HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA AN TOÀN THÔNG TIN

Báo cáo bài thực hành

Phát hiện và phản ứng sự cố SSH brute-force và persistence trái phép

Sinh viên thực hiện:

B20DCAT002 – Hoàng Thu Cúc

Giảng viên hướng dẫn: TS.Nguyễn Ngọc Điệp

HÀ NỘI 12-2025

MỤC LỤC

DANH MỤC CÁC HÌNH VẼ

NỘI DUNG THỰC HÀNH

1.1 Lab 1: Phát hiện và phản ứng sự cố SSH brute-force và persistence trái phép

1.1.1 Giới thiệu chung

Bài lab "Phát hiện và phản ứng sự cố tấn công SSH brute-force và persistence trái phép" được xây dựng giúp ta nhận ra 2 sự cố:

- brute-force SSH thất bại
- đăng nhập SSH thành công rồi thực hiện hành vi duy trì truy cập trái phép (persistence).

Bài thực hành hướng dẫn sinh viên :

- Cấu hình thu thập log để theo dõi sự kiện trên ELK
- Tạo Data View và alert trong Kibana
- Thực hiện phản ứng bằng fail2ban, và cuối cùng
- Xử lý hậu xâm nhập

1.1.2 Mục đích

Sinh viên biết cách

- triển khai hệ thống giám sát log với ELK
- theo dõi dấu hiệu xâm nhập
- thực hành thu thập bằng chứng, loại bỏ backdoor và khôi phục hệ thống.

1.1.3 Yêu cầu đối với sinh viên

- Hiểu về công cụ dò quét mạng phổ biến như hydra.
- Hiểu về bộ công cụ ELK (Elasticsearch, Logstash, Kibana)

1.1.4 Nội dung thực hành

- Trước khi khởi động bài lab, cần đảm bảo labtainer được cấu hình như sau:
    - Memory (RAM): 10GB
    - Hard Disk: Tối thiểu 80GB (khuyến nghị 100GB)
- Tải bài lab:

Vào /home/student/labtainer/labtainer-student và gõ lệnh sau trên terminal:

imodule https://github.com/anhdnmit/do_an_tot_nghiep/raw/refs/heads/main/idr_elk_sshbruteforce_new/imodule.tar

- Khởi động bài lab:

labtainer -r idr_elk_sshbruteforce

- Sau khi khởi động xong, ba terminal ảo sẽ xuất hiện: client, server, attacker
- Để kiểm tra địa chỉ IP của 3 máy, gõ lệnh sau trên từng máy:

ifconfig

1.1.4.1 Triển khai và cấu hình hệ thống

1.1.4.1.1 Trên máy server

Đầu tiên, ta phải bật cơ chế xác thực cho Elasticsearch và Kibana. Sinh viên mở file cấu hình /etc/elasticsearch/elasticsearch.yml để đảm bảo có dòng sau:

xpack.security.enabled: true

Tiếp đến sinh viên chạy lệnh sinh mật khẩu ngẫu nhiên cho các user của Elasticsearch:

sudo /usr/share/elasticsearch/bin/elasticsearch-setup-passwords auto

Ở Kibana, ta cần user riêng để kết nối Elasticsearch, chỉnh sửa ở file /etc/kibana/kibana.yml để đảm bảo có dòng sau:

elasticsearch.username: "kibana_system"

elasticsearch.password: "<PASS_KIBANA_SYSTEM>"

xpack.security.enabled: true

xpack.encryptedSavedObjects.encryptionKey:
"<PASS_ENCRYPTION_KEY>"

Đối với Logstash, ta hiểu luồng log đi từ Filebeat (trên client) → Logstash (trên server, port 5044) → Elasticsearch (trên server, port 9200). Do Logstash ghi vào Elasticsearch thì bắt buộc phải có user/password. Sinh viên tự chỉnh sửa file cấu hình /etc/logstash/conf.d/lab.conf theo mật khẩu đã sinh:

```
output {
  elasticsearch {
    hosts => ["http://127.0.0.1:9200"]
    user  => "elastic"
    password => "<PASS_ELASTIC>"
```

```
    index => "lab-%{+YYYY.MM.dd}"
  }
  stdout { codec => rubydebug }
}
```

Sau đó, khởi động lại 3 công cụ và kiểm tra các service đang lắng nghe trên port 5044, 9200, 5601:

sudo ss -lntp | egrep ':5044\b|:9200\b|:5601\b'

1.1.4.1.2 Trên máy client

Đã cài sẵn công cụ filebeat trên client nên chỉ cần khởi động và kiểm tra Filebeat có thiết lập kết nối thành công tới Logstash/Elasticsearch và sẵn sàng gửi sự kiện log hay chưa :

sudo filebeat test output -e

1.1.4.1.3 Tạo Data View trên Kibana

Trên server, mở trình duyệt và truy cập Kibana http://176.34.0.5:5601 . Trong Kibana, vào menu Stack Management để chọn Index Patterns. Đây là nơi cấu hình "Data View" để Kibana biết sẽ làm việc với những index nào trong Elasticsearch. Khai báo Data View cho index lab-*

- Name: lab
- Index pattern: lab-*
- Timestamp field: @timestamp

Sau đó chọn Create index pattern (hoặc Create data view).

Mở Discover để xem log. Nhấn vào biểu tượng 3 gạch (menu bên trái)

Chọn Discover. Ở góc trên, chọn Data View: lab. Khi đó, giao diện Discover sẽ hiển thị bảng log tăng dần theo thời gian (trục thời gian ở trên, bảng record ở dưới). Mỗi dòng tương ứng một event (ví dụ: log SSH, log từ Filebeat).

1.1.4.2 Phát hiện và phản ứng sự cố ssh brute force (Incident A)

Ở 2 máy client và attacker khởi động dịch vụ ssh để mô phỏng brute-force SSH thất bại. Trên attacker dùng hydra gửi hàng loạt mật khẩu sai.

hydra -l ubuntu -P /opt/wordlist.txt -V -t 8 -f -o hydra.out ssh://176.34.0.7

Lúc này, trên máy client sẽ ghi lại các sự kiện đăng nhập thất bại trong file ghi log hệ thống. Các dấu hiệu đặc trưng bao gồm chuỗi như "Failed password",

“authentication failure” hoặc “invalid user”. Đây là những dấu hiệu rõ ràng cho thấy có 1 quá trình tấn công brute-force đang xảy ra.

Khi các log thất bại được tạo ra trên client, Filebeat (đã được cài đặt sẵn trong máy client) sẽ tự động thu thập những log này và chuyển chúng đến server ELK thông qua Logstash. Tại đây, Logstash xử lý và đẩy dữ liệu vào Elasticsearch.

Giao diện quan sát Kibana sẽ nhận và hiển thị các log này thông qua Data View mà sinh viên đã cấu hình trước đó. Nhờ vậy, toàn bộ sự kiện brute-force được ghi nhận một cách trực quan, tập trung, và theo đúng dòng thời gian.

Sinh viên mở Kibana và truy cập mục **Discover**, nơi hiển thị toàn bộ dữ liệu log theo thời gian thực. Trong quá trình phân tích, sinh viên tìm kiếm các dấu hiệu đặc trưng của brute-force như:

- Các dòng log lặp lại “Failed password”
- Các thông báo “authentication failure”
- Các yêu cầu đăng nhập tới tài khoản không hợp lệ (“invalid user”)

Sinh viên tra ở ô KQL như sau để xem số lần xuất hiện dòng log “Failed password” lớn hơn 200 :

```
curl -sS -u elastic:<PASS_ELASTIC> \
  'http://127.0.0.1:9200/lab-*/_count?filter_path=count' \
  -H 'Content-Type: application/json' \
  -d '{
    "query":{"bool":{"filter":[
      {"term":{"host.name.keyword":"client"}},
      {"query_string":{"default_field":"message","query":"sshd
AND (Failed OR Invalid)"}},
      {"range":{"@timestamp":{"gte":"now-2h","lte":"now"}}}
    ]}}
  }' \
| jq -c '{count: .count}' | tee -a /home/ubuntu/evidence.json
```

Trên máy client, công cụ fail2ban và nftables cùng lúc để được cài rồi cấu hình để phản ứng tạm thời với sự cố.

Rồi cấu hình fail2ban như dưới:

```
sudo tee /etc/fail2ban/jail.local >/dev/null <<'CONF'
```

```
[sshd]
enabled   = true
port      = 22
filter    = sshd
logpath   = /var/log/auth.log
backend   = auto
maxretry  = 5
findtime  = 60
bantime   = 600
banaction = nftables-multiport
CONF
```

Khi fail2ban phát hiện đăng nhập thất bại nhiều, nó sẽ tự:

- Xác định địa chỉ IP đang gây ra brute-force
- Tạo một rule chặn tạm thời trong tường lửa (iptables)
- Ngăn chặn attacker tiếp tục thử mật khẩu

Sinh viên kiểm tra và xác nhận rằng fail2ban đã kích hoạt hành động này: một rule chặn mới xuất hiện, chứa địa chỉ IP của attacker.

sudo fail2ban-client status sshd

Điều này chứng minh rằng hệ thống đã thực hiện biện pháp containment tự động, cô lập nguồn tấn công và giảm thiểu rủi ro leo thang sự cố.

1.1.4.3 Phát hiện và phản ứng sự cố persistence trái phép (Incident B)

1.1.4.3.1 Khái quát về incident B

Incident B mô phỏng một kịch bản nâng cao hơn Incident A: attacker không chỉ brute-force thất bại mà lần này đăng nhập thành công vào máy client, sau đó thiết lập persistence (duy trì truy cập). Đây là tình huống phổ biến trong tấn công thực tế khi attacker đã vượt qua lớp bảo vệ ban đầu và bắt đầu thực hiện các hoạt động hậu xâm nhập (post-exploitation).

Sau khi Incident A kết thúc, IP của attacker đã bị chặn tự động bởi fail2ban.

Để mô phỏng một tình huống tấn công mới trong Incident B, sinh viên cần đảm bảo rằng cơ chế chặn tạm thời này không còn tác dụng. Khi fail2ban hoặc rule chặn bị vô hiệu hóa, attacker có thể thực hiện một phiên SSH mới mà không bị cản trở.

Khi fail2ban không còn chặn, attacker thử đăng nhập SSH bằng mật khẩu đúng.

Các sự kiện được ghi trong log như sau ở client:

- Thông báo xác thực thành công (“Accepted password”)
- Việc mở phiên làm việc SSH (“session opened”)

Những dòng log này thể hiện rằng attacker đã vượt qua giai đoạn brute-force thất bại và thực sự xâm nhập được vào hệ thống.

Ngay sau khi đăng nhập thành công, attacker thực hiện một trong các phương thức “bám trụ” để đảm bảo có thể quay lại hệ thống vào thời điểm khác mà không cần brute-force nữa.

Hai dạng persistence được mô phỏng trong bài lab:

- Thêm khóa SSH độc hại vào file authorized_keys của người dùng: Việc này cho phép attacker đăng nhập bằng key riêng, bỏ qua mật khẩu.
- Tạo một cron job nhẹ (one-time hoặc định kỳ): một script được cấu hình chạy khi hệ thống khởi động để tạo lại backdoor hoặc thực thi một hành động bất thường.

1.1.4.3.2 Trên attacker: thiết lập persistence

Trước khi thực hiện đăng nhập hợp lệ vào máy client, attacker tạo một cặp khóa SSH riêng để dùng làm “backdoor”.

ssh-keygen -t ed25519 -N '' -f ~/.ssh/lab_ed25519 -C 'attacker@lab'

- Một khóa riêng tư (private key) được lưu ở thư mục cá nhân của attacker. Đây là khóa bảo mật và attacker giữ lại để đăng nhập trái phép.
- Một khóa công khai (public key) tương ứng. Khóa này sẽ được chèn vào máy client nhằm cho phép attacker đăng nhập mà không cần mật khẩu.

Chuẩn bị xong cặp khóa SSH, attacker thực hiện đăng nhập vào máy client bằng mật khẩu hợp lệ để có thể thêm toàn bộ nội dung khóa công khai đã tạo ở bước trước vào file authorized_keys :

mkdir -p ~/.ssh && chmod 700 ~/.ssh

echo 'ssh-ed25519 <PUB_KEY> attacker@lab' >> ~/.ssh/authorized_keys

Từ một terminal khác, attacker thử kết nối vào client bằng khóa riêng tư đã tạo.

ssh -i ~/.ssh/lab_ed25519 ubuntu@176.34.0.7

Nếu đăng nhập thành công mà không cần mật khẩu, điều đó chứng minh backdoor đã được cài đặt đúng cách.

Sau khi có được phiên SSH hợp lệ và cài backdoor khóa SSH, attacker tiếp tục triển khai một cơ chế persistence thứ hai thông qua cron job đơn giản cho bài lab này.

```
JOB='* * * * * test -f "$HOME/.p_once" || { /usr/bin/logger -t persistence "persistence-once $(date +\%FT\%T) $(hostname) [cron]"; echo "persistence-once $(date +\%FT\%T) $(hostname) [cron]" >> /tmp/.once; touch "$HOME/.p_once"; }'
```

Attacker xây dựng một cron job với mục tiêu:

- Chạy mỗi phút.
- Kiểm tra xem một "file guard" đã tồn tại hay chưa. Nếu chưa thì sẽ :
    - Ghi một dòng sự kiện vào log hệ thống (thông qua logger) để đánh dấu hành vi persistence.
    - Tạo một bản ghi vào một tập tin tạm.
    - Tạo file guard để tránh chạy lại nhiều lần.

Cron job này đóng vai trò như một dấu hiệu rõ ràng của hành vi duy trì truy cập trên hệ thống.

Attacker nạp nhiệm vụ này vào crontab của người dùng.

```
( crontab -l 2>/dev/null; echo "$JOB" ) | crontab -
```

Khi đó, không làm gì cả thì cron vẫn sẽ cron job/phút.

1.1.4.3.3 Trên server: phát hiện sự cố

Để xác định liệu attacker có đăng nhập thành công vào SSH hay chưa, sinh viên mở mục Discover và tìm kiếm những chuỗi liên quan đến cơ chế xác thực SSH trên máy client. Các từ khóa cần dùng:

- sshd
- Accepted
- session opened

Khi lọc theo host.name = "client", sinh viên tìm các log thể hiện:

- "Accepted publickey": biểu thị đăng nhập thành công bằng khóa SSH
- "session opened for user …": một phiên SSH hợp lệ đã bắt đầu

Trên màn hình lệnh thì kiểm tra bằng câu lệnh sau:

```
curl -sS -u elastic:<PASS_ELASTIC> -H 'Content-Type: application/json' \
 'http://127.0.0.1:9200/lab-*/_count' -d '{
  "query": {"bool":{"must":[
```

```
   {"match_phrase":{"host.name":"client"}},
   {"match_phrase":{"message":"Accepted publickey"}},
   {"range":{"@timestamp":{"gte":"now-2h"}}}
  ]}}
}' | jq -c '{accepted_publickey: .count}' | tee -a /home/ubuntu/evidence.json
```

Persistence trong bài lab được mô phỏng bằng một tác vụ cron chạy định kỳ và tạo bản ghi nhận diện rõ ràng trong log.

Do đó, sinh viên cần tập trung phân tích hai dạng dấu hiệu:

(1) Log hoạt động CRON trên máy client. Trong Kibana, sinh viên tìm:

- Các bản ghi chứa CRON
- Các bản ghi chứa CMD (biểu thị lệnh cron được thực thi)
- Loại bỏ các bản ghi do run-parts tạo ra (vì là tác vụ hệ thống bình thường)

Những dòng còn lại giúp sinh viên phát hiện 1 cron job bất thường được chạy bởi người dùng mang dấu vết hành vi attacker tạo ra.

(2) Dấu hiệu đặc trưng "persistence-once". Trong bài lab, attacker cố tình chèn một chuỗi nhận diện rõ ràng trong log:

"persistence-once"

Khi tìm kiếm chuỗi này:

- Nếu log xuất hiện dòng chứa "persistence-once" → persistence đã thực thi.
- Nếu không xuất hiện → cron job chưa hoạt động hoặc chưa bị kích hoạt.

Phía server dùng truy vấn Elasticsearch để đếm chính xác số lượng bản ghi persistence.

```
curl -sS -u elastic:eJFGQt1v3k4LkCDjqMhB -H 'Content-Type: application/json' \
 'http://127.0.0.1:9200/lab-*/_count' -d '{
  "query": {"bool":{"must":[
   {"match_phrase":{"host.name":"client"}},
   {"match_phrase":{"message":"persistence-once"}},
   {"range":{"@timestamp":{"gte":"now-2h","lte":"now"}}}
  ]}}
}' | jq -c '{persistence_once: .count}' | tee -a /home/ubuntu/evidence.json
```

1.1.4.3.4 Trên client: thu thập chứng cứ và diệt bỏ dấu vết tấn công

Khi xác nhận rằng attacker đã đăng nhập thành công và thực thi persistence, bước tiếp theo là thu thập chứng cứ nhằm phục vụ cho việc phân tích và báo cáo. Việc thu thập chứng cứ được thực hiện trên cả client và server, mỗi nơi đảm nhiệm một vai trò khác nhau.

mkdir -p ~/ir-backup

cp -a ~/.ssh/authorized_keys ~/ir-backup/authorized_keys.bak.$(date +%s)

crontab -l > ~/ir-backup/crontab.bak.$(date +%s)

*sha256sum ~/ir-backup/**

Tiếp đến sinh viên cần phải loại bỏ cơ chế truy cập bí mật mà attacker đã cài đặt.

Kiểm tra file authorized_keys để xác định dòng chứa dấu hiệu backdoor (ví dụ: attacker@lab), rồi từ đó xoá dòng khóa SSH độc hại khỏi file.

sed -i '/attacker@lab/d' ~/.ssh/authorized_keys

Thiết lập lại quyền truy cập chuẩn cho file để SSH chấp nhận cấu hình mới, rồi lưu bản sao dự phòng trước khi thay đổi (để tham chiếu hoặc khôi phục nếu cần). File authorized_keys không còn chứa khóa backdoor và không còn cơ chế đăng nhập bằng private key của attacker.

Đối với việc loại bỏ hoàn toàn cơ chế tự động chạy mà attacker đã cài, sinh viên cần kiểm tra cron để tìm các tác vụ chứa nội dung “persistence-once” để loại bỏ dòng cron độc hại khỏi crontab của người dùng.

crontab -l | sed '/persistence-once/d' | crontab -

rm -f ~/.p_once /tmp/.once

Xóa các file guard được attacker tạo (thường ở thư mục cá nhân hoặc thư mục tạm) rồi mới khởi động lại dịch vụ cron để áp dụng cấu hình mới.

Cuối cùng, sinh viên cần đảm bảo rằng attacker không thể sử dụng lại con đường cũ để xâm nhập bằng cách đặt lại mật khẩu mới và mạnh cho tài khoản bị tấn công.

NEWPASS="<Mật khẩu mới>"

echo "New password will be: $NEWPASS"

echo "ubuntu:${NEWPASS}" | sudo chpasswd

Sinh viên đặt mật khẩu mạnh cho lần đặt mật khẩu mới này.

1.1.4.4 Theo dõi hậu sự cố

Sau khi xử lý sự cố ban đầu (Incident A – brute-force thất bại), sinh viên cần triển khai bước theo dõi hậu sự cố. Đây là nội dung quan trọng trong quy trình phản ứng sự cố vì cho phép phát hiện sớm khả năng tấn công lặp lại, kiểm tra tình trạng hệ thống sau khi khôi phục và tăng cường khả năng cảnh báo.

Trong giai đoạn này, cấu hình Logstash được mở rộng nhằm:

- Trích xuất chính xác hơn các trường liên quan đến SSH
- Chuẩn hóa dữ liệu để phục vụ cảnh báo
- Phân loại sự kiện theo đúng ngữ cảnh "authentication"
- Loại bỏ các tác vụ brute-force lặp lại khó quan sát bằng log thô

Vậy nên, sinh viên sửa /etc/logstash/conf.d/lab.conf để thêm filter (giữ nguyên input/output hiện có):

```
input {
  beats { port => 5044 }
}
filter {
  if "sshd" in [message] {
    grok {
      match => {
        "message" => [
          "Failed password for %{DATA:user.name} from %{IP:source.ip} port %{NUMBER} ssh2",
          "Invalid user %{DATA:user.name} from %{IP:source.ip}",
          "authentication failure%{DATA} rhost=%{IP:source.ip}%{GREEDYDATA}"
        ]
      }
      tag_on_failure => []
    }
    mutate { add_field => { "event.category" => "authentication" } }
  }
}
output {
  elasticsearch {
```

```
    hosts => ["http://127.0.0.1:9200"]
    index => "lab-%{+YYYY.MM.dd}"
  }
  # stdout { codec => rubydebug }  # tùy bật để debug
}
```

Rồi sau đó khởi động lại logstash, Log SSH trở nên dễ đọc, dễ lọc theo user, IP, loại sự kiện.

Sau đó, sinh viên tạo cảnh báo trong Kibana (Alerting) để phát hiện brute-force tái diễn. Vào tab Security → Rules → Create new rule → chọn Threshold (đừng chọn Query).

- mục Defination :
  - Index patterns: lab-* (đã có trước đó và bắt dùng)
  - KQL: host.name:"client" and message:(sshd and ("Failed password" or "Invalid user" or "authentication failure"))
  - Group by: source.ip.keyword
  - Threshold: is ≥ 1 within 1 minute (tùy đổi ngưỡng).
- About: Name: SSH brute-force failed (by IP); Severity: High (tùy).
- Schedule: Run every 1 minute, Look back 5 minutes.
- Actions: trong môi trường của bạn chỉ có “Index” → dùng luôn:
  - Chọn Index → tạo connector mới:
  - Connector name: SSH brute-force alerts
  - Index: ir_ssh_alerts
  - Index: lab-* (Kibana sẽ tự tạo).
  - Document to index:

{"@timestamp":"{{date}}","rule":"{{rule.name}}","results":"{{context.results }}"}

→ Save → Create & enable rule.

Ta kiểm tra với lệnh :

```
curl -sS -u elastic:eJFGQt1v3k4LkCDjqMhB \
'http://127.0.0.1:9200/.siem-signals-*/_search' \
-H 'Content-Type: application/json' -d '{
```

```
  "size": 0,
  "query": { "bool": { "filter": [
    { "term": { "signal.rule.name.keyword": "SSH brute-force failed (by IP)" } },
    { "term": { "signal.status": "open" } },
    { "range": { "@timestamp": { "gte": "now-1h", "lte": "now" } } }
  ]}},
  "aggs": { "ips": { "terms": { "field": "signal.threshold_result.terms.value.keyword", "size": 50 } } }
}'  | jq --arg time "$(date -Iseconds)" \
  '{rule:"SSH brute-force failed (by IP)",
    alerts:.hits.total.value,
    attackers:(.aggregations.ips.buckets|map(.key)),
    time:$time }' | tee -a /home/ubuntu/evidence.json
```

1.1.4.5 Kết thúc bài lab

Trên terminal đầu tiên sử dụng câu lệnh sau để kết thúc bài lab:

stoplab

Khi bài lab kết thúc, một tệp zip lưu kết quả được tạo và lưu vào một vị trí được hiển thị bên dưới stoplab.

Trong quá trình làm bài sinh viên cần thực hiện lại bài lab, dùng câu lệnh:

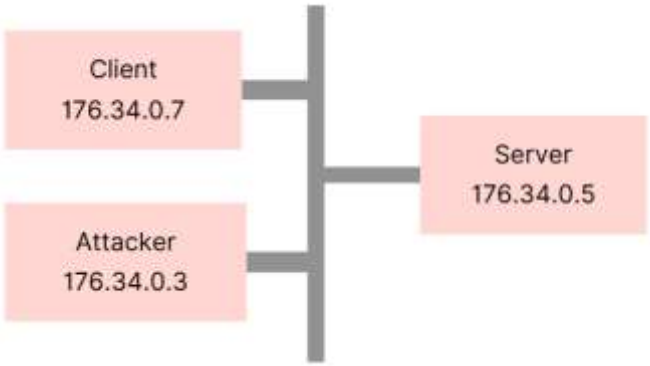
labtainer -r idr_elk_sshbruteforce

1.1.5 Thiết kế bài thực hành

1.1.5.1 Cấu hình docker

- Trong môi trường máy ảo Ubuntu, sử dụng docker tạo ra 3 container: : 1 container “client”, 1 container “server, 1 container “attacker”.
- Tạo mạng có cấu hình:
    - Subnet: 176.34.0.0/24
    - External Gateway: 176.34.0.1
- Cấu hình docker gồm có:
    - Server: Cấu hình cho máy server
        - Tên máy: server

        - Địa chỉ trong mạng LAN: 176.34.0.5
        - Gateway: 176.34.0.1
    - Client: Cấu hình cho máy client
        - Tên máy: client
        - Địa chỉ trong mạng LAN: 176.34.0.7
        - Gateway: 176.34.0.1
    - Attacker: Cấu hình cho máy tấn công 2 sự cố
        - Tên máy: attacker
        - Địa chỉ trong mạng LAN: 176.34.0.3
        - Gateway: 176.34.0.1



Hình 1 Topo mạng bài lab idr_elk_sshbruteforce

- config: lưu cấu hình hoạt động của hệ thống
- dockerfiles: mô tả cấu hình của các container, gồm:
    - server: sử dụng các thư viện mặc định hệ thống, cập nhật hệ thống và cài sẵn công cụ ELK.
    - client: sử dụng các thư viện mặc định hệ thống, cập nhật hệ thống và cài sẵn Filebeat và các file cấu hình.
    - attacker: sử dụng các thư viện mặc định hệ thống và cập nhật hệ thống, cài đặt thêm công cụ hydra.

1.1.5.2 Các nhiệm vụ cần thực hiện

- Đảm bảo hệ thống giám sát ELK trên server đang hoạt động và lắng nghe log.
- Đảm bảo log từ client được gửi tập trung về server để phân tích.
- Tạo đủ số lượng sự kiện tấn công để có dữ liệu log đáng kể cho việc phát hiện.
- Ghi nhận được chỉ số liên quan đến hành vi persistence để đánh giá mức độ hậu xâm nhập.

- Thể hiện cơ chế tự động chặn IP tấn công bằng hệ thống phòng thủ trên client.
- Thực hiện thao tác phân tích/chỉnh sửa dữ liệu liên quan attacker phục vụ điều tra.
- Thực hiện thao tác xóa bỏ thành phần persistence trong giai đoạn xử lý, làm sạch hệ thống.
- Ghi nhận hoặc đánh dấu thông tin liên quan đến tài khoản người vận hành trong quá trình xử lý.
- Đảm bảo hệ thống sinh ra đủ số lượng cảnh báo để minh họa việc giám sát liên tục.
- Tạo bản sao lưu log và kiểm tra tính toàn vẹn bằng mã băm để phục vụ điều tra, lưu trữ.

1.1.5.3 Các kết quả cần đạt được

Mỗi nhiệm vụ nhỏ sẽ được chia ra thành các mục chấm điểm để xác nhận sinh viên đã làm đúng các bước và hoàn thành bài thực hành hay chưa. Vì vậy, hệ thống sẽ ghi nhận các thao tác, sự kiện được mô tả theo bảng dưới đây để chấm điểm cho sinh viên:

Bảng 1. Bảng result bài idr_elk_sshbruteforce

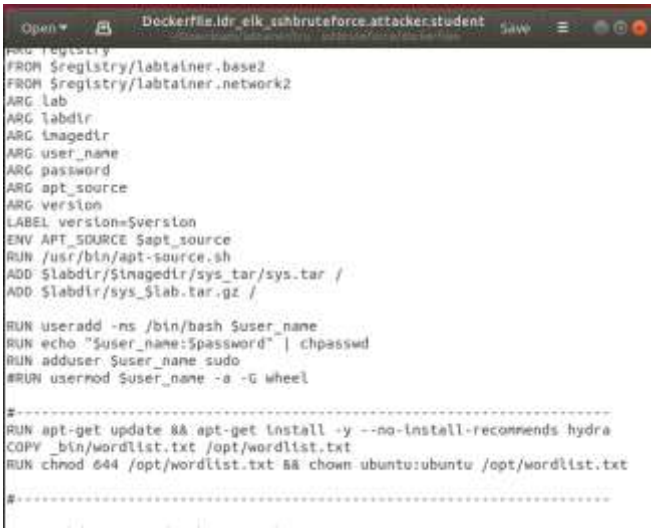
Result Tag	Container	File	File Type	Field ID	Timestamp Type
_server1	server	ss.stdout	FILE_REGEX	(?s)^(?=.*:5044\b)(?=.*:9200\b)(?=.*:5601\b).*\S	File
client1	client	filebeat.stdout	CONTAINS	talk to server	File
_detect1	server	evidence.json	FILE_REGEX	"count"\s*:\s*(1[1-9]|[2-9]\d{1,3})	File
_detect2	server	evidence.json	FILE_REGEX	"persistence_once"\s*:\s*([3-9]\d{0,3})	File
client2	client	fail2ban-client.stdout	CONTAINS	Banned IP	File
client3	client	sed.stdin	CONTAINS	attacker	File
client4	client	rm.stdin	CONTAINS	p_once	File
client5	client	echo.stdin	CONTAINS	ubuntu	File
_monitor	server	evidence.json	FILE_REGEX	"alert"\s*:\s*([2-9]|\d{2,})	File

client6	client	sha256sum.stdin	CONTAINS	backup	File

1.1.6 Cài đặt và cấu hình các máy ảo

Hình 2 Giao diện Labedit của bài lab

Hình 3 Result



Hình 4 Dockerfiles của máy attacker

Hình 5 Dockerfiles của máy client

Hình 6 Dockerfiles của máy server

1.1.7 Tích hợp và triển khai

1.1.7.1 Docker Hub

Hình 7 Bài thực hành được lưu trữ trên docker hub

1.1.7.2 Github

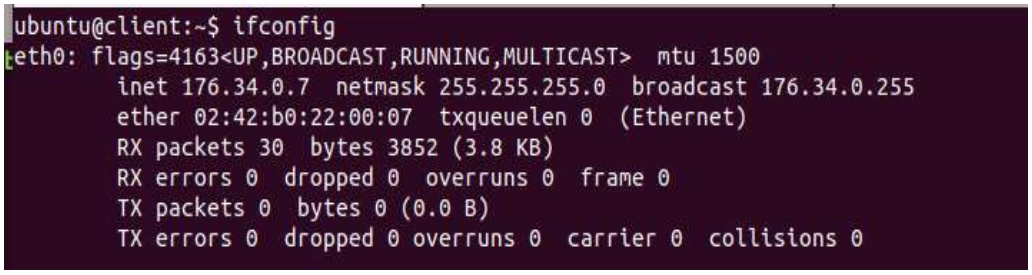
Link github:
https://github.com/anhdnmit/do_an_tot_nghiep/tree/main/idr_elk_sshbruteforce

Hình 8 Đẩy file imodule.tar lên github

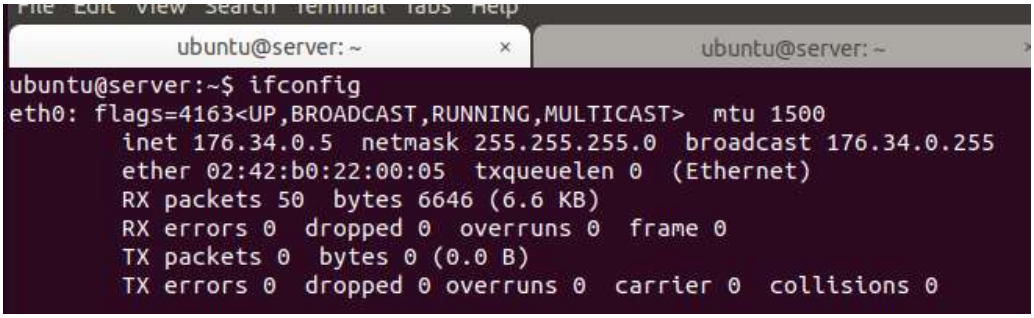
1.1.8 Thử nghiệm và đánh giá

Bài thực hành đã được xây dựng thành công. Bây giờ ta sẽ đánh giá về 1 trong các nhiệm vụ bài lab yêu cầu.

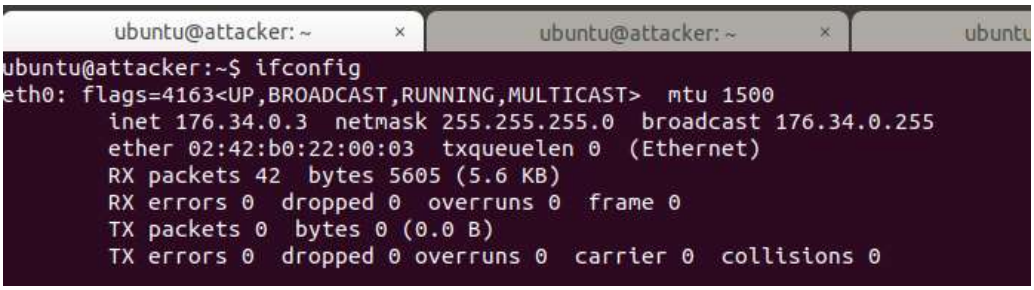
Trước hết, ta kiểm tra ip trên 3 máy.



Hình 9 IP client



Hình 10 IP server



Hình 11 IP attacker

Trong 10 nhiệm vụ bài lab đưa ra, ta đánh giá nhiệm vụ 1 làm nền tảng. Khi Elasticsearch, Kibana và Logstash đều được cấu hình đúng và mở cổng đầy đủ, các nhiệm vụ tiếp theo chỉ cần triển khai theo hướng dẫn chung đã đề ra.

Trước hết, ta chỉnh lại cấu hình của Elasticsearch để bật cơ chế bảo mật. Trên server, mở file cấu hình elasticsearch.yml lên để xem và bổ sung tham số cần thiết:

sudo nano /etc/elasticsearch/elasticsearch.yml

Trong file này, ta đảm bảo có dòng:

xpack.security.enabled: true

Sau khi lưu file, ta khởi động lại Elasticsearch để áp dụng cấu hình mới và kiểm tra trạng thái dịch vụ.

Tiếp đến, cần sinh mật khẩu cho các user nội bộ (trong đó quan trọng nhất là elastic và kibana_system). Ta dùng công cụ có sẵn của Elasticsearch để tạo mật khẩu ngẫu nhiên:

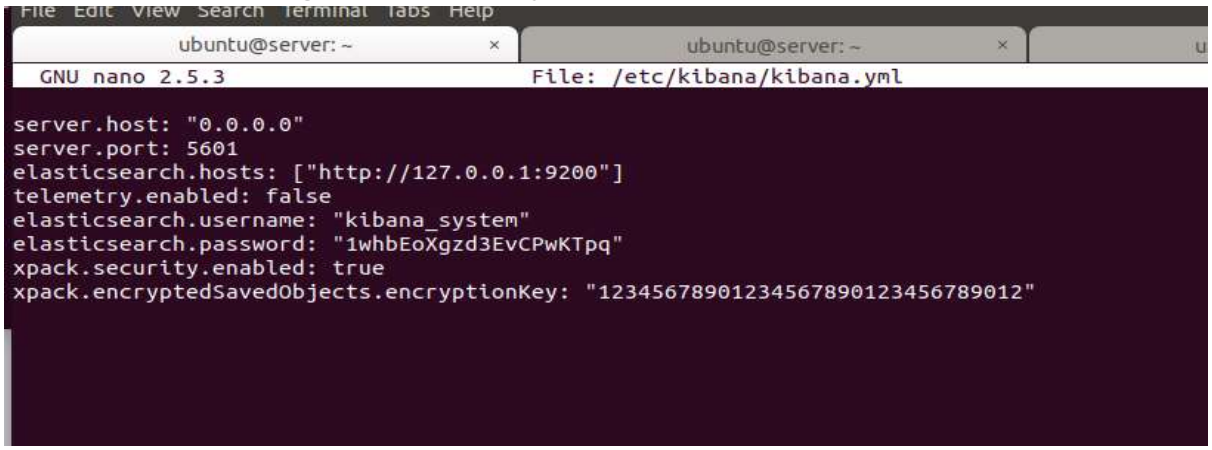
sudo /usr/share/elasticsearch/bin/elasticsearch-setup-passwords auto

Chương trình sẽ hỏi xác nhận, ta gõ y và nhấn Enter. Sau đó, màn hình sẽ in ra danh sách mật khẩu cho các user. Ở đây ta bắt buộc phải lưu lại hai mật khẩu của tài khoản elastic và kibana_system:

Hình 12 Danh sách tài khoản và mật khẩu

Sau đó, ta cấu hình Kibana để nó kết nối tới Elasticsearch bằng đúng tài khoản chuyên dụng kibana_system .

Trên server, mở file cấu hình Kibana, ta thêm các dòng sau để Kibana đăng nhập vào Elasticsearch bằng user kibana_system :



Hình 13 Chỉnh sửa file cấu hình kibana.yml

Sau khi chỉnh sửa xong, ta khởi động lại Kibana.

Khi Elasticsearch và Kibana đã sẵn sàng, ta chuyển sang Logstash. Dòng log trong bài lab sẽ đi theo luồng:

Filebeat trên client → Logstash trên server (cổng 5044) → Elasticsearch trên server (cổng 9200).

Filebeat gửi log tới Logstash mà không cần user/pass, nhưng khi Logstash ghi dữ liệu vào Elasticsearch thì phải dùng user elastic cùng mật khẩu tương ứng. Vì vậy ta kiểm tra và chỉnh lại file cấu hình của Logstash :

```
output {
  elasticsearch {
    hosts => ["http://127.0.0.1:9200"]
    user  => "elastic"
    password => "<PASS_ELASTIC>"
    index => "lab-%{+YYYY.MM.dd}"
  }
  stdout { codec => rubydebug }
}
```

Sau khi lưu file, ta khởi động lại Logstash.

Cuối cùng, để kết thúc Nhiệm vụ 1, ta xác nhận cả ba thành phần ELK đều đang lắng nghe trên đúng cổng:

```
ubuntu@server:~$ sudo ss -lnt 'sport = :9200 or sport = :5601 or sport = :5044'
State      Recv-Q Send-Q Local Address:Port               Peer Address:Port
LISTEN     0      4096         *:9200                     *:*
LISTEN     0      4096         *:5044                     *:*
LISTEN     0      511          *:5601                     *:*
ubuntu@server:~$
```

Hình 14 Kiểm tra trạng thái 3 cổng

Nếu thấy cả ba cổng đều có trạng thái LISTEN, ta có thể coi Nhiệm vụ 1 đã hoàn thành: hệ thống ELK đã được bảo vệ bằng cơ chế auth, dùng đúng tài khoản, và sẵn sàng nhận log từ các nhiệm vụ tiếp theo.

```
Results stored in directory: /home/ubuntu1/Labtainer_xfer/idr_elk_sshbruteforce
Labname idr_elk_sshbruteforce

Student          | server1 | server2 | server3 | server4 | client1 | client2 | client3 | client4 | client5 | client6 |
-----------------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+
b21dcat082       |    Y    |    Y    |    Y    |    Y    |    Y    |    Y    |    Y    |    Y    |    Y    |    Y    |
What is automatically assessed for this lab:
```

Hình 15 Checkwork bài idr_elk_sshbruteforce