Cybersecurity & SOC

The Security Operations Center Case

Bachelor in Electronics-ICT
Option: Cloud & Cybersecurity

Frederik Crauwels

Academic year: 2022-2023

Campus : Geel

LID VAN ASSOCIATIE KU LEUVEN

THOMAS MORE

Table of contents

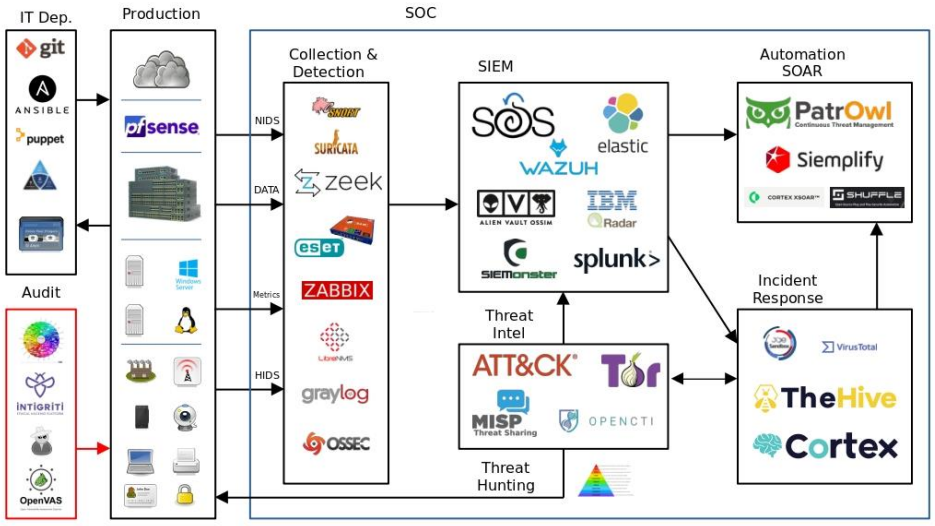
INTRODUCTION

This paper is an assignment for the course "Cybersecurity & SOC" at the Thomas More hogeschool. As one of the many subjects this project focusses primarily on blue team security and devops engineering. Simply put: a perfect combination of Cloud & Cybersecurity.

Devops (engineering) revolves around combining development and operations. It is a combination of people, processes and technology in order to create one, continuous stream of planning, coding, building, testing, implementing, deploying and monitoring. In essence DevOps revolves around automating as much as possible in order to generate a faster deployment and better application development. As this project requires the implementation of multiple software components, network(s) and assets, scalability, ease of (new) integration(s) and future-proofing the final solution are the most valuable aspects.

Security operations center (SOC) revolves around creating a safe and secure environment. Keywords are: visibility, security, protection and response. A SOC is mostly a combination of a multitude of abbreviations:

- **Hosts / Assets:** your hosts, or assets, are the cornerstone and frontline of your environment. These literally include ALL your personal / professional assets (end-user devices, servers, DC and even your entire SOC).
- **Log ingestion:** in order to obtain visibility – you need to obtain logging. Logging is available on both Windows and Linux assets in all kinds of shapes and forms.
- **SIEM:** Security Information and Event Management combines all ingested logs into one, "simple" environment. Every single source of logging and information coming from your assets are integrated in this solution.
- **SOAR:** Security Orchestration, Automation and Response provides "DevOps" in security. SOAR will make your SOC an automated powerhouse in which the opportunities are endless. Automatically handle events and incidents by implementing more third party solutions for further analysis – from IP reputation / fraud scoring to hash evaluations of files.
- **IM & IR:** Incident Management and Incident Response provide you with services once cyber threats become reality. It can both be handled manually or automatically, depending on your SOAR implementations. At the end of the line reported incidents or events need to be handled and analyzed.
- **Threat intelligence:** not every incident or event is a cyber theat. Obtaining threat intelligence by further analysis of logging provides visibility and knowledge of what you are dealing with. Is this IP malicious? Is this analyzed file hash linked to other malicious files? Is this document malware? These questions can further be handled by threat intelligence.

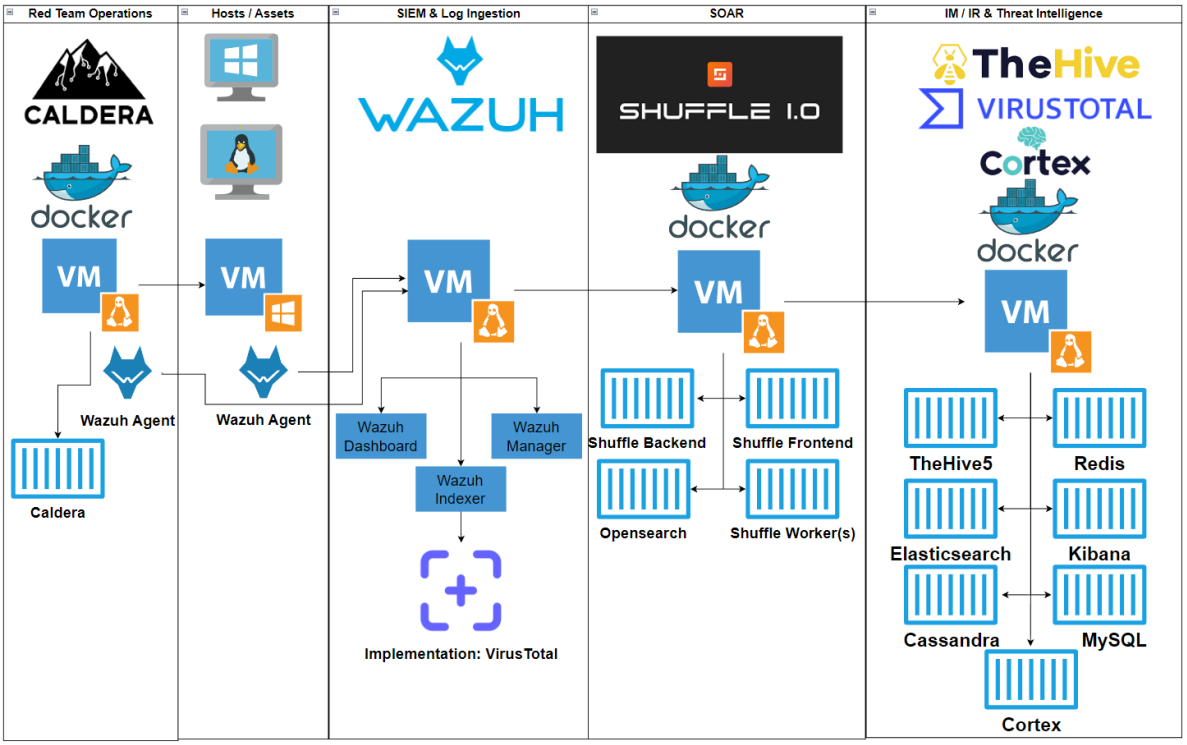


1 SECURITY OPERATIONS CENTER

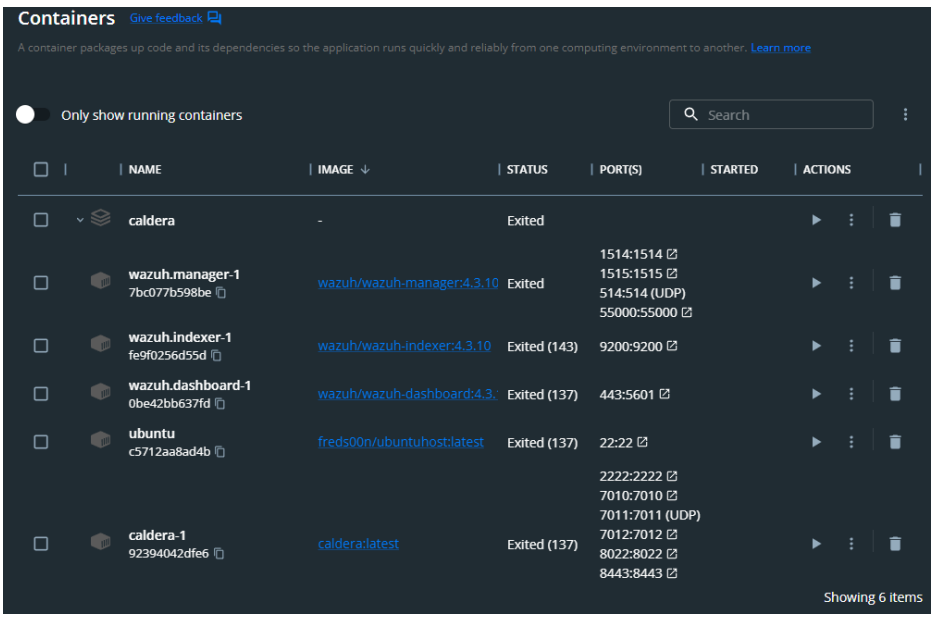
A security operations center (SOC) serves as a backend solution providing a general overview of cyberattacks on your assets. A SOC creates awareness and security against all kinds of cyber threats as a variety of solutions can be implemented to guarantee visibility and protection. This paper will provide a general overview of the entire implementation I have provided and all the lessons learned through the process of implementation.

1.1 Architectural design

DevOps starts with planning – and creating a general idea of what you SOC will look like in the end. An architectural design always you to obtain a high-level overview of your final solution.



While this has eventually become the final solution – the most important aspect of such an implementation: **scalability** and **performance**. A viable choice needs to be made: **local** or **cloud** building? Docker and Kubernetes are your preferred options for cloud.



While one asset could manage this entire stack – a **local deployment** only seems viable in case of a **Docker / Kubernetes deployment.** Virtual machines stacked on top of each other will only reduce scalability and performance.

Since both aspects are important for future-proofing this solution – the choice was simple: **Google Cloud Platform** (cloud). While technically speaking all the docker containers could be run locally via Docker Desktop – performance is down the drain. Scalability is also not the greatest option when selecting one, single asset.

This and the obvious advantage of having far more networking- and virtualization options makes a cloud deployment a very obvious choice.

Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP
✓	caldera	europe-west1-b			10.132.0.21 (nic0)	34.140.252.169 (nic0)
✓	soar-services	europe-west1-b			10.132.0.24 (nic0)	34.76.231.227 (nic0)
✓	soar-shuffle	europe-west1-b			10.132.0.23 (nic0)	35.233.6.158 (nic0)
✓	wazuh	europe-west1-b			10.132.0.20 (nic0)	35.187.97.89 (nic0)
✓	windowshostpro	europe-west1-b			10.132.0.13 (nic0)	130.211.58.26 (nic0)

One major factor needs to be considered when deploying to the cloud: **manage your firewall properly.** Four our sock this is technically speaking a perfect testing ground – open all ports and apply a range of 0.0.0.0/0 IPv4 adresses and your Wazuh agents are now officially honeypots.

Name	Type	Targets	Filters	Protocols / ports	Action	Priority	Network ↑	Logs	Hit count
shuffle-welcome	Egress	Apply to all	IP ranges: 35.233.6.0/24	all	Allow	1000	default	On	684
wazuh-agent-second	Egress	Apply to all	IP ranges: 10.132.0.0/24	all	Allow	1000	default	On	1
hostmachinepublic	Ingress	Apply to all	IP ranges: 84.197.9.0/24, 192.168.60.0/24	all	Allow	899	default	On	415
shuffle-find-me	Ingress	Apply to all	IP ranges: 35.187.97.0/24, 35.233.6.0/24	all	Allow	1000	default	On	3250
wazuh-access-shuffle	Ingress	Apply to all	Service account: 909318520781-compute(	all	Allow	1000	default	On	1
block-all	Ingress	Apply to all	IP ranges: 0.0.0.0/0	all	Deny	1001	default	On	660
default-allow-http	Ingress	http-server	IP ranges: 0.0.0.0/0	tcp:80	Allow	1002	default	On	3
default-allow-https	Ingress	https-server	IP ranges: 0.0.0.0/0	tcp:443	Allow	1002	default	On	2
default-allow-icmp	Ingress	Apply to all	IP ranges: 0.0.0.0/0	icmp	Allow	65534	default	On	0
default-allow-internal	Ingress	Apply to all	IP ranges: 10.128.0.0/9	tcp:0-65535 udp:0-65535 icmp	Allow	65534	default	On	0
default-allow-rdp	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:3389	Allow	65534	default	On	0

Primary steps for a secure starting environment:

- **Deploy a block-all:** since the standard Firewall settings basically allow ANY connection into your machines (especially SSH) – all your deployments basically become honeypots (even your security assets).
- ***Remove default-allow-ssh:*** the only reason why this comes in handy is because you can SSH through the browser / GCP console into any machine directly. Do you need this? No – you can utilize the Google Cloud SDK Shell for local connections + a "hostmachinepublic" IP whitelisting rule.
- **Deploy asset-specific ruling:** while /24 is not the best solution – it is intended only for your local machines to access other assets. *You can also apply one IP single-handedly if you have fixed IPs.*

Once networking and architecture have met – we are ready to build a SOC. As GCP offers a royal free-credit budget you can easily include your entire deployment.

1.2 Building blocks

This chapter will have a thorough overview of the mentioned building blocks in the architectural design. Every single asset will be discussed here.

1.2.1 Red team operations

This is where part of the exploitation starts. Red team operations is a completely different approach compared to penetration testing. Penetration testing revolves around finding **weaknesses** in the defense capabilities. Finding exploitable flaws in a minimum amount of time and a maximum amount of coverage. Red team operations has a similar goal: gaining access to data, but this data is pre-determined, for example. In essence it is an automation tool applying red team testing. It is well-rounded for thorough response capability testing, SOC readiness and security measures in place.

1.2.1.1 Caldera

MITRE Caldera is a readily-available red team tool in order to provide **automated adversary emulation**. Adversary-what? Technically speaking this tool provides you are deploying an automated penetration tester. It is perfect for training blue teams and detecting specific threats. Ideal for testing the capabilities of our SOC deployment, right? Caldera does not only have red teaming capabilities – it also has blue team capabilities:

- **Red team:** having a wide range of adversary profiles to your disposal – you can simulate any kind of real-life cyber-attack. Ideal for thorough defense testing and learning how to detect cyber threats.
- **Blue team:** the blue team aspects are even more interesting – it helps you in detecting gaps in your defenses. Effectively applying TTP (Tactics, Techniques and Procedures) which can be seen as behavior as a threat actor. Tactics are high-level descriptions, while techniques are mid-level details and procedures are thoroughly written-out techniques.

Rather than building your Kali / Parrot OS Linux – red team tools help you in saving time. Once a specific adversary profile or operation is defined – you are set to launch this attack on thousands of assets, training and testing all assets simultaneously (as well as your SOC and security response).

Remember scalability and an increase of performance? Then Caldera is exactly the tool you are looking for. It operates on an agent-basis making deployment and exploitation easier than ever.

1.2.2 Hosts / assets

Hosts are primarily your end-user devices… but don’t forget your SOC hosts either.

1.2.2.1 Windows vs Linux

As the primary test-surface end-user devices are at the forefront of your cyber warfare (security). Mass deployment of hundreds or thousands of end-user devices who each are different, one by one. OS differences, OS version differences, tooling differences,… the list is endless. Penetration testers can perform testing on one single device in order to obtain access, but what about the others? This is the reason included both Windows OS and Linux OS hosts is important. Each OS behaves differently – let alone OS version variations. One machine of each OS is included – of which the Linux host is the Caldera asset itself (not being pentested – only being monitored). Every single asset is important in having an effective monitoring and defense.

1.2.3 SIEM & Log ingestion

Security Information and Event Management go hand-in-hand with log ingestion in this deployment. Wazuh has it all – albeit not being the best at log ingestion with a variety of different options available. Implementation is made easier since Wazuh agents can collect logs directly forwarding them towards your SIEM solution, Wazuh.

1.2.3.1 Wazuh

Wazuh is an open source security platform providing primarily endpoint- and cloud security. As it is freely available for anyone to try it becomes an easy pick. Implementation are available completely out-of-the-box with Wazuh possibilities and other OpenSearch / ElasticSearch implementations, for example.

A second possibility is an ELK stack with Wazuh: ElasticSearch, Logstash, Kibana (and Beats). Choosing is losing – either options are very viable. Wazuh vs ELK is in this case more efficient since all Wazuh comes with all requirements in one deployment. ELK would just be an addition to the Wazuh Manager in this case, for example.

Wazuh has four main components:

- **Wazuh Dashboard:** this is effectively your Kibana visualization offering you all the visual beauty a SIEM has to offer.
- **Wazuh Indexer:** the index is comparable to ElasticSearch, as it is primarily build on top of OpenSearch (a different kind of "search"). In essence it is a search engine for your logs.
- **Wazuh Manager:** the manager runs all the operations in the background. Manager provides workers for your task-driven operations. It is effectively this component which makes Wazuh the SIEM solution itself. All other components can be swapped around for alternatives.
- **Wazuh Agent(s):** wazuh agents are comparable to Beat in which they effectively transport data back to your SIEM solution. Deploying these solutions on your end-user devices allow you to collect… logs! This provides a log ingestion solution – of which many others are available.

The combination of the above components effectively build your entire SIEM solution. Missing either one of those components (and possibly a Database / Cassandra) will make your SIEM solution less-effective. Each specific process (searching, visualization and workload) are split into different components allowing for better performance.

1.2.4 SOAR

Security Orchestration, Automation and Response provides a solution for further action regarding your logs, information and events. Once your SIEM has a good view on all your asset's logs your SOAR has the capability to provide actions for further remediation and response.

1.2.4.1 Shuffle.io

Shuffle.io is a SOAR solution providing automation in order to perform further analysis or response to an incident. The options are endless here – from threat intelligence to incident management and incident response. Essentially with any SIEM solution, shuffle.io creates a webhook to obtain all parameters from an event and parse those details bit by bit for further actions.

Many other SOAR solutions are available – making Shuffle.io just one of many options.

Shuffle has a variety of components:

- **Shuffle frontend:** the frontend is plain and simple: it provides an interactive web page in which you can program further actions.
- **Shuffle backend:** the backend provides further workload and actions with your OpenSearch engine and workers. Essentially performing all the tasks in the background, starting workers and triggering your OpenSearch engine.
- **Opensearch:** similar to Wazuh Indexer and Elasticsearch – OpenSearch is your search engine solution for this stack. In an ideal world and deployment – only one search engine is required for all stacks / assets.
- **Shuffle worker(s):** shuffle workers are your work horses. They execute all kinds of tasks in the background which you have programmed in the frontend.

Combining all of the above components provides you with a general SOAR solution.

1.2.5 IM / IR & Threat Intelligence

Incident Management and Incident Response give your incident managers, SOC analysts and service desk employees something valuable to look at. Once your SIEM has been fully programmed, and SOAR providing an open door for other applications, the threat intelligence and incident management can begin.

1.2.5.1 TheHive5

While most information available online revolves around TheHive4 – the open source solution provided by The Hive Project – TheHive5 has been the newest addition. While TheHive4 has been fully built by an open source community, TheHive5 is effectively being developed by a private company called StrangeBee.

Previous versions of TheHive were hereby mostly freely available with full capabilities. TheHive5 introduced a paywall for certain capabilities only including a free bundle called the "community" edition.

Either way you are choosing a "free" solution, or a solution starting from thousands of euros. As pictures speak louder than words – TheHive5 might not be the ideal solution for everyone. TheHive will be used to create alerts and cases for further analysis.

1.2.5.2 VirusTotal

Technically speaking VirusTotal is implemented within Wazuh, the SIEM solution, but should be included here as a threat intelligence tool. VirusTotal can analyze IPs and file hashes in order to obtain a better understanding of what you are dealing with.

Both of the above mentioned use-cases will result in one answer: is it malicious or not? Integrating VirusTotal can even go thus far you can create an automated response for malicious files, deleting the malicious file in the process. Thanks to its ideal API solution you can now integrate VirusTotal both in your SIEM and SOAR solution for threat intel or automated response.

1.2.5.3 Cortex

Last but not least, Cortex is a threat intelligence solution providing more detail to security researchers and analysts. Digital forensics and threat intelligence go hand in hand. As it is open source (developed by The Hive project / StrangeBee) it is freely available and doing similar tasks compared to VirusTotal. Analyzing IPs, email addresses, URLS, domain names, files and hashes in order to give your security team the most valuable information.

While cortex is included in the architecture and on the host machine – it is unfortunately not fully implemented and integrated in the SOAR. Implementation can be combined with TheHive5 and your SOAR providing further information to your incident alerts and cases.

1.3 Implementing your SOC

This chapter will provide further information about the entire implementation of the mentioned building blocks. Screenshots give you an idea of how the implementation is working and operating.

1.3.1 Caldera

Caldera is implemented via a docker-compose file, allowing you the ease of implementation.

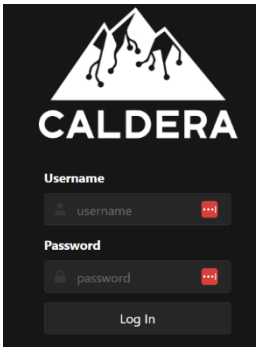
Simply git clone the Caldera repository and you are ready to go.

```
Fredson@caldera:/home/frederik_crauwels_be/caldera$ ls
CONTRIBUTING.md  conf                 plugins                  templates
Dockerfile       data                 requirements-dev.txt     tests
LICENSE          docker-compose.yml   requirements.txt         tox.ini
README.md        ftp_dir              server.py
SECURITY.md      package-lock.json    sonar-project.properties
app              package.json         static
```

Once you perform the docker compose, or docker-compose command you will have one machine operating all your red (and blue) teaming actions.

```
                         NAMES
2dd22aea3aa8   caldera:latest   "python3 server.py -…"   2 hours ago   Up 2 hour
s   0.0.0.0:2222->2222/tcp, :::2222->2222/tcp, 0.0.0.0:7010->7010/tcp, :::7010->
7010/tcp, 0.0.0.0:7012->7012/tcp, :::7012->7012/tcp, 0.0.0.0:8022->8022/tcp, :::
8022->8022/tcp, 0.0.0.0:8443->8443/tcp, :::8443->8443/tcp, 0.0.0.0:8853->8853/tc
p, :::8853->8853/tcp, 0.0.0.0:8888->8888/tcp, :::8888->8888/tcp, 0.0.0.0:7011->7
011/udp, :::7011->7011/udp   caldera-caldera-1
```

Once you have provided sufficient exposure to be able to access the container you can open up a web browser on port 8888.

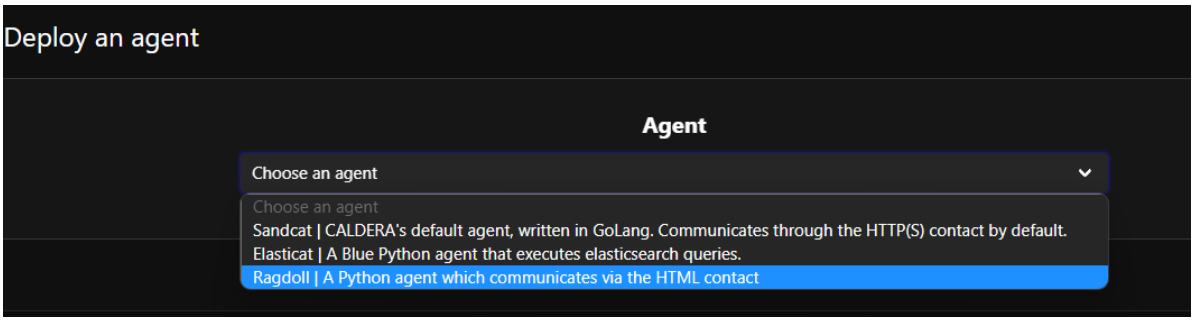


On this web page you are able to login. Credentials are provided within your deployment folder.

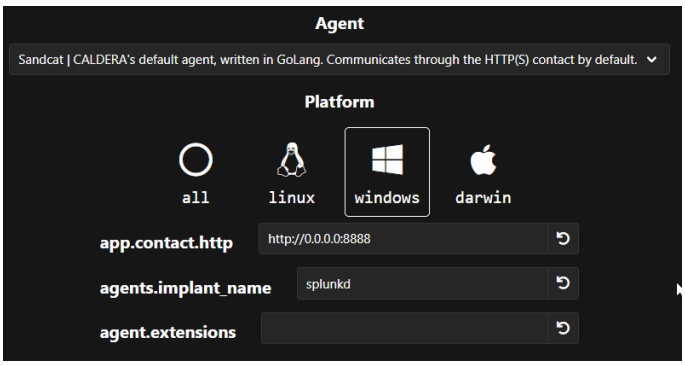
Caldera -> conf -> default.yml or local.yml files.

Once you have obtained your credentials you are ready to perform actions… but not so fast. Choose between the red or blue user. For our testing the blue user is better equipped as we want to test our gaps and defenses.

Secondly you simply need to deploy an agent similar to Wazuh, your SIEM. This can be just as easy as it sounds. Once you click on the "Agents" tab you can deploy an agent to any kind of OS.



Once you clicked on "deploy an agent" you can essentially deploy one of three agents. The explanation provided in the above picture provides a better understanding of the agent. Each agent communicates in an differently. For our deployment we will utilize the default agent.



Once you select an agent you need to provide the correct HTTP / URL of your Caldera deployment. This can easily be your localhost, or your public IP address depending on the deployment.

Underneath these options you obtain a thorough overview of how to deploy your agent in a variety of ways.

Once the agent is deployed on your host you are ready to run exploits. Running the exploits and testing the SOC will be a topic at the end of this documentation.

From my own experience the **last download option** is mostly the best solution possible for ease of implementation: **Deploy as a P2P agent with known peers included in compiled agent.** Simply follow the steps on your desired host:

```
PS C:\Windows\system32> $server="http://10.132.0.21:8888";
PS C:\Windows\system32> $url="$server/file/download";
PS C:\Windows\system32> $wc=New-Object System.Net.WebClient;
PS C:\Windows\system32> $wc.Headers.add("platform","windows");
PS C:\Windows\system32> $wc.Headers.add("file","sandcat.go");
PS C:\Windows\system32> $wc.Headers.add("gocat-extensions","proxy_http");
PS C:\Windows\system32> $wc.Headers.add("includeProxyPeers","HTTP");
PS C:\Windows\system32> $data=$wc.DownloadData($url);
PS C:\Windows\system32> get-process | ? {$_.modules.filename -like "C:\Users\Public\splunkd.exe"} | stop-process -f;
PS C:\Windows\system32> rm -force "C:\Users\Public\splunkd.exe" -ea ignore;
PS C:\Windows\system32> [io.file]::WriteAllBytes("C:\Users\Public\splunkd.exe",$data) | Out-Null;
PS C:\Windows\system32> Start-Process -FilePath C:\Users\Public\splunkd.exe -ArgumentList "-server $server -group blue -listenP2P -v" -WindowStyle hidden;
```

Returning to the dashboard – your agent is now active and alive:

id (paw)	host	group	platform	contact	pid	privilege	status	last seen
ppyvqx	DESKTOP-KBRQ0QU	blue	windows	HTTP	9740	Elevated	alive, trusted	33 seconds ago

Ready to test your defenses (and newly implemented SOC). This could, for example, be one of the installations that could be counted as suspicious and may not be installed if your SOC and security measures are not allowing this kind of deployment.

1.3.2 Windows (pro) host

The installation of a proper Windows OS is a hassle. Since Windows (for obvious reasons) does not allow you to deploy a Windows OS docker container (only servers) you need to have a work-around. Nonetheless the fruits of your labor can be utilized for an eternity to come.

Simply download a W10 iso file on the official Windows website. Once this ISO is being download you need to create a **local virtual machine** and perform a Windows 10 installation. *The reason I did not choose W11 is because the VM installation did not allow me to proceed with the installation due to not meeting "the requirements". Is this due to a VM installation, or the VM settings? Hopefully this does not become a blocking factor for the future!*

Once you have created your Windows 10 installation you simply need to **export** this VM and create an **OVA** file. OVA files are readily available VM images as to speak, which can be integrated in any VM virtualization solution such as the cloud or Vmware.

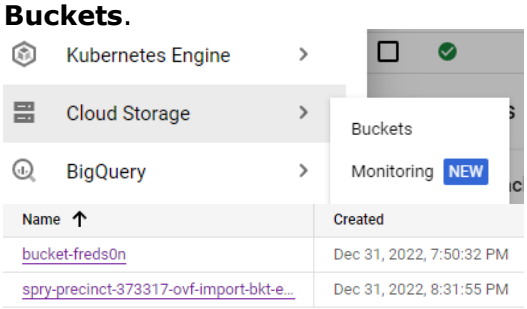
In this case we created the VM with Oracle VM Virtualbox Manager. Follow the steps mentioned for an OVA export:

1. Click on **File**.
2. Select **Export Virtual Appliance**.
3. Select your **Windows host**.
4. *Leave all settings at the first screen, next.*
5. *Leave all settings at the second screen, click **export**. Wait… done!*

It is as easy as that. This OVA file is essentially your Windows host in a nutshell allowing you to deploy Windows 10 hosts to any platform you wish. In this case – deployment to the cloud (GCP). In order to facilitate this you need a couple of nifty commands and a new shell. Google even provides thorough documentation themselves:

[Import an OVA file | Google Cloud]

In GCP you need to create a **virtual storage.** Go to **Cloud Storage** and click on **Buckets**.



Once arrived here you simply create a new bucket by clicking on **create**. Simply follow the steps – which for this tutorial do not matter at all. Simply click through the creation process.

Once created you have your bucket name (mine is bucket-freds0n).

You have two possibilities for uploading: command vs browser. If commands suit you better (advised for further use of GCP locally) [Install Gcloud CLI SDK].

In this case the uploading via command can be a lot easier as it will take some time (up to half an hour at least). Simply apply the following:

*gcloud compute instances import VM_NAME *

--source-uri=gs:PATH_TO_OVA_FILE

This is your upload command – an easy example: gcloud compute instances import windowshost.ova --source-uri=gs://bucket-freds0n/windows.ova

Sometimes you need to specify your OS version with the -- os flag.

Once you performed the upload – you now have a Windows.ova file in your cloud bucket. Now you simply need to deploy the .ova on GCP – create a virtual machine!

Name	Size	Type
Windows10-Host-Pro.ova	6.3 GB	application/x-virtualbox-ova
Windows10-Host.ova	5.4 GB	application/x-virtualbox-ova
wazuh.ova	2.5 GB	application/x-virtualbox-ova

Simply utilize the gcloud CLI and apply one more command:

*gcloud compute machine-images import MACHINE_IMAGE_NAME *

*--source-uri=gs:SOURCE_URI *

--os=OS

This is the basic command – but I strongly recommend you to utilize both the **–-zone** flag and the **–-machine-type** command in order to specify your machine settings.

Once both commands have been utilized you officially have a Windows 10 host in the cloud. *Do keep in mind: if you want to utilize RDP (Remote Desktop Protocol) you **must have a Windows Pro** version, as this is not available to the Home edition*.

1.3.3 Wazuh

Wazuh is our SIEM solution – having two major options for implementation: image installation or docker deployment. *In this case I choose for **image installation** since the machine is only going to run Wazuh (performance-wise),* allowing you to further secure your SIEM instance without having to deal with other assets on this VM.

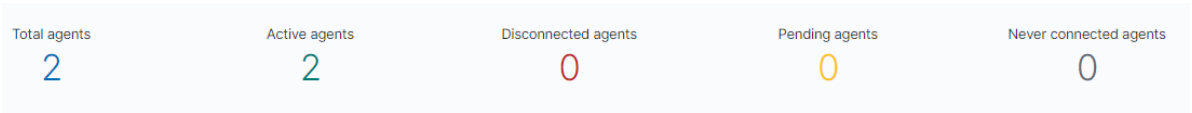
As you could see in the previous example – i imported an OVA file, which makes deployment easy: [download your Wazuh OVA file here!]

Simply follow the exact same steps for deployment. *Keep in mind this image has a CentOS (7) OS image*. *Vi(m) and RPM galore!*

Keep in mind – while the standard Wazuh user password is fairly easy to guess, this user has been disabled for SSH access. Only the local users are able to SSH into this machine. Resetting your passwords for this instance is a whole different story compared to the installation. *It is recommended to look into it nonetheless.*

Once you have provided your Wazuh VM you can connect through a local web browser. *Apply a networking rule which allows your PUBLIC ip address to access "all hosts". You could go further only specifying certain ports, but for a test deployment it will become tedious.*

Once you have access to the dashboard you can now deploy agents and start your SIEM adventures!



For this build I have included my **Windows Host** and my **Caldera host** (Linux) with an agent.

1.3.4 Shuffle.io

Once the "basics" have been provided, your actual threat intelligence and SOAR capabilities are becoming tremendously huge. You now have: multiple hosts with log collecting agents, Caldera for red team activities (testing your SOC deployment) and a SIEM. Shuffle.io will be the building block between your **SIEM** and your **threat intelligence / incident management solutions**.

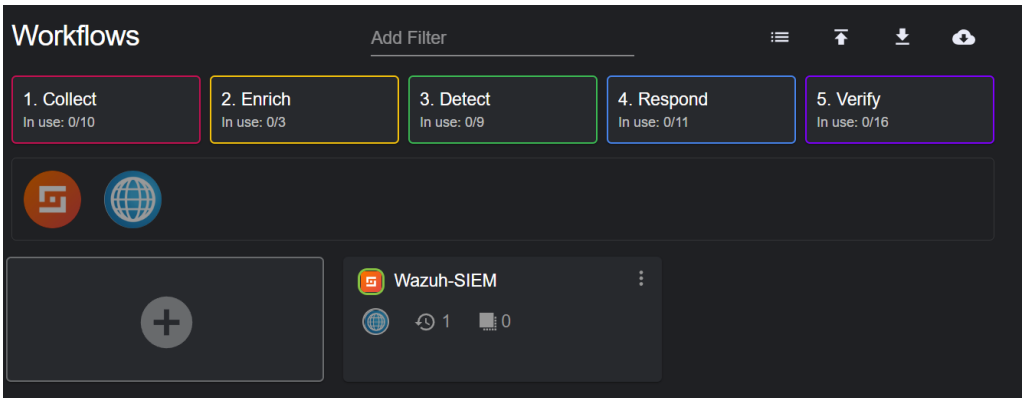
Simply look for the **newest shuffle** docker deployment. Kubernetes deployment is preferred – but is a lot more tedious to implement and manage. *I advise you to look for the newest shuffle version – since older shuffle versions will **look a lot more differently.** Visually the newer version offers more possibilities.*

Since the focus of this deployment revolves around scalability and security – my SOAR solution is separated from all other assets in one VM. SOAR requires a bunch of resources to handle your SIEM logs – effectively requiring memory and CPU. Once you implement more solutions you can easily scale horizontally or provide more nodes.

Once you have cloned your desired docker-compose directory simply deploy the solution on your desired OS:

```
Fredson@soar-shuffle:~$ sudo docker ps
CONTAINER ID   IMAGE                                   COMMAND                  CREATED        STATUS
PORTS                                                                           NAMES
5ea716caa497   ghcr.io/shuffle/shuffle-frontend:latest   "/entrypoint.sh ngin…"   16 hours ago   Up 15 hours
0.0.0.0:3001->80/tcp, :::3001->80/tcp, 0.0.0.0:3443->443/tcp, :::3443->443/tcp   shuffle-frontend
868d93ae71f9   ghcr.io/shuffle/shuffle-backend:latest    "./webapp"               16 hours ago   Up 15 hours
0.0.0.0:5001->5001/tcp, :::5001->5001/tcp                                       shuffle-backend
c17ba433eaf2   opensearchproject/opensearch:2.4.0       "./opensearch-docker…"   16 hours ago   Up 15 hours
9300/tcp, 9600/tcp, 0.0.0.0:9200->9200/tcp, :::9200->9200/tcp, 9650/tcp         shuffle-opensearch
64403a417362   ghcr.io/shuffle/shuffle-orborus:latest    "./orborus"              16 hours ago   Up 15 hours
                                                                                shuffle-orborus
```

You will not be dealing with most components – only the frontend. Once networking is applied and your containers are running it is time to open the web browser with **port https://<ip>:3443**.



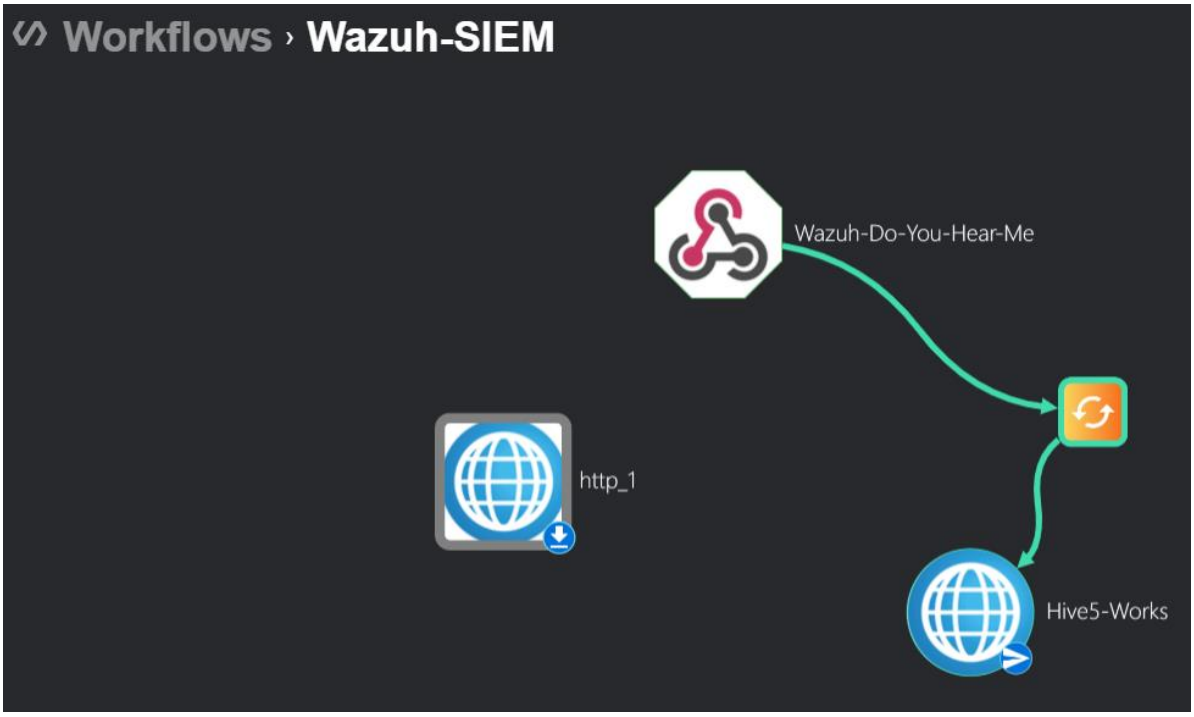
Here you will be able to build workflows for other solutions. As you can see – I have created my own Wazuh SIEM webhooks already. This part is more tricky. The first step in this process is **creating a Wazuh / SIEM webhook**. This webhook allows you to collect your SIEM event information.

Simply go to the following GitHub page: [Wazuh & Shuffle integration files | GitHub]

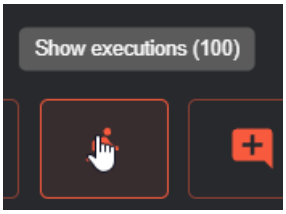
This handy page offers all you need for this integration. The best implementation guidance is provided by shuffle themselves: [Implement Wazuh & Shuffle integration now!] *Keep in mind – this deployment is the Wazuh user, not ossec*. These steps are required for **any further implementations**.

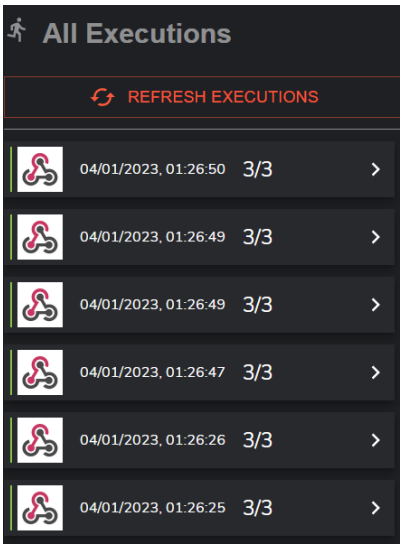
```
[Fredson@wazuh ~]$ sudo ls -all /var/ossec/integrations
total 40
drwxr-x---.  2 root wazuh  138 Jan  2 22:09 .
drwxr-x---. 19 root wazuh  242 Nov 11 14:22 ..
-rwxr-x---.  1 root wazuh  893 Jan  2 21:47 custom-shuffle
-rwxr-x---.  1 root wazuh 5326 Jan  2 21:49 custom-shuffle.py
-rwxr-x---.  1 root wazuh 4325 Nov 11 13:11 pagerduty
-rwxr-x---.  1 root wazuh 1020 Nov 11 13:11 slack
-rwxr-x---.  1 root wazuh 3809 Nov 11 13:11 slack.py
-rwxr-x---.  1 root wazuh 1020 Nov 11 13:11 virustotal
-rwxr-x---.  1 root wazuh 6439 Nov 11 13:11 virustotal.py
```

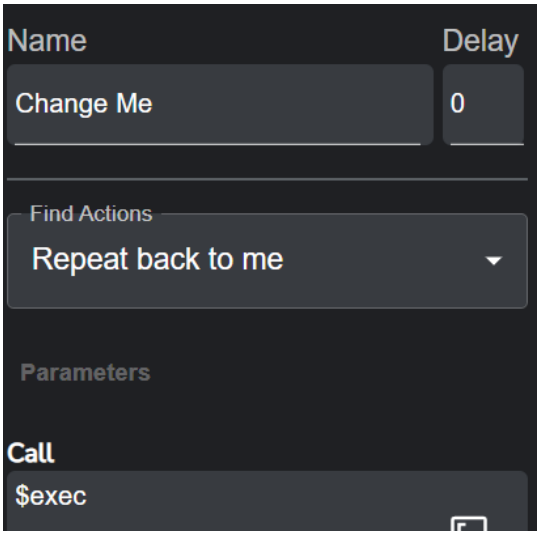
Once you have obtained all configurations you can now **test** your webhook in shuffle.



Simply click on the **Show executions** icon in order to **view** your webhook in action. For the building block it is best you implement a **shuffle tool(s)**. This nifty little block will count as a **repeater** so you can parse the data obtained further towards other implementations / building blocks. Simply apply the below settings:







Once this is done – we will have to implement TheHive5 for our first integration! Incident response, here we come.

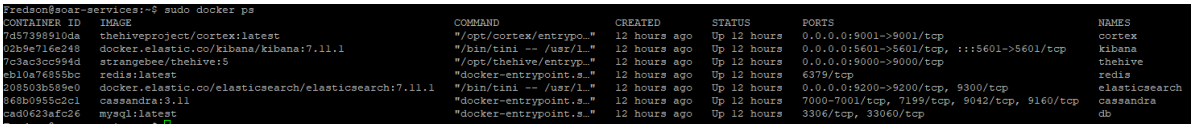
1.3.5 TheHive5

Why did i choose TheHive5? TheHive4 is, amongst all, features mostly in every single Youtube video and implementation document. I can assure you implementation of a new version is not easy without a lot of guidance – but not impossible. TheHive5 simply has options available in the frontend. Technically speaking you don't even need a SOAR anymore since TheHive5 can create its own webhooks towards Wazuh making ease of implementation even better. Obviously SOAR is still very handy for other tools such as Cortex and many others.

If you are not up for a challenge and a twist – I highly recommend you to implement TheHive4 as you will have a lot more comfort and guidance. Here goes nothing!

Simply utilize any of the TheHive4 docker compose deployments out there – for the easy of this tutorial (and your sake), I have provided my own public GitHub deployment of this exact implementation: Docker-SOC @ Defreddy | GitHub

This exact same file can be changed into a TheHive4 deployment as well – simply change the image name. It is that easy. Docker compose and you are ready to roll in the world of incident response. You can also **install a local version of TheHive5 on your desired host** which includes both Misp and Cortex modules. The reason I choose for docker deployment is due to scalability. If preferred – this definitely is a possibility: How to install TheHive5 | StrangeBee



A lot of components – only a handful of useful ones:

- **Cortex:** cortex is included and deployed – but not implemented. While it offers a lot of capabilities and especially **threat intelligence** – it is almost as easy as implementing the TheHive call. I definitely recommend you to give it a proper look!
- **Thehive:5:** our famous new TheHive container – we will be utilizing this container to access the frontend.
- **ElasticSearch:** basically the same as all other stacks – eventually you should work towards one ElasticSearch deployment. Multiple ElasticSearch / OpenSearch deployments are tedious but make sure environments are separated.
- **Databases:** MySQL and Cassandra are two databases. I highly recommend you to look into Cassandra as it is open source, NoSQL and highly praised for its scalability and availability.

Our main component is definitely **TheHive** – simply connect to the website: **http://<ip>:9000**. This is only the beginning – as we still need to retrieve data from our SOAR solution Shuffle. You need to login and create new organizations as this is required for your SOAR integration. Simply create a new organization and add all user roles. Once you are done adding users you log out and log back in with the **OrgAdmin** user role.

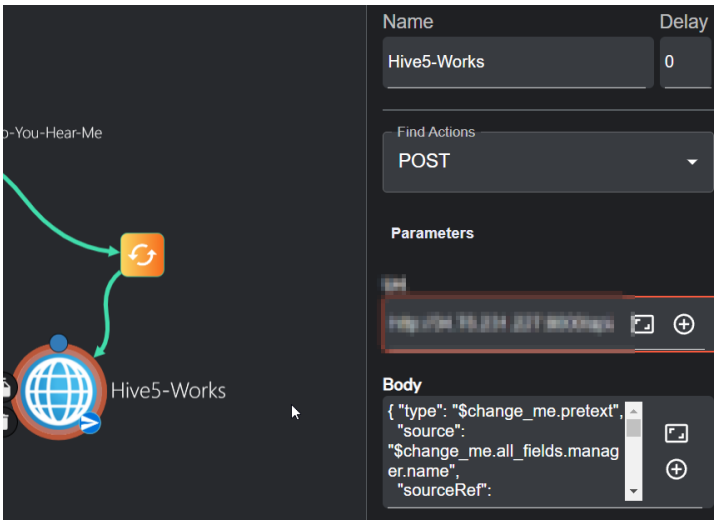
While this documentation is from TheHive4 – it is still fairly similar for a quick start: Quick setup | TheHive4

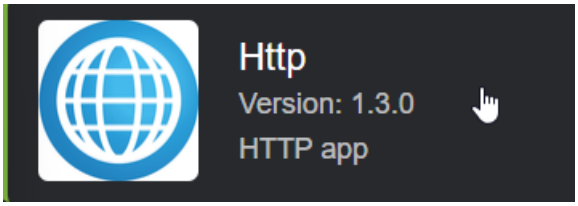
Once initialized we solely need **an API key** of this user (or the analyst user). From this point on it will get even more trickier.

Even the newest Shuffle has a TheHive5 app – but this one doesn't seem to work (for now?). Since we are basically connecting with an API – we can simply use API calls. On this part TheHive5 is very well-documented: [API building a request | StrangeBee].

In this deployment we only build an Alert – the first call for a security analyst something is wrong (without having to look into SIEM).

Simply follow the documentation provided – create your own POST request and select a **Http app**:





Select **POST** here and simply copy-paste the entire "With Curl" JSON string into the **Body** element (not the CURL elements!). The CURL elements need to be provided in the **Url** part and the **Headers** part – don't need a username or any other options underneath.

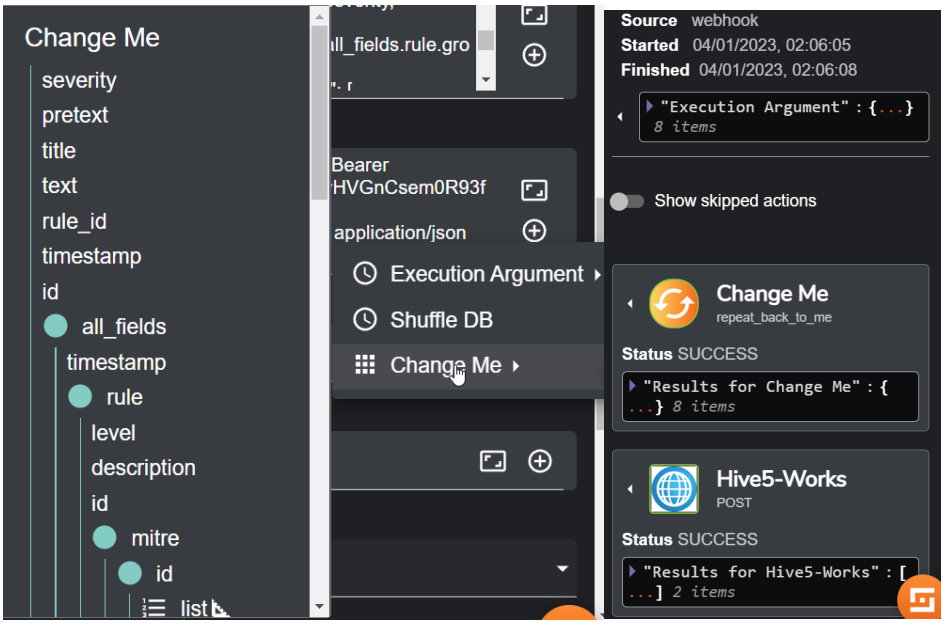
Below is an example of a TheHive5 body POST request:

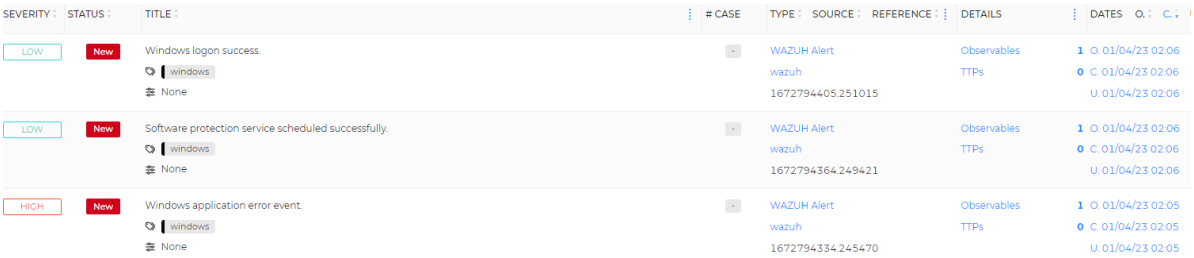
```
{ "type": "$change me.pretext",
  "source": "$change me.all fields.manager.name",
  "sourceRef": "$change me.id",
  "title": "$change me.title",
  "description":"$change me.all fields.agent.name",
  "severity": $change me.severity,
  "tags": [" $change_me.all_fields.rule.groups.# "] ,
  "observables": [
     { "dataType": "url", "data": "http://example.org" },
     { "dataType": "file", "attachment": "attachment0" }
  ]
}
```

By clicking on the **"+"** icon next to the **Body** section you can add new elements from your **Change Me** building block. Make sure to connect them both (as displayed above). Even easier is using the **extended window** button – highly recommended.

Once your **Url, Headers (with Bearer + API token) and Body** is ready you can successfully send your requests towards TheHive.

Click on the **Show Execution** button at the bottom again and take a look if your POST request is working. Once this is working – you now have a new alert in TheHive!





Now these incidents / events can be handled by a security analyst for further analysis!

1.3.6 VirusTotal

As VirusTotal is already directly implemented via Wazuh (integrated) – I will not implement it further within Shuffle – but it perfectly possible just like Cortex and Misp.

VirusTotal's implementation can easily be implemented within Wazuh for **file hash** monitoring. Once malicious files enter your system – VirusTotal will scan the file hash and compare it to other hashes. If it is malicious it will even remove the file as a whole due to an automated implementation.

Simply follow the documentation provided by Wazuh themselves: [File hash scanning and malicious file removal all-in-one | Wazuh]

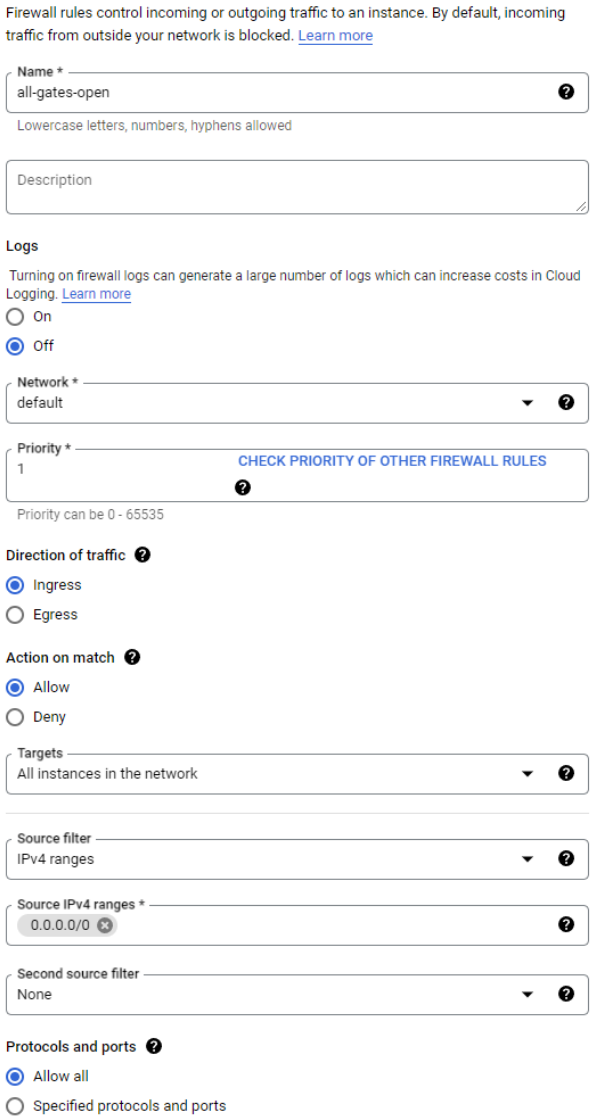
Register at VirusTotal for an API key and implement it directly withing Wazuh for ease of use – or implement it in Shuffle for more SOAR actions / automation.

1.4 Testing your SOC

This chapter will focus on testing our implemented SOC with our red team tooling (and a blue team agent) in order to generate new events which will eventually appear in our Incident Response platform TheHive5. The focus area for this test: SSH brute-forcing / generating login attempts and a malicious file upload.

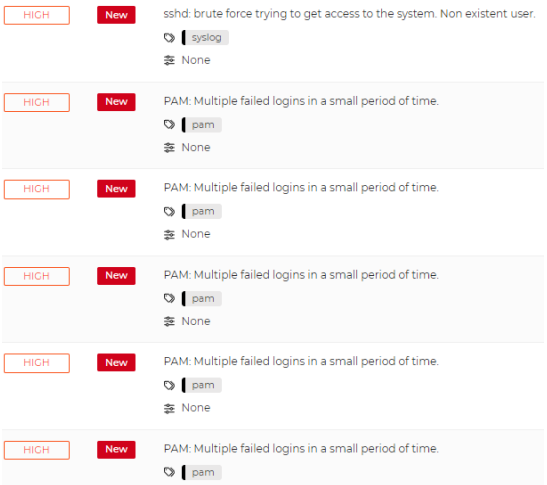
1.4.1 Leave the gates right open

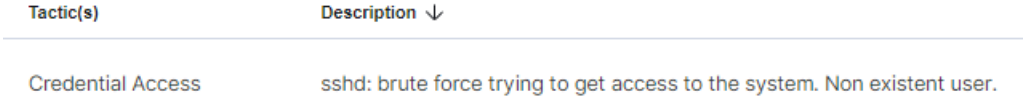
From a technical perspective you don't even have to perform testing yourself via tooling or Caldera. If you leave the "gates" to your environments right open you will notice the cyberspace world is a scary place.



If you let the **standard firewall ruling,** applied by GCP themselves, do their "job", your environments will be SSH brute forced. As there are a variety of botnets in the cyberspace and malicious threat actors (APTs), your environments will eventually fall for the SSH brute force attack.

Very effective in testing your environment – but absolutely **NOT RECOMMENDED**. Secure your environments as quickly as possible – disable the standard SSH ruling applied by GCP. The same goes for HTTP and HTTPS access, also applied by GCP.





@timestamp	2023-01-03T17:49:42.655Z
GeoLocation.country_name	China
GeoLocation.location.lat	34.7725
GeoLocation.location.lon	113.7266
_id	QtzAelUB2Hh08CX74eCI
agent.id	000
agent.name	wazuh
data.srcip	210.5.19.211
data.srcuser	usuario
decoder.name	sshd
decoder.parent	sshd
full_log	Jan 3 17:49:42 wazuh sshd[32271]: Failed password for invalid user usuario from 210.5.19.211 port 26200 ssh2

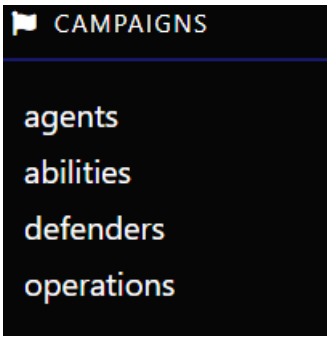
As you can clearly see – these "guests" have nothing to seek in your environments.

1.4.2 Putting Caldera to use

As the previous example is effective – we want a more controlled way of testing our environments which have Wazuh Agent applied.

Caldera is the tool we are looking for. As we already implemented the agent and selected the **blue** login – we can create specific abilities for our "threat actor" to exploit on our **windows host**.

Take a look at the **abilities, defenders and operations** in order to obtain a good understanding of your possibilities. Not limited to only those abilities – as you can easily create more yourself.

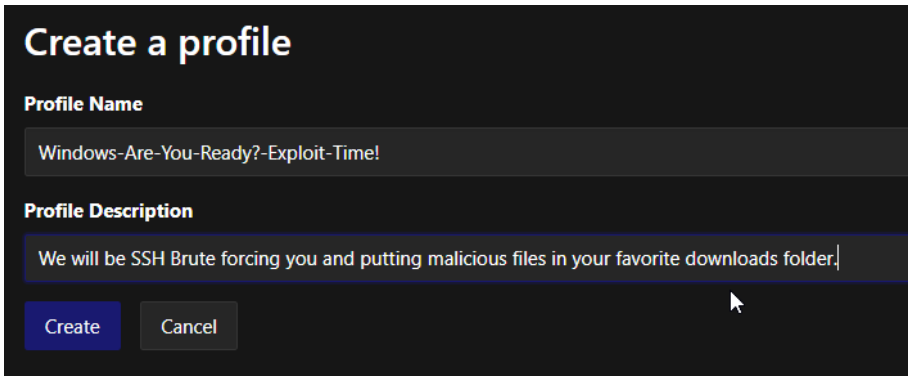


Abilities are possible activities / exploits readily available one by one.

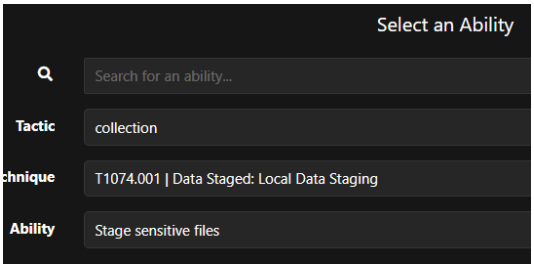
Defenders are specific profiles with a variety of **abilities** applied to them. You can easily utilize on of the defenders directly to apply a variety of exploits.

Operations are your launch trigger for exploits and defenders. Here you will launch your attack.

As we will be building a "SSH Brute-force" / login attempts and malicious file transfer ability ourselves – we need to create a new defender / profile first.

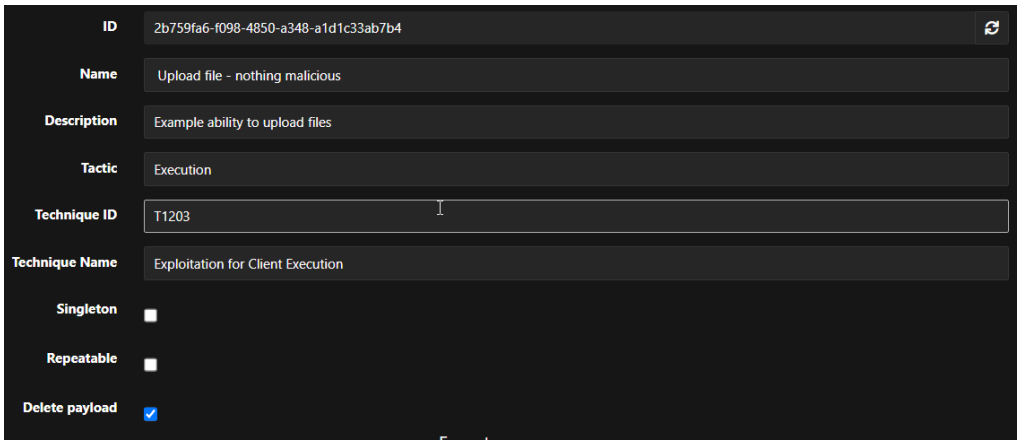


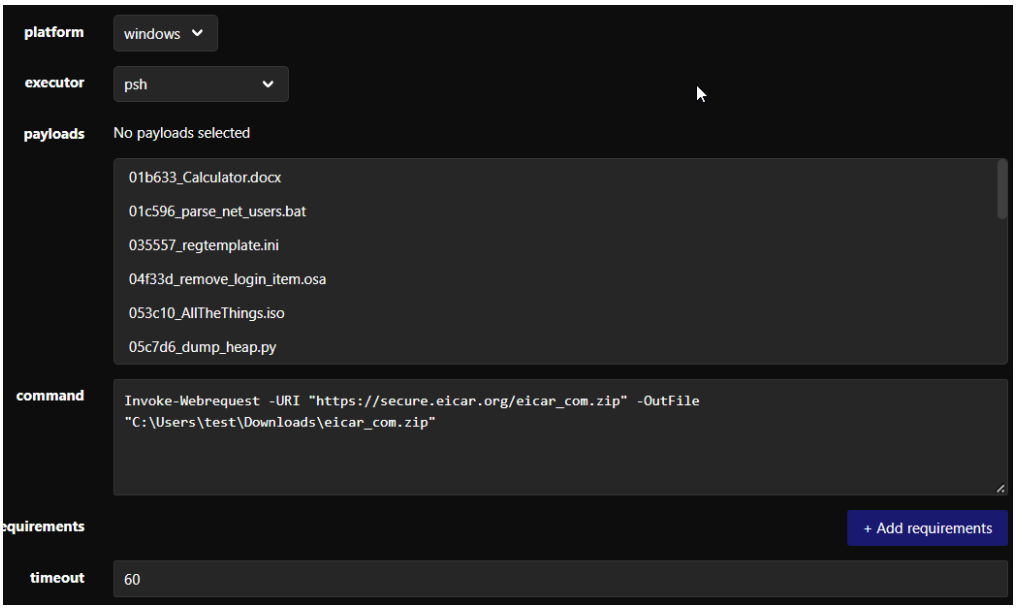
First of all – look for a suitable malicious file uploader. The blue team role has this readily available. Simply search for the ability "file" and select "**stage sensitive files"**:



Caldera has a variety of files available for deployment – and Wazuh is currently configured to investigate and delete any files staged into the **C:\Users\test\Downloads** folder. Simply scroll down to the **Windows** payloads and choose either psh or cmd. This is one example readily available – but you can also create your own abilities / exploits.

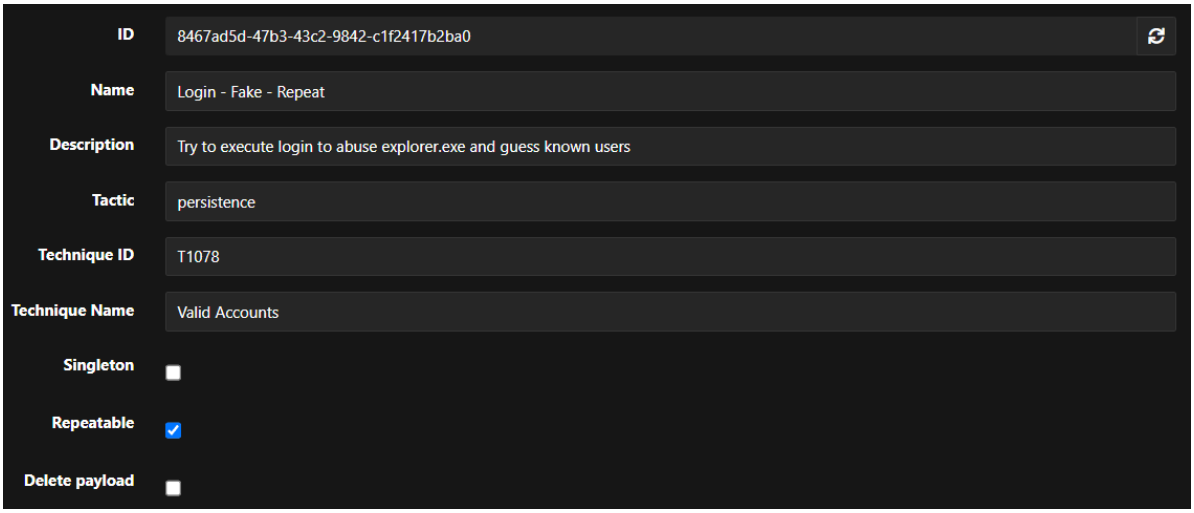
Simply go to **abilities** and click on **add ability**. Our first exploit is **uploading a malicious file!** Find more information about this type of exploit – the information is not of the importance if you want to include a test ability. *All fields are mandatory nonetheless.* Include the platform "Windows", executor PowerShell and the command.

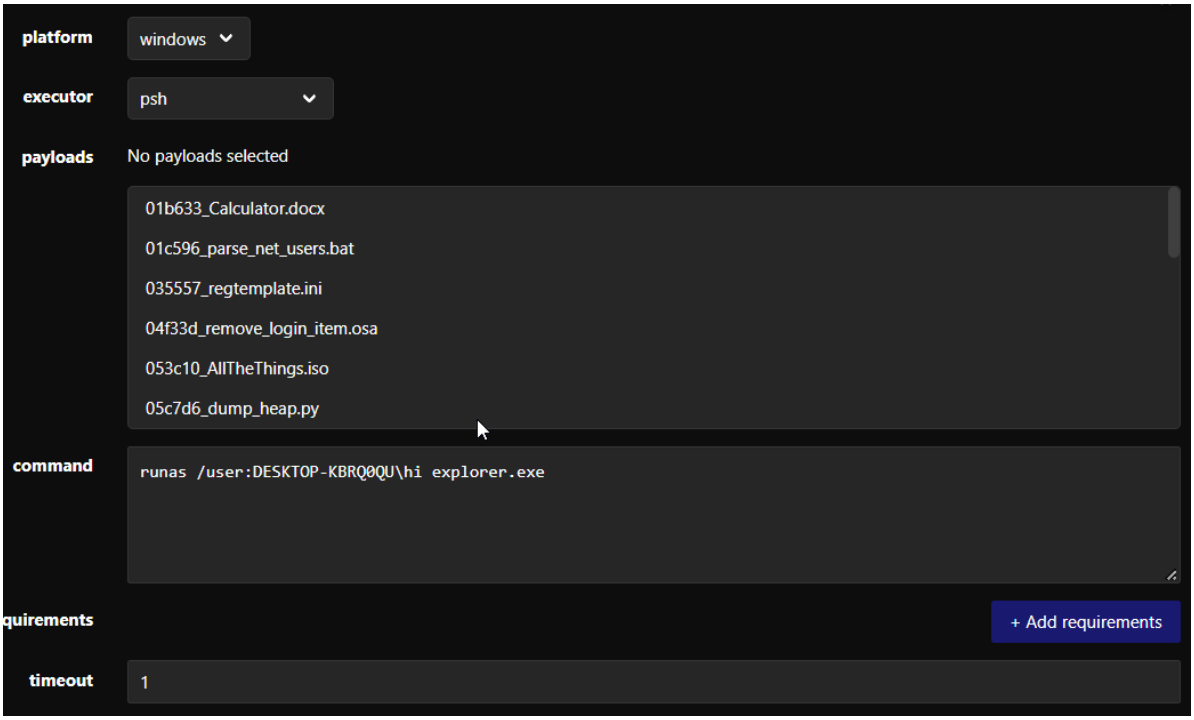




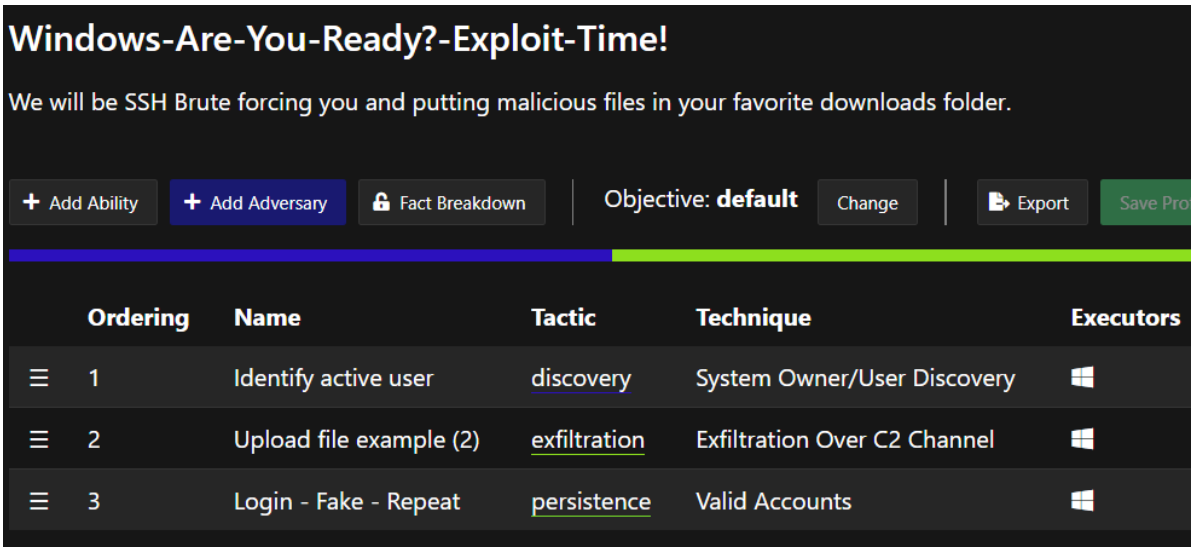
Now click on **Save** in order to be able to utilize this ability later on. This command effectively downloads a test malware file eicar_com.zip. Since we have enabled VirusTotal for our host in the Downloads folder our SOC should have no problem handling this issue.

The next exploit is nothing special either – triggering login commands on the host in order to guess users / credentials. You can put this ability on **Repeatable** – but it will not trigger very rapidly in succession – only every X seconds. Make sure **timeout** is as low as possible for a repeating factor. *But don't go lower than 1 second as the command will timeout.*

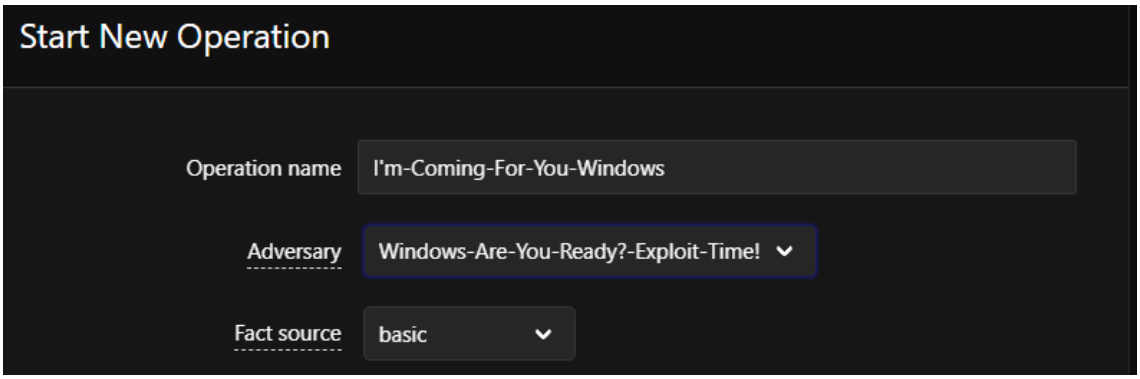




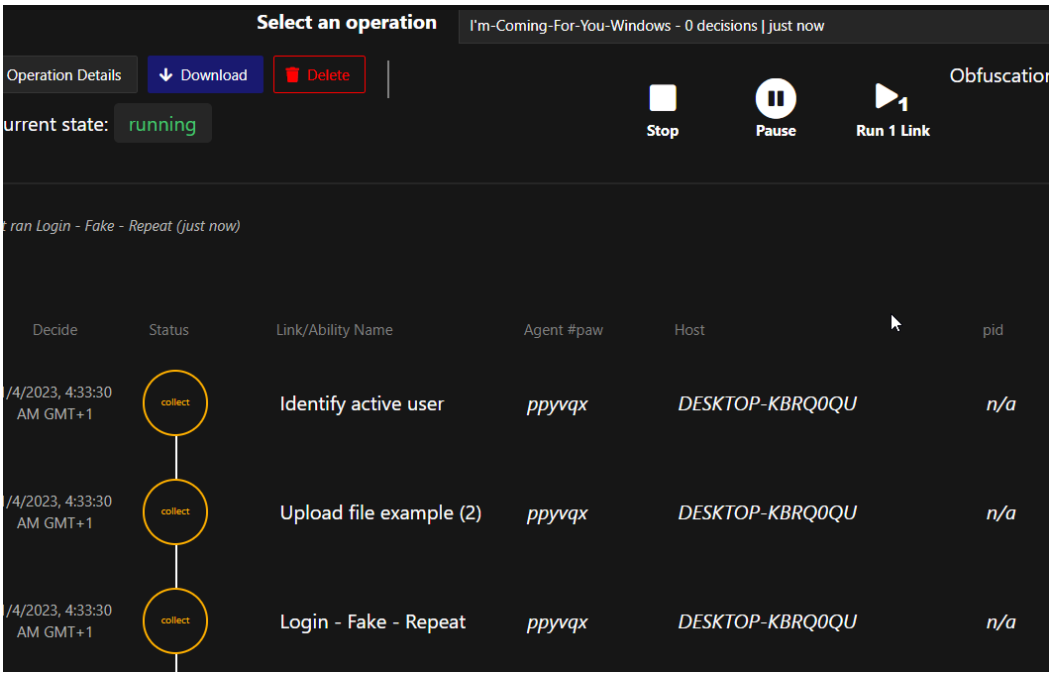
Click **save** and we now have two exploits which we can utilize and run on **every single host** in our command! Now **select** your profile and **add the abilities** by looking for their title / name. Once you have added your created abilities it should look like this:



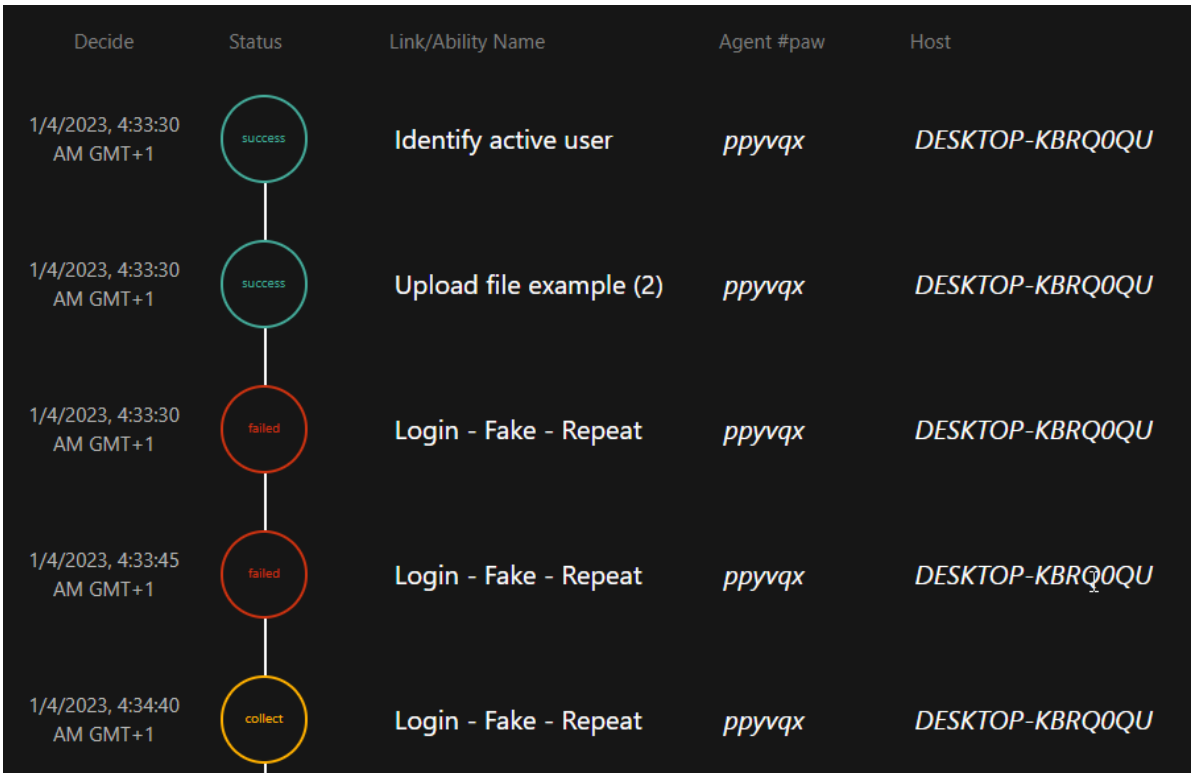
Now our defender / exploit profile is complete and ready for execution. Click on the **Operations** tab in order to launch an attack… or exercise. Click on **Create Operation,** name your operation and select your **adversary profile** we just made (above).



Locked and loaded – lets exploit our Windows host! **Start!** The profile will automatically start the exploits provided in your defenders profile.

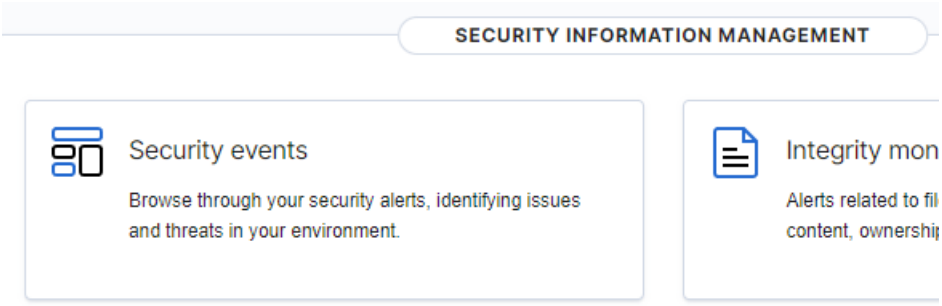


Once the abilities have completed their execution – you can simply look at the status:



It is normal the Login – Fake – Repeat fails. We need to insert a password… but we are not able to perform that action with our current ability!

Now check your SOC and look for the reports: Go to Wazuh and select **Security Events**:



If your SOC is configured correctly – we should see a variety of techniques, tactics and exploits.

- **Malicious file:** VirusTotal has scanned the file and removed the file automatically without manual intervention.
- **"Brute force" / login attempts:** Due to our attempts at logging in we see Login Failures arising all over the place.

Thus far we can only conclude this test **was a major success!** *But there is more. There should be logs generated in our Incident Response tool TheHive5.*

Wazuh / SIEM:

Time ↓	Agent	Agent name	Technique(s)	Tactic(s)	Description	Level	Rule ID
Jan 4, 2023 @ 04:36:08.967	001	DESKTOP-KBRQ0QU	T1078 T1531	Defense Evasion, Persistence, Privilege Escalation, Initial Access, Impact	Logon failure - Unknown user or bad password.	5	60122
Jan 4, 2023 @ 04:35:25.951	001	DESKTOP-KBRQ0QU	T1078 T1531	Defense Evasion, Persistence, Privilege Escalation, Initial Access, Impact	Logon failure - Unknown user or bad password.	5	60122
Jan 4, 2023 @ 04:34:40.919	001	DESKTOP-KBRQ0QU	T1078 T1531	Defense Evasion, Persistence, Privilege Escalation, Initial Access, Impact	Logon failure - Unknown user or bad password.	5	60122
Jan 4, 2023 @ 04:33:41.933	001	DESKTOP-KBRQ0QU	T1078 T1531	Defense Evasion, Persistence, Privilege Escalation, Initial Access, Impact	Logon failure - Unknown user or bad password.	5	60122
Jan 4, 2023 @ 04:33:41.886	001	DESKTOP-KBRQ0QU			Error removing threat located at c:\users\test\downloads\eicar_com.zip	12	100093
Jan 4, 2023 @ 04:33:40.364	001	DESKTOP-KBRQ0QU	T1203	Execution	VirusTotal: Alert - c:\users\test\downloads\eicar_com.zip - 60 engines detected this file	12	87105
Jan 4, 2023 @ 04:33:38.900	001	DESKTOP-KBRQ0QU			active-response/bin/remove-threat.exe removed threat located at c:\users\test\downloads\eicar_com.zip	12	100092
Jan 4, 2023 @ 04:33:38.586	001	DESKTOP-KBRQ0QU	T1070.004 T1485	Defense Evasion, Impact	File deleted.	7	553
Jan 4, 2023 @ 04:33:38.127	001	DESKTOP-KBRQ0QU	T1203	Execution	VirusTotal: Alert - c:\users\test\downloads\eicar_com.zip - 60 engines detected this file	12	87105
Jan 4, 2023 @ 04:33:37.612	001	DESKTOP-KBRQ0QU			VirusTotal: Alert - c:\users\test\downloads\eicar_com.zip - No positives found	3	87104
Jan 4, 2023 @ 04:33:36.079	001	DESKTOP-KBRQ0QU	T1565.001	Impact	Integrity checksum changed.	7	550
Jan 4, 2023 @ 04:33:36.040	001	DESKTOP-KBRQ0QU			File added to the system.	5	554

Incident Response / TheHive5:

MEDIUM	New	Logon failure - Unknown user or bad password. windows None	-	WAZUH Alert wazuh 1672803368.977162
MEDIUM	New	Logon failure - Unknown user or bad password. windows None	-	WAZUH Alert wazuh 1672803325.970423
MEDIUM	New	Logon failure - Unknown user or bad password. windows None	-	WAZUH Alert wazuh 1672803280.963684
MEDIUM	New	Logon failure - Unknown user or bad password. windows None	-	WAZUH Alert wazuh 1672803221.956945
MEDIUM	New	File deleted. ossec None	-	WAZUH Alert wazuh 1672803218.946864
MEDIUM	New	Integrity checksum changed. ossec None	-	WAZUH Alert wazuh 1672803216.942044
MEDIUM	New	File added to the system. ossec None	-	WAZUH Alert wazuh 1672803216.940863

Looks like both applications are correctly logging the events. This concludes our tests in order to obtain a clear view if our SOC is working.

1.5 Honorable mentions

This chapter will include some integration(s) or valuable additions to the entire SOC project as a whole. Some have not made it into the final solution but have been extensively tested for further implementation to no avail, or have a significant impact on the current build. Not all mentions have an impact on the implementation of the SOC build – but do have an impact on the general use of your environments.

1.5.1 Close the gates – how do you SSH into a machine

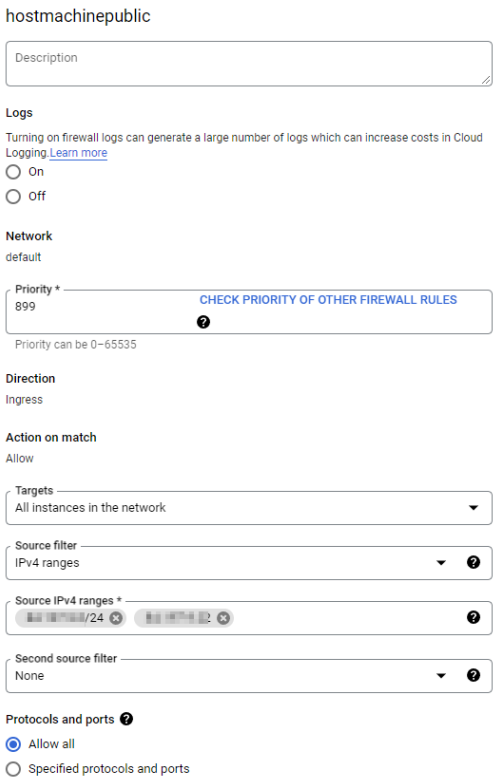
Since we have disabled the standard SSH ruling enabled by GCP we cannot utilize the handy SSH feature in the web browser anymore.

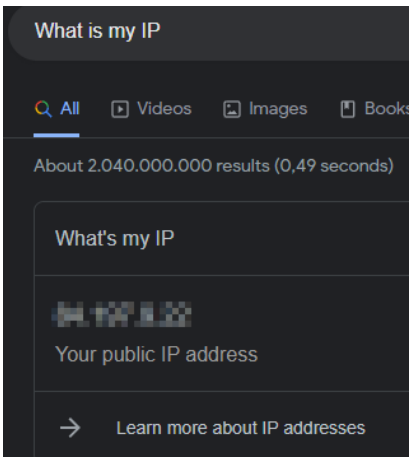
This is effectively what makes the GCP platform handy – easy access to SSH. Unfortunately, as we have seen in chapter 1.4, leaving the gates right open is absolutely not the best idea.

SSH: SSH is now not accessible anymore – or is it?
RDP: RDP is utilized by the Windows (Pro) Host. An important aspect of this implementation is the fact this is a **Windows PRO.** Windows Home does not have the addition of RDP access – which is absolutely not handy when testing your SOC and endpoint.

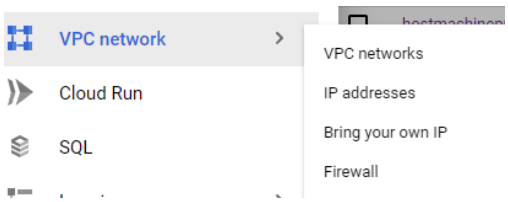
We did apply a specific ruling allowing your **public IP address** access to your environments. Simply create a rule allowing your public IP address into your machines:

1. **Public IP:** find your public IP address. The easiest way of doing so is to simply us Google dorking: "What is my IP".
2. **Firewall ruling:** Once you have an understanding of your public IP address we can set a specific firewall rule.
    a. Simply click on **VPC Network**.
    b. Click on **Firewall**.
    c. At the top of your screen you can **create a firewall rule**.
    d. Now apply the same settings as mentioned in the screenshot to the right. You can use a **range** of IP addresses, also called a **subnet bit**, or **one single IP address**.
    e. Make sure the priority is **lower** than your **block-all** ruling. ACLs work in a simple way: rules with lower priority come first.
    f. If applicable (but for ease of implementation not required) you can even select only specific ports and protocols utilized by your instances. Those are your ports specifically targeted at your **web port access** and obviously **SSH**.





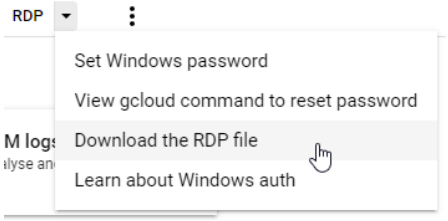
Once those settings are applied – you have obtained access to your assets!



1.5.1.1 Accessing GCP assets via SSH

As mentioned before – you are still not able to utilize the SSH function within the GCP dashboard. There is a workaround – connecting via your Google Cloud SDL local CLI. Since you have whitelisted your IP address to be able to access "all" ports and protocols you can also SSH locally into the host.

Windows is RDP – which is fairly straightforward:



1. **Download** an RDP file.
2. Set the **Windows password**.
3. Use the RDP file in order to obtain access.

For SSH you need to utilize a specific **command** or obtain your public SSH keys via the tool called PuTTy.

1. Open your **GCP cloud shell.**
2. Make sure you are **authenticated**:

   gcloud auth login --project=YOUR-PROJECT-ID-HERE

3. Look for your **instances**:

   gcloud compute instances list

4. Connect to your selected **instance:** It is **crucial** to use the **correct zone**. Without –-zone your SSH will abort since it will not connect to the correct zone. GCP will actually utilize PuTTy in order to create the SSH connection!

   gcloud compute ssh USER@instance_name_here --zone NOZE_NAME_HERE

5. Once you are done in your SSH session simply enter **exit** in your instance shell.

You now have successfully created an SSH connection to one of your instances.

1.5.2 Graylog – log ingestion done better

While WAZUH has agents available – this might not always be the best idea due to scalability and performance issues later on. When your SIEM has to handle a multitude of agents and logs – it can get crowded.

It is advised to utilize a separate log ingestion tool such as Graylog. During the implementation phase I have tried to establish Graylog log ingestion, directly forwarding the logs towards Wazuh. This creates a better layer between log ingesting and your SIEM – and helps in scaling your solutions later on.

While perfect in theory – one major issue came into the spotlight: **certificate management.** Since Wazuh is talking in secure HTTPS (HyperText Transfer Protocol Secure) – your graylog also needs to talk in HTTPS. Without proper certificate management you're not able to provide a solid connection between your log ingestion and your SIEM.

An early overview of your docker-compose.yml should include the Graylog instance + your Wazuh instances (Wazuh Indexer + Wazuh Manager + Wazuh Dashboard). *Noticed how I am clearly a fan of Docker (compose) deployments?*

```
mongodb:
  image: mongo:6.0.3
  #networks:
  #    - graylog
  #DB in share for persistence
  volumes:
    - ./mongo_data:/data/db

graylog:
  image: graylog/graylog:4.3-jre11
  #journal and config directories in local NFS share for persistence
  volumes:
    - ./graylog_journal:/usr/share/graylog/data/journal
  environment:
    # CHANGE ME (must be at least 16 characters)!
    - GRAYLOG_PASSWORD_SECRET=OQGWOEcM6R027M8rUnoZ
    # Password: admin
    - GRAYLOG_ROOT_PASSWORD_SHA2=cc98fd7aec95a055a76fd10c8d0cfa6e314b1a1fc5d6141544d29a0c1ba64945
    - GRAYLOG_HTTP_EXTERNAL_URI=http://0.0.0.0:9000/
    - GRAYLOG_ELASTICSEARCH_HOSTS="https://admin:SecretPassword@wazuh.indexer:9200"

  entrypoint: /usr/bin/tini -- wait-for-it https://wazuh.indexer:9200 --  /docker-entrypoint.sh
  #networks:
  #  - graylog
  links:
    - mongodb:mongo
    - wazuh.indexer
  restart: always
  depends_on:
    - mongodb
    - wazuh.indexer
  ports:
    # Graylog web interface and REST API
    - 9000:9000
    # Syslog TCP
    - 1516:1514
    # Syslog UDP
    - 1516:1514/udp
    # GELF TCP
    - 12201:12201
    # GELF UDP
    - 12201:12201/udp
```

This deployment is your **first option** – you can definitely select an **out of the box** "ova"-style deployment, similar to the Wazuh deployment in this SOC build.

- **Docker:** building your own image is definitely preferred – but might stir things up (certificate wise): My graylog-test docker image | Docker Hub
- **VM Linux installation:** Graylog installation manual | Graylog

After trying a multitude of solutions regarding the docker deployment Wazuh simply denied all connections coming from Graylog. Graylog effectively was and is running – but connecting HTTP to HTTPS is not that easy – and transforming Graylog docker containers to HTTPS is a tough nut to crack.

While Graylog has documentation readily available – and it seemed as easy as it looks – this definitely did not work out the way I intended it: How to make graylog talk HTTPS | Graylog

Even community pages stating it is "easy" to build a HTTPS Graylog instance: Community help - "easy" implementation | Graylog Community

After a multitude of attempts – every single certificate creation failed in which Graylog effectively had its self-signed certificates but could still not connect to Wazuh. Other implementations includes your certificate, but the key continuously failed to be created.

The long story short: **certificate management** is definitely not a piece of cake. In this case, the docker deployment might not be the best suitable solution. In the end this was in the early stages of the SOC build. At some point you simply leave it be and not implement is due to time constraints. Log ingestion, and definitely Graylog, is an absolute recommendation for your SOC. But make sure the first two building blocks are the primary focus: log ingestion and SIEM. Once both are working as intended other building blocks can be added. If this doesn't work – Wazuh does have log ingestion available via its excellent Wazuh Agents, forming your plan B.

Since I did not try the VM Linux installation – this might be a more suitable approach.

1.5.3 Local docker containers

While clearly a major fan of docker deployments, this SOC project leaves you with one major question: am I able to run this locally on VMs, or not? In my case the answer was clearly **not** thus shifting towards dockerizing was a more logical step.

Here is a GitHub example of how your final solution might look like: [Dockerizing your SOC build locally @ Defreddy | GitHub]

With a clear limitation in RAM memory (16 GB) – the entire docker build could run at rest. Meaning all containers run separately from one another and have no integrations included yet. Looking at Windows Task Manager the memory was starting to ramp up at 90% for just spinning all the containers locally.

That being said – dockerizing is preferred over VMs – but comes at a clear disadvantage: you only have so many ports on your host machine. This is one of the reasons why MISP, for example, also did not make it into the final solution. Since other instances had taken HTTP and HTTPS ports already – you couldn't even access the web browser via a standard docker-compose.yml deployment. Obviously workaround are here to stay – change your ports is highly advised if you are planning on running a local docker deployment.

1.5.3.1 MISP

As mentioned in the above section – MISP is included in most deployments. Both the local docker deployment and GCP deployment include MISP. The issue was ports. At this stage it was out of the question for a local docker container.

But why was it not included in the final solution? For some reason the docker-compose.yml deployment is not processing the UI correctly. While it is included on the soar-services host machine running all my SOAR services in GCP such as TheHive5 and Cortex, the web UI was not displaying as intended.

```
#  misp:
#    image: coolacid/misp-docker:core-latest
#    container_name: misp
#    restart: unless-stopped
#    depends_on:
#      - redis
#      - db
#    ports:
#      - "80:80"
#      - "443:443"
#    environment:
#      - "HOSTNAME=https://34.76.231.227/"
#      - "REDIS_FQDN=redis"
#      - "INIT=true"             # Initialze MISP, things includes, attempting to import SQL and the Files DIR
#      - "CRON_USER_ID=1"        # The MISP user ID to run cron jobs as
#      - "DISIPV6=true" # Disable IPV6 in nginx
#
#  misp-modules:
#    image: coolacid/misp-docker:modules-latest
#    container_name: misp-modules
#    environment:
#      - "REDIS_BACKEND=redis"
#    depends_on:
#      - redis
#      - db
```

The reasoning behind this is because some additional local settings (aside from your docker files) need to be adjusted to the correct base url (as you can see in the above environment variables).

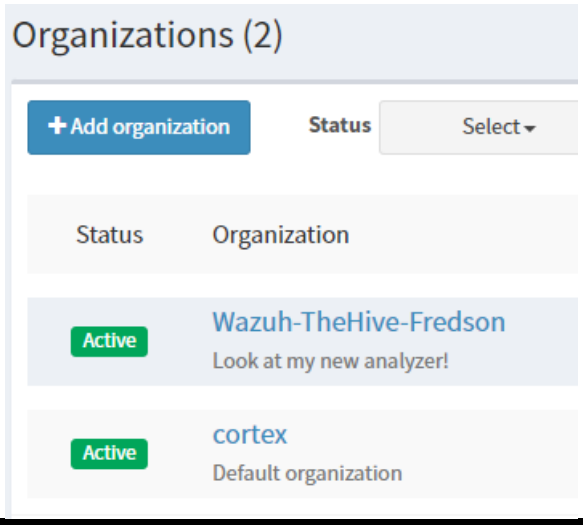
The same goes for MISP vs all other instances. You don't have to build it with docker: [How to deploy MISP on Ubuntu 20.04 | MISP]

1.5.4 Cortex

Last but not least – Cortex is effectively running in the final SOC solution – but not included in the process. While perfectly working and initial configuration is included:

Your SOAR (Shuffle) isn't always handling data correctly, it seems. When sending data (specifically non-parsed data resulting in "null" readings) it needs to be in the correct format. "Null" cannot be read and send to other building blocks thus generating an issue.

Since Wazuh's parameters are continuously shifting – null readings are also continuously presented. Sometimes you have an IP from your webhook, sometimes you don't.




```
Fredson@soar-services:~$ sudo docker ps
CONTAINER ID   IMAGE                                  COMMAND
       PORTS                                NAMES
7d57398910da   thehiveproject/cortex:latest           "/opt/cortex/entrypo…"
ours   0.0.0.0:9001->9001/tcp              cortex
```

In order to facilitate the IP-part you need to extend the shuffle workflow, similar to how a VirusTotal implementation would be setup. Instead of parsing the part in your "repeater" you simply build a new "repeater" in order to parse IPs. This would create / generate an entirely new flow and essentially an entirely new implementation and extension for your SOC.

While the implementation is very similar to TheHive5 (repeater + Cortex building blocks) it can absolutely be a wonderful extension for your SOC.

In the current SOC build hereby Cortex is not integrated in the workflow – but actively running in the background for a possible future implementation.

For more information – take a look at the beginning of this paper at point 1.5.4.1 – Cortex.

CONCLUSION

Building a SOC is definitely hard labor if you want to include a variety of aspects and integrations. Keeping in mind a SOC needs to be scalable for future implementations and providing sufficient abilities for vertical / horizontal scaling of single instances generates an interesting architectural design. The primary focus should be:

- **Scalability:** make sure your assets can scale horizontally and vertically. Avoid creating bottlenecks in your final solution by building every single aspect in one single VM, for example. Take your time to think of a logical architectural design.
- **Performance:** The entire SOC build requires a different kind of VM for every single instance. Wazuh and Caldera have requirements of at least 8 GB RAM, for example. If more aspects or integrations are provided on both of these VMs you will have to scale horizontally. Scaling horizontally is costly in a cloud environment and at some point you will reach a bottleneck since you can't scale horizontally forever. Mind your performance and think about the future: more hosts will be scanned by your SIEM, more logging will be integrated, more workload has to be handled.
- **Future-proofing:** as mentioned in the previous point – think about the future of your SOC implementation. Make sure implementations requiring high RAM or CPU capacity have a dedicated instance or VM. Keep implementations separated from one another by creating dedicated instances for each instance: make it logical. Keep in mind the bottleneck of scaling horizontally and obtaining optimal performance. Eventually the most efficient solution would be a Kubernetes cluster avoiding VMs altogether. Creating docker containers comes close – but still requires a variety of VMs separately due to load balancing. Kubernetes would avoid having to create VMs entirely making them redundant. Similar to docker deployments your code is essential Infrastructure As Code (IaC) making deploying, maintenance and future integrations easier than ever.

Taking in mind this primary focus – deploying to the cloud was the most optimal choice. Dockerizing as much as possible makes deploying easier due to Infrastructure As Code deployments. IaC makes life easy since you don't have to manage every single VM separately. New integration? Build or extend your docker-compose -> push to your code repository -> (deploy via your CI/CD pipeline) -> your machine is now fully operational, updated and easily maintainable for future patching and integrations.

Creating a Kubernetes cluster / setup is probably the most efficient way and also the toughest implementation. Every single aspect is literally code and makes patching, deploying and integrating easier. Scalability, flexibility and compatibility are major strengths for a Kubernetes cluster.

Some implementations require a virtual machine, some require a docker container and some require a Kubernetes cluster. Eventually one will replace the other – but a "hybrid" combination is essentially here to stay (for now).

The SOC build includes a red teaming tool, SIEM, SOAR and incident response hosted on GCP. Integrating all the pieces together creates an overall workflow in which cyber defenders obtain visibility of malicious threat actors abusing your assets. This is a crucial aspect – if I did not have SIEM enabled I might not even have noticed the brute force attacks, for example. Obtaining visibility in a modern day environment is crucial. A SOC is the perfect solution to provide just that.

Protect your dearest assets, obtain visibility, automate security and most definitely automate the response. Automated workflows blocking malicious IPs or removing malicious files are a must-have for modern environments but keep in mind your DevOps / architectural aspects. It is not solely a security implementation.

BIBLIOGRAPHY

/ D. (2019, March 19). *Graylog3 with https (easy tutorial)*. Graylog Community. https://community.graylog.org/t/graylog3-with-https-easy-tutorial/9519

/ F. (2021, December 14). *Integrating Shuffle with Virustotal and TheHive — Open Source SOAR part 3*. Medium. https://medium.com/shuffle-automation/integrating-shuffle-with-virustotal-and-thehive-open-source-soar-part-3-8e2e0d3396a9

/ F. (2022, April 14). *Real-time executions and IoC's with Shuffle, TheHive and MISP — Open Source SOAR part 4*. Medium. https://medium.com/shuffle-automation/indicators-and-webhooks-with-thehive-cortex-and-misp-open-source-soar-part-4-f70cde942e59

/ M. (n.d.). *GitHub - mitre/caldera: Automated Adversary Emulation Platform*. GitHub. https://github.com/mitre/caldera

/ S. (n.d.-a). *GitHub - Shuffle/Shuffle: Shuffle: A general purpose security automation platform. Our focus is on collaboration and resource sharing.* GitHub. https://github.com/Shuffle/Shuffle

/ S. (n.d.-b). *How do I import a OVA file in GCP*. Edureka Community. https://www.edureka.co/community/58630/how-do-i-import-a-ova-file-in-gcp

/ S. (n.d.-c). *Shuffle/functions/extensions/wazuh at main · Shuffle/Shuffle*. GitHub. https://github.com/Shuffle/Shuffle/tree/main/functions/extensions/wazuh

/ T. (n.d.-d). *Docker-Templates/README.md at main · TheHive-Project/Docker-Templates*. GitHub. https://github.com/TheHive-Project/Docker-Templates/blob/main/docker/thehive4-cortex3-misp-shuffle/README.md

/ W. (n.d.-a). *Detecting and removing malware using VirusTotal integration*. https://documentation.wazuh.com/current/proof-of-concept-guide/detect-remove-malware-virustotal.html

/ W. (n.d.-b). *How it works - VirusTotal integration · Wazuh documentation*. https://documentation.wazuh.com/current/user-manual/capabilities/virustotal-scan/integration.html

/ W. (n.d.-c). *Wazuh documentation*. https://documentation.wazuh.com/current/index.html

BlackPerl. (2021, December 31). *SOC Open Source, Build own SOAR with Shuffle, ELK-TheHive-Cortex-Teams Full Automation, Part 2*. YouTube. https://www.youtube.com/watch?v=Nb9_ahZMC5U

HackerSploit. (2021, October 29). *Red Team Adversary Emulation With Caldera*. YouTube. https://www.youtube.com/watch?v=EIHLXWnK1Dw

Import machine images from virtual appliances | Compute Engine Documentation |. (n.d.). Google Cloud. https://cloud.google.com/compute/docs/machine-images/import-machine-from-virtual-appliance

Import virtual appliances | Compute Engine Documentation |. (n.d.). Google Cloud. https://cloud.google.com/compute/docs/import/import-ovf-files

Installing CALDERA — caldera documentation. (n.d.). https://caldera.readthedocs.io/en/latest/Installing-CALDERA.html

Ishiaku, A. (2022, October 19). *Using Wazuh and TheHive for threat protection and incident response*. Wazuh. https://wazuh.com/blog/using-wazuh-and-thehive-for-threat-protection-and-incident-response/

MISP Web-interface not working as accepted · Issue #3272 · MISP/MISP. (n.d.). GitHub. https://github.com/MISP/MISP/issues/3272

Setup. (n.d.-a). https://go2docs.graylog.org/5-0/downloading_and_installing_graylog/installing_graylog_operations.html

Setup. (n.d.-b). https://go2docs.graylog.org/5-0/downloading_and_installing_graylog/installing_graylog_operations.html

StrangeBee - Docs. (n.d.-a). *TheHive 5 Documentation*. StrangeBee Docs. https://docs.strangebee.com/thehive/setup/installation/step-by-step-guide/

StrangeBee - Docs. (n.d.-b). *TheHive 5 Documentation*. StrangeBee Docs. https://docs.strangebee.com/thehive/setup/installation/docker/

Taylor Walton. (2021a, November 28). *Host Your Own SOAR - Shuffle Install*. YouTube. https://www.youtube.com/watch?v=YDUKZojg0vk

Taylor Walton. (2021b, December 13). *Shuffle + Wazuh + TheHIVE + Cortex = Automation Bliss*. YouTube. https://www.youtube.com/watch?v=FBISHA7V15c

Wazuh and Shuffle. (n.d.). Google Groups. https://groups.google.com/g/wazuh/c/yN5PQ-GpTls