Sprawozdanie z użycia i działania biblioteki Scapy (Python)

Bartosz Olszewski
17.03.2025

1 Scapy - narzędzie do analizy sieci

Scapy to potężne narzędzie do analizy, manipulacji i generowania pakietów sieciowych. Jest napisane w Pythonie i pozwala użytkownikom na przechwytywanie, modyfikowanie oraz wysyłanie pakietów w różnych protokołach sieciowych. Dzięki swojej elastyczności, Scapy jest szeroko wykorzystywane w testach penetracyjnych, diagnostyce sieci oraz edukacji.

Podstawowe funkcje Scapy

- Tworzenie i manipulacja pakietami sieciowymi.
- Przechwytywanie i analiza ruchu sieciowego.
- Wsparcie dla wielu protokołów (TCP, UDP, ICMP, ARP, DNS itp.).
- Integracja z innymi narzędziami sieciowymi.
- Możliwość rozszerzania funkcjonalności poprzez własne moduły.

2 Użycie Scapy

Podstawowe operacje na pakietach

Na początek przedstawiono przykład stworzenia pakietu IP z ustawionym polem TTL (Time To Live). Inicjalizacja pakietu odbywa się poprzez wywołanie:

```
a = IP(ttl=10)
a
# <IP ttl=10 |>
```

Na początku domyślne pole źródłowe (`src`) ustawione jest na adres 127.0.0.1:

```
a.src
# '127.0.0.1'
```

Można łatwo zmienić wartości pól, np. przypisać adres docelowy:

```
a.dst = "192.168.1.1"
a
# <IP ttl=10 dst=192.168.1.1 |>
```

Zmiana adresu źródłowego przebiega w sposób automatyczny:

```
a.src
# '192.168.8.14'
```

Aby zmienić adres źródłowy:

```
a.src = "192.168.1.2"
a
# <IP ttl=10 src=192.168.1.2 dst=192.168.1.1 |>
```

Usunięcie pola TTL jest równie proste:

```
del(a.ttl)
a
# <IP dst=192.168.1.1 |>
```

Przy odczycie pola TTL, Scapy przypisuje mu domyślną wartość, która wynosi 64:

```
a.ttl
# 64
```

3 Enkapsulacja protokołów i ich warstwowość w Scapy

W sieciach komputerowych dane są przesyłane w postaci pakietów zgodnych z modelem warstwowym. Każda warstwa dodaje własne nagłówki, enkapsulując dane z wyższej warstwy. Scapy pozwala na reprezentację tych warstw w prosty i czytelny sposób za pomocą operatora `/`, który umożliwia składanie różnych warstw protokołów.

3.1 Warstwowość protokołów i enkapsulacja

Każdy protokół sieciowy działa na określonej warstwie modelu OSI lub TCP/IP. Warstwy mogą być enkapsulowane w innych warstwach, co pozwala na budowanie struktury pakietów. Oto przykłady:

1. **Pojedyncza warstwa (IP):**

   ```
   IP()
   ```

   - **Wynik:** `<IP |>`
   - Tworzy nagłówek IP bez dodatkowych warstw.

2. **IP + TCP (protokół transportowy nad IP):**

   ```
   IP()/TCP()
   ```

   - **Wynik:** `<IP frag=0 proto=TCP |<TCP |»`
   - Warstwa TCP jest enkapsulowana w warstwę IP.

3. **Warstwa 2 (Ethernet) + Warstwa 3 (IP) + Warstwa 4 (TCP):**

   ```
   Ether()/IP()/TCP()
   ```

- **Wynik:** `<Ether type=0x800 |<IP frag=0 proto=TCP |<TCP |»`
- Ethernet dodaje nagłówek MAC, a następnie następują nagłówki IP i TCP.

4. **Pakiet z danymi (warstwa aplikacji HTTP):**

   ```
   IP()/TCP()/"GET / HTTP/1.0\r\n\r\n"
   ```

   - **Wynik:** `<IP frag=0 proto=TCP |<TCP |<Raw load='GET / HTTP/1.0˚˚' |»`
   - Warstwa aplikacji (surowe dane `GET / HTTP/1.0`) jest dodana na wierzchu stosu.

5. **Nietypowe połączenia (IP wewnątrz IP, czyli tunelowanie):**

   ```
   Ether()/IP()/IP()/UDP()
   ```

   - **Wynik:** `<Ether type=0x800 |<IP frag=0 proto=IP |<IP frag=0 proto=UDP |<UDP |»`
   - Wewnętrzna warstwa IP sugeruje tunelowanie IP przez IP.

6. **Zmiana domyślnego protokołu w nagłówku IP:**

   ```
   IP(proto=55)/TCP()
   ```

   - **Wynik:** `<IP frag=0 proto=55 |<TCP |»`
   - Wartość `proto=55` oznacza zmianę domyślnego protokołu transportowego.

3.2 Dlaczego enkapsulacja jest ważna?

- **Hierarchia warstw:** Każda warstwa ma swoje zadanie i dodaje własne informacje, np. adresy IP, porty, sumy kontrolne.
- **Bezpieczeństwo i tunelowanie:** Protokół IP może być enkapsulowany w innym IP, co umożliwia VPN-y i tunelowanie ruchu.
- **Elastyczność testowania w Scapy:** Możesz budować niestandardowe pakiety i testować różne scenariusze sieciowe.

Podsumowanie

- Scapy pozwala modelować stosy protokołów poprzez operator `/`, co ułatwia budowę i analizę pakietów.
- Możesz tworzyć różne kombinacje warstw (Ethernet, IP, TCP, UDP, ICMP itd.).
- Enkapsulacja protokołów jest kluczowa dla routingu, tunelowania i analizy sieci.
- Scapy daje możliwość eksperymentowania z pakietami, co jest przydatne w testach penetracyjnych i debugowaniu sieci.

4 Wysyłanie pakietów

Po zbudowaniu i analizie pakietów kolejnym krokiem jest ich wysłanie. Scapy udostępnia funkcje do wysyłki pakietów zarówno na warstwie 3 (używając `send()`) jak i na warstwie 2 (za pomocą `sendp()`).

Przykłady:

- **Wysyłanie na warstwie 3:**

```
1 send(IP(dst="1.2.3.4")/ICMP())
```

  Scapy wysyła pakiet ICMP i zwraca informację o wysłanych pakietach.

- **Wysyłanie na warstwie 2:**

```
1 sendp(Ether()/IP(dst="1.2.3.4", ttl=(1,4)), iface="eth1")
```

  Wysyłka pakietów na określonym interfejsie `eth1`.

- **Wysyłanie pakietów z pliku PCAP:**

```
1 sendp(rdpcap("/tmp/pcapfile"))
```

Opcjonalnie, funkcje `send()` i `sendp()` mogą zwracać listę wysłanych pakietów, jeżeli przekażemy parametr `return_packets=True`.

Multicast na warstwie 3: Scope Identifiers

Od wersji Scapy 2.6.0 możliwe jest precyzyjne określenie interfejsu, na którym mają być wysyłane pakiety multicast (IPv4) lub pakiety z adresami link-local (IPv6). Scapy, korzystając z tabeli routingu, domyślnie wybiera pierwszy interfejs, jednak dzięki tzw. identyfikatorom zakresu można wskazać konkretny interfejs. Przykłady:

```
conf.checkIPaddr = False  # Odpowiedź IP może być różna od żądanej

# Wysyłanie na interfejs 'eth0'
sr(IP(dst="224.0.0.1%eth0")/ICMP(), multi=True)
sr(IPv6(dst="ff02::1%eth0")/ICMPv6EchoRequest(), multi=True)
```

Adres `"224.0.0.1%eth0"` oznacza, że pakiet multicast zostanie wysłany z interfejsu o nazwie `eth0`. Alternatywnie, można użyć notacji z identyfikatorem interfejsu, np. `%15`.

5 Wyniki traceroute do google.com

Przykładowe wyniki:

```
Traceroute do 8.8.8.8
1: 10.138.0.1 (_gateway)
    Lokalizacja: Brak danych, Brak danych, Brak danych
    Operator: Brak danych
2: 195.136.81.241 (Nieznany)
    Lokalizacja: Gliwice, Silesia, PL
```

```
    Operator: AS35115 BESTGO.PL SP. Z O.O.
3: 88.220.36.153 (Nieznany)
    Lokalizacja: Warsaw, Mazovia, PL
    Operator: AS20804 Exatel S.A.
4: 88.220.196.247 (Nieznany)
    Lokalizacja: Rembertów, Mazovia, PL
    Operator: Brak danych
5: 209.85.168.100 (Nieznany)
    Lokalizacja: Warsaw, Mazovia, PL
    Operator: AS15169 Google LLC
6: 192.178.97.125 (Nieznany)
    Lokalizacja: Warsaw, Mazovia, PL
    Operator: AS15169 Google LLC
7: 108.170.234.247 (Nieznany)
    Lokalizacja: Warsaw, Mazovia, PL
    Operator: AS15169 Google LLC
8: 8.8.8.8 (dns.google)
    Lokalizacja: Mountain View, California, US
    Operator: AS15169 Google LLC
Osiągnięto cel!
```

Analiza wyników:

1. **1: 10.138.0.1 (_gateway)**
   Co to jest? To adres prywatny, typowo przypisywany do routera w Twojej sieci lokalnej (brama domyślna).
   Lokalizacja i operator: Brak danych – ponieważ jest to adres wewnętrzny, dane geolokalizacyjne nie są dostępne.

2. **2: 195.136.81.241 (Nieznany)**
   Lokalizacja: Gliwice, Silesia, PL.
   Operator: AS35115 BESTGO.PL SP. Z O.O.
   Interpretacja: Pierwszy przeskok poza siecią lokalną, kierowany przez dostawcę usług internetowych w Polsce.

3. **3: 88.220.36.153 (Nieznany)**
   Lokalizacja: Warsaw, Mazovia, PL.
   Operator: AS20804 Exatel S.A.
   Interpretacja: Kolejny węzeł w Polsce, już wewnątrz infrastruktury operatora.

4. **4: 88.220.196.247 (Nieznany)**
   Lokalizacja: Rembertów, Mazovia, PL.
   Operator: Brak danych.
   Interpretacja: Węzeł w obszarze Warszawy, gdzie dane o operatorze nie zostały pobrane.

5. **5: 209.85.168.100 (Nieznany)**
   Lokalizacja: Warsaw, Mazovia, PL.
   Operator: AS15169 Google LLC.
   Interpretacja: Przejście ruchu do infrastruktury Google.

6. **6: 192.178.97.125 (Nieznany)**
   Lokalizacja: Warsaw, Mazovia, PL.

Operator: AS15169 Google LLC.
Interpretacja: Kolejny węzeł wewnątrz infrastruktury Google.

7. **7: 108.170.234.247 (Nieznany)**
Lokalizacja: Warsaw, Mazovia, PL.
Operator: AS15169 Google LLC.
Interpretacja: Węzeł w sieci Google, przed opuszczeniem kraju.

8. **8: 8.8.8.8 (dns.google)**
Lokalizacja: Mountain View, California, US.
Operator: AS15169 Google LLC.
Interpretacja: Docelowy serwer DNS Google, znajdujący się w USA.

Podsumowanie wyników:

- **Początek trasy:** Pakiety zaczynają się od sieci lokalnej (adres prywatny gateway), gdzie dane geolokalizacyjne nie są dostępne.
- **Etap krajowy:** Pakiety przemieszczają się przez węzły w Polsce, przechodząc od lokalnego dostawcy do infrastruktury operatora (Exatel), a następnie do sieci Google.
- **Przejście do globalnej sieci:** Węzły zarządzane przez Google wskazują, że ruch został przekazany do globalnej infrastruktury.
- **Cel końcowy:** Docelowy serwer (8.8.8.8) znajduje się w USA, co potwierdza geolokalizacja.

6 Komendy związane ze sniffingiem

SNIFF:

```
sniff(filter="icmp", count=5)
```

Przy jednoczesnym uruchomieniu w innym terminalu polecenia:

```
ping 8.8.8.8
```

Aby wyświetlić przechwycone pakiety:

```
a = _
a.nsummary()
a[1]
```

Pseudo-WIRESHARK:

```
sniff(iface="wlp1s0", prn=lambda x: x.summary())
```

Traceroute:

```
traceroute(["www.yahoo.com", "www.altavista.com", "www.wisenut.com", "www.copernic.com"]
```