USE CASES

The Wazuh platform is used to protect and monitor systems in different ways. Due to all of its capabilities, it is often used for threat prevention, detection and response. Additionally, the Wazuh platform is used to meet regulatory compliance requirements (such as PCI DSS or HIPAA) and configuration standards (CIS hardening guides).

Wazuh is a popular security solution among IaaS users (e.g. Amazon AWS, Azure or Google cloud), used to monitor virtual machines and cloud instances. This is done at the system level using the Wazuh security agent and at the infrastructure level by pulling data directly from the cloud provider's API.

Additionally, Wazuh is often used to protect containerized environments by providing cloud-native runtime security. This feature is based on integration with Docker Engine API and Kubernetes API. Additionally, for added protection, the Wazuh security agent can run on the Docker host, providing a full set of threat detection and response capabilities.

In this section, you will find a brief example of some of the most common use cases of the Wazuh solution.

Table 1: SIEM Wazuh use cases

Log data analysis	File integrity monitoring
Rootkit Detection	active response
Configuration Assessment	System inventory
Vulnerability detection	Cloud Security Monitoring
Container Security Monitoring	Regulatory conformity

1- Log data analysis

Automated log management and analysis speeds threat detection. In many cases, evidence of an attack can be found in device, system, and application log messages. Wazuh can be used to automatically aggregate and analyze log data.

The Wazuh agent, running on the monitored endpoint, is usually the one responsible for reading messages from the operating system and the application log, passing themto the Wazuh server, where the analysis takes place. When no agent is deployed, the server can also receive data via syslog from network devices or applications.

Wazuh uses decoders to identify the source application of the log message. Then it analyzes the data using application-specific rules. Here is an example rule used to detect SSH authentication failure events:

```
<rule id="5716" level="5">
<if_sid>5700</if_sid>
<match>^Failed|^error: PAM: Authentication</match>
<description>SSHD authentication failed.</description>
<miter>
<id>T1110</id>
</mitre>
authentication_failed .1,tsc_CC6.8,tsc_CC7 .2,tsc_CC7.3,</group>
</rule>
```

Rules include a match field, used to define the pattern the rule will match. They also have a level field that specifies the priority of resulting alerts. In addition, the rules enrich events with technical identifiers from the Miter ATT&CK framework and witha match to regulatory compliance checks.

The manager will generate an alert each time an event, collected by one of the agents or via Syslog, matches a rule whose priority level is higher than a predefined threshold(3 by default).

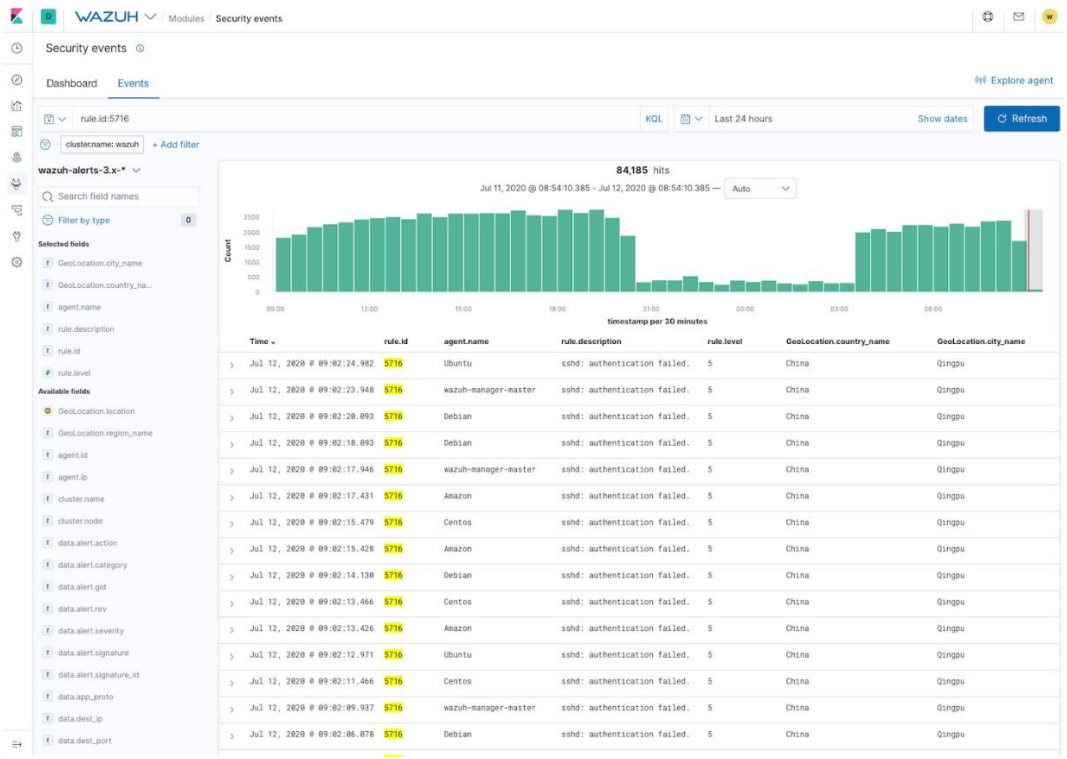
Here is an example found in /var/ossec/logs/alerts/alerts.json (some fields removed for brevity):

Output


```
{
  "agent": {
      "id": "005",
      "ip": "10.0.1.175",
      "name": "Centos"
  },
  "predecoder": {
      "hostname": "ip-10-0-1-175",
      "program_name": "sshd",
      "timestamp": "Jul 12 15:32:41"
  },
  "decoder": {
      "name": "sshd",
      "parent": "sshd"
  },
  "full_log": "Jul 12 15:32:41 ip-10-0-1-175 sshd[21746]: Failed password fo
  "location": "/var/log/secure",
  "data": {
      "dstuser": "root",
      "srcip": "61.177.172.13",
      "srcport": "61658"
  },
  "rule": {
      "description": "sshd: authentication failed.",
      "id": "5716",
      "level": 5,
```


Picture 1: Capturing log data analysis alerts

Once generated by the Wazuh server, the alerts are sent to the component Elastic- Stackwhere they are enriched with geolocation information, stored and indexed. Kibana can then be used to search, analyze and visualize the data. Below is an alert as displayedin the interface:



Picture 2: Visualization of generated alerts in the Kibana interface

Wazuh provides a default set of rules, updated periodically, with more than 3,000 rules for different systems and applications. In addition, it allows the creation of custom rules.

2- File integrity monitoring

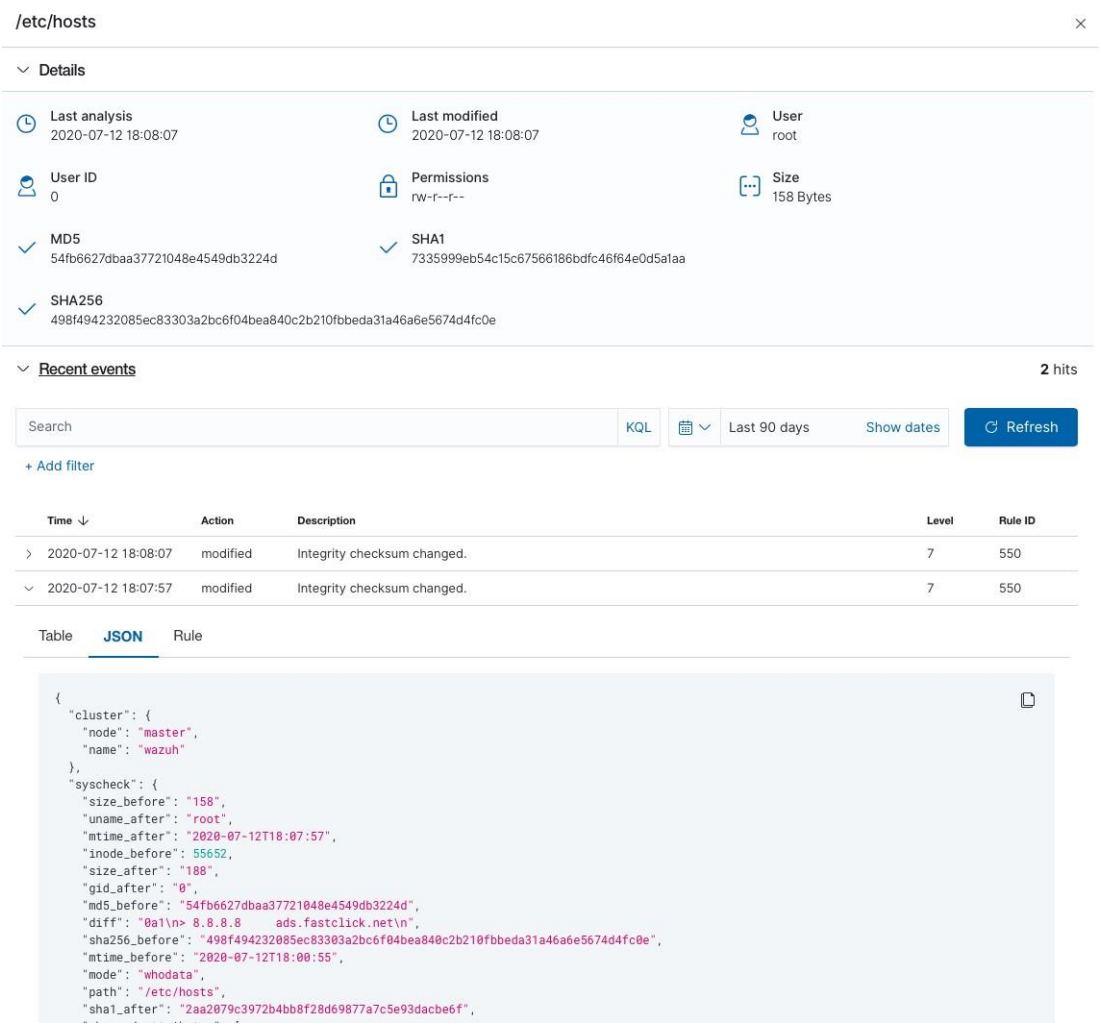
The File Integrity Monitoring (FIM) component detects and alerts when operating system or application files change. This capability is often used to detect access or changes to sensitive data. If your servers are PCI DSS compliant, requirement 11.5 states that a file integrity monitoring solution must be installed in order to pass the audit.

Below is an example of an alert generated when a monitored file changes. Metadata includes MD5, SHA1 and SHA256 checksums, file size (before and after modification), file permissions, file owner, content modifications and user who made these modifications (data who).

Picture 3: Capture of alerts generated when modifying files

A good summary of file changes can be found in the FIM Dashboard, which provides drill-down capabilities to view full details of triggered alerts.

Use cases



Picture 4: Visualization of modification alerts generated in the Kibana interface

3- Rootkit Detection

agent wazuhPeriodically scans the monitored system for rootkits at the kernel and userspace level. This type of malware usually replaces or modifies existing operat ing system components, in order to change the behavior of the system. Rootkits can hide other processes, files, and network connections. Wazuh uses different detection mechanisms to look for system anomalies or well-known intrusions. The Rootcheck component does this periodically:

Table 2: Detection mechanisms used by wazuh to search for systemanomalies

Shares	**Detection mechanism**	**Binary**	**System call**
Detection of hidden processes	Comparison of system output binaries and system calls	PS	setid
			getpgid
			kill
Detection of hidden files	Comparison of outputfrom binary systems and system call.	ls	Statistical
			opendir
			Playback directory
	Scan /dev	ls	opendir
Detection of hidden ports	Comparison of binary system output and system calls	Netstat	Bind
Detection of known rootkits	Use a malicious file database	-	Statistical
			Open
			opendir
	Inspect file contents using signatures	-	Open
	Detect file permission and ownership anomalies	-	Statistical

Below is an example of an alert generated when a hidden process is found. In this case, the affected system is running a Linux kernel-level rootkit (named Diamorphine):

Picture 5: Capturing alerts generate on detection of a hidden process

4- Active response

Agent Wazuhautomates response to threats by performing actions when detected. The agent has the ability to block network connections, stop running processes, or delete malicious files, among other actions. Additionally, it can also run custom scripts developed by the user (e.g. Python, Bash, or PowerShell).

To use this feature, users define conditions that will trigger scripted actions, which typically involve detecting and assessing threats. For example, a user can use log analysis rules to detect an intrusion attempt and an IP reputation database to assess the threat by finding the source IP address of the login attempt.

In the scenario described above, when the source IP address is recognized as malicious (low reputation), the monitored system is protected by automatically configuring a firewall rule to drop all traffic from the attacker. Depending on the active response, this firewall rule is temporary or permanent.

Below is the configuration used to define two scripts used for automated login blocking.

On Linux systems, the Wazuh agent typically integrates with the local

Iptables firewall for this purpose. On Windows systems, it uses the null routing technique (also known as blackholing) instead:

```
<command>
<name>firewall-drop</name>
<executable>firewall-drop.sh</executable>
<expect>srcip</expect>
<timeout_allowed>yes</timeout_allowed>
</command>
<command>
<name>win_route-null</name>
<executable>route-null.cmd</executable>
<expect>srcip</expect>
<timeout_allowed>yes</timeout_allowed>
</command>
```

In addition to defined commands, active responses define conditions that must be met to trigger them. Below is an example configuration that would trigger the firewall-drop command when the log scan rule matches 100100.

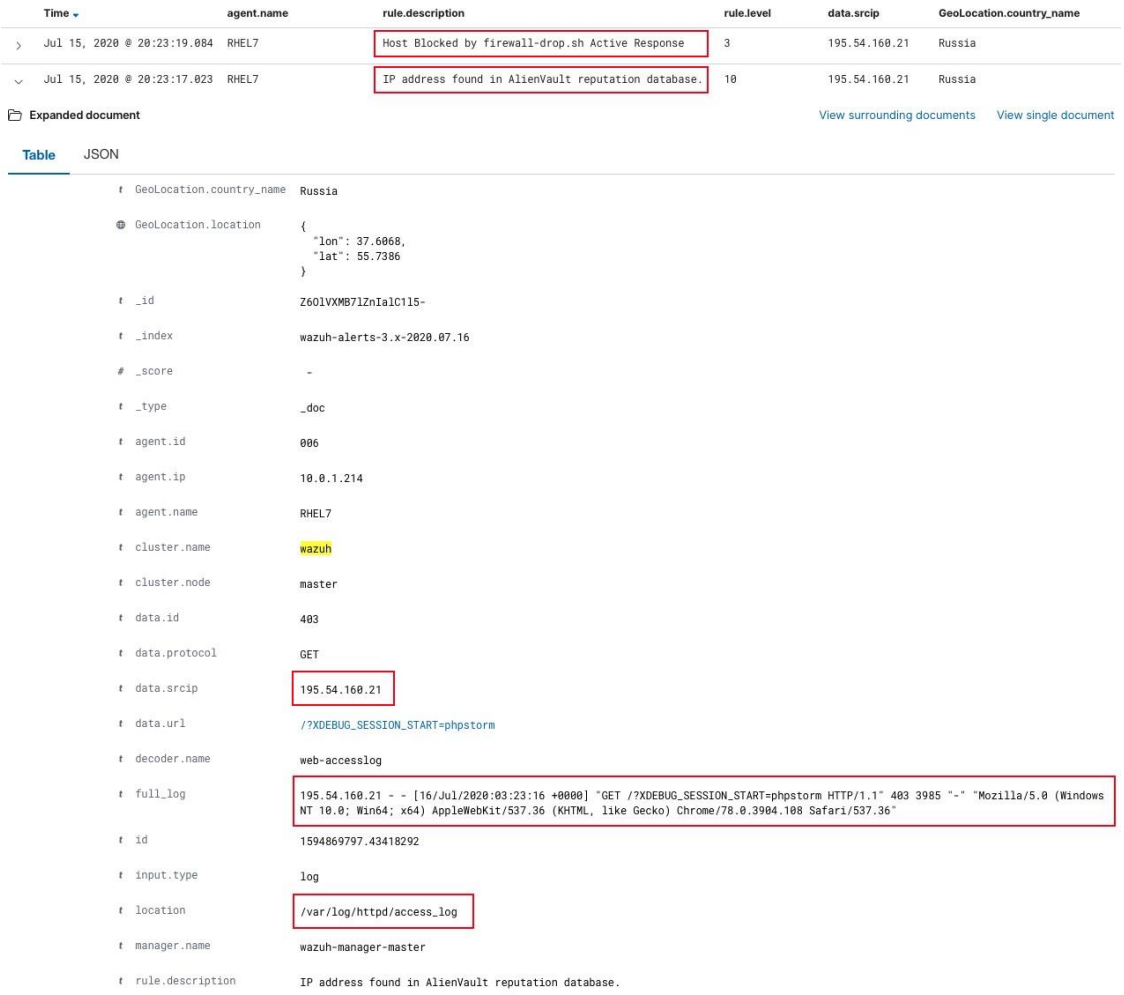
```
<active-response>
<command>firewall-drop</command>
<location>local</location>
<rules_id>100100</rules_id>
<timeout>60</timeout>
</active-response>
```

In this case, rule 100100 is used to check for alerts when the source IP address is part of a well-known IP reputation database:

```
<rule id="100100" level="10">
<if_group>web|attack|attacks</if_group>
<list field="srcip" lookup="address_match_key">etc/lists/blacklist-alienvault</list>
<description>IP address found in AlienVault reputation database. </description>
```

</rule>

Below is a screenshot with two Wazuh alerts: the one triggered when a web attack is detected (attempt to exploit a PHP server vulnerability), and the one that informs that the malicious actor has been blocked.



Time	agent.name	rule.description	rule.level	data.srcip	GeoLocation.country_name
Jul 15, 2020 @ 20:23:19.084	RHEL7	Host Blocked by firewall-drop.sh Active Response	3	195.54.160.21	Russia
Jul 15, 2020 @ 20:23:17.023	RHEL7	IP address found in AlienVault reputation database.	10	195.54.160.21	Russia

Expanded document — View surrounding documents — View single document

Table | JSON

GeoLocation.country_name	Russia
GeoLocation.location	{ "lon": 37.6068, "lat": 55.7386 }
_id	Z6OlVXMB71ZnIalC115-
_index	wazuh-alerts-3.x-2020.07.16
_score	-
_type	_doc
agent.id	006
agent.ip	10.0.1.214
agent.name	RHEL7
cluster.name	wazuh
cluster.node	master
data.id	403
data.protocol	GET
data.srcip	195.54.160.21
data.url	/?XDEBUG_SESSION_START=phpstorm
decoder.name	web-accesslog
full_log	195.54.160.21 - - [16/Jul/2020:03:23:16 +0000] "GET /?XDEBUG_SESSION_START=phpstorm HTTP/1.1" 403 3985 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36"
id	1594869797.43418292
input.type	log
location	/var/log/httpd/access_log
manager.name	wazuh-manager-master
rule.description	IP address found in AlienVault reputation database.

Picture 6: Capture generate when detecting and blocking a web attack

5- Configuration Assessment

Automated Security Configuration Assessment (SCA) is a critical capability to improve an enterprise's security posture and reduce its attack surface. The Wazuh SCA module makes it possible to maintain a standard configuration via the monitored terminals. This is done through predefined checks based on Center of Internet Security (CIS) hardening guides.

When the SCA module is enabled, the Wazuh agent periodically performs scans, reporting misconfigurations in the monitored system. These scans evaluate the system configuration using policy files, which contain a set of checks to run. For example, an SCA check can inspect the file system configuration, check for the presence of a

software update or security patch, verify that the local firewall is enabled, identify services that are running unnecessary execution or check the password policy of users. Policies for SCA scans are written in YAML format, allowing users to understand them quickly. Using the SCA syntax, users can extend existing policies to meet their needs or write new ones. Each policy contains a set of controls, and each control has one or more rules. For example, a rule can be used to check for the existence of a file, directory, Windows registry key, or running process, among other things. It is also possible to run a command and check its output against a regular expression.

- Example Linux SCA rule:

- id: 5546

title: "Ensure IP forwarding is disabled"

description: "The net. ipv4.ip_forward flag is used to tell the system whether it can forward packets or not."

rationale: "Setting the flag to 0 ensures that a system with multiple interfaces (for example, a hard proxy), will never be able to forward packets, and therefore, never serve as a router."

remediation: "Set the following parameter in /etc/sysctl.conf or a /etc/sysctl.d/* file: net.ipv4.ip_forward = 0 and set the active kernel parameters."

Compliant :

-cis:["3.1.1"]

- cis_csc:["3", "11"]

- pci_dss: ["2.2.4"]

- nist_800_53: ["CM.1"]

condition: all

rules:

- 'c:sysctl net.ipv4.ip_forward -> r:^net.ipv4.ip_forward\s*=\s*0$'

- 'c:grep -Rh net\.ipv4\.ip_forward /etc/sysctl.conf /etc/sysctl.d -> r:^net.ipv4.ip_forward\s*=\s*0$'

- Example Windows SCA rule:

- id: 14038

title: "Ensure Microsoft Firewall is enabled"

Compliant :

- pci_dss: ["10.6.1", "1.4"]

- hipaa: ["164.312.b", "164.312.a.1"]

- nist_800_53: ["AU.6", "SC.7"]

-tsc: ["CC6.1", "CC6.8", "CC7.2", "CC7.3", "CC6.7"]

condition: all

rules:

- **'r:HKEY_LOCAL_MACHINE\software\policies\microsoft\windowsfirewall\domainprofile -> enablefirewall -> 1'**

- Example macOS SCA rule:

- id: 8522

title: "Ensure nfs server is not running"

description: "macOS can act as an NFS fileserver. NFS sharing could be enabled to allow someone on another computer to mount shares and gain access to information from the user's computer. File sharing from a user endpoint has long been considered questionable and Apple has removed that capability from the GUI.NFSD is still part of the Operating System and can be easily turned on to export shares and provide remote connectivity to an end user computer."

rationale: "File serving should not be done from a userdesktop, dedicated servers should be used. Open ports make it easier to exploit the computer."

remediation: "Ensure that the NFS Server is not running and is not set to start at boot Stop the NFS Server: sudo nfsd disable Remove the exported Directory listing: rm /etc/export"

compliance:

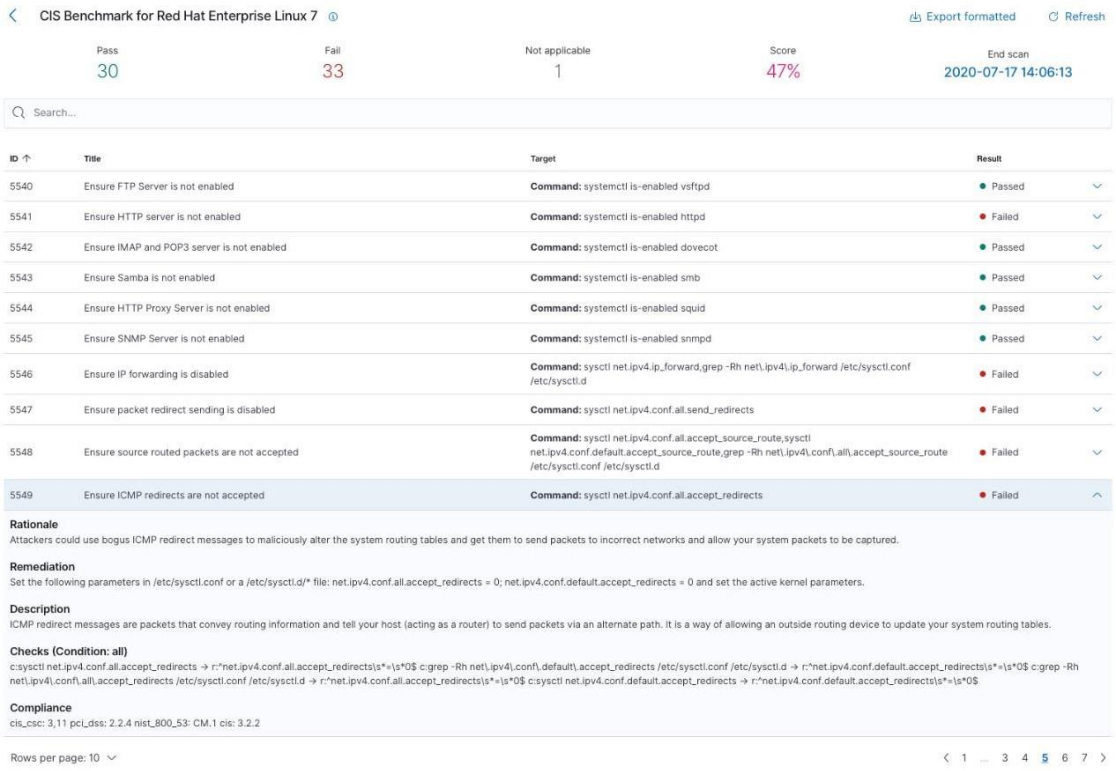
- cis: ["4.5"]

condition: none

rules:

- **'p:nfsd'**
- **'p:/sbin/nfsd'**
- **'f:/etc/exports'**

Below is an example of the results of a configuration assessment. These can be obtained through the web UI or dir ectly through the Wazuh RESTful API.



Picture 7 : Visualization of configuration assessment by Wazuh

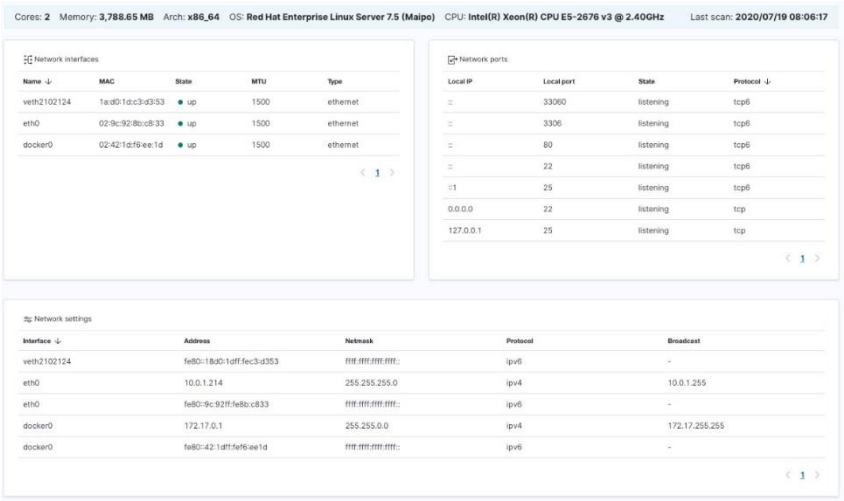
6- System inventory

The Wazuh Agent System Inventory module collects hardware and software information from the monitored system. This capability helps identify resources and assess the effectiveness of patch management.

The inventory data collected, for each of the monitored endpoints, can be queried through the Wazuh RESTful API and from the web UI. This includes memory usage, disk space, processor specifications, network interfaces, open ports, running processes, and a list of installed applications.

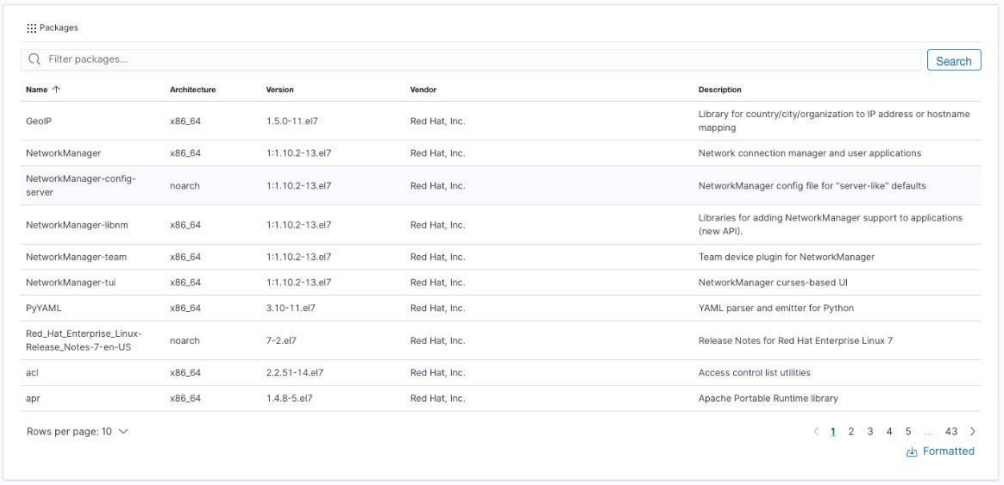
In order to collect the data, the Wazuh agent runs periodic scans (the time interval is configurable). After the scan completes, the agent compares the new inventory data with the old from the previous scan. In this way, the agent identifies system events, for example when a new port has been opened, a process has been terminated or a new application has been installed.

Example of hardware inventory, network interfaces, open ports, and network settings:



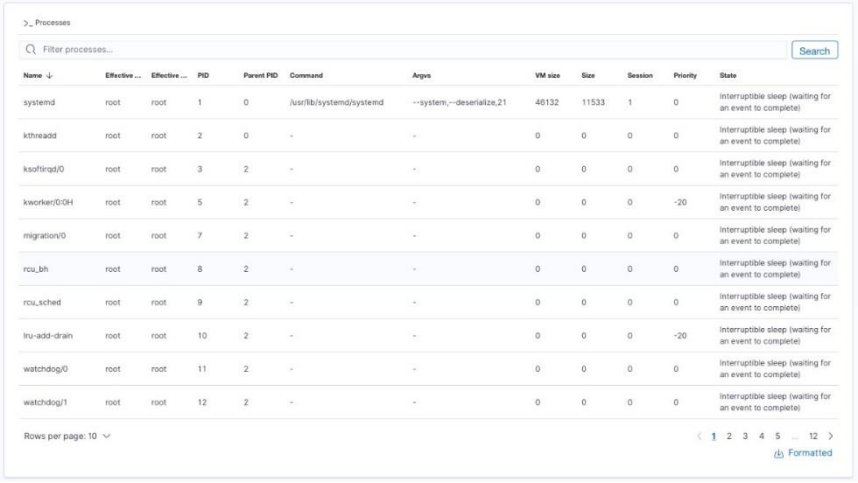
Picture 9: Viewing the system inventory of a wazuh agent

Example of software inventory:



Picture 10: Viewing Wazuh Agent Software Inventory

Example of a running process:



Picture 11: Viewing the inventory of processes running on a Wazuh agent

7- Vulnerability detection

Vulnerable software applications are typically targeted by attackers to compromise endpoints and gain a persistent presence on targeted networks. The Wazuh platform, leveraging its software inventory capabilities, maintains an up-to-date list of allapplications installed on endpoints with the Wazuh agent installed. By correlating this information with the National Vulnerability Database (NVD) and with information collected from different operating system vendors, Wazuh is able to identify vulnerable applications and produce risk reports.

In order to detect vulnerable software, Wazuh uses a database of common vulnerabilities and exposures (CVE) created automatically using data extracted from the following sources:

- CVE for Ubuntu Linux distributions
- CVE for Red Hat and CentOS Linux distributions
- CVE for Debian Linux distributions
- National Vulnerability Database CVE
- Microsoft Security Response Center

In order to enable vulnerability detection, users must configure the agentWazuhto collect software inventory data, and theWazu serverto extract CVE information from different vulnerability streams. Here is an example of a vulnerability detection alert:

```
{
 "agent": {
 "id":"003",
 "ip":"10.0.1.102",
 "name":"Windows"
},
 "rental":"vulnerability-detector",
 "data": {
 "vulnerability": { "assign":"
 cve@mitre.org ","CV":"CVE-
 2020-12395",
 "cve_version":"4.0",
 "cvss": {
 "cvss2": {
 "base_score":"10",
 vector: {
 "access_complexity":"low",
 "attack_vector":network,
 "authentication":"none",
 "availability":"complete",
 "privacy_impact":"complete",
 "integrity_impact":"complete"
}
}
},
 "cwe_reference":"CWE-119",
 "package": {
 "architecture":"x86_64",
 "condition":"less than 68.8.0",
 "generated_cpe":"a:mozilla:thunderbird:68.0:::::x86_64:",
 "name":"Mozilla Thunderbird 68.0 (x64 en-US)",
 "version":"68.0"
},
 "published":"2020-05-26",
 "references": [
"https://bugzilla.mozilla.org/buglist.cgi?bug_id=1595886%2C1611482%2C1614704%2C1624098
%2C1625749%2C1626382%2C1628076%2C1631508",
```

"https://security.gentoo.org/glsa/202005-03",
"https://security.gentoo.org/glsa/202005-04",
"https://usn.ubuntu.com/4373-1/",
"https://www.mozilla.org/security/advisories/mfsa2020-16/",
"https://www.mozilla.org/security/advisories/mfsa2020-17/",
"https://www.mozilla.org/security/advisories/mfsa2020-18/","https://nvd.nist.gov/vuln/detail/CVE-2020-12395"

],

"severity":"High",

"title":"Mozilla developers and community members reported memory safety bugs present in Firefox 75 and Firefox ESR 68.7. Some of these bugs showed evidence of memory corruption and we presume that with enough effort some of these could have been exploited to run arbitrary code. This vulnerability affects Firefox ESR < 68.8, Firefox < 76, and Thunderbird < 68.8.0.", "updated":"2020-06-12"

}

},

"rules": {

"description":"CVE-2020-12395 affects Mozilla Thunderbird 68.0 (x64 en-US)",
"id":"23505",

"level":10

},

"timestamp":"2020-07-20T00:41:36.302+0000"

}

Picture 12: Vulnerability detection dashboard

8- Cloud Security Monitoring

The Wazuh Security Platform provides threat detection, configuration compliance, and continuous monitoring for multicloud and hybrid environments. It protects cloud workloads by monitoring the infrastructure at two different levels:

- **Endpoint level**: Monitoring cloud instances or virtual machines using Wazuh lightweight security agent.
- **Infrastructure level**cloud: monitoring cloud services and activity by collecting and analyzing provider API data. Amazon AWS, Microsoft Azure and Google Cloud Platform are supported.
    - ❖ Amazon web-service

The Wazuh agent also provides a module to monitor and secure your AWS c loud infrastructure. This module collects AWS service log data from S3 buckets and forwards the collected log messages to the Wazuh server, where events are analyzed using out-of-the-box Wazuh rules for AWS.

The following list describes the AWS services that Wazuh is able to monitor:

- **Amazon Guard duty**A threat detection service that continuously monitors malicious

activity and unauthorized behavior to protect your AWS accounts, workloads, and data stored in Amazon S3.

- **Amazon Inspector**: An automated security assessment service that helps improve the security and compliance of applications deployed on AWS.

- **Amazon Key Management Service (KMS):** used to create and manage cryptographic keys, and to control their use across a wide range of AWS services and in your applications.

- **Amazon Macie**: Fully managed data security and privacy service. It automatically detects unencrypted S3 buckets, publicly accessible buckets, and buckets shared with external AWS accounts.

- **Amazon Virtual private cloud (VPC)**: Provisions a logically isolated section of the AWS Cloud where AWS resources can be launched on a user-defined virtual network.

- **AWS Config**: analyze, audit and evaluate the configurations of your AWS resources. Config continuously monitors and logs configurations of AWS resources allowing automation to evaluate saved configurations against desired ones.

- **AWS Cloudtrail**: Enables governance, compliance, operational auditing, and risk auditing of your AWS account. With CloudTrail, you can record, continuously monitor, and persist stock-related account activity on your AWS infrastructure.
- **AWS Trusted Advisor**: Helps users reduce costs, increase performance, and improve security by optimizing their AWS environment. It provides real -time guidance to help users provision their resources according to AWS best practices.

- **AWS Web Application Firewall (WAF)**: Helps protect your web applications or APIs from common web exploits that can affect availability, compromise security, or consume excessive resources.

Example alert when an AWS security group is deleted:


```
{
        "agent": {
 "id":"000",
 "name":"wazuh-manager-master"
},
 "data": {
 "aws": { "awsRegion":"us-
 west-1",
 "aws_account_id": "1234567890",
 "eventID":"12ab34c-1234-abcd-1234-123456789",
 "eventName":"DeleteSecurityGroup",
 "eventSource":"ec2.amazonaws.com",
 "eventTime":"2020-08-06T15:13:07Z",
 "eventType":"AwsApiCall",
 "eventVersion":"1.05",
 "recipientAccountId":"0987654321",
 "requestID":"12345678-abcd-efgh-1234-
 123456789","requestParameters": {
 "groupId":"sg-12345678901234567"
},
 "responseElements": {
 "_return":"true",
 "requestId":"12345678-abcd-efgh-1234-123456789"
},
 "source":"CloudTrail",
 "sourceIPAddress":"cloudformation.amazonaws.com"
 ,"userAgent":"cloudformation.amazonaws.com",
 "userIdentity": {
 "accountId":"1234567890",
 "arn":"arn:aws:iam::1234567890:user/john.doe
 ",
 "invokedBy":"cloudformation.amazonaws.com",
 "mainId":"ABCDEFGHIJKLMNH",
 "sessionContext": {
 "attributes": {
 "creationDate":"2020-08-06T09:08:14Z",
```

"mfaAuthenticated":"false"

}

},

"kind":"IAMUser",
"userName":"john.doe
"

}

},

"integration":"aws"

},

"rules": {

"description":"AWS Cloudtrail: ec2.amazonaws.com - DeleteSecurityGroup.",
"id": "80202",

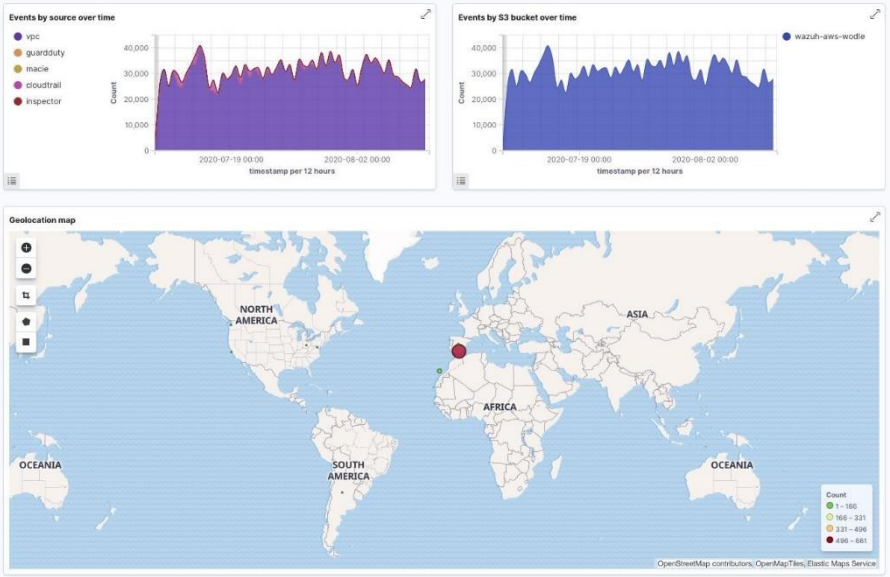
"level": 3

}

"timestamp": "2020-08-06T15:47:14.334+0000"

}

Example AWS Dashboard :



❖ **Microsoft Azure:**

The Wazuh agent module for Microsoft Azure makes it easy to extract logs from the Azure platform. In particular, it is designed to obtain data from the following services:

- **Log Analytics API**: The Log Analytics API is a central component of the Azure Monitor service, which is used to aggregate and analyze log data. The sources of this data are cloud applications, operating systems, and Azure resources. The Wazuh module for Azure is able to query the Log Analytics API, extracting logs collected by the Azure monitoring service.

- **Blob storage API**: Azure service logs are optionally forwarded to Azure Blob Storage. Concretely, it is possible to configure an Azure service to export logs to a container in a storage account created for this purpose. Then the Wazuh agent will download these logs through its integration with the Blob Storage API.

- **Active Directory Graph API**: The Azure Active Directory Graph API provides access to AZURE AD through REST API endpoints. It is used by Wazuh to monitor Active Directory events (e.g., creating a new user, updating a user's properties, disabling a user's account, etc.)

Here is an example rule alerted by Azure.


```
{
 "agent": {
 "id":"000",
 "name":"wazuh-manager-master-0"
},
 "data": {
 "Category":"Administrative",
 "ResourceProvider":"Microsoft.Compute",
 "TenantId":"d4cd75e6-7i2e-554d-b604-3811e9914fea",
 "ActivityStatus":"Started",
 "Kind":"AzureActivity",
 "OperationId":"d4elf2e7-65d8-2824-gf44-37742d81c00f",
 "ResourceId":"/WazuhGroup/providers/Microsoft.Compute/virtualMachines/Logstash",
 "OperationName":"Microsoft.Compute/virtualMachines/start/action",
 "CorrelationId":"d4elf2e7-65d8-2824-gf44-37742d81c00f",
 "Resource":"Logstash",
 "Level":"Informational",
 "Call":" john.doe@email.com ",
 "TimeGenerated":"2020-05-25T15:43:16.52Z",
 "ResourceGroup":"WazuhGroup",
 "SubscriptionId":"v1153d2d-ugl4-4221-bc88-40365el115gg",
 "EventSubmissionTimestamp":"2020-05-25T15:43:36.109Z",
 "CallerIpAddress":"83.49.98.225",
 "EventDataId":"69db115c-45ds-664b-4275-a684a72b8df2",
 "SourceSystem":"Azure"
},
 "rules": {
 "description":"Azure: Log analytics:
 Microsoft.Compute/virtualMachines/start/action","id":"62723",

 "level":3

},

 "timestamp":"2020-05-25T15:45:51.432+0000"

}
```


- **Google Cloud Platform:**

Wazuh monitors Google Cloud services by pulling events from the Google Pub/Sub messaging service, which is used as middleware for event ingestion and delivery. This integration helps detect threats targeting your Google Cloud resources.

The following example shows an alert generated when a known bad actor (a low reputation source IP address) tries to get a list of pods running in Google Kubernetes Engine (GKE):

```
                    {
                    "agent": {
                "id":"000",
                "name":"wazuh-manager-master"
                    },
                    "data": {
                    "insertId":"b2c2e792-aaa9-4422-82d0-
                    de32940b1234","labels": {
 "authorization": {
 "k8s": {
 "io/decision":"allow"
}
}
                    },
                    "logName":"projects/gke-audit-
logs/logs/cloudaudit.googleapis.com%2Fdata_access"
,
                    "operation": {
                "first":"true",
                "id":"b2c2e792-aaa9-4422-82d0-de32940b1234",
                "last":"true",
                 "to produce":"k8s.io"
                    },
                    "protoPayload": {
                "@kind":"type.googleapis.com/google.cloud.audit.AuditLog",
                "authenticationInfo": {
 "mainEmail":" john.doe@email.com "
},
 "authorizationInfo": [{
 "granted":true,
 "permission":"io.k8s.core.v1.pods.list",
 "resource":"core/v1/namespaces/default/po
 ds"
}],
 "methodName":"io.k8s.core.v1.pods.list"
 ,"requestMetadata": {
 "callerIp":"35.195.195.195",
```

```
"callerSuppliedUserAgent":"kubectl/v1.18.6 (linux/amd64) kubernetes/dff82dc"
},
"resourceName":"core/v1/namespaces/default/pod
s","serviceName":"k8s.io"
},
"receiveTimestamp":"2020-08-17T17:09:19.068723691Z",
"resource": {
"labels": {
"cluster_name":"wazuh",
"rental":"us-central1-c",
"project_id":"gke-audit-logs"
},
"kind":"k8s_cluster"
},
"timestamp":"2020-08-17T17:09:05.043988Z"
},
"rules": {
"description":"Malicious GKE request origin for io.k8s.core.v1.pods.list operation.",
"id":"400003",
"level":10
},
"timestamp":"2020-08-17T17:09:25.832+0000"
```

}

8- Container Security Monitoring

Wazuh is used to monitor signs of container security incidents, alerting in real time. Wazuh protects container workloads at two different levels:

- Infrastructure level

Wazuh provides the following mechanisms to monitor Docker hosts or Kubernetes nodes:

- **Integration with Docker engine and Kubernetes APIs:** in this scenario, the Wazuh module for Docker acts as a subscriber. It listens to Docker or Kubernetes events, being able to alert when an anomaly or a security incident is detected.
- **Wazuh agent deployment on Docker hosts and Kubernetes nodes:** for a self-managed infrastructure, wazuh agent deployment provides a comprehensive setof security features, such as malware detection, file integrity monitoring, configuration assessment, data analysis log, vulnerability detection and active responses.
- **Integration with hosted infrastructure providers (e.g., Google GKE, Amazon EKS, etc.):**in this case, the Wazuh modules for cloud security monitoring download the audit logs from the managed service for security analysis.

Table 3: Example of infrastructure-level security alerts

A Docker image is downloaded or updated	A container is running in privileged mode
Kubernetes configuration is changed	Hardening checks fail for Docker host
A new container or pod is created	A user executes a command or a shell inside a container
A new application is installed on the Docker host	Vulnerabilities are detected on the Docker host

- Container level:

In order to get container-level visibility, you can deploy the Wazuh agent to a Kubernetes DaemonSet container. This type of deployment ensures that the Wazuh agent will run on all nodes in your Kubernetes cluster. Also, other Kubernetes pods will be able to send data (e.g., application log messages) to the DaemonSet container, so that the agent can process it and pass it to the Wazuh server for security analysis.

Table 4: Example of container-level security alerts

New process create in a container	File Integrity Monitoring Alerts
New application installed in a container	Vulnerability detected in a container
Log analysis alert (e.g. Nginx event)	Hardening verification failed in a container

9- Regulatory conformity

The Wazuh platform is often used to meet the technical aspects of regulatory compliance standards. Wazuh not only provides the necessary security controls such as intrusion detection, configuration assessment, log analysis, vulnerability detection, among others to meet compliance requirements, but also utilizes its SIEM capabilities to centralize, analyze and enrich security data.

In order to provide regulatory compliance support, Wazuh rules have been carefully mapped to compliance requirements. So, when an alert is generated (a rule condition has been met), it automatically includes compliance information. Here is the list of currently supported standards:

- Payment Card Industry Data Security Standard (PCI DSS)
- General Data Protection Regulation (GDPR)
- NIST 800-53 Special Publication (NIST 800-53)
- Good Practice Guide 13 (GPG13)
- Trust Services Criteria (TSC SOC2)
- Health Insurance Portability and Accountability Act (HIPAA)

Additionally, Wazuh rules include mapping to the MITER ATT&CK Framework, which is used for alert taxonomy and to provide better security context. Here is an example of a detection rule used to detect access to forbidden directories in Apache web servers:

```
<rule id="30306" level="5">
<if_sid>30301</if_sid>
<id>AH01276</id>
<description>Apache: Attempt to access forbidden directory index.
</description>
<miter>
<id>T1190</id>
</mitre>
access_denied tsc_CC6.1,tsc_CC6 .8,tsc_CC7.2,tsc_CC7.3,</group>
</rule>
```

Example alert for the rule:

Apache: Attempt to access forbidden directory index.

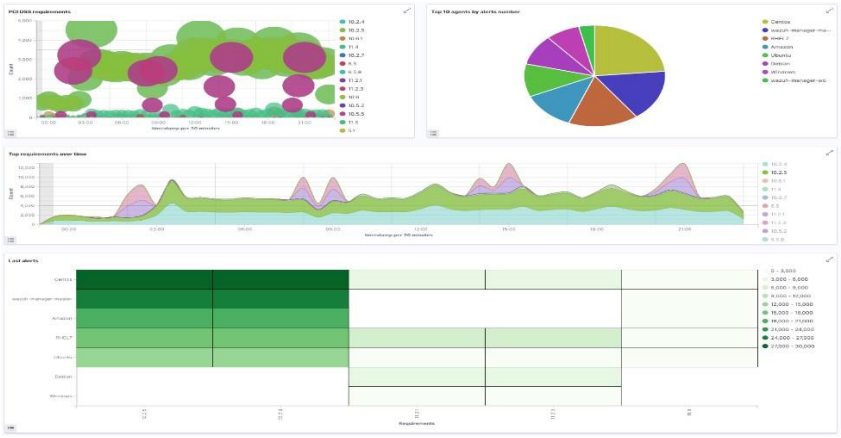
output


```
{
 "agent": {
 "id":"006",
 "ip":"10.0.1.214",
 "name":"RHEL7"
},
 "data": {
 "id":"AH01276",
 "scip":"24.4.35.192",
 "sport":"61844"
},
 "full_log":"[Fri Sep 04 06:08:51.973988 2020] [autoindex:error] [pid 28811] [client 24.4.35.192:61844] AH01276: Cannot serve directory /var/www/html/: No matching Directory Index (index.html ) found, and server-generated directory index forbidden by Options directive",
 "rental":"/var/log/httpd/error_log
 ","rules": {
```

```
"miter": {
"id": [
"T1190"
],
"tactical": [
"Initial Access"
],
"technical": [
"Public-Facing Application Exploit"
]
},
"gdpr": [
"IV_35.7.d"
],
"hipaa": [
"164.312.b"
],
"nist_800_53": [
"SA.11",
"AU.14",
"AC.7"
],
"pci_dss": [
"6.5.8",
"10.2.4"
],
"tsc": [
"CC6.6",
"CC7.1",
"CC6.1",
```

```
"CC6.8",
"CC7.2",
"CC7.3"
]
},
"timestamp":"2020-09-04T06:08:53.878+0000"
}
```

Sample Regulatory Compliance Dashboard for PCI DSS:



10- Generation of the Map of the current detection capacityof ourSIEM

i. Introduction

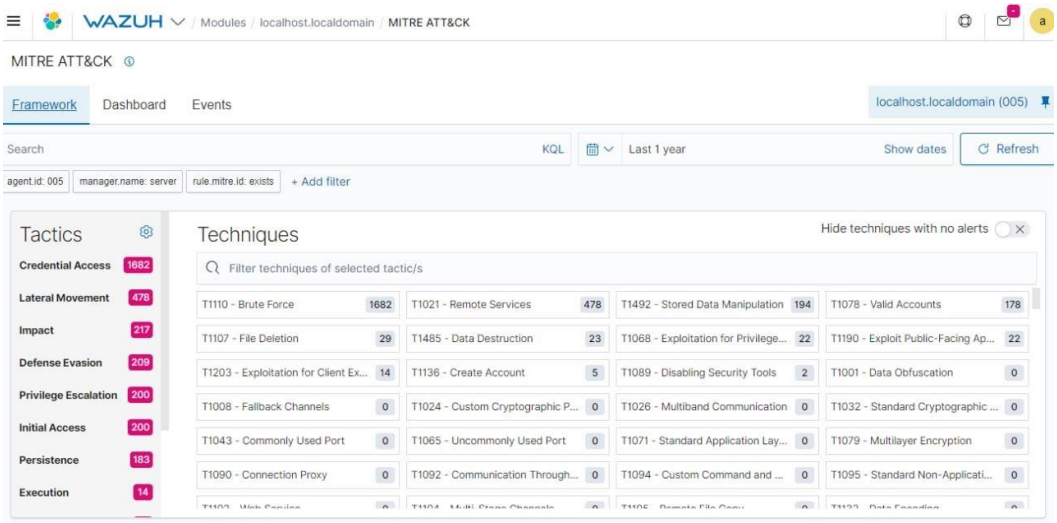
Attack emulation plays an important role in identifying the techniques, tactics, and procedures (TTPs) used by adversaries. Projects like Atomic Red Team (ART) can help automate emulation while conflicting activity can be detected using Wazuh.
The MITER ATT&CK Framework, which stands for MITER Adversial Tactics, Techniques, and Common Knowledge (ATT&CK), is a knowledge base for modeling the behavior of a cyber-adversary. However, it is useful to automate the process of different types of attack techniques in order to better prevent and defend our information system in the event of any compromise.

ii. Wazuh MITER ATT&CK Module

Wazuh comes with the out-of-the-box MITER ATT&CK module and threat detection rules mapped to their corresponding MITER technique IDs. This module has four components which are:

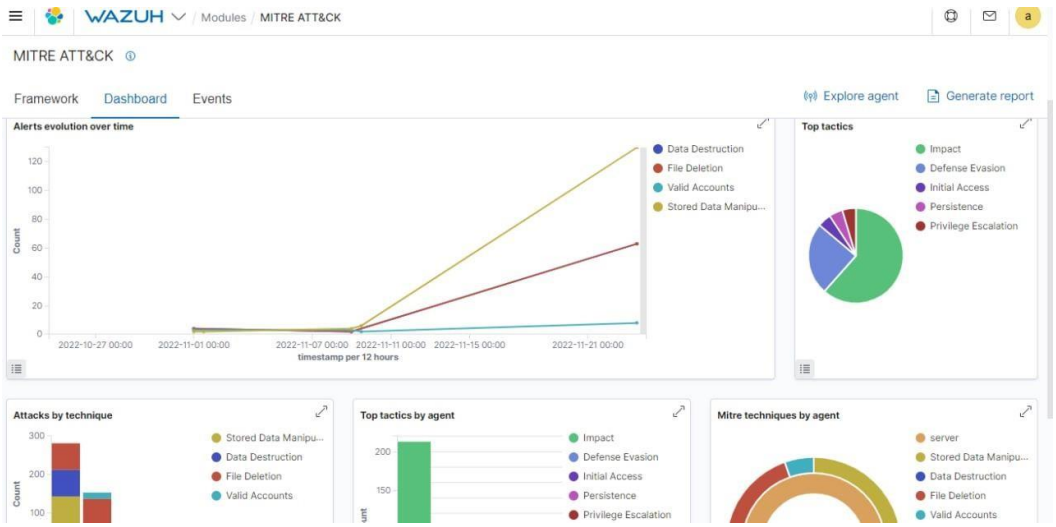
1. **The intelligence component of the Wazuh MITER ATT&CK module: Contains**detailed information on threat groups, mitigation, software, tactics and techniques used in cyberattacks. This component helps threat hunters identify and classify the different TTPs used by adversaries.

2. **The Framework component of the Wazuh MITER ATT&CK module: helps**threat hunters to reduce threats or compromised endpoints. This component uses specific techniques to see all events related to this technique and the endpoints where these events occurred.



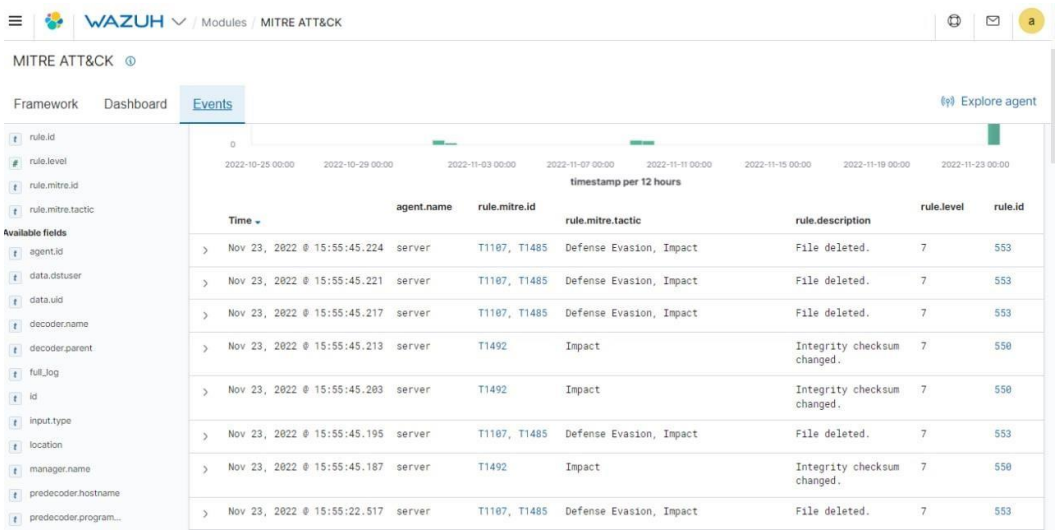
Picture 13: Wazuh SCA Dashboard

3. **The dashboard component of the MITER ATT&CK module: Helps to** summarize all events in graphs to help threat hunters get a quick overview of MITER-related activities in an infrastructure



Picture 14: Wazuh MITER ATT&CK frame

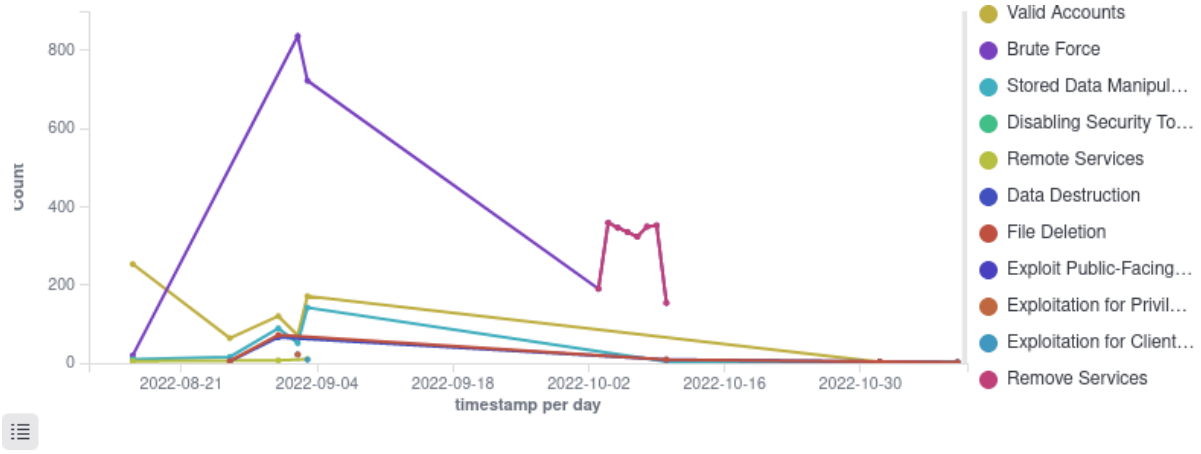
4. **The Wazuh MITER ATT&CK event component: Displays the** real-time events, with their respective MITER IDs, to better understand each reported alert.



Picture 15: Events Wazuh MITER ATT&CK

iii. Map of the detection capacity of our SIEM

❖ **<u>Evolution of MITER alerts (the analysis will be done in number of alerts associated with the timestamp per day):</u>**

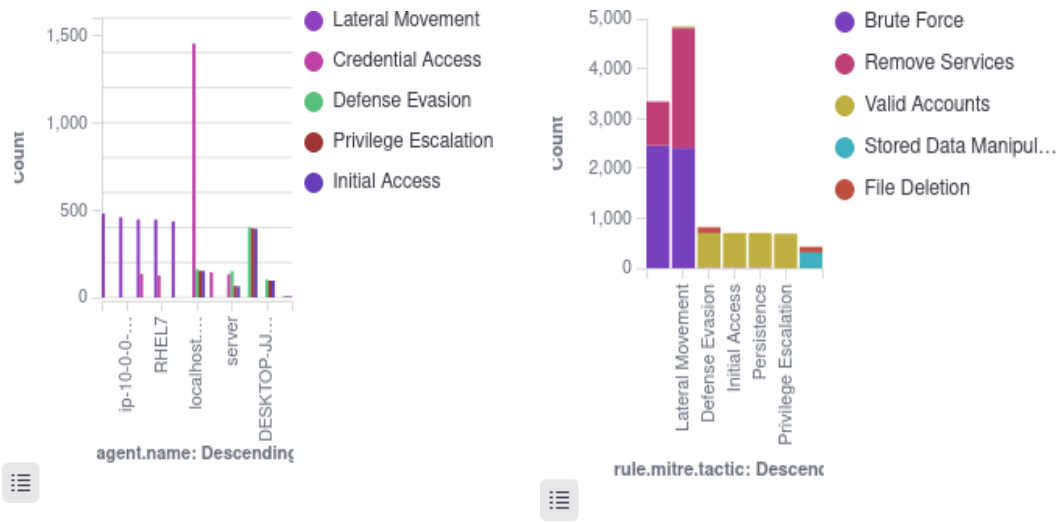


Picture 16: Alert graph generated by MITER ATT&CK

➢ **Analysis of the graph (Case of alerts of attempts by Brute Force)**

In our case, referring to our graph if we take for example the brute force attack which is associated with the purple color, we can see that the number of brute force attempt alerts varies from 0 to 800 from 11 /08/22 to 04/09/22 however from 04/09/22 to 02/10/22 it decreases from 700 attempts to 200 attempts!

❖ **Overview of the tactics and techniques most used by agents:**



Picture 17: Graphs of tactics and techniques generated by MITER ATT&CK

➢ Analysis of the graphs above:

On the left we have the most used tactics on our agents according to the mapping of the MITER Framework which are specified at the bottom of the graph and on the right we have the most used techniques on our agents also specified at the bottom of the graph.

- Tactical sides when we take for example access to credentials named Credential Access on our graph, we can see that it was detected about 1400 times on our Red Hat agent named localhost. localdomain on the graph!

- Technical sides when we take for example lateral movement which is the process by which attackers spread from an entry point to the rest of the network, raw we see according to MITER Framework mapping that this technique has been used over 2000 times to perform the brute force attack on our agents. However, this same technique has been used about 5000 times to perform the service removal attack!

iv. Conclusion

The MITER ATT&CK framework helps classify and properly identify threats based on discovered TTPs. Wazuh uses its dedicated MITER ATT&CK components to display information about how endpoint security data matches TTPs. Wazuh's threat hunting capabilities help cybersecurity analysts detect apparent cyberattacks as well as underlying infrastructure compromises.

b. Other Use Cases

- Telegram bot for sending alerts from our information system in real time

Wazuh can also allow us to send alerts from our information system in real time via an instant messaging application of our choice, namely Gmail, Telegram, etc. In our case we had to integrate an API allowing us to manage alerts from our information system in real time via Telegram!

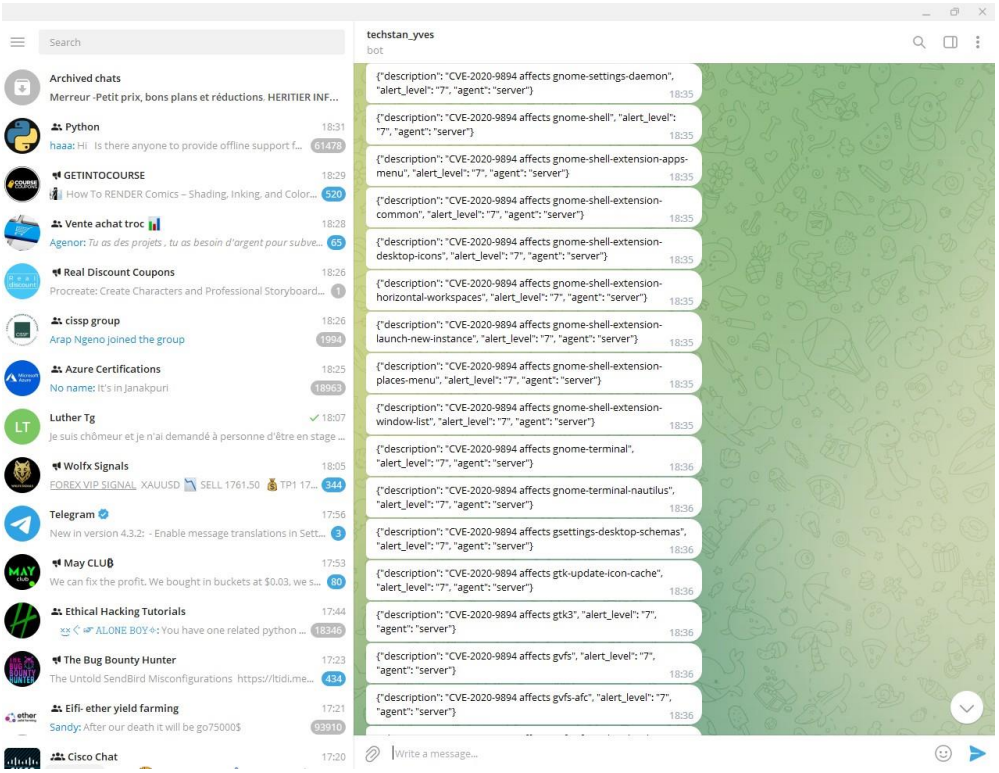
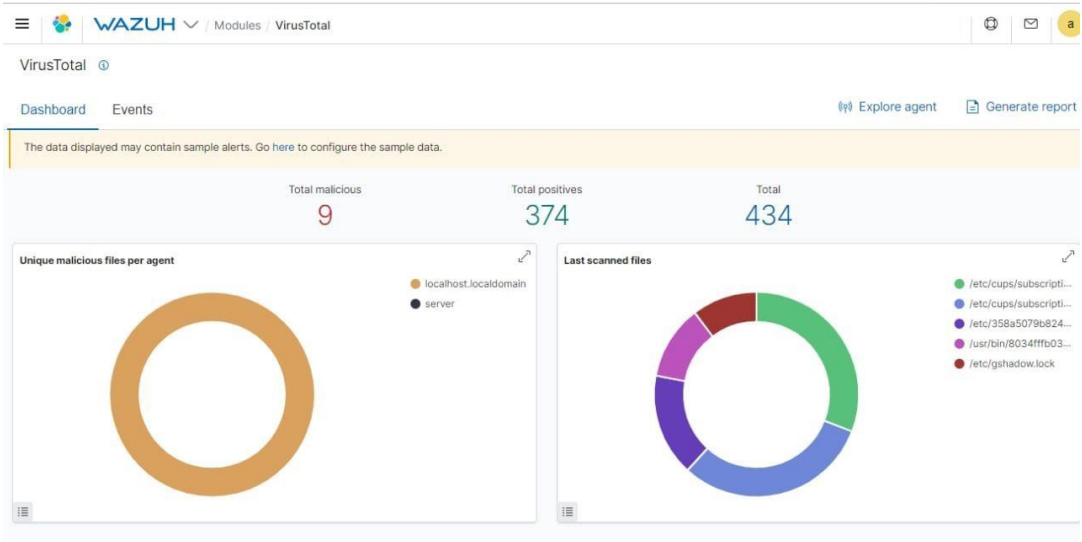


Figure 18: Capture of alerts generated by the Telegram bot

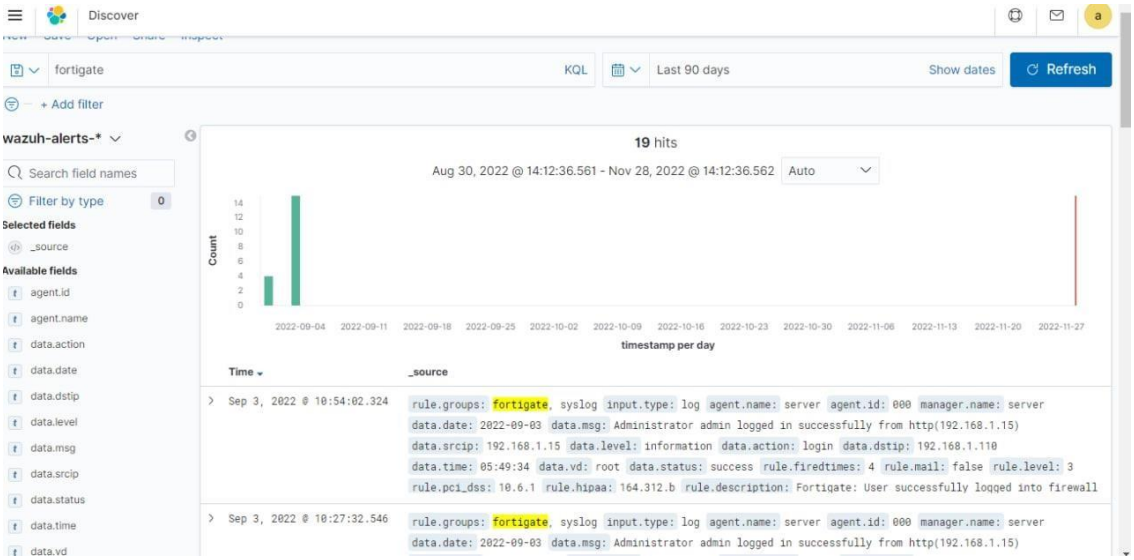
- Integration of Virus Total into our SIEM

Virus Total is an online service owned by Google that allows the analysis ofsuspicious files and facilitates the rapid detection of viruses, worms, Trojan horses and all kinds of malware detected by antivirus engines. In our case, too, we have integrated the Total virus API into our SIEM in order to facilitate the detection of malicious files in our information system!



Picture 19 : Total Virus Dashboard Capture

- Monitoring a firewall
  Wazuh Grace can be used to monitor certain network infrastructures, namely: routers, switches, firewalls, etc. In our case we have monitored one of the most used firewalls in information systems today, namely Fortigate.



Picture 20: Capturing the kibana interface while monitoring the Fortigate firewall