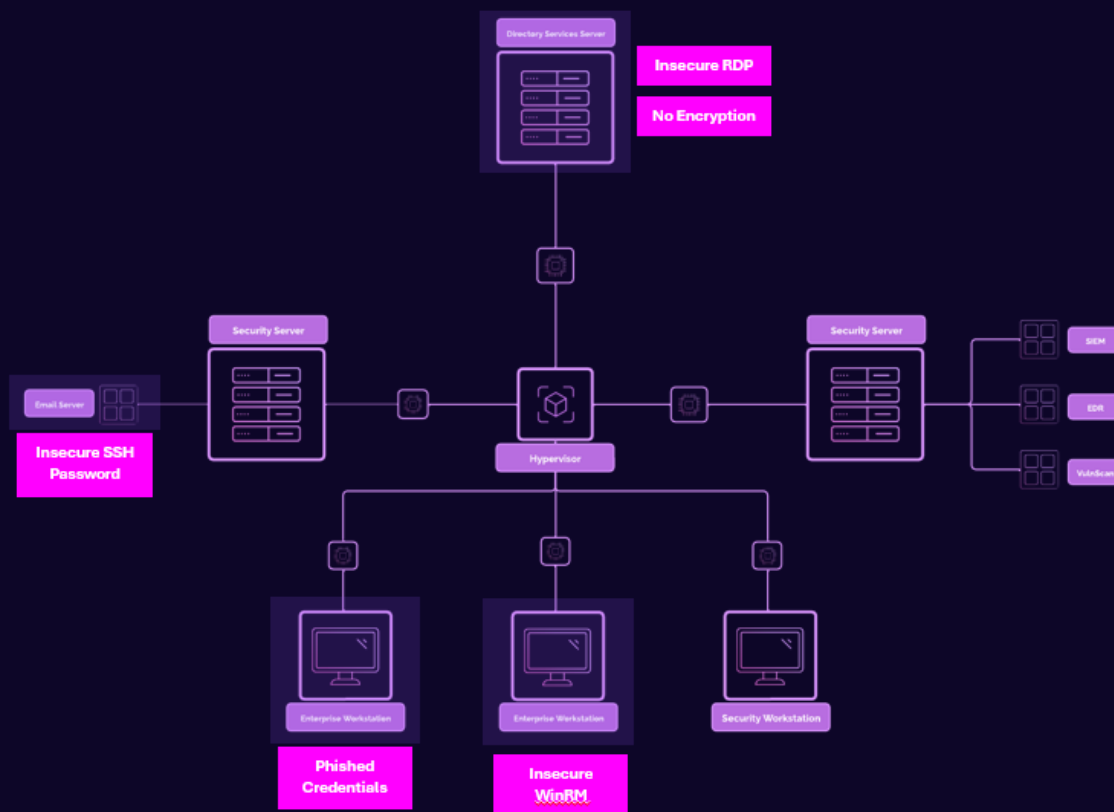
---

Table of Contents .....	2
Prerequisites .....	3
Network Topology .....	3
Vulnerable Environment .....	3
Overview.....	3
Open SSH on [project-x-email-svr] .....	4
Detection Integration.....	5
Open SSH on [project-x-linux-client] .....	5
Detection Integration.....	7
Create Detection Alert .....	8
Configure Email Connection from [project-x-email-svr] to [project-x-linux-client] .....	12
Detection Integration.....	15
Enable WinRM on [project-x-win-client].....	15
Detection Integration.....	15
Create Detection Alert .....	16
Enable RDP on [project-x-dc] .....	20
Detection Integration.....	21
Setup “Sensitive File” [project-x-dc] .....	22
Detection Integration.....	22
Create Detection Alert .....	25
Exfiltration to [project-x-attacker].....	29
Detection Integration.....	29

## 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
```

```
email-svr@smtp:~$ sudo ufw status  
Status: active  
  
To Action From  
--  
Postfix ALLOW Anywhere  
25 ALLOW Anywhere  
587 ALLOW Anywhere  
143 ALLOW Anywhere  
993 ALLOW Anywhere  
22 ALLOW Anywhere  
Postfix (v6) ALLOW Anywhere (v6)  
25 (v6) ALLOW Anywhere (v6)  
587 (v6) ALLOW Anywhere (v6)  
143 (v6) ALLOW Anywhere (v6)  
993 (v6) ALLOW Anywhere (v6)  
22 (v6) ALLOW Anywhere (v6)
```

Verify SSH is running:

```
sudo systemctl status ssh
```

Enable Password Authentication. Open the SSH configuration file:

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

```
email-svr@smtp:~$ sudo systemctl status ssh
• ssh.service - OpenBSD Secure Shell server
  Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
  Active: active (running) since Sat 2024-12-14 03:05:48 UTC; 8min ago
```

Locate the line for PasswordAuthentication. Uncomment if commented.

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

```
# To disable tunneled clear text communications
#PasswordAuthentication yes
```

```
PasswordAuthentication yes
```

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

```
#PermitRootLogin prohibit-password
```

```
PermitRootLogin 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
```

```
email-svr@smtp:~$ sudo ufw status
Status: active

To Action From
--
Postfix ALLOW Anywhere
25 ALLOW Anywhere
587 ALLOW Anywhere
143 ALLOW Anywhere
993 ALLOW Anywhere
22 ALLOW Anywhere
Postfix (v6) ALLOW Anywhere (v6)
25 (v6) ALLOW Anywhere (v6)
587 (v6) ALLOW Anywhere (v6)
143 (v6) ALLOW Anywhere (v6)
993 (v6) ALLOW Anywhere (v6)
22 (v6) ALLOW Anywhere (v6)
```

Verify SSH is running:

```
sudo systemctl status ssh
```

Enable Password Authentication. Open the SSH configuration file:

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

```
email-svr@smtp:~$ sudo systemctl status ssh
• ssh.service - OpenBSD Secure Shell server
  Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
  Active: active (running) since Sat 2024-12-14 03:05:48 UTC; 8min ago
```

Locate the line for PasswordAuthentication. Uncomment if commented.

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

```
# To disable tunneled clear
#PasswordAuthentication yes

PasswordAuthentication yes
```

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

```
#PermitRootLogin prohibit-password

PermitRootLogin yes_
```

Restart SSH service:

```
sudo systemctl restart ssh
```

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

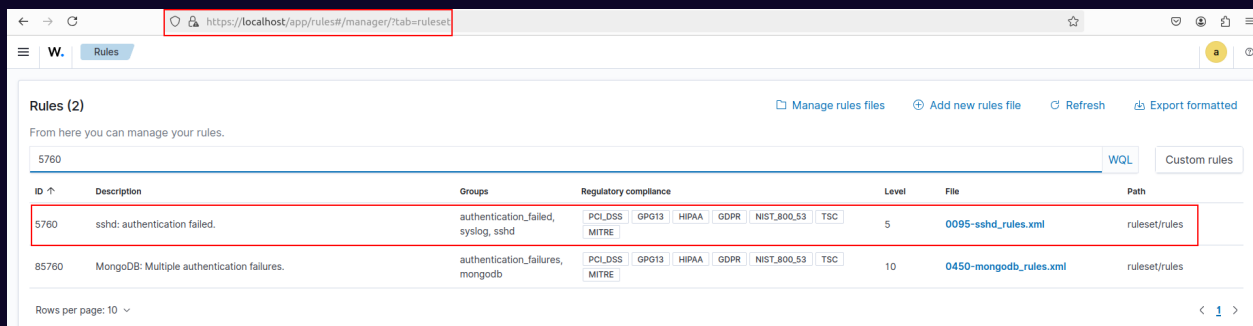
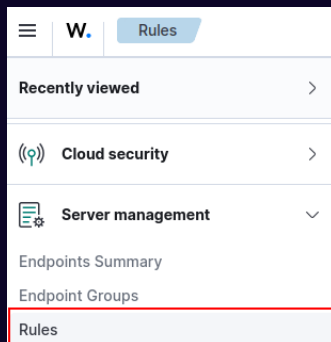
```
sudo passwd root
```

## 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”.

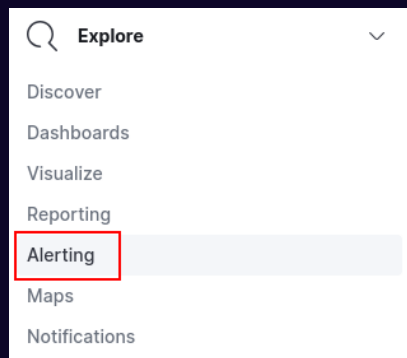
👉 You must have a failed login attempt.

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

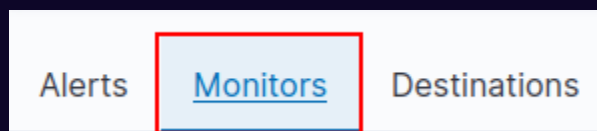
## Create Detection Alert

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".



Create monitor

Here we can create a new monitor.

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

**Monitor details**

**Monitor name**

3 Failed SSH Attempts

**Monitor type**

☒ **Per query monitor**  
Per query monitors run a query and generate alerts based on trigger criteria that match query results.

☐ **Per bucket monitor**  
Per bucket monitors run a query that evaluates trigger criteria based on aggregated values in the dataset.

☐ **Per cluster monitor**  
Per cluster monitors run a query that evaluates trigger criteria based on aggregated values in the dataset.

☐ **Composite monitor**  
Composite monitors chain the outputs of different monitor types and focus trigger conditions to reduce alert noise and generate finer results.

**Monitor defining method**  
Specify the way you want to define your query and triggers. [Learn more](#)

☒ **Visual editor** ☐ **Extraction query editor** ☐ **Anomaly detector**

**Schedule**

**Frequency**

By interval

**Run every**

1 Minute(s)

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

**Data source**

**Index**

wazuh-alerts-4.x-\*

You can use a \* as a wildcard or date math index resolution in your index pattern

**Time field**

@timestamp

Choose the time field you want to use for your x-axis

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

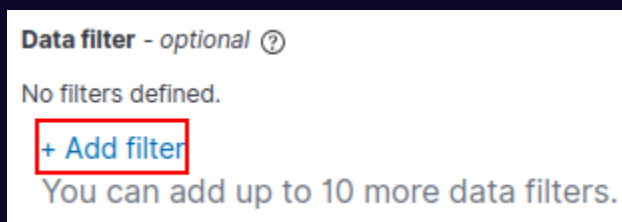
Based on a sample log of a failed ssh attempt, we can construct a query to monitor specific field / value key-pairs.

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.

**ADD DATA FILTER**

decoder.name

is

sshd

**ADD DATA FILTER**

rule.groups

contains

authentication\_failed

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

**Time range for the last** ?  

1

hour(s)

**Data filter - optional** ?  

decoder.name is sshd

rule.groups contains authentication\_failed

+ Add filter

You can add up to 8 more data filters.

Let’s add a Trigger.

**No triggers**  
Add a trigger to define conditions and actions.  

Add 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.

**3 Failed SSH Attempts**

**Trigger name**  

3 Failed SSH Attempts

**Severity level**  

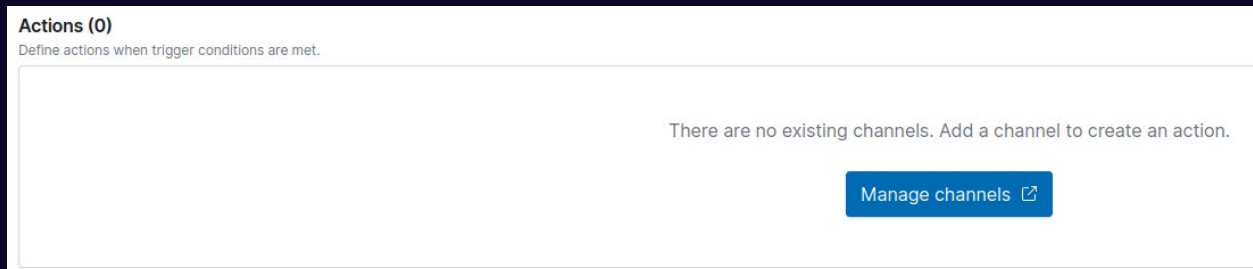
3 (Medium)

**Trigger condition**  

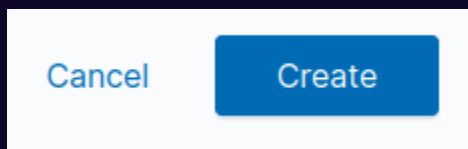
IS ABOVE

2

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.

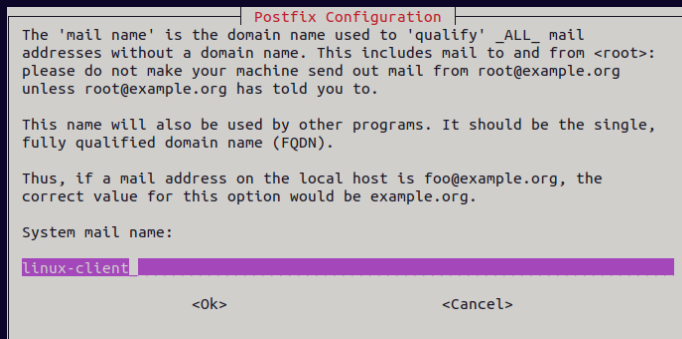
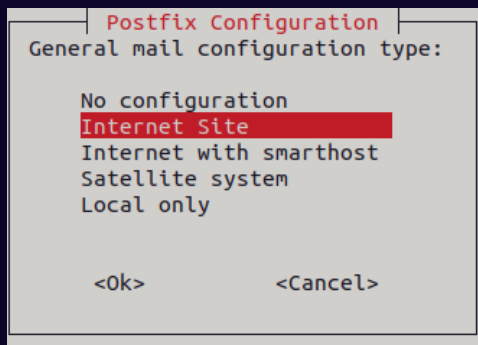
Let's setup and configure postfix on [project-x-linux-client] so we can send emails to the email server, leveraging Postfix again.

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 4625 (for unsuccessful).

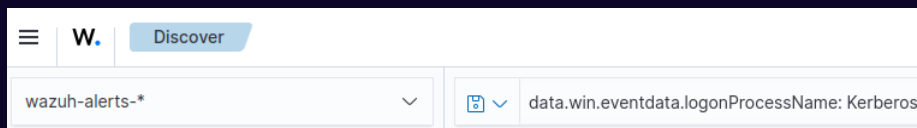
- Wazuh Rule ID: 60106
- Description: User: Windows Logon Success

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 login.

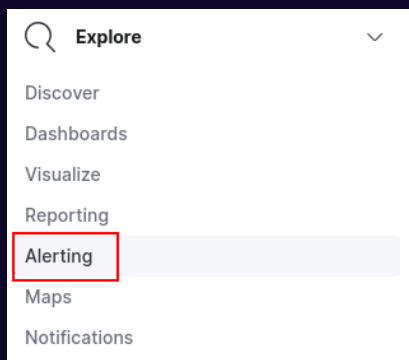
#	data.win.eventdata.logonProcessName	Kerberos
#	data.win.eventdata.logonType	3
#	data.win.eventdata.processId	0x0
#	data.win.eventdata.subjectLogonId	0x0
#	data.win.eventdata.subjectUserSid	S-1-0-0
#	data.win.eventdata.targetDomainName	CORP.PROJECT-X-DC.COM
#	data.win.eventdata.targetLinkedLogonId	0x0
#	data.win.eventdata.targetLogonId	0x5ee78d0
#	data.win.eventdata.targetUserName	WIN-40EF77PVH6Q\$
#	data.win.eventdata.targetUserSid	S-1-5-18
#	data.win.eventdata.virtualAccount	%1843
#	data.win.system.channel	Security
#	data.win.system.computer	WIN-40EF77PVH6Q.corp.project-x-dc.com
#	data.win.system.eventID	4624
#	data.win.system.eventRecordID	108296
#	data.win.system.keywords	0x8020000000000000
#	data.win.system.level	0
#	data.win.system.message	> "An account was successfully logged on." Subject: Security ID: S-1-0-0 Account Name: - Account Domain: - Login ID: 0x0

## Create Detection Alert

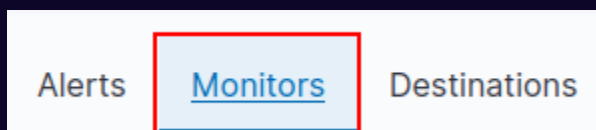
Let's create an alert for WinRM. We will do something similar to the Failed SSH attempts.

Go to “Explore” → “Alerting”.

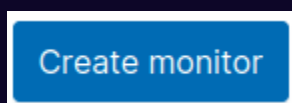




Select the “Monitors” tab on the top left.



Select “Create monitor”.



Here we can create a new monitor.

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

A screenshot of the 'Create monitor' form. The form has a title 'Create monitor' and a section 'Monitor details'. Under 'Monitor details', there is a 'Monitor name' field with the text 'WinRM Logon'. Below that is the 'Monitor type' section, which contains five options: 'Per query monitor' (selected), 'Per bucket monitor', 'Per cluster metrics monitor', 'Per document monitor', and 'Composite monitor'. Each option has a brief description. At the bottom is the 'Monitor defining method' section, which contains three options: 'Visual editor' (selected), 'Extraction query editor', and 'Anomaly detector'.

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

**Data source**

**Index**

wazuh-alerts-4.x-\* ✕

You can use a \* as a wildcard or date math index resolution in your index pattern

**Time field**

@timestamp |

Choose the time field you want to use for your x-axis

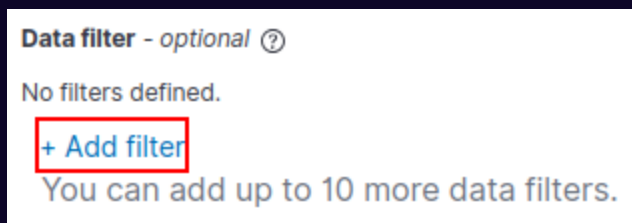
Next, we can add a query to select what logs and log fields we would like to monitor.

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

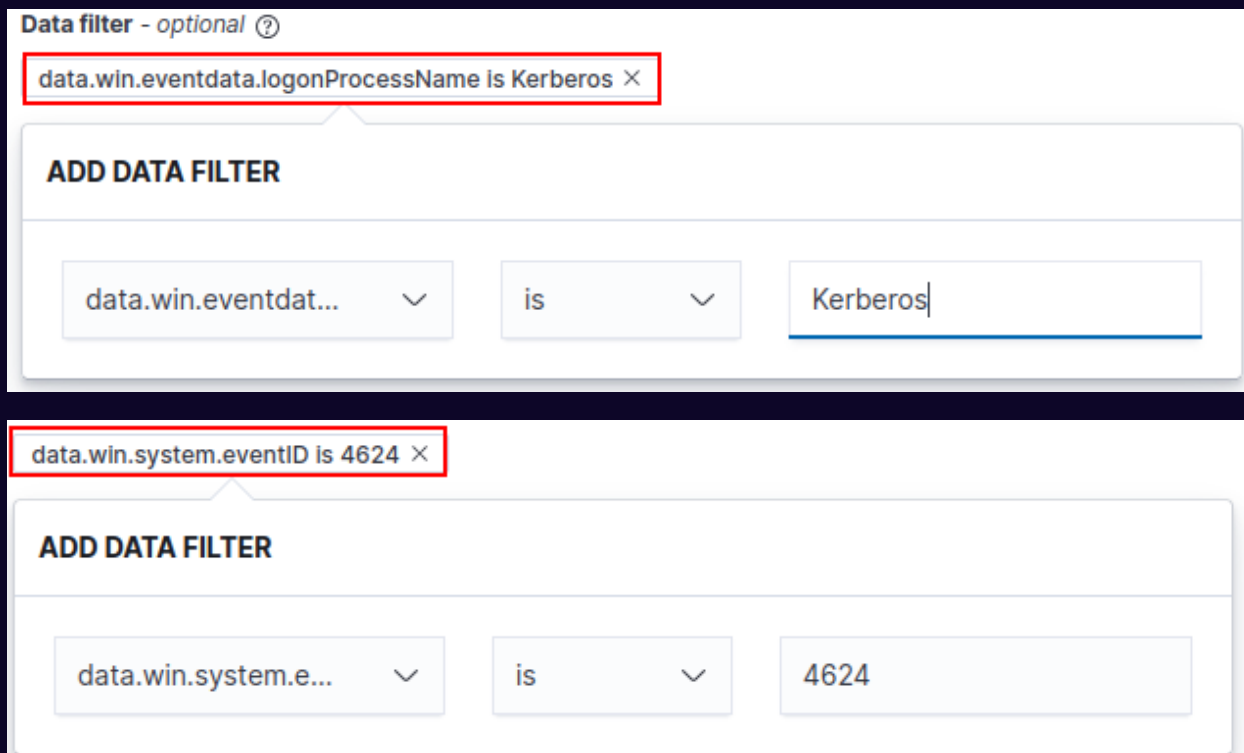
#	data.win.eventdata.logonProcessName	Kerberos
#	data.win.eventdata.logonType	3
#	data.win.eventdata.processId	0x0
#	data.win.eventdata.subjectLogonId	0x0
#	data.win.eventdata.subjectUserSid	S-1-0-0
#	data.win.eventdata.targetDomainName	CORP.PROJECT-X-DC.COM
#	data.win.eventdata.targetLinkedLogonId	0x0
#	data.win.eventdata.targetLogonId	0x5ee78d0
#	data.win.eventdata.targetUserName	WIN-40EFT7PVH6Q\$
#	data.win.eventdata.targetUserSid	S-1-5-18
#	data.win.eventdata.virtualAccount	%1843
#	data.win.system.channel	Security
#	data.win.system.computer	WIN-40EFT7PVH6Q.corp.project-x-dc.com
#	data.win.system.eventID	4624
#	data.win.system.eventRecordID	108296
#	data.win.system.keywords	0x8020000000000000
#	data.win.system.level	0
#	data.win.system.message	> "An account was successfully logged on."
	Subject:	
	Security ID:	S-1-0-0
	Account Name:	-
	Account Domain:	-
	Logon ID:	0x0

👉 A WinRM Logon attempt log.

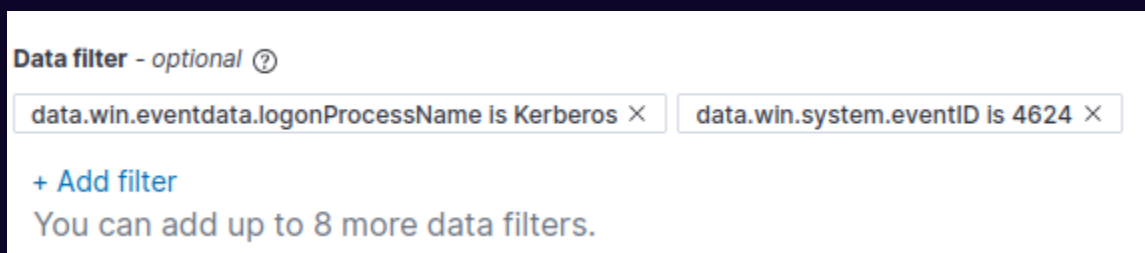
Navigate to the “Data filter” → “+ Add filter”.



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.

No triggers

Add a trigger to define conditions and actions.

Add trigger

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

WinRM Logon

Trigger name

WinRM Logon

Severity level

3 (Medium)

Trigger condition

IS ABOVE

1

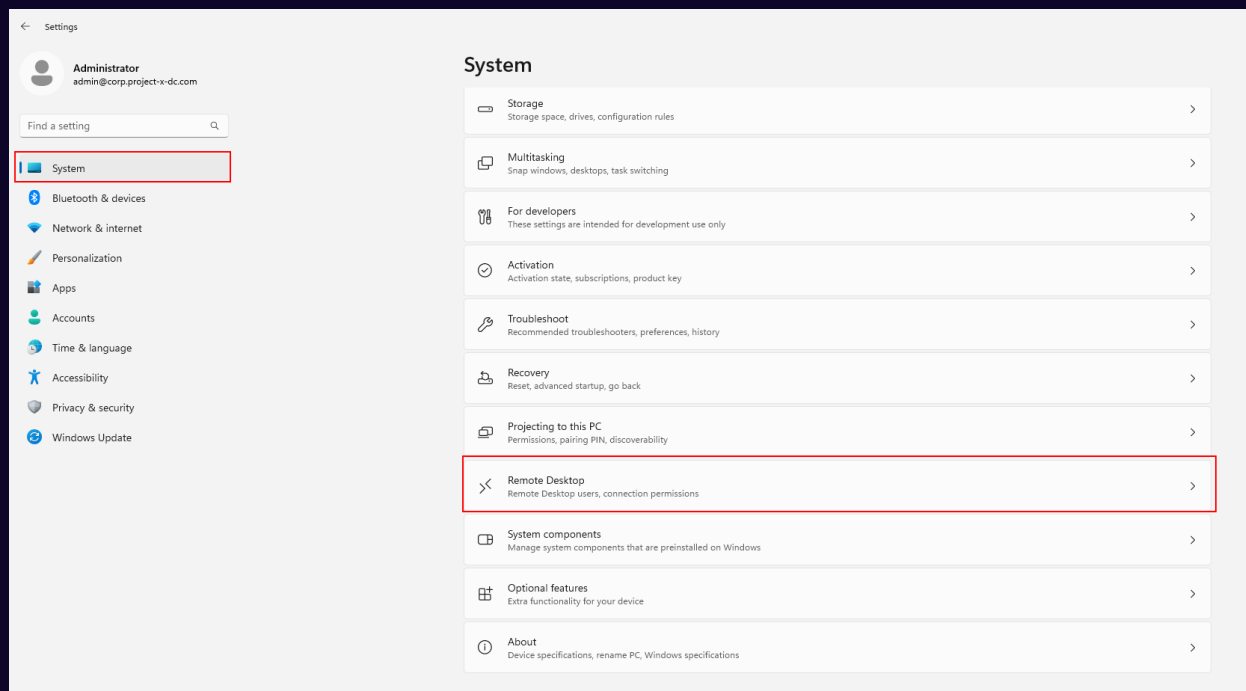
Scroll to the bottom and Select “Create”.

Cancel

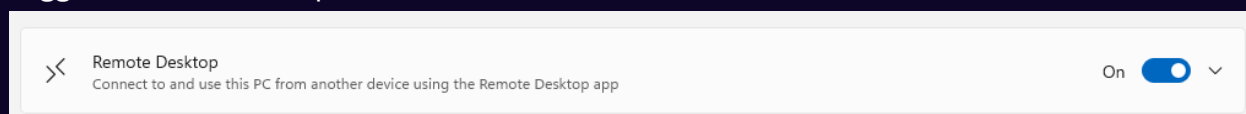
Create

## 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 4625 (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
```



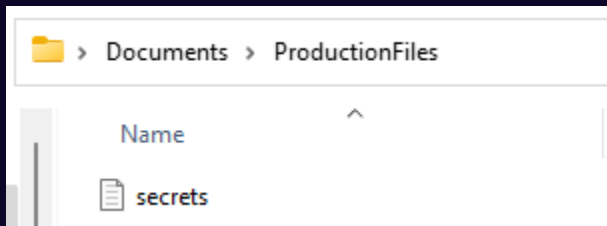
1

<b>full_log</b>	{ "win":{"system":{"providerName":"Microsoft-Windows-Security-Auditing","providerGuid":"","(4549625-5478-4994-a5ba-3e3b0328c30d)","eventID":4544,"version":"3","level":"0","task":"12544","opcode":"0","keywords":"0x8020000000000000","systemTime":"2024-12-27T21:53:46.58745512","eventRecordID":"35813","processID":"808","threadID":"5876","channel":"Security","computer":"project-x-win-client.corp.project-x-dc.com","severityValue":"AUDIT_SUCCESS","message":"'\\\"An account was successfully logged on.\\r\n\\r\nSubject:\\r\n\\nSecurity ID:\\t\\tS-1-5-18\\r\n\\nAccount Name:\\t\\tPROJECT-X-WIN-CS\\r\n\\nAccount Domain:\\t\\tCORP\\r\n\\nLogon ID:\\t\\t0x3E7\\r\n\\nLogon Information:\\r\n\\nLogon Type:\\t\\t10\\r\n\\nRestricted Admin Mode:\\tNo\\r\n\\nRemote Credential Guard:\\tNo\\r\n\\nVirtual Account:\\t\\tNo\\r\n\\nElevated Token:\\t\\tYes\\r\n\\nImpersonation Level:\\t\\tImpersonation\\r\n\\nNew Logon\\r\n\\nSecurity ID:\\t\\tS-1-5-21-1113131282-3d122231d8-29b76d8f18-f6b0\\r\n\\nAccount"
<b>id</b>	1735336427.896199
<b>input.type</b>	log
<b>location</b>	EventChannel
<b>manager.name</b>	secbox
<b>rule.description</b>	User: CORP\Administrator logged using Remote Desktop Connection (RDP) from ip:10.0.0.5.
<b># rule.firedtimes</b>	2
<b>rule.groups</b>	win_evt_channel, windows
<b>rule.id</b>	92653
<b># rule.level</b>	3
<b>rule.mail</b>	false
<b>rule.mitre.id</b>	T1021.001, T1078.002
<b>rule.mitre.tactic</b>	Lateral Movement, Defense Evasion, Persistence, Privilege Escalation, Initial Access
<b>rule.mitre.technique</b>	Remote Desktop Protocol, Domain Accounts
<b>timestamp</b>	Dec 27, 2024 @ 15:53:47.067

## 10

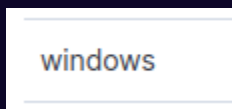
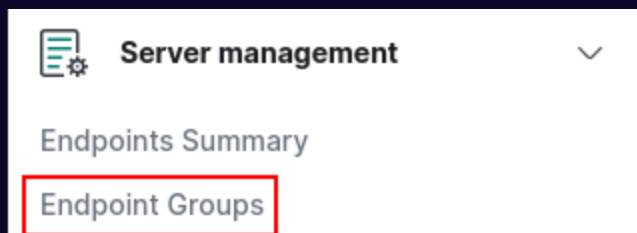
Log

Nav

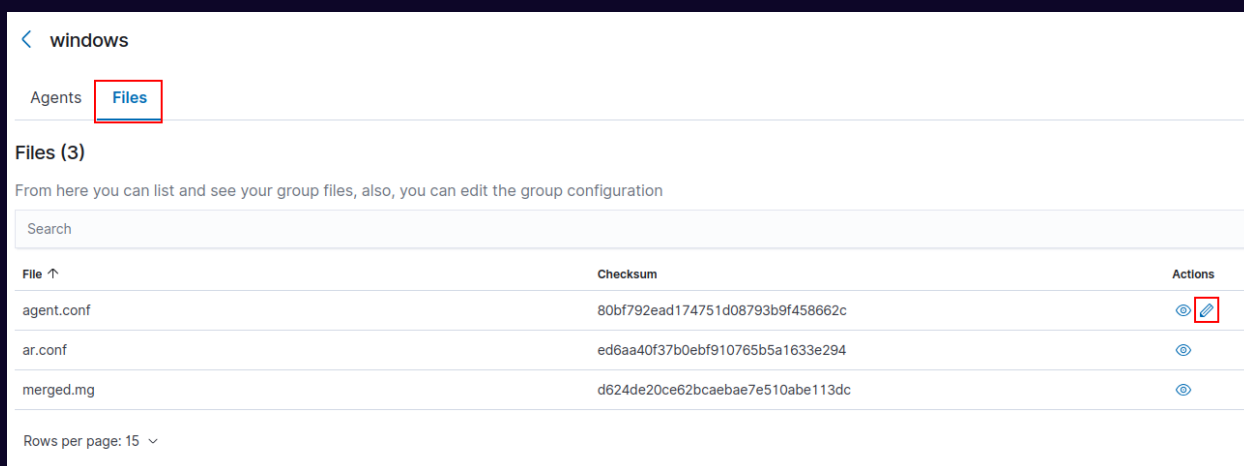


## De

Nav



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>

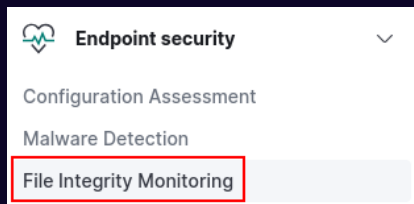
```
<directories check_all="yes" report_changes="yes"
realtime="yes">C:\Users\Administrator\Documents\ProductionFiles</
directories>
```

```
<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.

A screenshot of the 'File Integrity Monitoring' 'Inventory' page. The page shows a list of files monitored by the 'project-x-dc' agent. The 'File' column is highlighted with a red box, showing the path 'c:\users\administrator\documents\productionfiles\secrets.txt'. Other columns include 'Last Modified', 'User', 'User ID', 'Group', 'Group ID', and 'Size'. The table has 11 rows of data. The 'project-x-dc (004)' agent is selected on the right side of the page.

File	Last Modified	User	User ID	Group	Group ID	Size
c:\users\administrator\documents\productionfiles\secrets.txt	Dec 28, 2024 @ 13:05:51.000	Administrators	S-1-5-32-544			83
c:\windows\regedit.exe	Sep 5, 2024 @ 23:07:44.000	TrustedInstaller	S-1-5-80-95600888...			606208
c:\windows\system.ini	Apr 1, 2024 @ 02:01:26.000	SYSTEM	S-1-5-18			219
c:\windows\system32\drivers\etc\hosts	Apr 1, 2024 @ 02:01:27.000	SYSTEM	S-1-5-18			824
c:\windows\system32\drivers\etc\lmhosts.sam	Apr 1, 2024 @ 02:01:27.000	SYSTEM	S-1-5-18			3683
c:\windows\system32\drivers\etc\networks	Apr 1, 2024 @ 02:01:27.000	SYSTEM	S-1-5-18			407
c:\windows\system32\drivers\etc\protocol	Apr 1, 2024 @ 02:01:27.000	SYSTEM	S-1-5-18			1358
c:\windows\system32\drivers\etc\services	Apr 1, 2024 @ 02:01:27.000	SYSTEM	S-1-5-18			17635
c:\windows\system32\windowspowershell\v1.0\powershell.exe	Apr 1, 2024 @ 02:00:33.000	TrustedInstaller	S-1-5-80-95600888...			454656
c:\windows\system32\winrm.vbs	Apr 1, 2024 @ 02:00:38.000	TrustedInstaller	S-1-5-80-95600888...			204072
c:\windows\win.ini	Apr 1, 2024 @ 02:01:26.000	SYSTEM	S-1-5-18			92

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.

A screenshot of the 'File Integrity Monitoring' 'Events' page. The page shows a list of events triggered by the 'project-x-dc' agent. The table has columns: 'timestamp', 'agent.name', 'syscheck.path', 'syscheck.event', 'rule.description', and 'rule.level'. The event for 'c:\users\administrator\documents\productionfiles\secrets.txt' is highlighted with a red box, showing a 'modified' event with 'rule.level' 7.

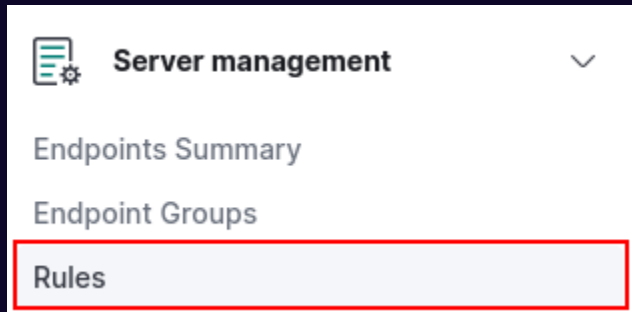
timestamp	agent.name	syscheck.path	syscheck.event	rule.description	rule.level
Dec 28, 2024 @ 13:11:27.778	project-x-dc	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\DFSR\Access Chec...	modified	Registry Key Integrity Checksum C...	5
Dec 28, 2024 @ 13:06:43.844	project-x-dc	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\bam\State\UserSet...	modified	Registry Value Integrity Checksum ...	5
Dec 28, 2024 @ 13:06:43.841	project-x-dc	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\bam\State\UserSet...	modified	Registry Value Integrity Checksum ...	5
Dec 28, 2024 @ 13:06:43.833	project-x-dc	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\bam\State\UserSet...	modified	Registry Value Integrity Checksum ...	5
Dec 28, 2024 @ 13:06:43.833	project-x-dc	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\bam\State\UserSet...	modified	Registry Key Integrity Checksum C...	5
Dec 28, 2024 @ 13:05:54.866	project-x-dc	c:\users\administrator\documents\productionfiles\secrets.txt	modified	Integrity checksum changed.	7
Dec 28, 2024 @ 13:03:27.896	project-x-dc	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\SharedAccess\Epo...	modified	Registry Value Integrity Checksum ...	5



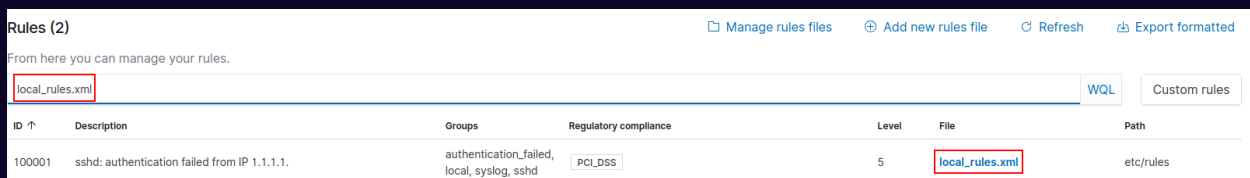
## Create Detection Alert

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>
```

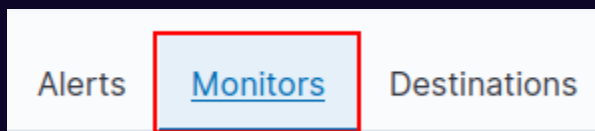
Click "Save" → "Restart".

```
< local_rules.xml
Ruleset Test Save
Changes will not take effect until a restart is performed. Restart
2
3 <!-- Modify it at your will. -->
4 <!-- Copyright (C) 2015, Wazuh Inc. -->
5
6 <!-- Example -->
7 <group name="local,syslog,sshd,">
8
9 <!--
10 Dec 10 01:02:02 host sshd[1234]: Failed none for root from 1.1.1.1 port 1066 ssh2
11 -->
12 <rule id="100001" level="5">
13 <if_sid>5716</if_sid>
14 <srcip>1.1.1.1</srcip>
15 <description>sshd: authentication failed from IP 1.1.1.1.</description>
16 <group>authentication_failed,pci_dss_10.2.4,pci_dss_10.2.5,</group>
17 </rule>
18
19 </group>
20
21 <group name="syscheck">
22 <rule id="100002" level="10">
23 <field name="file">secrets.txt</field>
24 <match>modified</match>
25 <description>File integrity monitoring alert - access to sensitive.txt file detected</description>
26 </rule>
27 </group>
28 </group>
```

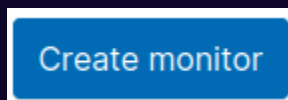
After the console restarts.

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.

Create monitor

Monitor details

Monitor name

File Accessed

Monitor type

☒ Per query monitor

Per query monitors run a query and generate alerts based on trigger criteria that match query results.

☐ Per bucket monitor

Per bucket monitors run a query that evaluates trigger criteria based on aggregated values in the dataset.

☐ Per cluster metrics monitor

Per cluster metrics monitors run API requests to monitor the cluster's health.

☐ Per document monitor

Per document monitors run queries that return individual documents matching the trigger conditions.

☐ Composite monitor

Composite monitors chain the outputs of different monitor types and focus trigger conditions to reduce alert noise and generate finer results.

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

>projectsecurity\_

wazuh-alerts-4.x-\*

For “Time Field” select:

@timestamp

### Data source

**Index**

wazuh-alerts-4.x-\* X

You can use a \* as a wildcard or date math index resolution in your index pattern

**Time field**

@timestamp |

Choose the time field you want to use for your x-axis

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.

<b>full_log</b>	> 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: 'A12rh0Q880ad2a0c521fa7Acb22a0a5c'
<b>id</b>	1735604014.17034950
<b>input.type</b>	log
<b>location</b>	syscheck
<b>manager.name</b>	secbox
<b>rule.description</b>	Integrity checksum changed.
<b>rule.firedtimes</b>	1
<b>rule.gdpr</b>	II_5.1.f
<b>rule.gpg13</b>	4.11
<b>rule.groups</b>	ossec, syscheck, syscheck_entry_modified, syscheck_file
<b>rule.hipaa</b>	164.312.c.1, 164.312.c.2
<b>rule.id</b>	550
<b>rule.level</b>	7
<b>rule.mail</b>	false
<b>rule.mitre.id</b>	T1565.001
<b>rule.mitre.tactic</b>	Impact
<b>rule.mitre.technique</b>	Stored Data Manipulation

📁 secrets.txt file modified sample log.

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

**Data filter** - optional ?

No filters defined.

[+ Add filter](#)

You can add up to 10 more data filters.

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.

**ADD DATA FILTER**

full\_log contains secrets.txt

**ADD DATA FILTER**

syscheck.event is modified

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

**Data filter** - optional ?

full\_log contains secrets.txt × syscheck.event is modified ×

[+ Add filter](#)

You can add up to 8 more data filters.

Let's add a Trigger.

**No triggers**

Add a trigger to define conditions and actions.

[Add trigger](#)

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

File Accessed

Trigger name

File Accessed

Severity level

2 (High)

Trigger condition

IS ABOVE 1

Scroll to the bottom and Select “Create”.

Cancel 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

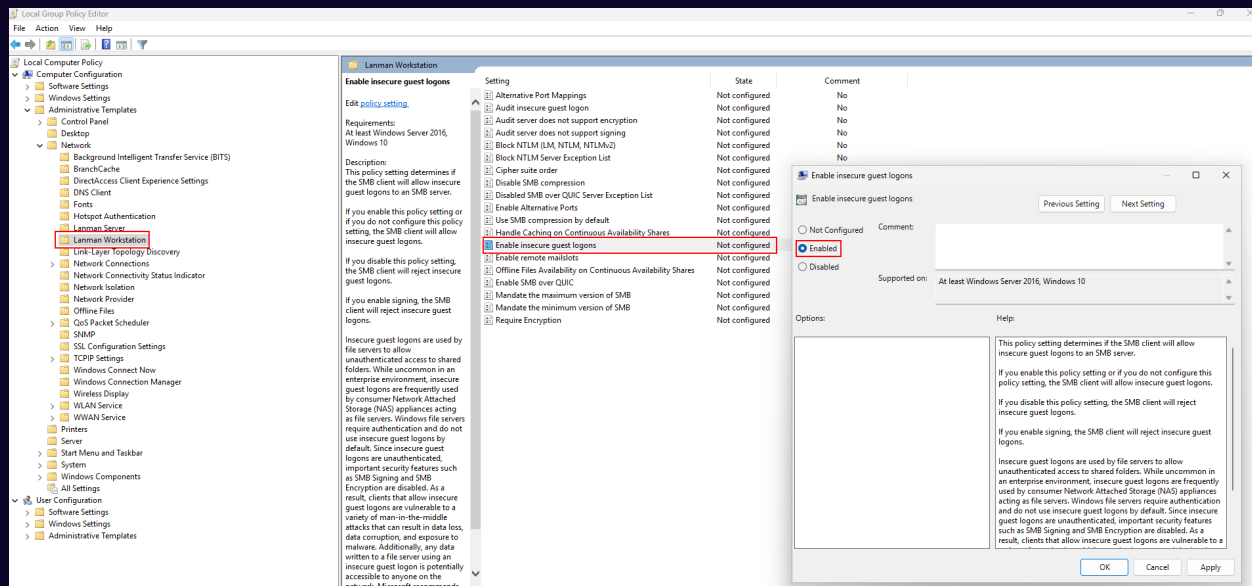
The above detection will be relied upon to detect changes to the secrets.txt file.

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
```