VM User Behavior Analysis using OSSEC on SAVI Testbed

Rajsimman Ravichandiran
Kristina Dzeparoska

Executive Summary

This project aims to showcase the need for Host-based Intrusion Detection Systems (HIDS) which allow internal host inspection, as opposed to network-based intrusion detection systems. OSSEC is a scalable, multi-platform, open source HIDS which is capable of performing powerful correlation and analysis, file integrity checking, centralized policy enforcement, rootkit detection, real-time alerting and active responses. Our project's main focus is to extend the capabilities of OSSEC to perform User Behaviour Analysis based on shell commands entered on the hosts. We implement a Supervised Machine Learning Model (Naive Bayes) to profile users. We also utilize public UNIX dataset produced by Purdue University to train the model. When evaluated, our model's prediction accuracy is 69%. Although, better than weak classifiers, it can be definitely improved by using larger datasets or complex models. Moreover, the responses can be expanded from notifying the system administrators to blocking malicious code execution or blocking intruders on victim hosts.

Introduction

Security is a major challenge in the cloud environment requiring diverse approaches to achieve fortified protection at various layers in systems. Currently, Network based IDS (such as Snort) is prominent in industries where it is used for inspecting network traffic and for protection purposes. However, a major flaw to this approach exists; if malicious users are able to bypass NIDS, there are no defense mechanisms in place to detect and remove unwanted threats.

Hence, we consider Host-based Intrusion Detection System (HIDS) to monitor the systems. Since it resides inside the system, it provides abundant information about the system and it's current states (such as call distributions, memory usage etc.).

OSSEC provides the following: log analysis, integrity checking, rootkit detection, time-based alerting, and active response. Furthermore, it has a centralized, cross-platform architecture allowing multiple systems to be easily monitored and managed. It consists of a main application, agents, and a web interface.

Project Description

In this project, we extend OSSEC functionalities to provide User Behaviour Analysis on SAVI Testbed. We implement a Machine Learning technique (Naive Bayes Supervised Learning model) to detect unauthorized users on the VMs.

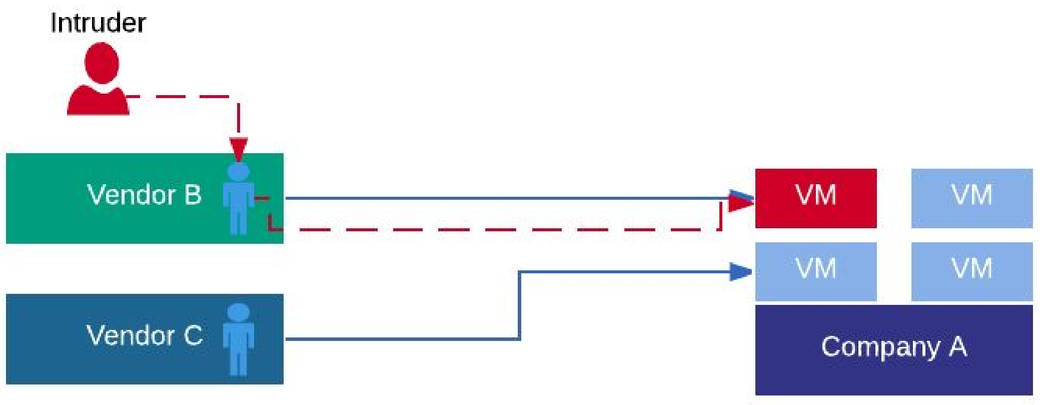
Project Use-case

Our primary purpose of our project is to address the issue of identity detection for users working on cloud infrastructures. If there are private companies that host cloud systems and share those resources with other vendors, it is critical for system admins (of that cloud owner company) to know who used the system and detect any suspicious behaviour.

For example, if company A owns a private cloud and vendors B and C deploy their products onto company A's cloud infrastructure, then system admins of company A must know the users interacting with the system.

Note: VMs created on the cloud will have 'root' user access for the vendors (for deploying products). Hence, detecting unauthorized access by using usernames is superfluous.

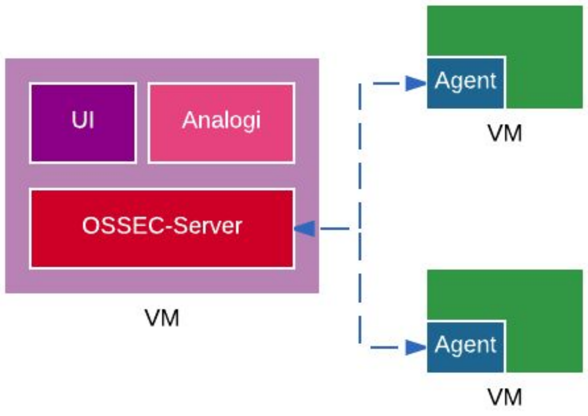
Therefore, we characterize the identity of the users based on how they interact with the system (i.e., user(s)' shell commands). By using this input, we can characterize the behaviour of the user and detect any malicious intent or anomalous behaviour.



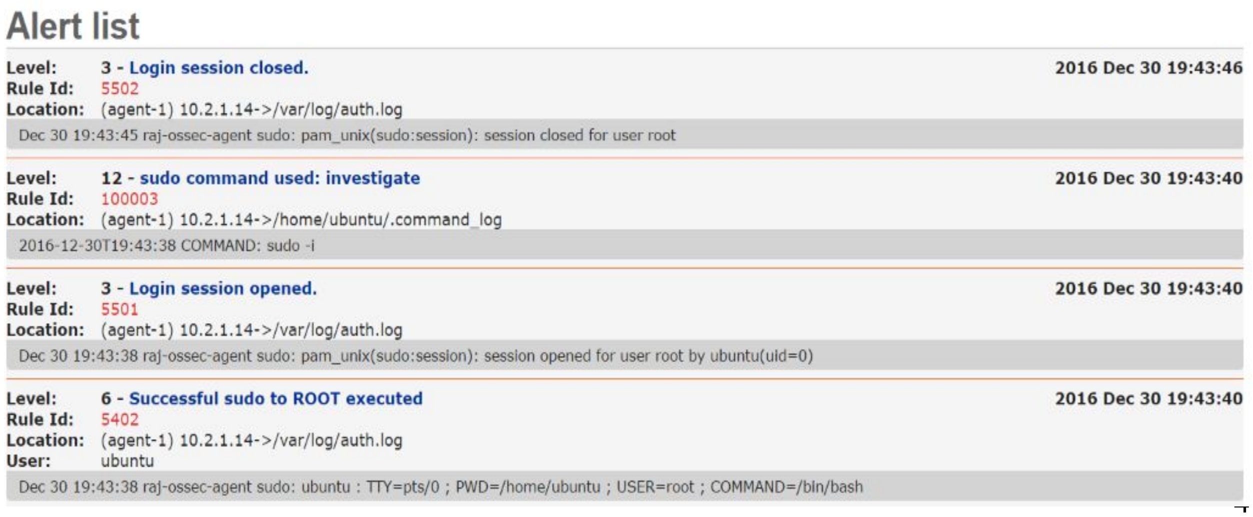
OSSEC background

OSSEC contains 2 main components of its architecture: server and agents. The server is the main framework that is responsible for scanning, monitoring, detecting and responding to threats found on hosts. The agent resides inside each VM to feed system data back to the server.

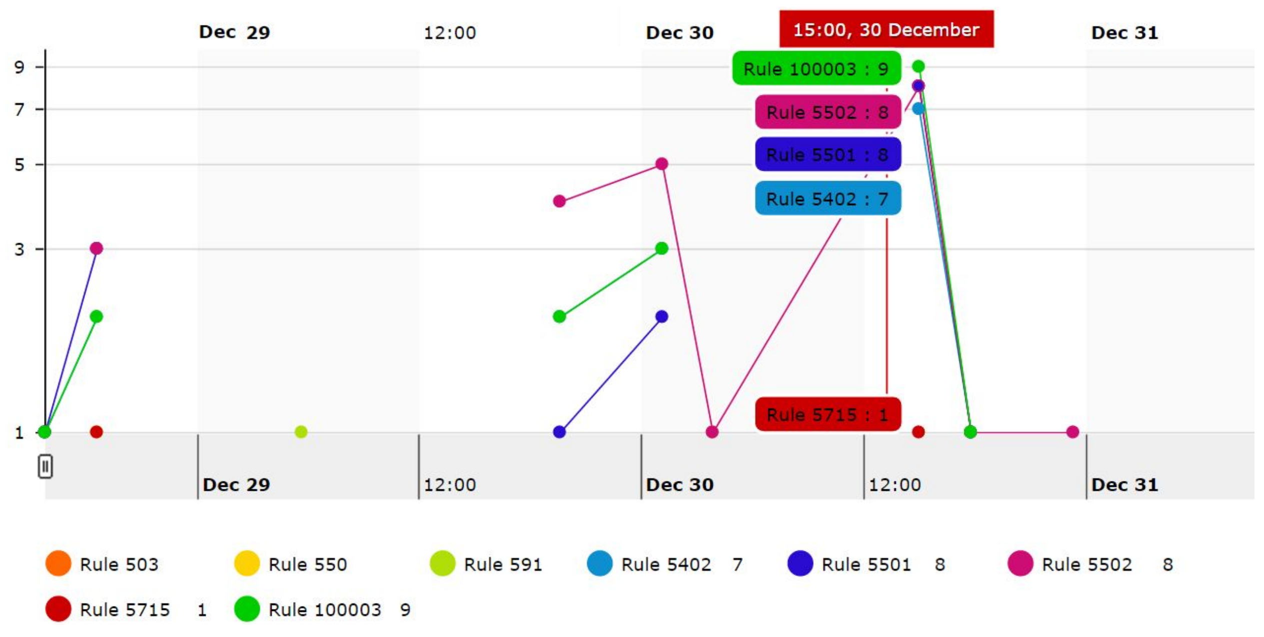
For our project, we deployed OSSEC server on a VM and agents to 2 VMs (as shown on the picture below).



Along with the server and agents, OSSEC contains two other components: UI and Analogi (as shown above). Both are available open source web user interfaces for OSSEC. OSWUI (OSSEC Web User Interface) is the standard web interface for OSSEC. While it offers great search functions, it does not offer graphical representations. AnaLogi (Analytical Log Interface), an additional web UI which offers an informative dashboard with visual information. A couple of diagrams of OSWUI and Analogi graphs are shown below. Detailed information regarding these components can be found on the Appendix.

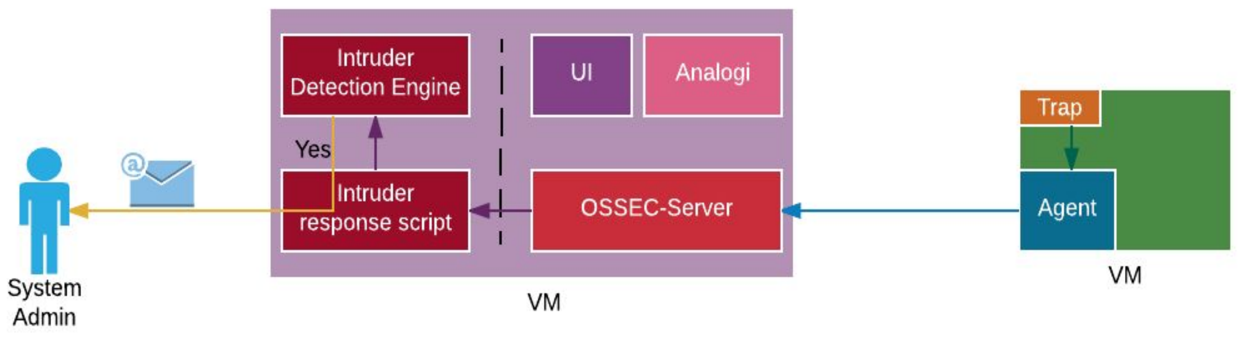


```
Alert list
Level:     3 - Login session closed.                                2016 Dec 30 19:43:46
Rule Id:   5502
Location:  (agent-1) 10.2.1.14->/var/log/auth.log
  Dec 30 19:43:45 raj-ossec-agent sudo: pam_unix(sudo:session): session closed for user root

Level:     12 - sudo command used: investigate                      2016 Dec 30 19:43:40
Rule Id:   100003
Location:  (agent-1) 10.2.1.14->/home/ubuntu/.command_log
  2016-12-30T19:43:38 COMMAND: sudo -i

Level:     3 - Login session opened.                                2016 Dec 30 19:43:40
Rule Id:   5501
Location:  (agent-1) 10.2.1.14->/var/log/auth.log
  Dec 30 19:43:38 raj-ossec-agent sudo: pam_unix(sudo:session): session opened for user root by ubuntu(uid=0)

Level:     6 - Successful sudo to ROOT executed                     2016 Dec 30 19:43:40
Rule Id:   5402
Location:  (agent-1) 10.2.1.14->/var/log/auth.log
User:      ubuntu
  Dec 30 19:43:38 raj-ossec-agent sudo: ubuntu : TTY=pts/0 ; PWD=/home/ubuntu ; USER=root ; COMMAND=/bin/bash
```



Platform Architecture

As mentioned before, we extend the capabilities of OSSEC to perform identity detection. We primarily use input shell commands to model user behaviour on the VM. Our detection engine is triggered only when a user enters a **sudo** command (i.e. *sudo netstat -tulpn*). As shown in the diagram below, we implemented 3 main components: Trap (on the monitored VM), Intruder Response script, and Intruder Detection Engine (on the OSSEC-Server VM). In the following sections, we describe each component in-detail.



Trap

In order to collect user input commands, we implement a script to log every command entered on the VM on a log file. We insert the following code snippet on the user's **bashrc** file (located on the monitored VM):

```
# log every command typed and when
if [ -n "${BASH_VERSION}" ]; then
    trap "caller >/dev/null || \
printf '%s\\n' \"\$(date '+%Y-%m-%dT%H:%M:%S')\
 COMMAND: \${BASH_COMMAND}\" 2>/dev/null >>~/.command_log" DEBUG
fi
```

This will record each command on a log file which we let OSSEC Server to monitor.

Intruder Response Script

OSSEC provides a method to create custom rules for detection purposes. Moreover, this allows us to execute custom scripts when an alert is triggered. In our project, we developed a shell script to perform a few steps:

1. Log the alert when a remote user (on the monitored VM) enters a **sudo** command
2. Send the command to Intruder Detection Engine to find any suspicious behaviour
3. If intruder is detected, notify system administrator

Implementation

The Intruder Response Script is entirely developed using a shell script. The script then executes the Intruder Detection Engine (written in python) along with the command. If the engine detects the intruder, the script notifies the system administrator (using linux sendmail command).

Intruder Detection Engine

The Intruder Detection Engine is used to learn and profile the users based on their historical data of input commands. The engine incorporates a supervised machine learning technique (Naive Bayes) to detect anomalies in the behaviour.

Assumptions

In this platform, we assume a couple of factors in order to accurately detect intruders:

1. We already have historical data of users' shell commands
2. We have a mapping of which user is responsible for the VM

Both these factors have advantages and disadvantages and will be explained in further sections.

Historical Data

In order for our model to perform accurate detection, it requires large datasets of input commands. For our project, we use a public UNIX User Data Set. This data set was part of 9 computer users from Purdue University. Therefore, we use this data set to train our model.

An example of the input command session is shown below:

```
**SOF** whoami pwd ls dir vi source <1> source <1> exit **EOF**
```

The dissection of the command session as follows:

SOF	- flag for start of session
<1>	- argument, such as file_name
EOF	- flag for end of session

Based on our experimentations, we realized that at least 200 command sessions is required (for each user) in order for our model to moderately predict the user.

Naive Bayes Model

Our core component of the detection engine is a simple supervised learning model used for text based classification. It is often used as a baseline method for text based categorization (spam detection, text frequencies etc.). Hence, we chose this model to classify input commands.

Model Accuracy Evaluation

Using the public data set, we trained our model to perform multi-class classification (for multiple users). We perform some data preprocessing techniques such as data sanitization (remove SOF, EOF commands, etc.) and uniform distribution of data set.

Furthermore, we perform cross-validation techniques between training, valid and testing datasets, when tuning hyperparameters for our model. This prevents our model to overfit based on only training data. With 5 kfold cross validation, we achieved an accuracy of 69.1% .

A Confusion Matrix shows how well our model classified accurately. An example of the Confusion Matrix for our model is shown on the next page.

```
             precision    recall  f1-score   support

          0       0.48      0.96      0.64        73
          1       0.88      0.37      0.52        38
          2       0.57      0.92      0.70        89
          3       1.00      0.83      0.91        53
          4       0.86      0.74      0.79        34
          5       0.94      0.49      0.65        63
          6       0.80      0.36      0.50        44
          7       0.79      0.73      0.76       103
          8       0.97      0.50      0.66        64

avg / total       0.78      0.69      0.69       561
```

Note: Each row represents the User ID in the dataset (9 in total)

The Precision score represents the fraction of retrieved instances that are relevant.
`Precision = True Positive / (True Positive + False Positive)`

The Recall score represents the fraction of relevant instances that are retrieved.
`Recall = True Positive / (True Positive + False Negative)`

The F1-score considers both Precision and Recall to compute the score.
`F1 = Precision * Recall / (Precision + Recall)`

From the confusion matrix and kfold cross validation , we can interpret that our model accuracy is **69%**. We are prone to higher False Negatives than False Positives because our Precision is higher than Recall scores (**78%** and **69%** respectively). This proves that our classifier is better than a weak classifier (>= **50%**), but, cannot be considered as a strong classifier. Stronger classifiers will have better Precision and Recall scores (> 80%) and will have less False Positives and False Negatives.

Our model can be definitely improved by either:
- using complex classifiers (neural networks), ensemble methods (random forests) or
- using larger amount of datasets (at least more than 1000 command sessions per user, instead of 200)

Implementation

The Intruder Detection Engine is completely developed in python. We use sklearn machine learning libraries to implement Naive Bayes model. As mentioned before, we use Public UNIX User Data set provided by 9 users from Purdue University.

Demo Workflow

In the demo presented on the video, we show how our model detects an intruder based on an entered sudo command on a monitored VM. In this section, we explain the workflow in detail.

Firstly, our model contains the following in-memory table (shown below) that maps respective users to the VMs. This can be interpreted as follows: User ID # 7 is the original root user for the VM: 10.2.1.14. Note, the User ID field is the same ID used on the public dataset. Hence, if a (sudo) command entered on the monitored VM is not classified as User ID # 7, then an intruder is inside the system. Once an intruder is detected, an email notification is sent to the system administrator of our infrastructure to start the in-detail investigation.

VM IP Address	User ID
10.2.1.14	7
10.2.1.15	4

Future Development

As mentioned in previous sections, our platform assumes that historical data is available and the model has the knowledge of which users are responsible for respective VMs. In the real deployment, both these factors have to be addressed.

In future development, we propose a model that learns based on unsupervised clustering algorithms which do not depend on complete historical datasets. The model can improve the accuracy as more commands are entered on the host in real-time. For the second issue, the Intruder Response Script can incorporate Openstack commands to gather more information about the VM and create an internal map of the user to a specific VM.

Furthermore, we can also improve the responses when the intruder is inside the system. Instead of just notifying the system administrator, the Intruder Response Script should either block the user or the execution of the entered command. This will prevent any maliciously intended execution on the victim host.

Conclusion

In this project, we deploy an HIDS system named OSSEC on Savi Testbed. We extend its functionalities to incorporate User Behaviour Analysis to detect intruder based on input shell commands on the hosts. We use machine learning techniques to implement the detection engine and its prediction accuracy is 69%. In future development, we propose to make a stronger classifier by using complex models or using larger datasets. We can also expand our responses to block code execution or boot users out of systems.

References

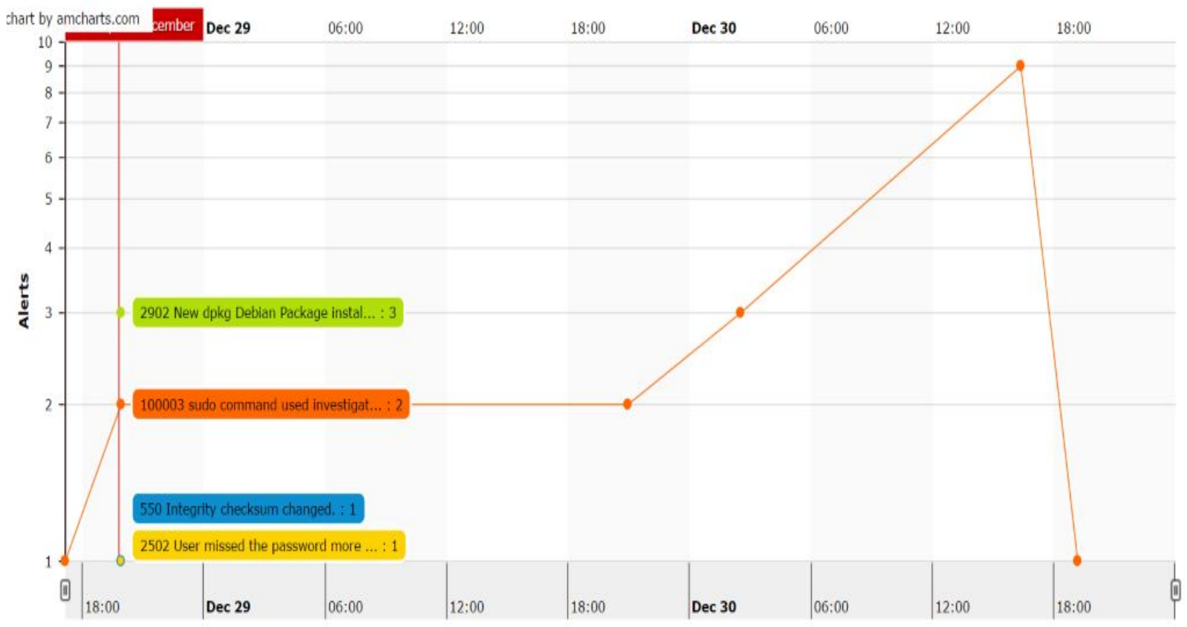
1. Public UNIX User Data Set - https://archive.ics.uci.edu/ml/datasets/UNIX+User+Data
2. Sklearn Naive Bayes Library - http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
3. OSSEC Homepage - http://www.ossec.net

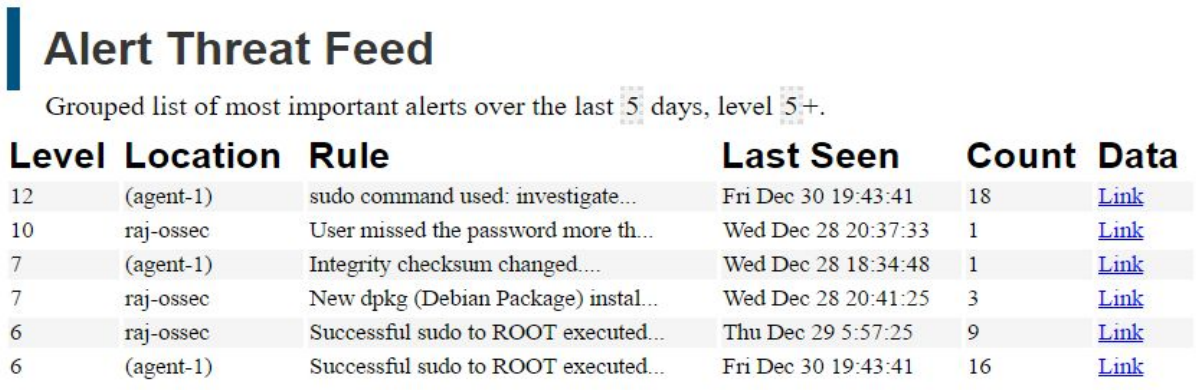
Appendex

AnaLogi

The main tab on the AnaLogi dashboard features the most recently triggered alerts and it offers filtering options for querying. The next tab groups the most important alerts triggered, the trending IPs and it offers trend analysis to be performed. The rest of the tabs provide different filtering and querying options to display data visually, to inspect the logs and export .csv files. The IP Info tab provides details and geo-location for a specific IP. The management tab offers management functionalities, such as overview of the most triggered alerts to investigate any rules that need fine-tuning, and additionally it gives a summary of the database. It allows to perform database cleanup and removal of locations that are no longer to be monitored by OSSEC.

The image below shows rules that have triggered in the period of Dec 29th - Dec 30th. Below the graphical representation we review a grouped list of the most important alerts triggered during the past five days.



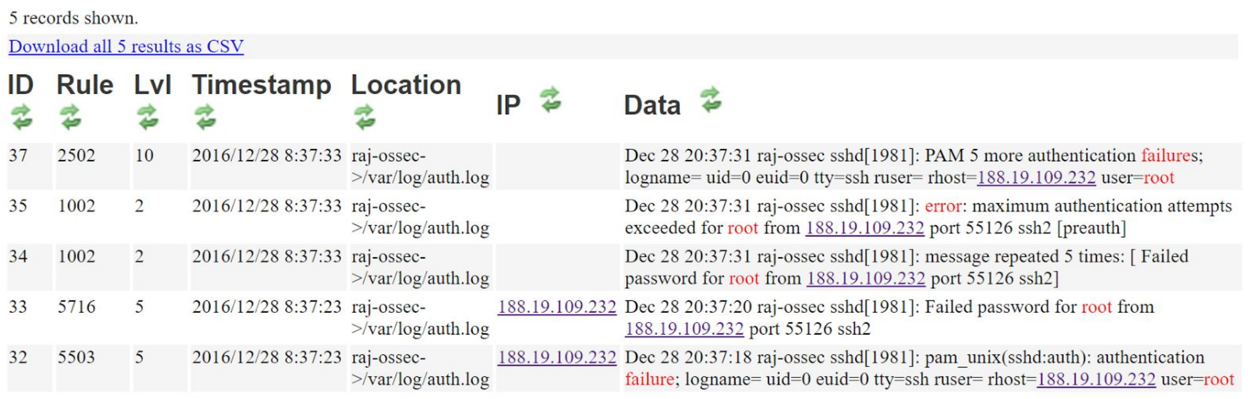


Alert Threat Feed

Grouped list of most important alerts over the last 5 days, level 5+.

Level	Location	Rule	Last Seen	Count	Data
12	(agent-1)	sudo command used: investigate...	Fri Dec 30 19:43:41	18	Link
10	raj-ossec	User missed the password more th...	Wed Dec 28 20:37:33	1	Link
7	(agent-1)	Integrity checksum changed....	Wed Dec 28 18:34:48	1	Link
7	raj-ossec	New dpkg (Debian Package) instal...	Wed Dec 28 20:41:25	3	Link
6	raj-ossec	Successful sudo to ROOT executed...	Thu Dec 29 5:57:25	9	Link
6	(agent-1)	Successful sudo to ROOT executed...	Fri Dec 30 19:43:41	16	Link

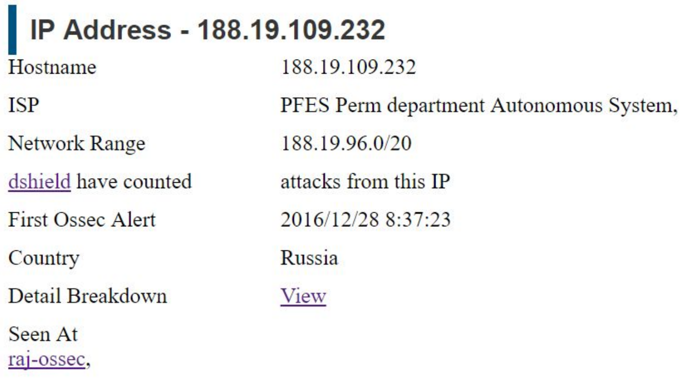
Initially, we assigned a floating IP to test the functionality of the web UI. We discovered during the brief online period we had been attacked by the IP below. This event was logged and reported by OSSEC. The next two images below are displaying the logs that were generated during the failed attempt. Additionally, we have released the floating IP and enabled port-forwarding for local and secure access of the web user interfaces.



5 records shown.

Download all 5 results as CSV

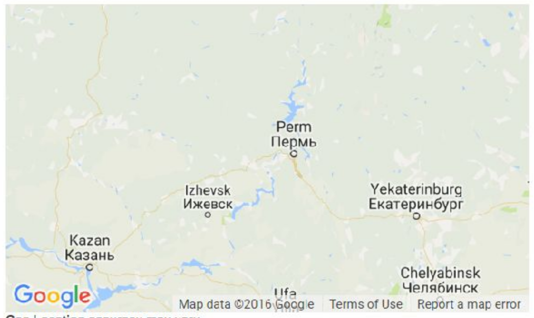
ID	Rule	Lvl	Timestamp	Location	IP	Data
37	2502	10	2016/12/28 8:37:33	raj-ossec->/var/log/auth.log		Dec 28 20:37:31 raj-ossec sshd[1981]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=188.19.109.232 user=root
35	1002	2	2016/12/28 8:37:33	raj-ossec->/var/log/auth.log		Dec 28 20:37:31 raj-ossec sshd[1981]: error: maximum authentication attempts exceeded for root from 188.19.109.232 port 55126 ssh2 [preauth]
34	1002	2	2016/12/28 8:37:33	raj-ossec->/var/log/auth.log		Dec 28 20:37:31 raj-ossec sshd[1981]: message repeated 5 times: [Failed password for root from 188.19.109.232 port 55126 ssh2]
33	5716	5	2016/12/28 8:37:23	raj-ossec->/var/log/auth.log	188.19.109.232	Dec 28 20:37:20 raj-ossec sshd[1981]: Failed password for root from 188.19.109.232 port 55126 ssh2
32	5503	5	2016/12/28 8:37:23	raj-ossec->/var/log/auth.log	188.19.109.232	Dec 28 20:37:18 raj-ossec sshd[1981]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=188.19.109.232 user=root



IP Address - 188.19.109.232

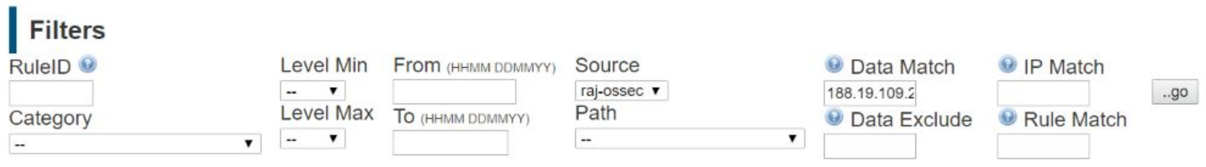
Hostname	188.19.109.232
ISP	PFES Perm department Autonomous System,
Network Range	188.19.96.0/20
dshield have counted	attacks from this IP
First Ossec Alert	2016/12/28 8:37:23
Country	Russia
Detail Breakdown	View

Seen At
raj-ossec,



Geo Location accuracy may vary

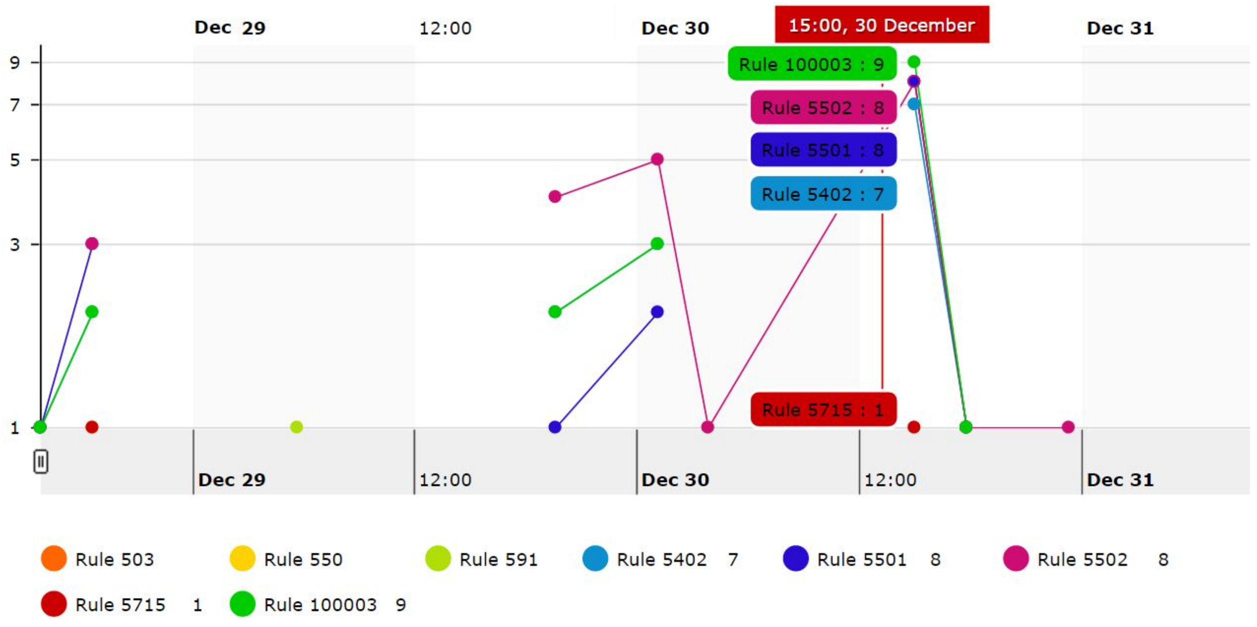
The next image provides an overview of the filtering options presented by AnaLogi.

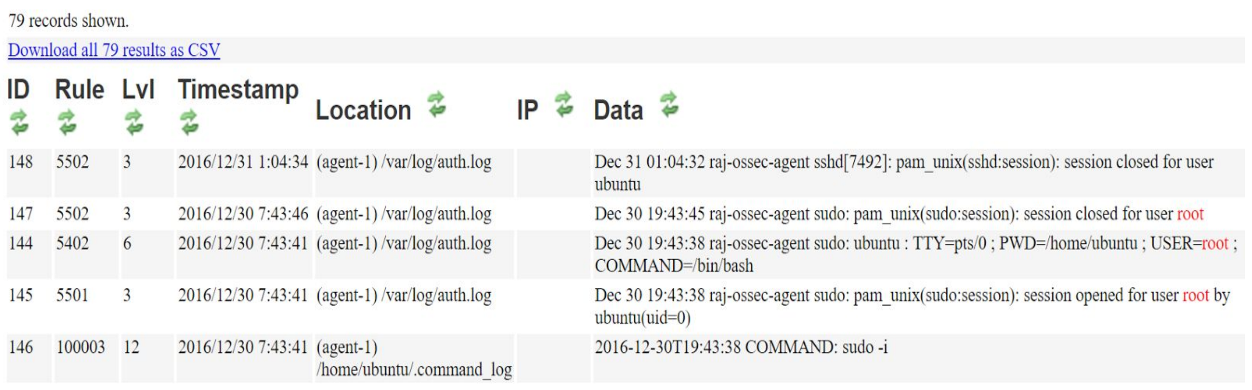


Filters

RuleID	Level Min --	From (HHMM DDMMYY)	Source raj-ossec	Data Match 188.19.109.2	IP Match	..go
Category --	Level Max --	To (HHMM DDMMYY)	Path --	Data Exclude	Rule Match	

Finally, we present one more graph with rules triggered for the location: **agent1** for a specific period (Dec 29th - Dec 31st). The graphical representation can be further expanded or narrowed down by the hour, the logs below the graph can be inspected and exported for more details.





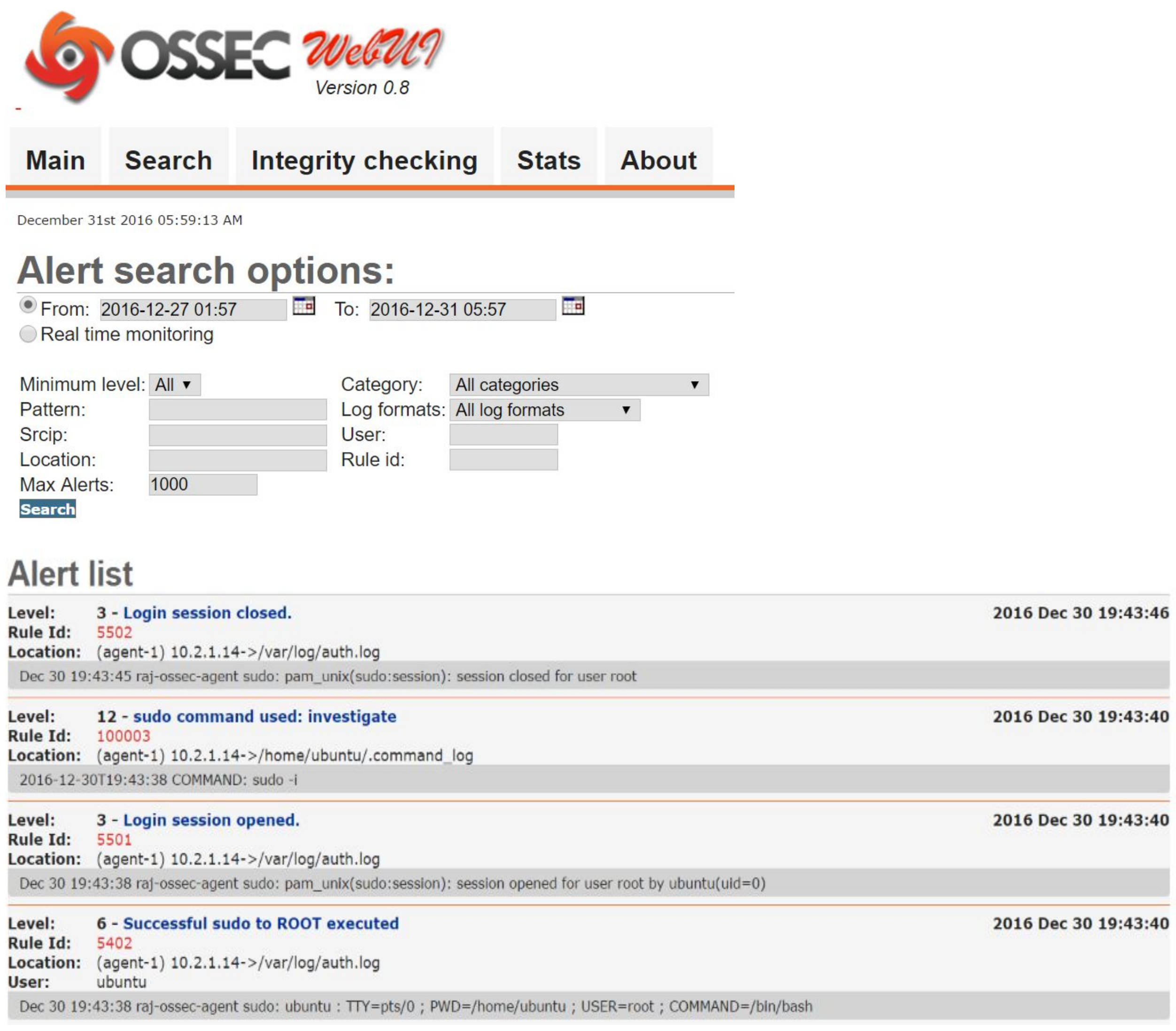
79 records shown.

Download all 79 results as CSV

ID	Rule	Lvl	Timestamp	Location	IP	Data
148	5502	3	2016/12/31 1:04:34	(agent-1) /var/log/auth.log		Dec 31 01:04:32 raj-ossec-agent sshd[7492]: pam_unix(sshd:session): session closed for user ubuntu
147	5502	3	2016/12/30 7:43:46	(agent-1) /var/log/auth.log		Dec 30 19:43:45 raj-ossec-agent sudo: pam_unix(sudo:session): session closed for user root
144	5402	6	2016/12/30 7:43:41	(agent-1) /var/log/auth.log		Dec 30 19:43:38 raj-ossec-agent sudo: ubuntu : TTY=pts/0 ; PWD=/home/ubuntu ; USER=root ; COMMAND=/bin/bash
145	5501	3	2016/12/30 7:43:41	(agent-1) /var/log/auth.log		Dec 30 19:43:38 raj-ossec-agent sudo: pam_unix(sudo:session): session opened for user root by ubuntu(uid=0)
146	100003	12	2016/12/30 7:43:41	(agent-1) /home/ubuntu/.command_log		2016-12-30T19:43:38 COMMAND: sudo -i

OSWUI

We proceed briefly by presenting the standard OSSEC dashboard - OSWUI, version 0.8. Both OSWUI and AnaLogi are using the same OSSEC database that was initially set up. While OSWUI does not offer graphical representations of data, it does offer better search queries to be performed. Also, it offers its form of real-time monitoring in the search options as shown on the image below. The next two images showcase the web UI and the search options, along with a partial view of search results obtained from within OSWUI.



December 31st 2016 05:59:13 AM

Alert search options:

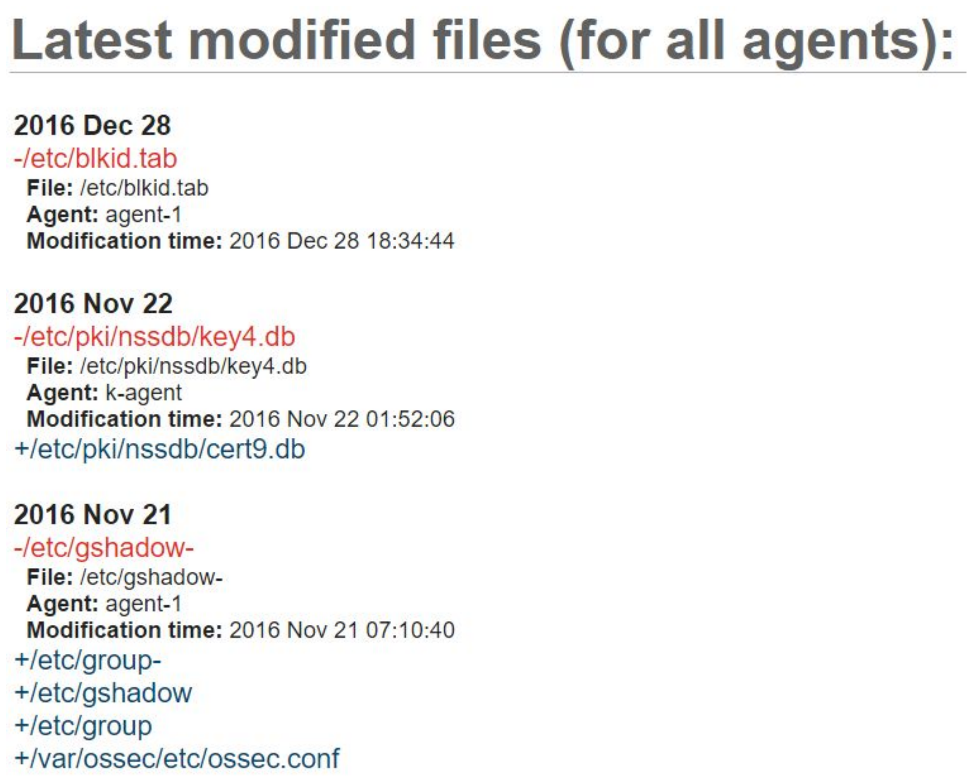
From: 2016-12-27 01:57 To: 2016-12-31 05:57
Real time monitoring

Minimum level:	All	Category:	All categories
Pattern:		Log formats:	All log formats
Srcip:		User:	
Location:		Rule id:	
Max Alerts:	1000		

Alert list

```
Level:    3 - Login session closed.                                    2016 Dec 30 19:43:46
Rule Id:  5502
Location: (agent-1) 10.2.1.14->/var/log/auth.log
Dec 30 19:43:45 raj-ossec-agent sudo: pam_unix(sudo:session): session closed for user root

Level:    12 - sudo command used: investigate                          2016 Dec 30 19:43:40
Rule Id:  100003
Location: (agent-1) 10.2.1.14->/home/ubuntu/.command_log
2016-12-30T19:43:38 COMMAND: sudo -i

Level:    3 - Login session opened.                                    2016 Dec 30 19:43:40
Rule Id:  5501
Location: (agent-1) 10.2.1.14->/var/log/auth.log
Dec 30 19:43:38 raj-ossec-agent sudo: pam_unix(sudo:session): session opened for user root by ubuntu(uid=0)

Level:    6 - Successful sudo to ROOT executed                         2016 Dec 30 19:43:40
Rule Id:  5402
Location: (agent-1) 10.2.1.14->/var/log/auth.log
User:     ubuntu
Dec 30 19:43:38 raj-ossec-agent sudo: ubuntu : TTY=pts/0 ; PWD=/home/ubuntu ; USER=root ; COMMAND=/bin/bash
```

OSWUI provides a list of the available agents, as well as lists of recently modified files through OSSEC's integrity checking service. The next two images offer a view of recently modified files from different locations (agents or servers) that are monitored by OSSEC, as well as a closer view of a specific location and its modified files.
In addition, OSWUI allows us to review the checksums (SHA1 and MD5) for all monitored files for each location by dumping data from the database. It also provides the view from before and after the change. Finally, OSWUI also allows statistics to be calculated and presented regarding total number of alerts, sources of trigger, daily statistics and similar.



Latest modified files (for all agents):

2016 Dec 28

-/etc/blkid.tab

File: /etc/blkid.tab
Agent: agent-1
Modification time: 2016 Dec 28 18:34:44

2016 Nov 22

-/etc/pki/nssdb/key4.db

File: /etc/pki/nssdb/key4.db
Agent: k-agent
Modification time: 2016 Nov 22 01:52:06

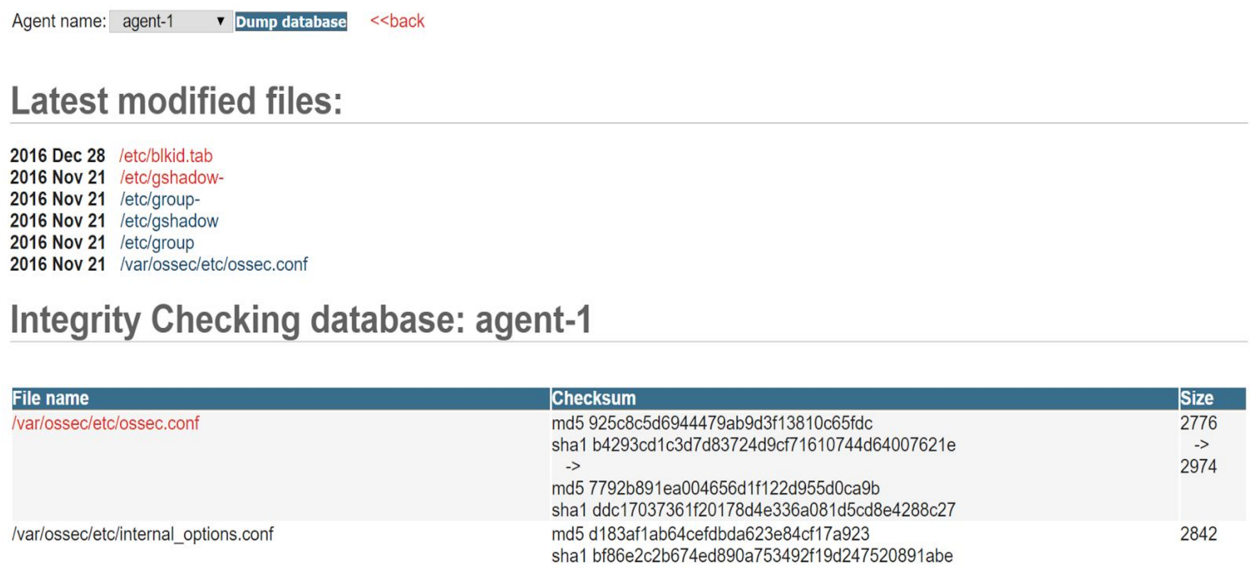
+/etc/pki/nssdb/cert9.db

2016 Nov 21

-/etc/gshadow-

File: /etc/gshadow-
Agent: agent-1
Modification time: 2016 Nov 21 07:10:40

+/etc/group-
+/etc/gshadow
+/etc/group
+/var/ossec/etc/ossec.conf



Latest modified files:

2016 Dec 28	/etc/blkid.tab
2016 Nov 21	/etc/gshadow-
2016 Nov 21	/etc/group-
2016 Nov 21	/etc/gshadow
2016 Nov 21	/etc/group
2016 Nov 21	/var/ossec/etc/ossec.conf

Integrity Checking database: agent-1

File name	Checksum	Size
/var/ossec/etc/ossec.conf	md5 925c8c5d6944479ab9d3f13810c65fdc sha1 b4293cd1c3d7d83724d9cf71610744d64007621e -> md5 7792b891ea004656d1f122d955d0ca9b sha1 ddc17037361f20178d4e336a081d5cd8e4288c27	2776 -> 2974
/var/ossec/etc/internal_options.conf	md5 d183af1ab64cefdbda623e84cf17a923 sha1 bf86e2c2b674ed890a753492f19d247520891abe	2842