**Analysing the Vulnerabilities and Behaviour of a Linux-based EC2 Instance using Amazon Inspector**

**Table of contents**

|  |  |
| --- | --- |
| 1. | Architectural Overview of Amazon Inspector Assessment Runs |
| 2. | Installing SSM and Inspector agent on the EC2 instance |
| 3. | Troubleshooting the Unknown Agent status with AWS Support Team |
| 4. | Monitoring Amazon Inspector Using Amazon CloudWatch |
| 5. | Monitoring Amazon Inspector Using Amazon CloudTrail |
| 6. | Report findings and Remediation |
| 7. | Lambda for E-mail notification Automatic Updation |
| 8. | Pricing Incurred in Task |

1. **Architectural overview of the Amazon Inspector assessment runs**

Amazon Inspector performs security assessments of [Amazon EC2](https://aws.amazon.com/ec2/) instances by using AWS managed [rules packages](http://docs.aws.amazon.com/inspector/latest/userguide/inspector_rule-packages.html#InspectorRulePackages) such as the [Common Vulnerabilities and Exposures](http://docs.aws.amazon.com/inspector/latest/userguide/inspector_cves.html) (CVEs) package. Amazon Inspector is a service designed for analysing AWS resources in order to identify potential security issues.

By creating templates and specifying targets you can easily assess how secure your AWS resources are. During the assessment process a wide range of elements are scanned and all collected data is then analysed and compared to a set of security rules specified in the template. As a result you get a detailed report indicating what security problems your resources potentially suffer from and what should you do to fix them.

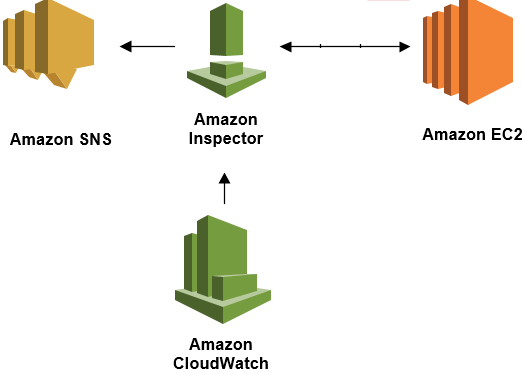
****

Figure 1. Architectural Overview

1. **Installing SSM and Inspector agent on the EC2 instance**

Amazon Inspector uses assessment targets to designate the AWS resources to evaluate and t**o** create an assessment target and install a Systems Manager Agent and inspector agent on the EC2 instance using run command which will be restricted otherwise**.** To verify that the agent is installed and running, sign in to your EC2 instance and run the following command:

sudo /opt/aws/awsagent/bin/awsagent status

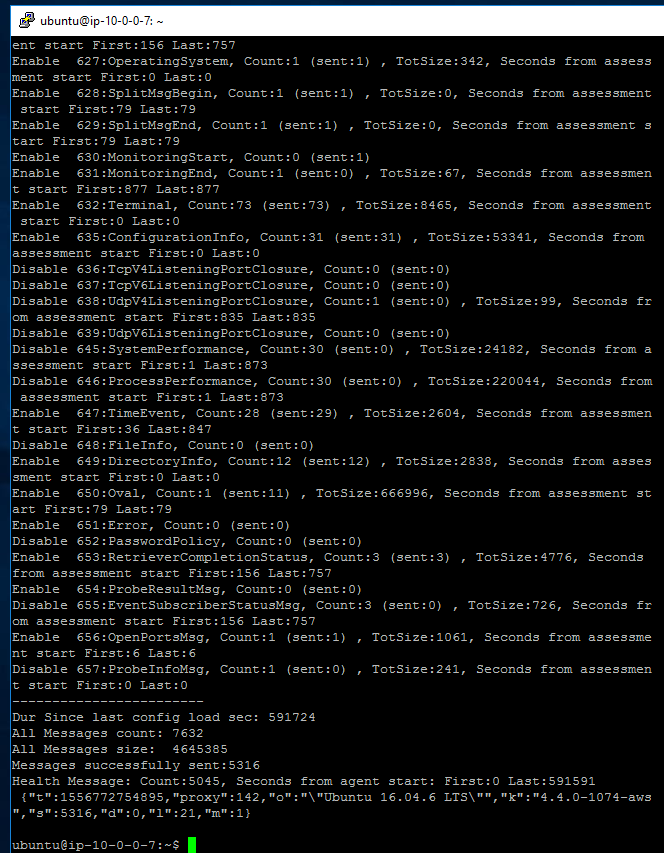


Figure 2. Messages exchanged between agent and inspector

This command returns the status of the currently running agent, on checking the status it is observed in the screenshot that messages are being exchanged between the agent installed on ec2 machine and amazon inspector.

AWS Systems Manager Agent (SSM Agent) is Amazon software that can be installed and configured on an Amazon EC2 instance, an on-premises server, or a virtual machine (VM). SSM Agent makes it possible for Systems Manager to update, manage, and configure ec2 instances. SSM Agent is installed, by default in some but in some like the machine tested on it had to be manually installed.

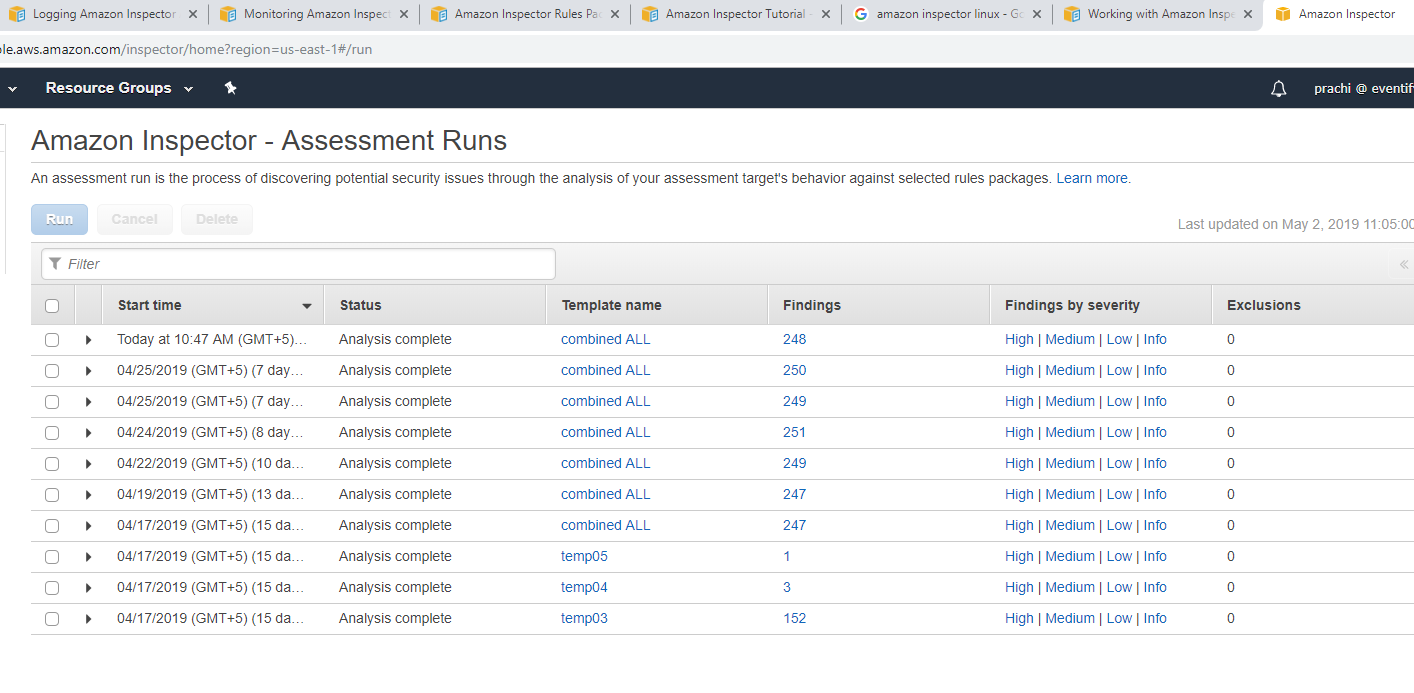


Figure 3. Assessment runs on target machine

1. **Troubleshooting the unknown agent status with AWS Support Team**

I contacted AWS Support because the agent status for egsp linux instance "i-083952770d8693fd1" was being shown as "UNKNOWN". We performed the following troubleshooting actions from within the instance:

1) Checked the kernel version which comes out be 4.9.32-15.41.amzn1.x86\_64 (lies in kernel versions list that are compatible with an Amazon Inspector agent running on Linux)

2) Installed inspector agent again.

3) sudo /opt/aws/awsagent/bin/awsagent status

4) telnet arsenal.us-west-2.amazonaws.com 443

5) curl -vk <https://arsenal.ap-south-1.amazonaws.com>

6) telnet s3.dualstack.us-west-2.amazonaws.com 443

7) curl -vk <https://s3.dualstack.ap-south-1.amazonaws.com>

Since, we were not able to figure out the possible issue causing this error, I reached out to our internal service team to further troubleshoot the issue. As per the internal team, there wasn't seem to be an issue with our connection to arsenal. Hence, they would require the Agent logs of the instance with this issue which were captured using the following steps:

1) Copy the agent.cfg file located at "/opt/aws/awsagent/etc/agent.cfg" to "/opt/aws/awsagent/etc/agent\_bckup.cfg".

2) Stop the agent with command: sudo /etc/init.d/awsagent stop

3) Edit the agent.cfg file located at "/opt/aws/awsagent/etc/agent.cfg".

4) Use the below configuration OR append these settings to your existing configuration:

{

"SubSystems" : "ALL",

"LogLevels" : "LogAll",

"LogFile" : "/tmp/agent\_test.log"

}

5) Restart the agent so that agent reads the correct configuration using the following command: sudo /etc/init.d/awsagent start

6) Wait 15 minutes or run a 15 minute assessment

7) After 15 minutes or the 15 minute assessment is over, stop the agent once again with:

sudo /etc/init.d/awsagent stop

Collect the file /tmp/agent\_test.log and send it back to us.

Copy back "/opt/aws/awsagent/etc/agent\_bckup.cfg" to "/opt/aws/awsagent/etc/agent.cfg"

8) Restart the agent so that agent reads the original configuration again:

sudo /etc/init.d/awsagent start

The agent logs obtained by following the steps mentioned above helped us narrow down the issue and then we captured a packet trace at agent start using the following steps:

1) Check if tcpdump is installed on the system: which tcpdump

2) If tcpdump is not installed then: sudo yum install -y tcpdump

3) Stop the agent with command: sudo /etc/init.d/awsagent stop

4) Start network capture to a file: sudo tcpdump -i any -w agent.pcap

5) Start the agent: sudo /etc/init.d/awsagent start

6) Wait 30 seconds

7) Stop network capture: Ctrl+C

8) Attach agent.pcap to this support case.

The case is still open and the issue remains unresolved as of now but I came across a blog dated Jan 2019 which states that CIS Operating System Configuration findings is currently not supported for Amazon Linux versions in Inspector yet.

1. **Monitoring Amazon Inspector Using Amazon CloudWatch**

The Amazon Inspector namespace includes the following metrics. And can be monitored for real-time metrics using Amazon CloudWatch, which collects and processes raw data into readable. By default, Amazon Inspector sends metric data to CloudWatch in 5-minute periods. And can be used with the AWS Management Console, the AWS CLI, or an API to view the metrics that Amazon Inspector sends to CloudWatch. Here, console is used.

1. AssessmentTargetARN metrics:

|  |  |
| --- | --- |
| Metric | Description |
| TotalMatchingAgents | Number of agents that match this target |
| TotalHealthyAgents | Number of agents that match this target that are healthy |
| TotalAssessmentRuns | Number of assessment runs for this target |
| TotalAssessmentRunFindings | Number of findings for this target |

1. AssessmentTemplateARN metrics:

|  |  |
| --- | --- |
| Metric | Description |
| TotalMatchingAgents | Number of agents that match this template |
| TotalHealthyAgents | Number of agents that match this template that are healthy |
| TotalAssessmentRuns | Number of assessment runs for this template |
| TotalAssessmentRunFindings | Number of findings for this template |

1. Aggregate metrics

|  |  |
| --- | --- |
| Metric | Description |
| TotalAssessmentRuns | Number of assessment runs in this AWS account |

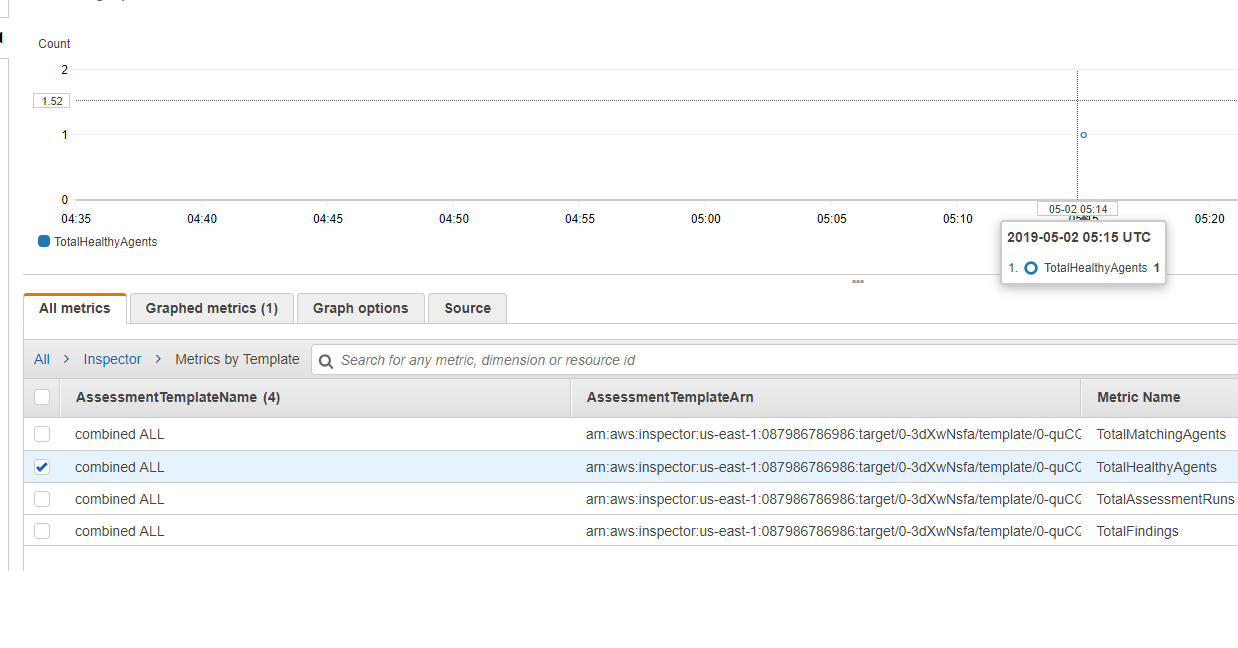


Figure 4. Graph metrics for total healthy agents

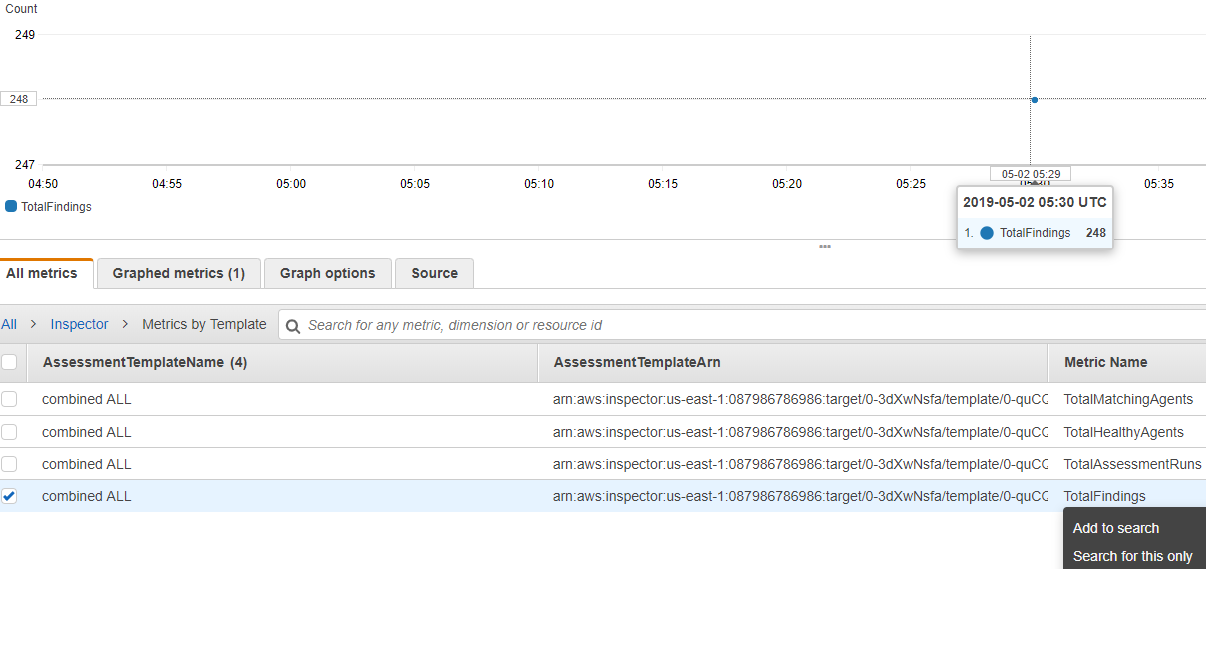


Figure 5. Graph metrics for total findings

1. **Monitoring Amazon Inspector Using Amazon CloudTrail**

Amazon Inspector is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Amazon Inspector. CloudTrail captures all API calls for Amazon Inspector as events, including calls from the Amazon Inspector console and code calls to the Amazon Inspector API operations.

The major difference noticed between CloudWatch and CloudTrail monitoring is that Cloudwatch logs focus on what is happening, which resources and services are being used. Whereas CloudTrail focusses on revealing who did the activity and when was it done.

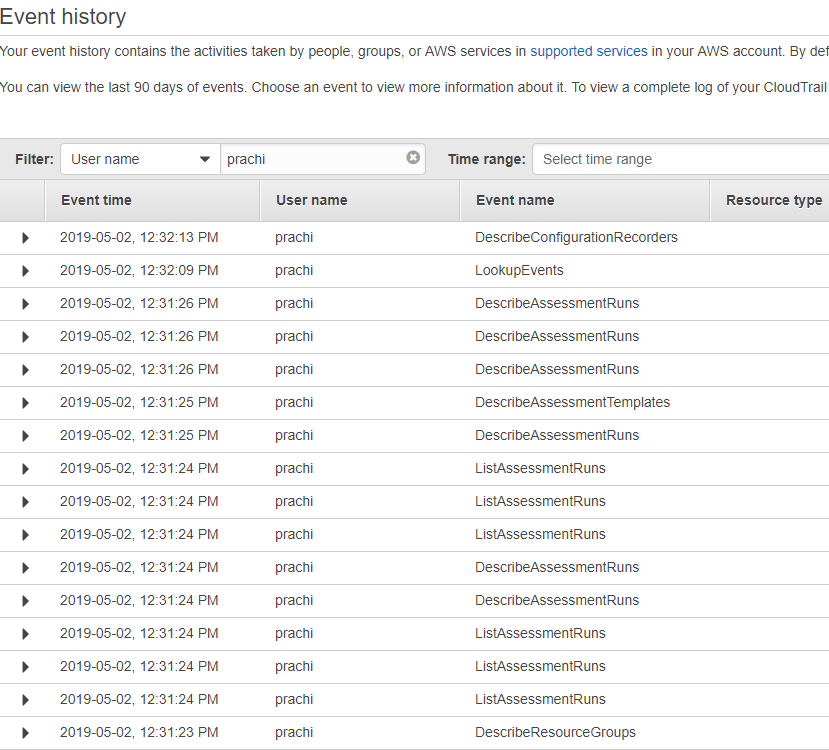


Figure 6. CloudTrail event history for Amazon Inspector

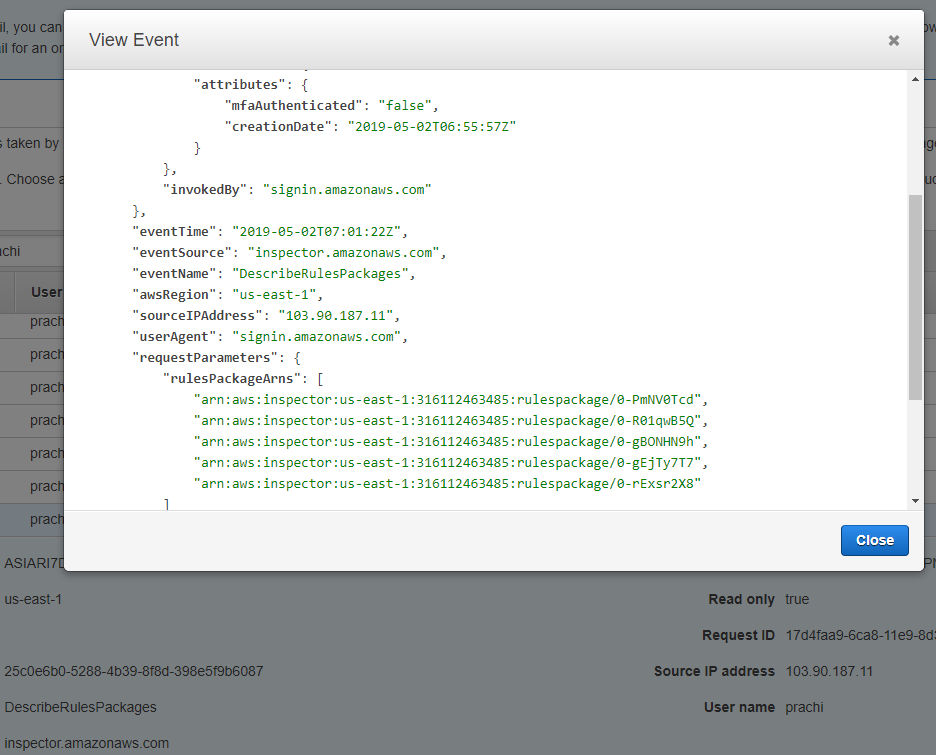


Figure 7. CloudTrail event triggered details

1. **Report findings and Remediation**

The reports generated were studied and the remediations were segregated and proceeded as per below priorities:

Priority 1:          Critical Risk Profile and can be addressed Quick in Time

Priority 2:           High Risk Profile and can be addressed Quick in Time

In parallel of above, exercise is performed to logically group all the findings irrespective of their Risk Profile rating and Time to fix. (Here preferences should be given which are exposed over internet and the vulnerability is old in age – can be extracted from CVE info)

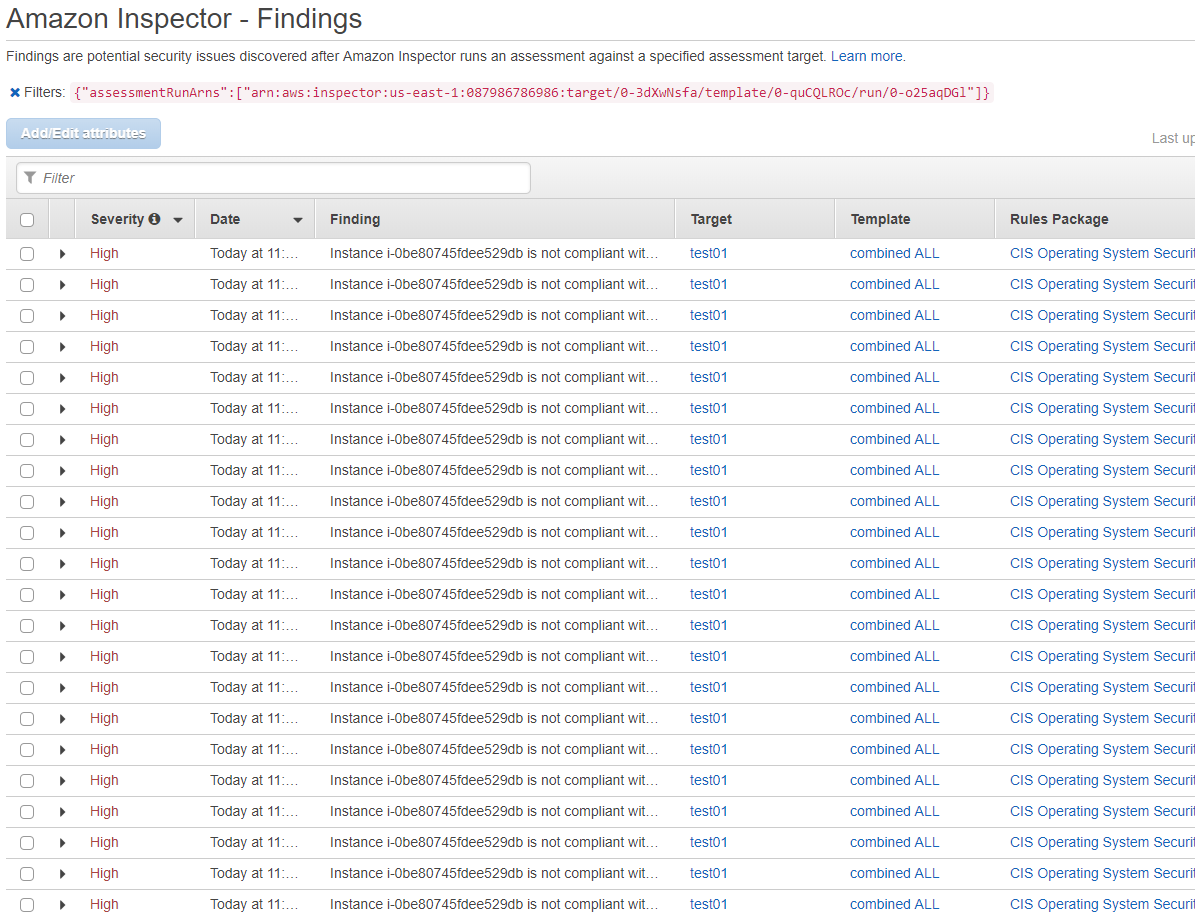


Figure 8. Operating system vulnerabilities

For e.g.

1. There are several vulnerabilities in CVE section in relation to PHP which can be fixed by updating it to the latest version, to eliminate multiple linked vulnerabilities.
2. Multiple vulnerabilities due to older Linux version, on updating the version and it will fix the multiple linked vulnerabilities.
3. There are multiples ports and services being set as open by Amazon, and are being highlighted by them as vulnerable so I checked with the application owners that what ports and services are required open to enable the accessibility of application and rest of the ports and services can be shut to reduce down the list of vulnerabilities

These are just examples, and such relations can be found out throughout the list of vulnerabilities being generated. After addressing the P1 and P2, the ec2 instances was rescanned and reconciled output had lesser number of vulnerabilities left.

Further, it is important to know what all applications and services are running on the EC2 instance being monitiored because accordingly the priority of remediations will change. The contact us eventifyd instance had a tomcat server and sql server majorly running to host a webpage. . Port number 8080 and 81 were open so they were closed. 8080 was also not required because the application had tomcat hosted on port 80. And the webpage [http://contactus.minddemo.cloud](http://contactus.minddemo.cloud/) is not hosted on https.

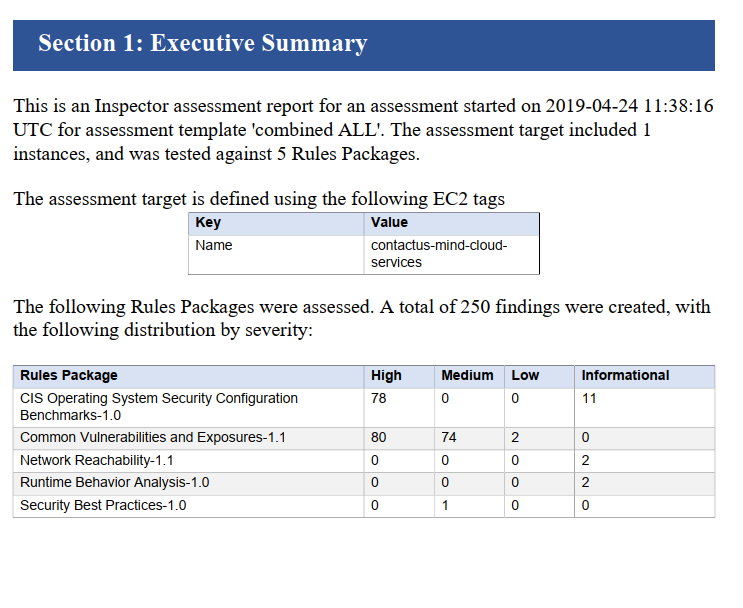


Figure 9. Findings report summary

1. **Lambda for email notification automatic updation**

### 1) Create an SNS topic to which Amazon Inspector will publish messages:

Amazon SNS uses topics, communication channels for sending messages and subscribing to notifications. An SNS topic was created for this solution to which Amazon Inspector publishes messages whenever there is a security finding. Then a Lambda function is created that subscribes to this topic and receives a notification whenever a new security finding is generated.

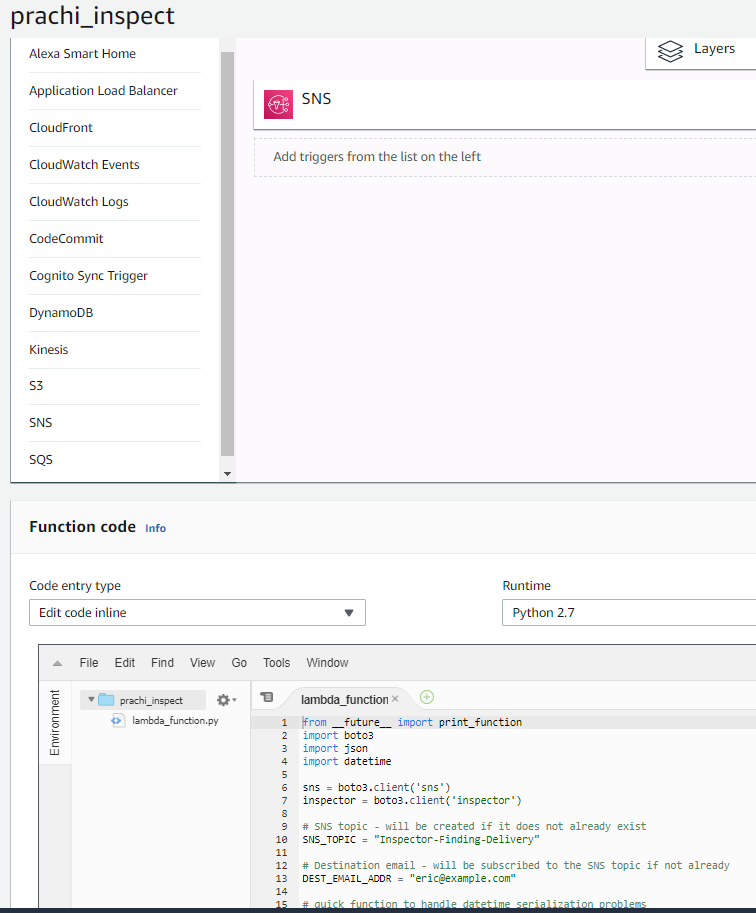


Figure 10. Lambda for automated updation

### 2) Configure an Amazon Inspector assessment template to post finding notifications to the SNS topic:

An assessment template is a configuration that tells Amazon Inspector how to construct a specific security evaluation. For example, an assessment template can tell Amazon Inspector which EC2 instances to target and which rules packages to evaluate. You can configure a template to tell Amazon Inspector to generate SNS notifications when findings are identified. In order to enable automatic remediation, you either create a new template or modify an existing template to set up SNS notifications to the SNS topic that you just created.

### 3)  Create the Lambda auto-remediation function:

Now, create a Lambda function that listens for Amazon Inspector to notify it of new security findings, and then tells the EC2 SSM agent to run the appropriate system update command.

1. **Pricing Incurred in task**

On calculating the bill using simple AWS monthly calculator the following is the cost estimated for the task.

1. Amazon Inspector Assessment runs-

Amazon Inspector assessments with the network reachability rules packages are priced per instance per assessment (instance-assessment) per month. For example, if you run 1 assessment against 1 instance, that is 1 instance-assessment. If we run 1 assessment against 10 instances, it is 10 instance-assessments. The pricing starts at $0.15 per instance-assessment per month but the first 250 assessment runs are free. Around 15 runs were made so the costing stands at zero.

1. CloudWatch-

Detailed ec2 monitoring is not enabled so the price $2.10 per instance per month and goes down to $0.14 per instance at the lowest priced tier.

1. CloudTrail-

There was not any need to create additional trail so the costing stands at zero.

1. Lambda, SNS service and AWS Business Support is also utilized.
2. EC2 Instances-

The below is the monthly cost and they were used for 5 days so it adds up to 1.5 dollars.

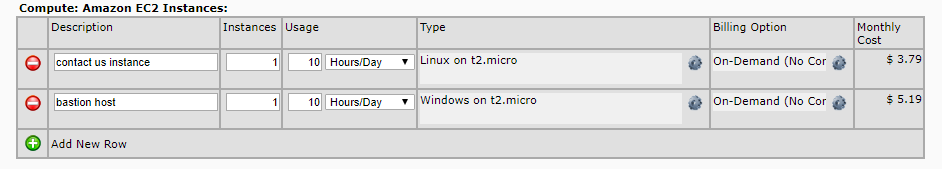


Figure 11. EC2 instance usage