User: root

Password: toor

Elastalert :

این برنامه لاگ ها رو به صورت هیستوریکال بررسی میکند. قبلا برنامه ای که برای لاگ کورولیشن استفاده میشد به صورت inline بود یعنی تمام لاگ ها را دونه به دونه وقتی وارد میشدند بررسی میکرد و چون بعضی از رول ها به صورت sequence دار هستند، کورولیشن inline سورس زیادی مصرف میکرد و بهینه نبود. و ما اومدیم لاگ ها را به صورت هیستوریکال بررسی میکنیم یعنی ما یک query داریم و اگر ان را ریکوعست بدیم سمت سرور الستیک سرچمان اگر ان کوعری وجود داشته باشد به ما یک پاسخی میدهد. هیستوریکال کورولیشن مزایای زیادی دارد ازجمله نیاز به ریسورس های خیلی کمتری دارد، false positive های کمتری دارد، سرعت خیلی بهتری دارد و خیلی از مزیت های دیگر.

<https://github.com/Yelp/elastalert>

قسمت هایی از پروژه که تغییر داده شده است:

فایل اصلی برنامه در مسیر /root/Elastalert/elastalert/elastalert/elastalert.py میباشد. در این فایل، فانکشن get\_hits() اصلی ترین تغییرات را دارد. در این قسمت ما eql , kql را اضافه کردیم (قبلا فقط dsl ساپورت میکرد)

قسمت بعدی get\_alert\_body() میباشد که وقتی یک رول مچ بخورد این alert\_body را که اینجا تعریف کردیم را میسازد و به ایندکس elastalert\_status میفرستد. مثلا در اینجا میتوان ANPquery را اضافه کرد یا match\_body را عوض کرد و مطابق با هر خواسته ای میشود انرا کاستومایز کرد.

اجرای برنامه:

فایل اصلی برنامه در مسیر /root/Elastalert/elastalert/elastalert/elastalert.py میباشد. برای شروع برنامه الست الرت باید دستور python -m elastalert.elastalert در مسیر /root/Elastalert/elastalert/elastalert را زد. این دستور همه ی رول های داخل پوشه example\_rules را اجرا میکند. اگر بخواهیم یک رول را اجرا کنیم باید این دستور را زد:

python -m elastalert.elastalert –rule example\_rules [rule\_name]

ساختار فایل های رول:

این فایل های yaml هستند. فایل رول چند قسمت مختلف دارد. اولین قسمت ANPdata میباشد که اطلاعاتی مثل تاریخ نوشته شدن رول را نوشته. قسمت بعدی ANPrule میباشد. این قسمت اطلاعات کلی راجع به رول میدهد. مثلا قسمت مچ شدن رول با فریمورک MITRE ATT&CK را نشان میدهد و یا اینکه severity رول را نشان میدهد. قسمت های بعدی name, index رول میباشد. اگر رول kql باشد میتوان ایندکس را wildcard (\*) زد و اگر eql باشد میتوان اسم alias را زد (نمیتوان \*را استفاده کرد) در هر دوحالت eql و kql میتوان یک ایندکس را نوشت. در eql یک ویژگی اضافه شده که اگر اسم 2 یا بیشتر از 2 ایندکس را نوشتیم و با , (, + space) جدا کردیم، نتایج آن ایندکس ها را جمع میکند و نشان میدهد. قسمت بعدی type میباشد. Type های مختلفی داریم:

* Match where there are at least X events in Y time" (frequency type)
* Match when the rate of events increases or decreases" (spike type)
* Match when there are less than X events in Y time" (flatline type)
* Match when a certain field matches a blacklist/whitelist" (blacklist and whitelist type)
* Match on any event matching a given filter" (any type)
* Match when a field has two different values within some time" (change type)
* Match when a never before seen term appears in a field" (new\_term type)
* Match when the number of unique values for a field is above or below a threshold (cardinality type)

قسمت بعدی query میباشد. بنا به اینکه کوعری eql باشد یا kql در قسمت query نوشته میشود.

قسمت بعدی alert میباشد. وقتی یک رول مچ بخورد به الرتی که توی رول تعریف شده است اطلاعات رو میفرستد. Alert های مختلفی داریم مثل:

* Email
* JIRA
* OpsGenie
* Commands
* HipChat
* MS Teams
* Slack
* Telegram
* GoogleChat
* AWS SNS
* VictorOps
* PagerDuty
* PagerTree
* Exotel
* Twilio
* Gitter
* Line Notify
* Zabbix

قسمت های name, index, query, alert حتما باید باشند. اگر وجود نداشته باشند برنامه ارور میدهد. همچنین اگر name یک رول با رول دیگر مثل هم باشند برنامه خطا میدهد.

بک گراند EQL و : KQL

Kql به صورت key & value میباشد که باید در یک query\_sring قرار بگیر مثلا

"query": {"query\_string": {"query": "event.category:process and event.type:start and process.name:nslookup.exe and process.args:(-querytype=\* or -qt=\* or -q=\* or -type=\*)" }  }

در kql نمیتوان event های پی در پی (sequence دار) را تشخیص داد ولی در eql میشود. دو مثال از eql :

"query": "process where event.type in (\"start\", \"process\_started\") and\n process.name: (\"powershell.exe\", \"pwsh.exe\") and process.args : \"New-MailboxExportRequest\*\""

curl -X GET "localhost:9200/my-data-stream/\_eql/search?pretty" -H 'Content-Type: application/json' -d'

{

"query": "sequence\n

[ process where process.name == \"regsvr32.exe\" ]\n

[ file where stringContains(file.name, \"scrobj.dll\") ]"

{

'

ما در eql 2 تا فیلد اساسی داریم. Event\_category , condition

چند نوع event\_condition وجود دارد:

'any'

'process'

'network'

'file'

'registry'

'authentication'

'configuration'

'database'

'driver'

'host'

'iam'

'intrusion\_detection'

'malware'

'package'

'session'

'threat'

'web'

کاندیشن هم بر اساس استاندارد ECS (elastic common schema) میباشد.

توجه شود که result های هر کدام از eql ها فرق میکند با هم (sequence دار و بدون sequence)

Elasticsearch’s Detection Rules:

<https://github.com/elastic/detection-rules/tree/main/rules>

در این مسیر تمام رول های مختلف ویندوزی، لینوکسی، و غیره میباشد که بروزرسانی میشود. فایل های تعریف شده در این مسیر همه toml میباشند که باید تبدیل به yaml شوند که قابل خواندن با برنامه elastalert باشند. لینک یک convertor برای تبدیل toml به yaml : <https://www.convertsimple.com/convert-toml-to-yaml/>

پروژه جنگو:

ابتدا تو مسیر /root/Elastalert/backapi/rest/rule رفته و دستور

python manage.py runserver 192.168.250.122:9500 را زده. توجه شود که میتوان هر پورتی را داد. اگر superuser تعریف نشده است میتوان ان را تعریف کرد با دستور python manage.py createsuperuser و سپس یوزر و پسورد را تعریف میکنیم. با این کار میشود صفحه ی ادمین را دید. مسیر <http://192.168.250.122:9500/admin/> را بزنید و یوزر و پسورد admin admin بزنید.

این پروژه جنگو چه کاری انجام میدهد؟ با استفاده از api های تعریف شده در این پروژه میتوان رول ها و استراتژی ها را اضافه، حذف، و تغییر داد. Api ها در قسمت urls تعریف شده اند.

مراحل نصب :

apt install python-pip

pip install elasticsearch

pip install -r requirements.txt

pip install -r requirements-dev.txt

pip install "setuptools>=11.3"

python setup.py install

update-alternatives --install /usr/bin/python python /usr/bin/python3.7 2

apt-get install python3-pip

python -m pip install elasticsearch==7.13.4

pip install -r requirements.txt

pip install -r requirements-dev.txt

pip install -r requirements-ex.txt

python -m pip install --upgrade requests

python -m pip install --upgrade urllib3

Made a workaround by editing my /home/pi/.local/lib/python3.7/site-packages/elastalert/alerts.py

Changing lines:

26 from jira.client import JIRA

27 from jira.exceptions import JIRAError

to:

26 #from jira.client import JIRA

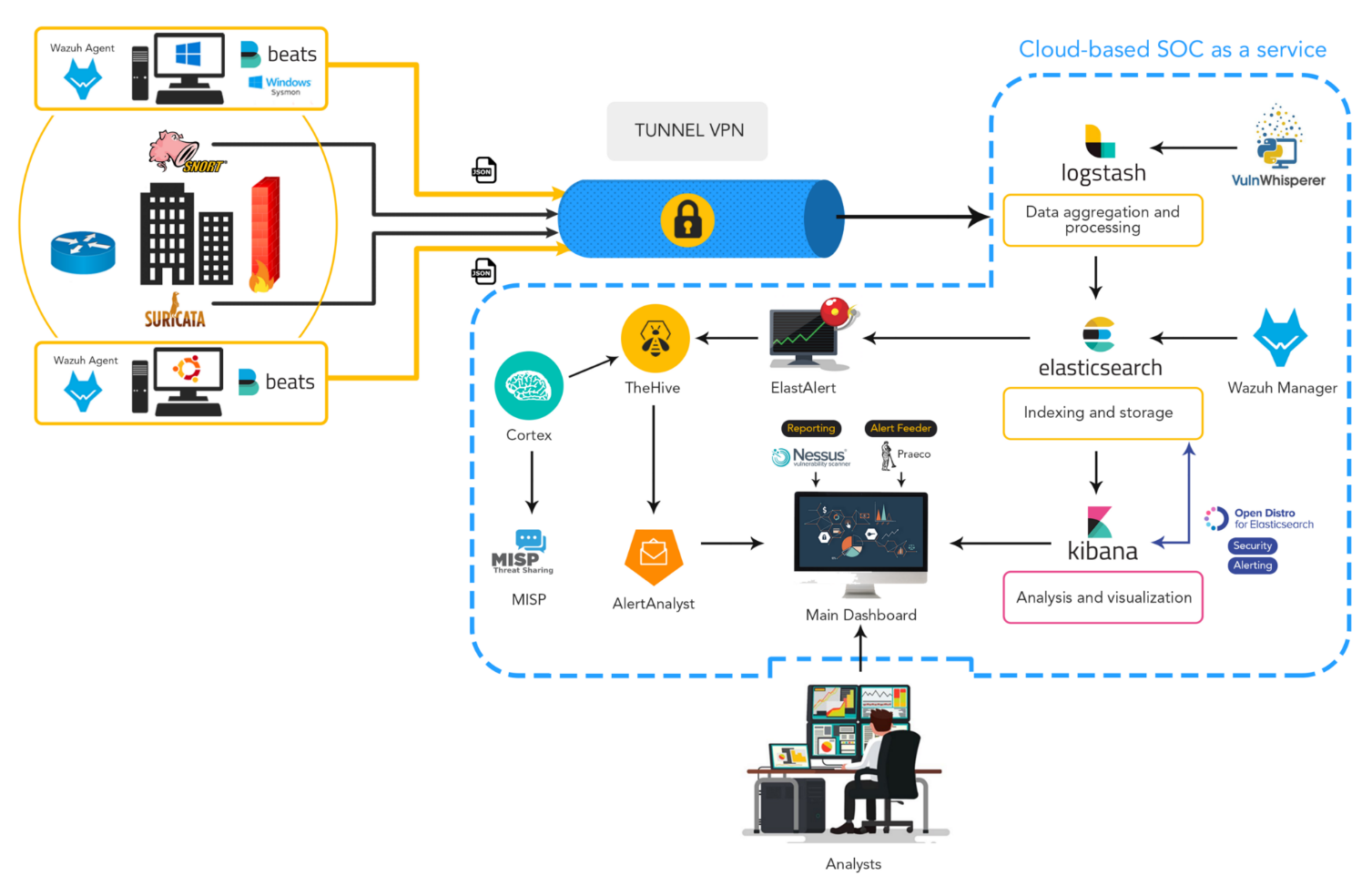
27 #from jira.exceptions import JIRAError

28 JIRAError = KeyError

pip install pyflowchart

NOTE: you should have a same ip address range with the elasticsearch that you send query. You can add ip address on your interfaces by going to this path: nano /etc/network/interfaces





اخرین تستی که انجام شد این رول ها مچ خوردند.

