Implement ELK as Cluster

Secure API with certificate

Hosein Tahaee
TAHAEE.H@GMAIL.COM

راه اندازی الستیک به صورت کلاستر و secure کردن ارتباطات بین نودها

اهداف:

1. **راه‌اندازی کلاستر Elasticsearch** با سه نود (Node1, Node2, Node3).
2. **راه‌اندازی Kibana** که به‌صورت امن به کلاستر متصل شود.
3. **راه‌اندازی Logstash** که به‌صورت امن به Elasticsearch ارسال/دریافت لاگ‌ها را انجام دهد.
4. استفاده از **گواهی‌های SSL/TLS** برای ارتباط **HTTP** (REST API) و **Transport** (ارتباط داخلی نودهای Elasticsearch).

بهتر است برای راه اندازی و درادامه آن امن سازی ارتباط از نام دامنه FQDN سیستم ها استفاده شود تا در صورت تعویض آی پی، تنظیمات Certification تغییری نکند.

Full Domain	Hostname	IP
ELK-node1.tosinso.local	ELK-node1	192.168.104.175
ELK-node2.tosinso.local	ELK-node2	192.168.104.176
ELK-node3.tosinso.local	ELK-node3	192.168.104.177
ELK-kibana.tosinso.local	ELK-kibana	192.168.104.178
ELK-logstash.tosinso.local	ELK-logstash	192.168.104.179

بخش اول: نصب و پیکربندی اولیه Elasticsearch

در اینجا آخرین نسخه الستیکس یعنی **Elasticsearch 8.x** یا مشابه آن فرض شده است (نسخه‌های x.7 به بالا روش کمابیش یکسانی دارند).

1. نصب Elasticsearch روی هر سه سرور

روی هر سه نود (ELK-node1, ELK-node2, ELK-node3) Elasticsearch را نصب کنید (از مخازن رسمی یا بسته .deb/.rpm). پس از نصب، فایل پیکربندی اصلی در مسیر `/etc/elasticsearch/elasticsearch.yml` را ویرایش می‌کنیم.

2. تنظیمات اولیه در فایل elasticsearch.yml

برای هر نود (با توجه به IP و نام خودش) حداقل موارد زیر را قرار می‌دهیم:

Node1 (`/etc/elasticsearch/elasticsearch.yml`):

```
cluster.name: "my-elk-cluster"
node.name: "ELK-node1"

# network
network.host: 0.0.0.0
http.port: 9200
transport.port: 9300

# discovery and cluster
discovery.seed_hosts: ["192.168.104.176", "192.168.104.177"]
cluster.initial_master_nodes: ["ELK-node1", "ELK-node2", "ELK-node3"]

# xpack security (فعال‌سازی امنیت)
xpack.security.enabled: true
xpack.security.transport.ssl.enabled: true
xpack.security.http.ssl.enabled: true
```

Node2 (`/etc/elasticsearch/elasticsearch.yml`):

```
cluster.name: "my-elk-cluster"
node.name: "ELK-node2"

network.host: 0.0.0.0
http.port: 9200
transport.port: 9300

discovery.seed_hosts: ["192.168.104.175", "192.168.104.177"]
cluster.initial_master_nodes: ["ELK-node1", "ELK-node2", "ELK-node3"]

xpack.security.enabled: true
xpack.security.transport.ssl.enabled: true
xpack.security.http.ssl.enabled: true
```

Node3 (`/etc/elasticsearch/elasticsearch.yml`):

```
cluster.name: "my-elk-cluster"
node.name: "ELK-node3"

network.host: 0.0.0.0
http.port: 9200
transport.port: 9300

discovery.seed_hosts: ["192.168.104.175", "192.168.104.176"]
cluster.initial_master_nodes: ["ELK-node1", "ELK-node2", "ELK-node3"]

xpack.security.enabled: true
xpack.security.transport.ssl.enabled: true
xpack.security.http.ssl.enabled: true
```

نکته: اگر در حال راه‌اندازی اولیهٔ کلاستر هستید، فقط روی یکی از نودها (مثلاً نود1) `cluster.initial_master_nodes` را قرار دهید. اما اگر قبلاً کلاستر را راه‌اندازی کرده‌اید، همان مقادیر قبلی را حفظ کنید و فقط تنظیمات SSL را اضافه کنید.

تا اینجای کار، Elastic امنیت را فعال کرده ولی هنوز گواهی‌های SSL و مقادیر کلید و گواهی‌ها (certificate/key path) را تعیین نکرده‌ایم. در ادامه سراغ ساخت گواهی‌ها می‌رویم.

بخش دوم: ساخت گواهی‌ها با elasticsearch-certutil

1. ساخت CA (گواهی مرجع)

ابتدا روی یکی از سرورهای Elasticsearch (ترجیحاً نود مستر، مثلاً Node1) دستور زیر را اجرا کنید. محل نصب Elasticsearch ممکن است متفاوت باشد (مثلاً `usr/share/elasticsearch/` یا `opt/elasticsearch`)، اما فرض می‌کنیم پوشهٔ باینری‌ها اینجاست:

```
cd /usr/share/elasticsearch
sudo bin/elasticsearch-certutil ca
```

در ادامه از شما می‌پرسد که یک **Password** برای CA تنظیم کنید! توصیه می‌شود رمز قرار دهید. در نهایت یک فایل با پسوند `.p12` ایجاد می‌شود (مثلاً `elastic-stack-ca.p12`).

این فایل را در مسیری امن (مثلاً `/etc/elasticsearch/certs/`) قرار دهید.

2. ساخت گواهی برای نودها (Node1, Node2, Node3) و سرویس‌ها (Kibana, Logstash)

بعد از ساخت CA، دستور زیر را اجرا می‌کنیم تا برای چندین هاست، گواهی صادر شود:

```
sudo bin/elasticsearch-certutil cert --ca /etc/elasticsearch/certs/elastic-stack-ca.p12
```

پس از زدن دستور فوق

1. از شما پرسیده می‌شود آیا می‌خواهید چندین گواهی برای هاست‌های مختلف صادر کنید یا یکی‌یکی. گزینه "multiple" را انتخاب کنید (در اکثر نسخه‌ها با پرسش "Generate a certificate per node?" مواجه می‌شوید).
2. حال باید یک لیست از **hostname** و **SAN** ارائه دهید. به هر هاست یک Label اختصاص داده و IP/Hostname آن را وارد کنید.

به عنوان مثال (نسخه‌های اخیر) یک پرسش تعاملی دارد شبیه زیر:

Please enter the desired output file [certs.zip]

فایل خروجی نهایی یک archive هست (مثلاً `certs.zip`). سپس از شما می‌خواهد اطلاعات هر هاست را وارد کنید. شما می‌توانید برای هر هاست یک Label وارد کنید (مثال):

```
Enter name for this instance: node1
Enter IP Addresses for instance (comma-separated if more than one) [None]: 192.168.104.175
Enter DNS names for instance (comma-separated if more than one) [None]: ELK-node1.tosinso.local,ELK-node1
...
Enter name for this instance: node2
Enter IP Addresses for instance [None]: 192.168.104.176
Enter DNS names for instance [None]: ELK-node2.tosinso.local,ELK-node2
...
Enter name for this instance: node3
Enter IP Addresses for instance [None]: 192.168.104.177
Enter DNS names for instance [None]: ELK-node3.tosinso.local,ELK-node3
...
Enter name for this instance: kibana
Enter IP Addresses for instance [None]: 192.168.104.178
Enter DNS names for instance [None]: ELK-kibana.tosinso.local,ELK-kibana
...
Enter name for this instance: logstash
Enter IP Addresses for instance [None]: 192.168.104.179
Enter DNS names for instance [None]: ELK-logstash.tosinso.local,ELK-logstash
```

در نهایت با فشردن Enter فرآیند ساخت گواهی‌ها تمام می‌شود و فایلی به نام مثلاً `certs.zip` ایجاد می‌شود که در داخلش پوشه‌هایی (بر اساس همان Label ها) حاوی `node.key`, `node.crt` و CA Chain خواهند بود.

3. انتقال گواهی‌ها به سرورهای موردنظر

فایل `certs.zip` تولیدشده را به هر سرور منتقل کنید و در مسیر مناسب قرار دهید. برای مثال:

- محتوای مربوط به نود1 (داخل فولدر `node1`) را به `etc/elasticsearch/certs/node1/` کپی کنید.

نود2 را به `etc/elasticsearch/certs/node2/`

.. و الی آخر

همینطور برای Kibana و Logstash

همچنین مطمئن شوید دسترسی فایل‌ها محدود باشد و Elasticsearch/Kibana/Logstash (با کاربر `elasticsearch`، `kibana`، یا `logstash`) بتواند آن‌ها را بخواند.

بخش سوم: تنظیمات SSL در Elasticsearch

حالا می‌خواهیم **هم در لایه Transport** و **هم در لایه HTTP** گواهی را مشخص کنیم.

در مثال زیر برای Node1 فرض می‌کنیم فایل گواهی و کلید نام‌های `node.crt` و `node.key` دارند، و CA با نام `ca.crt` در همان فولدر موجود است.

Node1 (`/etc/elasticsearch/elasticsearch.yml`):

```
xpack.security.transport.ssl.enabled: true
xpack.security.transport.ssl.verification_mode: certificate
xpack.security.transport.ssl.key: /etc/elasticsearch/certs/node1/node.key
xpack.security.transport.ssl.certificate: /etc/elasticsearch/certs/node1/node.crt
xpack.security.transport.ssl.certificate_authorities: [ "/etc/elasticsearch/certs/node1/ca.crt" ]

xpack.security.http.ssl.enabled: true
xpack.security.http.ssl.key: /etc/elasticsearch/certs/node1/node.key
xpack.security.http.ssl.certificate: /etc/elasticsearch/certs/node1/node.crt
xpack.security.http.ssl.certificate_authorities: [ "/etc/elasticsearch/certs/node1/ca.crt" ]
```

برای Node2 و Node3 دقیقاً همین مقادیر را می‌گذاریم، اما مسیر فایل‌ها را مطابق پوشه مربوط به خودشان تغییر می‌دهیم:

```
xpack.security.transport.ssl.key: /etc/elasticsearch/certs/node2/node.key
xpack.security.transport.ssl.certificate: /etc/elasticsearch/certs/node2/node.crt
...
xpack.security.http.ssl.key: /etc/elasticsearch/certs/node2/node.key
xpack.security.http.ssl.certificate: /etc/elasticsearch/certs/node2/node.crt
...
```

و به همین ترتیب برای Node3.

> **نکته مهم**: قبل از ری‌استارت کردن، مطمئن شوید که **فایل‌های گواهی** را در مسیر جدید کپی کرده، دسترسی‌ها (permission) درست هستند و نام فایل‌ها با پیکربندی همخوانی دارد.

حالا هر سه سرویس Elasticsearch را **ری‌استارت** کنید تا کلاستر با تنظیمات SSL بالا بیاید:

```
sudo systemctl restart elasticsearch
```

بخش چهارم: تنظیمات Kibana با SSL

روی سرور (192.168.104.178) Kibana فایل‌های مربوط به `/kibana` (که در مراحل ساخت با certutil ساخته شد) را در مثلاً `/etc/kibana/certs/` قرار دهید:

```
kibana.crt
```

```
kibana.key
ca.crt
```

سپس فایل پیکربندی اصلی Kibana (معمولاً /etc/kibana/kibana.yml) را ویرایش کنید:

```
elasticsearch.hosts: ["https://ELK-node1.tosinso.local:9200","https://ELK-node2.tosinso.local:9200","https://ELK-node3.tosinso.local:9200"]
elasticsearch.ssl.certificateAuthorities: [ "/etc/kibana/certs/ca.crt" ]
elasticsearch.ssl.certificate: "/etc/kibana/certs/kibana.crt"
elasticsearch.ssl.key: "/etc/kibana/certs/kibana.key"
elasticsearch.ssl.verificationMode: certificate

server.ssl.enabled: true
server.ssl.certificate: "/etc/kibana/certs/kibana.crt"
server.ssl.key: "/etc/kibana/certs/kibana.key"
```

در نهایت Kibana را ری استارت کنید:

```
sudo systemctl restart kibana
```

حالا Kibana از طریق HTTPS در پورت 5601 (به‌صورت پیش‌فرض) در دسترس خواهد بود و برای برقراری ارتباط به Elasticsearch از گواهی استفاده می‌کند.

بخش پنجم: تنظیمات Logstash با SSL

روی سرور (192.168.104.179) Logstash نیز فایل‌های logstash.crt, logstash.key, ca.crt را در مثلاً مسیر /etc/logstash/certs/ قرار می‌دهیم.

دو حالت رایج برای امنیت Logstash وجود دارد:

1. **Logstash به Elasticsearch** خروجی می‌فرستد (Output).
2. **Logstash ورودی از Beats** یا سرویس‌های دیگر دارد. (این‌جا هم می‌توان SSL را فعال کرد.)

5.1. امن‌سازی ارتباط Logstash → Elasticsearch

در فایل کانفیگ Logstash (مثلاً /etc/logstash/conf.d/output.conf) بخشی شبیه زیر خواهید داشت:

```
output {
  elasticsearch {
    hosts => ["https://ELK-node1.tosinso.local:9200","https://ELK-node2.tosinso.local:9200","https://ELK-node3.tosinso.local:9200"]
    user => "elastic"
    password => "YourPassword"
    ssl => true
    cacert => "/etc/logstash/certs/ca.crt"
    ssl_certificate => "/etc/logstash/certs/logstash.crt"
    ssl_key => "/etc/logstash/certs/logstash.key"
  }
}
```

نکته: در بسیاری از سناریوها برای خروجی صرفاً نیاز است ssl => true و cacert => "/path/to/ca.crt" استفاده کنید و لزومی ندارد کلید خصوصی (logstash.key) برای outbound استفاده شود. اما اگر احراز هویت دوطرفه (mutual TLS) می‌خواهید، پارامترهای ssl_certificate و ssl_key را هم قرار می‌دهید.

5.2. امن‌سازی ورودی (اختیاری)

اگر شما از **Logstash Beats Input** یا **TCP Input** استفاده می‌کنید و می‌خواهید ارتباط کلاینت‌ها (مثلاً Filebeat) با Logstash هم امن باشد، در بخش input کانفیگ Logstash به‌شکل زیر عمل می‌کنید:

```
input {
  beats {
    port => 5044
    ssl => true
    ssl_certificate => "/etc/logstash/certs/logstash.crt"
    ssl_key => "/etc/logstash/certs/logstash.key"
    # در صورت نیاز اگر بخواهید client certificate شود چک شود آنها:
    ssl_verify_mode => "force_peer"
    cacert => "/etc/logstash/certs/ca.crt"
  }
}
```

نکته: این را فقط در صورتی نیاز دارید که می‌خواهید از Filebeat/Metricbeat و... به Logstash با TLS وصل شوید.

در انتها سرویس Logstash را ری‌استارت می‌کنیم:

```
sudo systemctl restart logstash
```

در انتها این مورد به صورت جامع تشریح شده است.

بخش ششم: اعتبارسنجی و تست نهایی

وضعیت کلاستر Elasticsearch:

از یکی از نودها یا حتی سیستم کلاینت، دستور زیر را با curl انجام دهید (دقت کنید که الان HTTPS فعال شده است و پورت 9200 روی SSL کار می‌کند):

```
curl -k --user elastic:YourPassword https://192.168.104.175:9200/_cluster/health?pretty
```

اگر OK و سبز (یا زرد یا قرمز اما بدون خطای SSL) جواب داد، یعنی ارتباط SSL برقرار است.

(در محیط Production توصیه می‌شود از `-k` استفاده نکنید و به‌جای آن CA را به سیستم عامل یا پارامتر `--cacert` بدهید.)

دسترسی Kibana:

در مرورگر به آدرس:

```
https://192.168.104.178:5601
```

رفته و مطمئن شوید Kibana بالا می‌آید و به Elasticsearch وصل است. اگر از شما کاربر و پسورد پرسید (مثل `elastic` یا هر کاربر دیگری که ساختید) وارد شوید و اطلاعات کلاستر را ببینید.

ارسال لاگ از Logstash به Elasticsearch:

لاگ را به پورت 5044 (اگر Beats است) یا هر پورت ورودی دیگر ارسال کنید؛ خروجی هم به Elasticsearch امن اشاره می‌کند و نباید خطای SSL دریافت کنید.

نکات:

- **استفاده از نام دامنه (FQDN) یا Subject Alternative Name** در هنگام ساخت گواهی‌ها بسیار مهم است. اگر نام یا IP جدید اضافه می‌کنید، لازم است گواهی جدید صادر کنید.
- بهتر است برای راحتی مدیریت، یک DNS داخلی راه‌اندازی شود و هر نود با نام FQDN خودش شناسایی شود تا با تغییر IP، مجبور به ساخت مجدد گواهی نشوید.
- اگر در آینده بخواهید یک نود جدید اضافه کنید، کافی‌ست از همان CA استفاده کنید و برای نود جدید با `elasticsearch-certutil` یک گواهی جدید بسازید و آن را به تنظیماتش اضافه کنید.

اگر مایلید ارتباط بین Filebeat و مقصد (Elastic یا Logstash) هم امن باشد، باید در پیکربندی Filebeat بخش مربوط به **SSL** را فعال کنید و گواهی مناسب را به آن بدهید. به صورت کلی دو سناریو رایج داریم:

1. **Filebeat →\rightarrow→ Logstash**
2. **Filebeat →\rightarrow→ Elasticsearch**

در هر دو حالت اگر می‌خواهید ارتباط رمزگذاری‌شده (TLS/SSLTLS/SSLTLS/SSL) باشد و (در صورت نیاز) گواهی کلاینت هم بررسی شود، باید در فایل کانفیگ ‘filebeat.yml‘ پارامترهای SSL را مقداردهی کنید.

1) اگر خروجی Filebeat به Logstash است

فرض می‌کنیم Logstash را روی پورت ۵۰۴۴ با SSL فعال کرده‌اید. در تنظیمات Logstash هم چیزی شبیه این داشتید:

```
input {
  beats {
    port => 5044
    ssl => true
    ssl_certificate => "/etc/logstash/certs/logstash.crt"
    ssl_key => "/etc/logstash/certs/logstash.key"
    # اگر تمایل دارید کلاینت‌ها هم با گواهی معتبر شناسایی شوند:
    #ssl_verify_mode => "force_peer"
    #cacert => "/etc/logstash/certs/ca.crt"
  }
}
```

حالا در فایل `filebeat.yml` (یا در ماژول‌های Filebeat) باید خروجی را با SSL فعال کنید:

```
output.logstash:
  hosts: ["192.168.104.179:5044"]
  ssl.enabled: true
  # اگر فقط نیاز دارید Certificate Authority لاگ‌استش را تأیید کند:
  ssl.certificate_authorities: ["/etc/filebeat/certs/ca.crt"]

  # اگر Mutual TLS می‌خواهید و لاگ‌استش ssl_verify_mode => "force_peer" گذاشته:
  # ssl.certificate: "/etc/filebeat/certs/filebeat.crt"
  # ssl.key: "/etc/filebeat/certs/filebeat.key"
```

مقدار `hosts` باید حاوی آدرس و پورت Logstash باشد.

- در `ssl.certificate_authorities` مسیر همان CA مرجع (یا گواهی سرور لاگ‌استش) را می‌دهید تا Filebeat بتواند اعتبار گواهی Logstash را بررسی کند.

نکته درباره Mutual TLS

اگر در Logstash از `"ssl_verify_mode => "force_peer` استفاده کرده‌اید یا می‌خواهید **دوطرفه** احراز هویت شود، لازم است Filebeat نیز گواهی و کلید اختصاصی خود را داشته باشد و در کانفیگش مقدار دهید:

```
ssl.certificate: "/etc/filebeat/certs/filebeat.crt"
ssl.key: "/etc/filebeat/certs/filebeat.key"
```

و در سرور Logstash نیز `"cacert => "/etc/logstash/certs/ca.crt` و `"ssl_verify_mode => "force_peer` قرار دهید تا گواهی کلاینت را تأیید کند.

2) اگر خروجی Filebeat به Elasticsearch است

گاهی Filebeat مستقیماً به Elasticsearch متصل می‌شود و دیگر نیازی به Logstash ندارید. در این صورت، اگر Elasticsearch را روی HTTPS پیکربندی کرده‌اید (پورت 9200)، فایل `filebeat.yml` شما باید شامل تنظیمات زیر باشد:

```
output.elasticsearch:
  hosts: ["https://192.168.104.175:9200", "https://192.168.104.176:9200", "https://192.168.104.177:9200"]
  username: "elastic"
  password: "YourPassword"
  ssl.certificate_authorities: ["/etc/filebeat/certs/ca.crt"]
  # ssl.certificate: "/etc/filebeat/certs/filebeat.crt"
  # ssl.key: "/etc/filebeat/certs/filebeat.key"
```

حتماً در `hosts` از `https://` استفاده کنید.

در `ssl.certificate_authorities` مسیر CA همان Elasticsearch (یا CA داخلی Elastic) را قرار دهید.

اگر از `xpack.security.http.ssl.client_authentication: required` در Elasticsearch استفاده می‌کنید (یعنی **کلاینت هم باید گواهی داشته باشد**)، پس در Filebeat باید گواهی و کلید خودش را تنظیم کنید.

نکات:

مسیر گواهی‌ها در Filebeat: اگر فایل گواهی CA یا گواهی/کلید خود کلاینت را جداگانه روی سرور دارید، مسیرها را در Filebeat دقیق تنظیم کنید. همچنین سطح دسترسی فایل‌ها طوری باشد که فایل‌بیت (کاربر filebeat) بتواند آن‌ها را بخواند.

سطح تأیید SSL: اگر به دلایلی نیاز دارید تأیید را ساده‌تر کنید (برای محیط تست) می‌توانید `ssl.verification_mode: none` بگذارید. ولی در محیط Production توصیه نمی‌شود.

تهیه گواهی Filebeat: در صورتی که می‌خواهید احراز هویت دوجانبه (mutual TLS) داشته باشید، نیاز است هنگام تولید گواهی‌ها (با `elasticsearch-certutil` یا ابزار CA دیگر) برای Filebeat نیز یک گواهی صادر کنید تا Logstash یا Elasticsearch آن را تأیید کنند.

اررورهای رایج SSL: اگر mismatch در Subject Alternative Name رخ دهد (مثلاً شما IP در گواهی قرار نداده باشید ولی با IP وصل شوید، یا نام دامنه با گواهی نخورد)، Filebeat خطای SSL می‌دهد. در این صورت باید یا از DNS/FQDN مطابق گواهی استفاده کنید یا SAN گواهی را اصلاح کنید.

بخش زیر به صورت کامل به احراز هویت دوجانبه میپردازد.

احراز هویت دوجانبه

در احراز هویت دوجانبه (Mutual TLS/MTLS)، هم کلاینت (اینجا Filebeat) و هم سرور (Logstash یا Elasticsearch) باید گواهی‌های معتبر ارائه کنند تا هویت یکدیگر را تصدیق کنند. در ادامه، مراحل کامل برای سناریوهای Filebeat → Logstash و Filebeat → Elasticsearch را بررسی می‌کنیم.

بخش اول: پیش‌نیاز و ساخت گواهی‌ها

1. ساخت یک CA (گواهی ریشه)

فرض می‌کنیم قبلاً با ابزار `elasticsearch-certutil` یا ابزار دیگر، یک CA (گواهی ریشه) ساخته‌اید. مثلاً خروجی زیر را دارید:

`elastic-stack-ca.p12`

یا کلید و گواهی‌های معادل آن (مثلاً `ca.crt` و `ca.key`).

2. صدور گواهی برای Filebeat و سرور مقصد (Logstash/Elasticsearch)

حالت الف: استفاده از `elasticsearch-certutil`

1. همان‌جا که CA دارید (مثلاً `/etc/elasticsearch/certs/elastic-stack-ca.p12`)، دستور زیر را اجرا کنید تا گواهی سرویس‌ها و کلاینت‌ها (از جمله Filebeat) ساخته شود:

```
sudo bin/elasticsearch-certutil cert
--ca /etc/elasticsearch/certs/elastic-stack-ca.p12
```

به سؤالات تعاملی پاسخ دهید. برای هر هاست/کلاینت یک برچسب (Label) و IP/DNS وارد کنید. مثلاً:

```
Enter name for this instance: filebeat
Enter IP Addresses for instance [None]: 192.168.104.180
Enter DNS names for instance [None]: filebeat.tosinso.local
```

برای Elasticsearch، Logstash یا هر سرویس دیگر هم مسیر مشابه.

1. در انتها فایلی مثل `certs.zip` خواهید داشت که درون آن فولدرهایی بر اساس Label ساخته شده است (مثلاً `/filebeat` حاوی `filebeat.crt, filebeat.key, ca.crt`).
2. فایل مربوط به `/filebeat` را روی سروری که Filebeat اجرا می‌شود کپی کرده و در مسیر امنی مثلاً `/etc/filebeat/certs/` قرار دهید.

نکته: اگر پیش‌تر برای Logstash یا Elasticsearch هم گواهی ساخته‌اید، از همان‌ها استفاده کنید (نیاز به ساخت مجدد نیست)؛ فقط حتماً در Logstash یا Elasticsearch نیز از همین CA یا CA همسان استفاده شده باشد تا دو طرف همدیگر را بشناسند.

بخش دوم: تنظیمات احراز هویت دوجانبه Filebeat → Logstash

در این سناریو، Logstash روی پورت 5044 با Beats input راه‌اندازی می‌شود و MTLS برقرار است.

1. پیکربندی Logstash (سمت سرور)

فایل کانفیگ ورودی در Logstash

فرض کنیم فایل کانفیگ ورودی در مسیر `logstash.conf` یا `input.conf-01` باشد:

```
input {
  beats {
    port => 5044
    ssl => true
    ssl_certificate => "/etc/logstash/certs/logstash.crt"
    ssl_key => "/etc/logstash/certs/logstash.key"

    # فایل گواهی CA ریشه که Logstash می‌کند با آن گواهی کلاینت را اعتبارسنجی:
    cacert => "/etc/logstash/certs/ca.crt"

    # الزامی کردن ارائه گواهی از سمت کلاینت:
    ssl_verify_mode => "force_peer"
  }
}
```

`ssl_verify_mode => "force_peer"` به این معناست که Logstash، **گواهی** Filebeat را هم بررسی کند و بدون ارائه گواهی معتبر، اتصال رد شود.

- مقدار `cacert` همان CA یا مجموعه CAهایی است که گواهی Filebeat را تأیید می‌کند.

مهم: پوشه `/etc/logstash/certs/` باید حاوی `logstash.crt`, `logstash.key` (گواهی سرور) و `ca.crt` (CA) باشد؛ یا حداقل `cacert` => به مسیر درست فایل CA اشاره کند.

در نهایت، سرویس Logstash را ری‌استارت کنید:

```
sudo systemctl restart logstash
```

2. پیکربندی Filebeat (سمت کلاینت)

فایل کانفیگ `filebeat.yml`

کافی است در قسمت `output.logstash`، علاوه بر فعال کردن SSL، گواهی و کلید خصوصی Filebeat را هم معرفی کنید تا Filebeat در هنگام handshake، گواهی خودش را ارائه بدهد. به‌صورت نمونه:

```
filebeat.inputs:
  - type: log
    enabled: true
    paths:
      - /var/log/syslog
      - /var/log/auth.log

output.logstash:
  hosts: ["192.168.104.179:5044"]   # IP یا نام سرور Logstash
  ssl.enabled: true
  # CA ریشه جهت اعتبارسنجی گواهی سرور (Logstash)
  ssl.certificate_authorities: ["/etc/filebeat/certs/ca.crt"]

  # گواهی و کلید Filebeat (کلاینت):
  ssl.certificate: "/etc/filebeat/certs/filebeat.crt"
  ssl.key: "/etc/filebeat/certs/filebeat.key"
```

`ssl.certificate_authorities` باید به همان CA مرجعی اشاره کند که گواهی **Logstash** با آن امضا شده است.

- `ssl.certificate` و `ssl.key` گواهی و کلید خصوصی **Filebeat** هستند که در مرحله تولید گواهی برای `filebeat` Label ساخته‌اید.
- سطح دسترسی فایل‌ها باید طوری باشد که کاربر `filebeat` (یا هر کاربری که سرویس را اجرا می‌کند) قابلیت خواندن کلید خصوصی را داشته باشد.

در پایان سرویس Filebeat را ری‌استارت کنید:

sudo systemctl restart filebeat

حالا اگر در لاگ Filebeat یا لاگ Logstash پیغام خطای SSL مشاهده نکنید، یعنی ارتباط برقرار است. می‌توانید با ابزارهایی مانند `tcpdump` یا با بررسی Logstash برقراری session TLS موفق را رصد کنید. همچنین در لاگ Logstash خواهید دید که کلاینت با یک CN یا SAN خاص (مطابق گواهی Filebeat) متصل شده است.

در صورتی که از Logstash استفاده نمیکنید و مستقیم لاگ را به سوی الستیک میفرستید :

تنظیمات احراز هویت دوجانبه Filebeat → Elasticsearch

اگر به‌جای Logstash، مقصد Filebeat مستقیماً Elasticsearch باشد و بخواهید MTLS برقرار کنید، باید:

1. در **Elasticsearch** لایه HTTP به‌صورت TLS راه‌اندازی شده باشد.
2. **تنظیمات احراز هویت دوجانبه** در Elasticsearch فعال باشد:
   - در فایل `elasticsearch.yml`:

```
xpack.security.http.ssl.enabled: true
xpack.security.http.ssl.client_authentication: required
xpack.security.http.ssl.key: /etc/elasticsearch/certs/elasticsearch.key
xpack.security.http.ssl.certificate: /etc/elasticsearch/certs/elasticsearch.crt
xpack.security.http.ssl.certificate_authorities: [ "/etc/elasticsearch/certs/ca.crt" ]
```

مقدار `client_authentication: required` یعنی کلاینت هم موظف است گواهی ارائه دهد.

در **Filebeat**، مشابه بالا اما این بار در `output.elasticsearch`:

```
output.elasticsearch:
  hosts: ["https://192.168.104.175:9200"]
  username: "elastic"
  password: "YourPassword"

  ssl.certificate_authorities: ["/etc/filebeat/certs/ca.crt"]
  ssl.certificate: "/etc/filebeat/certs/filebeat.crt"
  ssl.key: "/etc/filebeat/certs/filebeat.key"
  # در صورت لزوم
  # ssl.verification_mode: full
```

حالا Elasticsearch انتظار دارد که کلاینت با یک گواهی معتبر (امضاشده توسط CA خودش) وصل شود و در غیر این صورت اجازه دسترسی را نخواهد داد.

نکات مهم در احراز هویت دوجانبه

1. **انتخاب CA واحد**

اطمینان حاصل کنید هم سرور و هم Filebeat از یک مرجع گواهی (یا CAهایی که در زنجیره اعتماد یکسانی باشند) استفاده می‌کنند. اگر Filebeat با CA متفاوتی امضا شده باشد، سرور (Logstash/Elasticsearch) گواهی Filebeat را قبول نخواهد کرد.

2. **نام‌ها و SAN**

هنگام ساخت گواهی IP، Filebeat یا hostname واقعی آن را در SAN (Subject Alternative Name) درج کنید تا mismatch رخ ندهد. همین‌طور برای سرور نیز اگر نام DNS استفاده می‌کنید، باید در SAN گواهی سرور باشد یا از IP استفاده کنید.

3. **رمز فایل‌های کلید**

اگر موقع ساخت گواهی، رمز (passphrase) روی `key.` یا `p12.` گذاشته باشید، باید در سرویس مربوطه (Logstash یا Elasticsearch یا Filebeat) هم پیکربندی کنید که رمز را بداند. در ساده‌ترین سناریوها، اغلب بدون رمز استفاده می‌شود.

4. **تنظیم سطح دسترسی فایل‌ها**

فایل‌های کلید (خصوصاً کلید خصوصی) نباید قابل خواندن توسط همه کاربران سیستم باشد. معمولاً سطح دسترسی 0400 یا 0600 و مالکیت کاربر سرویس مربوطه (elasticsearch, logstash, filebeat) کافی است.

5. اگر CA درست نباشد، پیام `unknown ca` یا `certificate signed by unknown authority` به چشم می‌خورد