به نام خدا

تمرین سری دوم درس امنیت شبکه پیشرفته

مجتبی موذن  شماره دانشجویی :۹۹۲۱۰۶۲۳

---

**بخش نظری :**

**تمرین ۱ - قسمت ۱ :**

این مقاله راه حل حمله ای را برای تحلیل ترافیک ورودی شبکه ی تر بررسی میکند . در این مقاله عنوان شده است که به منظور تحلیل سایت هایی که کاربران مورد بازدید قرار میدهند راه حلی را یافته اند که با استفاده از شبکه های عصبی عمیق کار میکند . هدف حمله این است که در دو جهان بسته ( یعنی سایت هایی که کاربر بازدید میکند محدود است ) و در جهان باز ( سایت هایی که کاربر بازدید میکند محدود نیست ) ترافیک کاربر را در حین انتقال بین کاربر و شبکه ی تر مورد بررسی قرار دهند و بفهمند که کاربر چه سایت هایی را بازدید کرده است . حمله کننده مابین شبکه ی تر و کاربر قرار دارد یعنی میتواند در ISP ( تامین کننده ی اینترنت) و یا gateway کاربر که میتواند مودم او باشد حضور داشته باشد . این حمله میتواند به صورت اسکریپت در یکی از نقاط تعیین شده بین کاربر و ورودی شبکه ی تر اجرا شود و ترافیک

رمز شده ی کاربر را مورد بررسی قرار دهد . طبق راه حل موجود در مقاله از یک شبکه ی عصبی عمیق که شامل دو بخش " استخراج ویژگی " و دیگری " کلاس بندی" استفاده شده است . در بخش استخراج ویژگی با استفاده از ورودی های شبکه ویژگی سایت های مشابه پس از چندین دور تکرار این مرحله استخراج شده و در ورودی لایه ی کلاس بندی قرار میگیرد . در لایه ی کلاس بندی با توجه به ویژگی های استخراج شده تفکیک سایت ها انجام می گیرد . ورودی های شبکه در لایه ی ابتدایی طبق گفته ی مقاله در ابتدا یک وکتور از ‹timestamp, ±packet size› تشکیل شده است . که یکی زمان کپچر کردن پکت را نشان می دهد و دیگری سایز پکت را نشان میدهد . مثبت و منفی بودن سایز پکت جهت خروجی یا ورودی را نشان میدهد ( خروجی یا ورودی به کاربر ) . ولی با توجه به مقاله فقط از اعداد بین +۱ تا -۱ برای جهت به عنوان ورودی استفاده شده است ( علت این موضوع این طور بیان شده است که استفاده از سایز بسته تاثیری در روند نداشته است و (احتمالا از نظر من ) باعث پردازش بیشتر ورودی شبکه شده است بنابراین مثل کار های قبلی شده در مقاله فقط از یک وکتور -1 , +1 به عنوان ورودی های شبکه استفاده شده است. خروجی این شبکه ی عمیق شامل ضرایب هر کدام از ویژگی های استخراج شده در بخش ویژگی های استخراج شده است که با استفاده یک لایه ی اضافه به عنوان output prediction برای تشخیص سایت ها استفاده میشود . در واقع در این شبکه های عمیق ما یک بخش به عنوان یادگیری داریم و خروجی ضرایب ویژگی هر سایت را به یک شبکه ی تک لایه میدهیم که بتواند تشخیص دهد کدام سایت بازدید شده است و پکت مربوط به ورودی مربوط به کدام سایت است. ( با توجه به پکت ورودی و ضرایب استخراج شده در لایه های قبلی این موضوع میتواند حل شود )

**تمرین ۱ - قسمت ۲ :**

طبق این مقاله در یک محیط باز یعنی محیطی که انتظار بازدید هر سایتی را توسط کاربر داریم این روش حمله بدین شکل کار میکند که یک مجموعه ی محدود از سایت های اصطلاحا monitored را نگهداری

میکند و در قسمت تشخیص و کلاس بندی میتواند با یک محدوده ی رنج محدود بگوید که آیا این سایتی که توسط کاربر بازدید شده به این مجموعه ی monitored اختصاص دارد یا خیر در غیر این صورت خروجی برای یک سایت unmonitored خواهد بود . پس در واقع سناریوی جهان باز در این مقاله یک جهان بسته هست که یک مجموعه سایت unmonitored نیز به آن اضافه شده است . این خود یک مشکل اساسی است زیرا باید پیش از یادگیری توسط یادگیری عمیق مجموعه ای از داده های سایت هایی که کاربر ممکن است به آن مراجعه کند را در اختیار داشته باشیم ( تشکیل مجموعه ی monitored ) انجام این کار کمی سخت است و نیاز به اطلاعاتی از جانب کاربر دارد . در واقع در این روش ما برای هر کاربر به طور خاص باید یک پروفایل از سایت هایی که ممکن است آن ها را بازدید کند تحت عنوان مجموعه ی داده های monitored جمع آوری کنیم و جمع آوری این مجموعه نیازمند شناخت کاربر و حدسیات میباشد ( کما این که این شناخت هم یک چیزی نسبی است و نمیتوان به طور صد در صد فهمید کاربر دقیقا چه سایت هایی را بازدید میکند ) در کنار این موضوع اگر بخواهیم این مشکل را با افزایش مجموعه ی monitored حل کنیم خود یک مشکل پردازشی به علت تعدد گره های شبکه در لایه های یادگیری می باشد و باعث کندی در یادگیری میشود .

مشکل بعدی جمع آوری داده از کاربر است . جمع آوری داده هایی از یک کاربر خاص برای یادگیری نیاز به پردازش چندین کامپیوتر به مدت چندین روز میباشد که ممکن است در این فاصله ی زمانی کاربر نیز تغییر کند بنابراین رفتار کاربر نیز عوض میشود ( در سایت هایی که آن ها را بازدید کرده ) . بنابراین جمع آوری داده های یادگیری از یک کاربر علاوه بر زمان زیاد نیاز به مطمئن بودن از عدم تغییر کاربر دارد . علاوه بر این برای مقابله با این حمله میتوان از روش های یادگیری ماشین خصمانه استفاده کرد . علاوه بر آن این نوع از حمله در مقابل روش های دفاعی واکی تاکی کم کارکرد است و موجب از بین رفتن تاثیر حمله تا حدی زیاد و افزایش زیاد فراخوان میشود که موجب میشود تعداد زیادی از سایت هایی که کاربر بازدید می کند نتواند مورد بررسی قرار گیرد .

در مورد دقت این روش و این که آیا دقت این روش در محیط های واقعی نیز قابل قبول است نیز به نظر من نه این دقت در محیط های واقعی کمتر از مقداری است که در این مقاله گفته شده است. یکی از مشکلات این روش این است که تشکیل داده های سایت های monitored کاری سخت است و اگر کاربر از آن سایت ها بازدیدی نداشته باشد یا رفتار خود را تغییر داده باشد این دقت باعث میشود حجم زیادی از داده های مهاجم unmonitored لقب بگیرند . که این یعنی بسیاری از سایت ها جزو داده های monitored نبوده و عملا رفتار کاربر ناقض حدس زده میشود و دقت پایین می آید . در یک محیط واقعی کاربر ممکن است از میلیون ها سایت بازدید داشته باشد و مراجعه ی مجدد به آن ها مدت ها طول کشید به همین علت دریافت داده های unmonitored نیاز به زمان زیادی برای یادگیری دارند و عملا دقت تشخیص در یک محیط واقعی با یک کاربر واقعی به شدت کاهش می یابد .

---

**سوال ۲ - قسمت ۱ :**

به طور کلی -D یک روش برای پنهان کردن ip شما در هنگام اسکن میباشد . این ویژگی که چندین آدرس ip در ورودی میگیرد اینطور عمل میکند که چندین سیستم مهاجم بی گناه را ( که از نظر سامانه ی IDS مقصد تعیین بی گناه بودن آن ها دشوار است ) طوری پوشش میدهد که هر کدام یک پورت را اسکن کنند و سامانه ی IDS مقصد نمی تواند تشخیص دهد که کدام یک واقعا قصد اسکن داشته و کدام یک بی گناه است . این روش یکی از روش های پنهان سازی سامانه ایست که واقعا در حال اسکن پورت هاست . برای مثال دستور زیر اینطور عمل میکند :

```
nmap -n -D 192.168.1.5,10.5.1.2,172.1.2.4,3.4.2.1 192.168.1.5
```

در این دستور 192.168.1.5 مورد هجوم قرار میگیرد و IDS چهار پورت اسکن را گزارش می دهد ولی نمیتواند تعیین کند کدام یک واقعی و کدام یک بی گناه است . یکی از راه های جلوگیری از این نوع پورت اسکن استفاده از مسیریابی بسته هاست .

**قسمت ۲ :**

sI یک روش اسکن پیشرفته برای اسکن پورت های TCP است که به صورت کور انجام میشود . یعنی هیچ بسته ای از سمت مهاجم به سمت مقصد ارسال نمیشود . این روش اسکن از یک حمله ی هوشمندانه ی کانال جانبی بهره میبرد که بدون نیاز به فرستادن هیچ پیامی به سمت سیستم هدف پورت را اسکن کند و اطلاعات مربوط به پورت های باز سیستم هدف را برگرداند . مشکلات روش های اسکن تقریبا همگی شناسایی سیستم مهاجم در سمت IDS است که این خوب نیست ولی با این روش درصد شناسایی به شدت کاهش می یابد. در این روش اسکن به صورت غیر مستقیم به وسیله ی یک سیستم قربانی که zombie نامیده می شود انجام میشود . معمولا این سیستم ها قربانی طوری انتخاب می شوند که مورد اعتماد سیستم هدف باشند .

نحوه ی کارکرد :

هر بسته ی ip دارای یک Fragment Identification Number میباشد که مهاجم در سیستم زامبی با استفاده از آن و فرستادن یک بسته ی syn به قربانی ( هدف اسکن) متوجه زیاد شدن این مقدار ID میشود . پس از این که زامبی syn را ارسال کرد اگر پورت مورد اسکن باز باشد ID دوتا اضافه میشود ( یعنی قربانی پاسخ مثبت داده و منتظر وصل شدن ارتباط است ) و اگر ID یکی اضافه شود قربانی RST ارسال کرده (در این مواقع یعنی پورت مقصد بسته است که قربانی RST ارسال کرده است ) پس پورت مورد نظر اسکن شده و بسته است . کار مهاجم ( اسکنر ) این است که در سیستم زامبی به دنبال ID بسته های IP

بگردد و قدیمی و جدید را با هم مقایسه کند . علاوه بر این از این نوع حمله میتوان استفاده کرد تا سیستم زامبی را به عنوان مهاجم به سامانه ی تشخیص نفوذ معرفی کرد .

از مشکلات این روش این است که باید سیستم زامبی وجود داشته باشد که بیکار باشد ( زیرا پیدا کردن توالی ID بسته های IP در یک سیستم مشغول سخت است و زمان اسکن را افزایش میدهد) علاوه بر آن زامبی باید تا حد امکان به شبکه ی سیستم هدف نزدیک باشد تا تاخیر زیاد نشود بنابراین نمی توان از هر سیستمی برای زامبی استفاده کرد . یکی از راه های پیدا کردن زامبی مناسب استفاده از اسکریپت مورد اشاره در سوال است . این اسکریپت میتواند توالی شناسه ی بسته های IP میزبان را طبقه بندی کند . تا با استفاده از آن بتوانیم زامبی خوب را پیدا کنیم . این اسکریپت میتواند به صورت موازی علیه میزبان های زیادی اجرا شود تا زودتر به یک میزبانی زامبی دست پیدا کنیم . خروجی اسکریپت روشی است که میزبان از آن برای تولید ID در بسته های IP استفاده میکند مثلا میتواند تصادفی یا ضربی باشد .

**قسمت ۳ :**

هر کدام از اسکریپت های همراه به یک دسته از طبقه بندی تعلق دارند . این طبقه بندی بر اساس عملکرد اسکریپت ها را درون خود جا میدهند . هر اسکریپت متعلق به یک یا بیش از چند دسته بندی است . دسته بندی های اسکریپت های nmap در ۱۴ دسته ی **auth, broadcast, brute, default. discovery, dos, exploit, external, fuzzer, intrusive, malware, safe, version, vuln** میشود . هر کدام از این دسته بندی ها شامل تعدادی از اسکریپت ها میباشد که اهداف مشابهی را دنبال میکنند به همین علت در یک دسته قرار گرفتند.

**قسمت ۴ :**

این اسکریپت که در دسته ی ***brute, intrusive*** قرار میگیرد برای حسابرسی رمز عبور در سرور های SMTP ایمیل استفاده میشود . از آن جایی که این سرور ها اطلاعات حساسی را درون خود جای داده اند برای حسابرسی به آن ها از حملات جامع Brute force بر روی آن ها استفاده میشود این دستور یک حمله ی جامع dictionary را علیه سرور مورد نظر آغاز میکند تا رمز عبور های ضعیف را کشف کند .

نحوه ی کارکرد آن بدین شکل است که از dictionary attack که یک حمله ی آزمون جامع است برای حمله استفاده میکند در این حمله کلماتی که بیشتر استفاده میشود و ساده هستند معمولا استفاده میشود . در این حمله از روش های مختلف متن خام یا Digest-md5 و یا NTLM و CRAM-MD5 و همچنین LOGIN برای احراز هویت استفاده میشود. همچنین این اسکریپت به صورت پیشفرض از فایل های nselib/data/usernames.lst/ و nselib/data/passwords.lst برای ورودی ها استفاده میکند که البته میتوان این فایل ها را هم تغییر داد .

برای تشخیص این نوع از اسکن در شبکه به این شکل عمل میکنیم که پورت مربوط به SMTP را حسابرسی می کنیم و تعداد تلاش های ناموفق هر آدرس را در یک محدوده بررسی کرده و اگر مقدار تلاش ها از یک حد آستانه بیشتر شد آدرس مورد نظر مشکوک و آن را میبندیم بدین ترتیب از اجرای این اسکن روی سرور خود جلوگیری میکنیم .

**قسمت ۵ :**

رکورد های یک سرور DNS اطلاعات مفید و جالبی را میتوانند نشت دهند . این اسکریپت برای حمله ی آزمون جامع بر روی DNS انجام میشود . برای مثال در رکورد DNS میتوان با کلید واژه ی MAIL در پسوند آدرس فهمید که رکورد متعلق به یک سرور SMTP میباشد و یا در سرور های زیر ساخت Cloudflare میتوان فهمید که کدام آدرس ها توسط آن محافظت می شود . نحوه ی عملکرد آن به این شکل است که سعی میکند در داخل لیست hostname جست و جو را انجام داده و در صورتی که یک ورودی معتبر باشد آن را نشان میدهد . این اسکریپت در دسته ی ***intrusive***, ***discovery*** قرار میگیرد . این حمله برای مهاجمان در واقع یک پیش درآمد برای حمله های دیگر است . مهاجم از طریق این حمله میتواند یک دامنه و تمام زیر دامنه های آن را به دست آورد و پس از آن بتواند برای حملات دیگری از آنها استفاده کند . از جمله می تواند ساختار دامین ها را شناسایی و نقاط ضعف سیستم را بهتر بدست بیاورد . این حمله علاوه بر آن می تواند اطلاعاتی را راجع به نوع سروری که نام دامنه متعلق به آن است نشت دهد .

**قسمت۶ :**

dns-nsec-enum اسکریپتی است که برای جست و جو و مکان یابی تمامی سرور ها و اطلاعات آن ها استفاده میشود که از سرویس dns استفاده می کنند ولی در dns-brute ما رکورد های dns جست و جو و مورد حمله ی آزمون جامع قرار میدهیم . در dns-nsec-enum با استفاده از پروتکل nsec تمام zone های یک dns و زیر دامنه های آن استخراج میشود در حالی که در dns-brute رکورد های dns استخراج شده است .

**قسمت ۷ :**

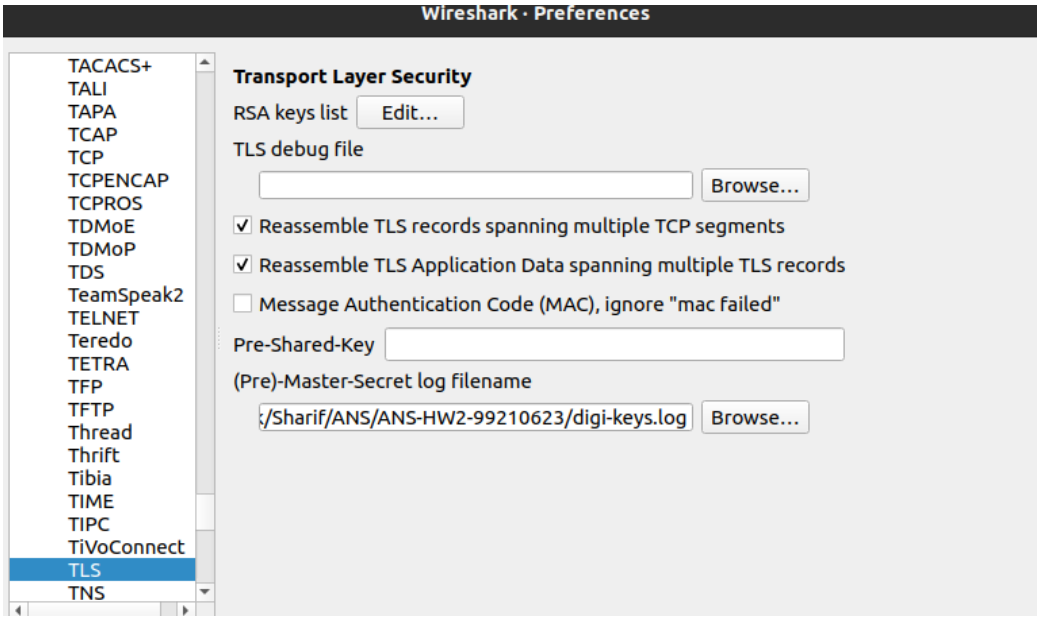
همانطور که در قسمت قبل دیدیم با مکانیزم nsec میتوان مناطق ( zones ) را پیمود و رکورد های dns را استخراج کرد . این مشکل باعث شد تا nsec3 به وجود بیاید که در مثال این حمله مقاوم است . nsec ساده تر از nsec3 است زیرا در nsec3 از پروتکل های رمزگذاری و هش استفاده شده است و این باعث سختی پردازش شده است . هر دوی این پروتکل ها برای احراز اصالت و شناسایی پاسخ های یک سرور dns استفاده می شود ولی nsec مشکل پیمایش مناطق را به همراه دارد . nssec3 از توابع یک طرفه ی hash استفاده میکند در این حالت در پاسخ به یک درخواست به جای نام اصلی که در nssec از آن استفاده میشد از توابع یک طرفه استفاده میشود و در پاسخ hash یک نام برای جست و جو کننده فرستاده میشود .

---

## بخش عملی

### سوال ۱ ) تحلیل ترافیک

برای حل این سوال ابتدا فایل ترافیک را با استفاده از منوی file -> open باز میکنیم و پس از آن باید فایل کلید ها را در قسمت edit-> preferences و در قسمت protocols مطابق شکل پایین در بخش TLS ( چون ترافیک رمز شده بر روی TLS انجام شده است ) بارگذاری کنیم . شکل پایین نشان میدهد که فایل کلید ها را به عنوان کلید اصلی به این پروتکل جهت رمزگشایی معرفی میکنیم .




پس از تایید این قسمت پروتکل TLS با کلید ها رمز گشایی میشود و پیام هایی دارای پروتکل HTTP2 رمزگشایی شده نمایش داده میشوند .

| | | | | | |
|---|---|---|---|---|---|
| 5848 37.290984 | 92.114.19.26 | 192.168.1.3 | TLSv1.3 | 308 | Server Hello, Change Cipher Spec, Encrypted Extensions, Finis… |
| 5849 37.291550 | 192.168.1.3 | 92.114.19.26 | TLSv1.3 | 134 | Change Cipher Spec, Finished |
| 5850 37.291755 | 192.168.1.3 | 92.114.19.26 | HTTP2 | 146 | Magic, SETTINGS[0], WINDOW_UPDATE[0] |
| 5851 37.292965 | 192.168.1.3 | 92.114.19.26 | HTTP2 | 1211 | HEADERS[1]: GET /static/files/1fdcadec.woff2 |
| 5852 37.293029 | 192.168.1.3 | 92.114.19.26 | HTTP2 | 158 | HEADERS[3]: GET /static/files/fe12f13a.woff2 |
| 5853 37.293072 | 192.168.1.3 | 92.114.19.26 | TCP | 159 | 5390 → 443 [PSH, ACK] Seq=2046 Ack=255 Win=131584 Len=105 |
| 5854 37.293114 | 192.168.1.3 | 92.114.19.26 | HTTP2 | 158 | HEADERS[7]: GET /static/files/b8d3de59.woff2 |
| 5855 37.293164 | 192.168.1.3 | 92.114.19.26 | HTTP2 | 211 | HEADERS[9]: GET /static/files/bc60cf05.svg |
| 5856 37.293205 | 192.168.1.3 | 92.114.19.26 | HTTP2 | 157 | HEADERS[11]: GET /static/files/7c53fd0f.svg |
| 5857 37.293249 | 192.168.1.3 | 92.114.19.26 | HTTP2 | 185 | HEADERS[13]: GET /static/files/1c93eb76.svg |
| 5858 37.293294 | 192.168.1.3 | 92.114.19.26 | HTTP2 | 156 | HEADERS[15]: GET /static/files/9252b9fc.png |
| 5859 37.293338 | 192.168.1.3 | 92.114.19.26 | HTTP2 | 156 | HEADERS[17]: GET /static/files/cbaed462.png |
| 5860 37.293385 | 192.168.1.3 | 92.114.19.26 | HTTP2 | 179 | HEADERS[19]: GET /static/merged/0df728a6.css |
| 5861 37.293429 | 192.168.1.3 | 92.114.19.26 | HTTP2 | 157 | HEADERS[21]: GET /static/merged/0d0f06be.css |

سپس در این قسمت برای پیدا کردن محصولاتی که کاربر از آن ها بازدید کرده باید با استفاده از ابزار اسکریپت wireshark تعداد جست و جو ها را کاهش دهیم به همین علت در قسمت filter فیلتر زیر را اعمال می کنیم . لازم به ذکر است با پیدا کردن یکی از بسته ها متوجه آدرس سرور و آدرس کاربری که اطلاعات را مشاهده کرده است شدیم و بنابراین در فیلتر جست و جو این آدرس را اعمال کردیم . آدرس کاربر و سرور به شکل زیر است :

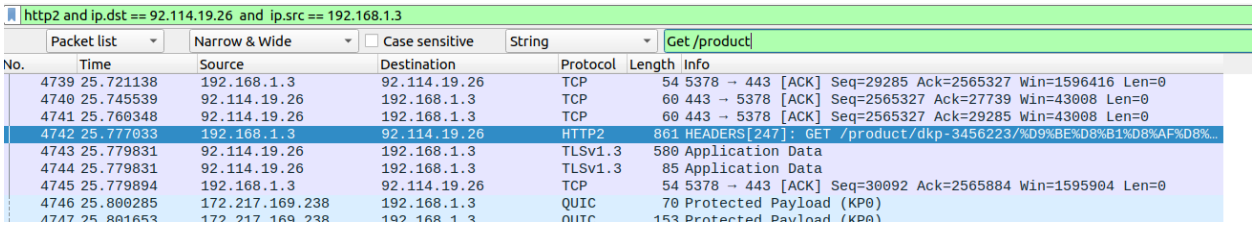
```
dst ip = 92.114.19.26
src ip = 192.168.1.3
```

سپس فیلتر زیر را در قسمت فیلتر در بالای صفحه قرار میدهیم :

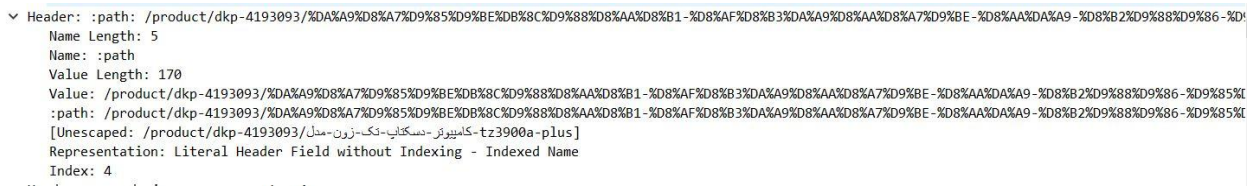
```
http2 and ip.dst == 92.114.19.26  and  ip.src == 192.168.1.3
```

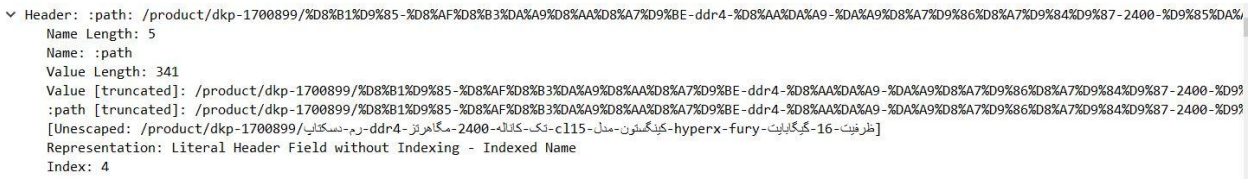
پس از اعمال فیلتر بالا در قسمت فیلتر تمامی بسته هایی که از پروتکل http 2 و آدرس مبدا و مقصد استفاده کرده اند فیلتر میشوند ( بسته های http2 پس از رمزگشایی توسط کلید دیکد شدند )

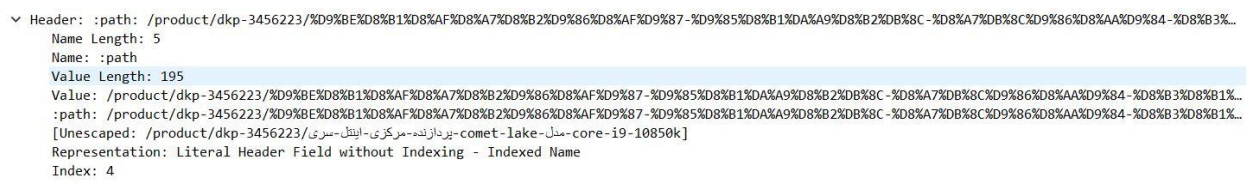
پس از اعمال فیلتر و پیدا شدن بسته های مورد نظر از آنجایی که بسته ی اول را پیدا کرده بودیم متوجه شدیم که در قسمت info بسته ی اول دارای توضیحات "GET /product " می باشد بنابراین با استفاده از گزینه ی edit->find packets بر روی محتوای بسته ها نیز فیلتر اعمال میکنیم و در نهایت سه بسته را پیدا میکنیم . برای این کار پس از کلیک بر روی find packets گزینه ی string را انتخاب کرده و محتوای فیلتر را اعمال میکنیم :




به این ترتیب توانستیم بسته های دیگر را با استفاده از فیلتر بسته ها و فیلتر محتوا پیدا کنیم . سه بسته در نهایت پیدا شد که به شکل زیر محتوای آن ها در قسمت header نمایش دهنده ی محصولی است که کاربر از آن بازدید داشته است . در شکل های زیر سه محصول و محتوای header آن ها نمایش داده شده است .




همانطور که در شکل بالا مشاهده میکنید کاربر از سه محصول بازدید کرده است :

1. کامپیوتر دسکتاپ تک زون tz3000a plus
2. رم دسکتاپ ۱۶ گیگابایت کینگستون تک کاناله با فرکانس ۲۴۰۰ مگاهرتز
3. پردازنده ی مرکزی اینتل سری core i9 10850k مدل comet lake

بدین ترتیب در این تمرین توانستیم به کمک نرم افزار وایرشارک و اعمال فیلتر ها بر روی بسته ها محصولاتی که کاربر مشاهده کرده است را پیدا کنیم .

## سوال ۲ ) کشف کرم

### قسمت ۱ :

برای حل این قسمت از سوال یک اسکریپت پایتون نوشته شد . در این اسکریپت ابتدا فایل را با استفاده از کتابخانه ی pyshark در پایتون که یک کتابخانه برای پارس کردن فایل های ترافیک شبکه است باز میکنیم و در قسمت فیلتر فیلتر های خود را اعمال می کنیم . اصولا بسته های udp همواره بخش کمی از ترافیک موجود در شبکه ها را تشکیل میدهند . دلیل این امر سرویس های کمی است که از این پروتکل استفاده میکنند . کرم ها غالبا از این پروتکل برای پرکردن ظرفیت پهنای باند کانال های شبکه استفاده می کنند . به همین علت در این تمرین با استفاده از فیلتری که در بخش قبلی صحبت کردیم و با pyshark بسته هایی را که udp هستند ولی در پورت های DHCP و DNS که به ترتیب ۶۷ و ۶۸ و ۵۳ هستند فرستاده نشده اند(زیرا این پورت ها برای سرویس های عادی استفاده میشود ) فیلتر کرده و پس از آن بسته های مشکوک که توسط کرم ها احتمالا صادر شده اند خروجی این قسمت از کد هستند . خواندن فایل ترافیک به صورت زیر با pyshark انجام میشود .

```
caps = pyshark.FileCapture('200302161400.dump' ,  display_filter='(udp) and (not udp.port==53 and not
udp.port==67 and not udp.port==68 and not icmp)')
```

همانطور که مشاهده می شود با استفاده از آرگومان دوم تابع File Capture فیلتر های این فایل را میدهیم این فیلتر ها همان فیلتر های Display filter هستند که در وایرشارک و یا خط فرمان tshark نیز استفاده میشود. در این فیلتر مشخص کرده ایم بسته های که udp هستند و icmp نیستند و همچنین مربوط به پورت های مشهور نیستند را خروجی دهد . در ادامه برای تحلیل بسته های مشکوک و پیدا کردم کرم ها آدرس و پورت مبدا و مقصد را در یک فایل csv ذخیره میکنیم .

```
with open('data.csv', 'w', encoding='UTF8') as f:
    writer = csv.writer(f)
    writer.writerow(['srcip' , 'dstip' , 'srcport' , 'dstport'])
    for c in caps:

        row = []
        row.append(c['ip'].src)
        row.append(c['ip'].dst)
        row.append(c['udp'].srcport)
        row.append(c['udp'].dstport)
        writer.writerow(row)
```

پس از ذخیره سازی کلیه ی بسته های مشکوک ( که تعریف مشکوک را در بند قبل گفتیم ) در یک فایل CSV از این فایل برای شمردن تعداد بسته ها استفاده میکنیم . هر بسته در یک خط ذخیره شده است و تعداد خطوط نشان دهنده ی کلیه ی بسته های مشکوک است .

برای شمردن تعداد خطوط در اسکریپت پایتونی از دستور wc و کتابخانه ی os در پایتون استفاده کرده ایم .

```
sh("wc -l data.csv")
```

```
def sh(script):
os.system("bash -c '%s'" % script)
```

تعداد کل بسته های مشکوک در فایل ترافیک داده شده بر اساس تعریف بسته ی مشکوک ( UDP باشد ولی روی پورت های مشهور نباشد و ICMP هم نباشد ) **396515** میباشد . همچنین در قسمت دوم سوال که گفته شده است ۱۰ تا از پربازدیدترین پورت های مقصد را پیدا کنیم در این اسکریپت از کتابخانه ی collection و تابع counter استفاده شده است . با استفاده از این تابع یکی از ستون های فایل csv را به آن میدهیم و این تابع n تا از پرتکرار ترین اعداد داخل آن ستون را بر میگرداند .

```
collections.Counter(line.rstrip().rpartition(',')[-1] for line in f).most_common(10)
```

در این تابع جداکننده همان جداکننده ی فایل csv در نظر گرفته شده است . محتوای فایل csv به شکل زیر است :

```
srcip,dstip,srcport,dstport
28.83.233.157,140.220.54.123,51855,137
219.16.145.215,140.220.157.52,6257,6257
5.84.41.17,140.220.33.184,6257,6257
95.86.58.170,150.49.162.211,1036,1434
208.9.196.200,165.142.12.141,1550,1434
217.97.148.78,144.94.182.74,1167,137
26.194.47.108,162.145.251.117,1026,137
212.92.90.26,160.199.175.116,1120,1434
5.84.41.17,140.220.33.184,6257,6257
95.194.11.203,150.49.18.249,1025,137
5.84.41.17,140.220.33.184,6257,6257
166.174.78.100,192.168.1.1,3252,161
```

در نهایت خروجی این کد ۱۰ تا از پر تکرار ترین پورت های باز دیده شده ی مقصد به همراه تعداد بسته های مشکوک میباشد :

```
➜  ANS-HW2-99210623 python3 6-1.py
number of suspicious packages is ==>
396515 data.csv
most 10 dst port use in network ==>  [('1434', 113834), ('38293', 99209), ('137', 89890), ('6257',
77839), ('3973', 1681), ('41170', 1553), ('1801', 1479), ('6970', 1446), ('2002', 1328), ('123', 554)]
```

پورت ۱۴۳۴ از همه ی پورت ها بیشتر تکرار شده است .

**قسمت ۲ :**

برای پیدا کردن کردم تولید کننده ی پیام ها از روش جست و جو استفاده کردم و سعی کردم با استفاده از شماره ی پورت مقصدی که کرم بیشتر از همه به آن مراجعه کرده است استفاده کنم . پورت شماره ی ۱۴۳۴ از همه بیشتر درخواست داشته است و با جست و جو متوجه شدم که sql slammer کرمی است که بیشترین درخواست را به این پورت ارسال میکند . این کرم با ارسال ترافیک زیاد به سمت سرور ها سعی در اقدام پیش زمینه ای در جهت حملات منع سرویس خدمت دارد و از پورت ۱۴۳۴ برای این کار استفاده میکند . این کرم از آسیب پذیری در SQL سرور مایکروسافت استفاده می کند ( البته برای این آسیب پذیری در سال های اخیر وصله های امنیتی ایجاد شده است ) . کرم SQL slammer از درگاه ۱۴۳۴ تحت پروتکل UDP

استفاده میکند .منبع

**قسمت ۳ :**

برای پیدا کردن تعداد بسته هایی که از یک کرم ارسال شده است به این صورت عمل کردیم که تعدا بسته های ارسال شده از هر آدرس را شمارش کرده و آن را به عنوان تعداد بسته های ارسال شده توسط هر کم در نظر گرفتیم

```
most 10 ip in network ==>  [('208.132.163.23', 99209), ('212.92.90.26', 14720), ('100.151.
51.188', 9738), ('140.220.33.184', 9350), ('140.220.157.52', 8442), ('140.220.206.121', 72
99), ('150.49.103.221', 6335), ('140.220.107.210', 4894), ('205.101.101.177', 4786), ('27.
8.8.238', 4764)]
```

با توجه به شمارش تعداد بسته های ارسالی پر شمار ترین آدرس های ip با تعداد بسته های ارسالی در شکل بالا آمده است .

**قسمت ۴ :**

در این قسمت با توجه به روندی که تا به حال طی شده تمامی آدرس های مبدا و مقصد بسته ها را در یک فایل به نام Part4 ذخیره میکنیم .

```
with open('Part4.csv', 'w', encoding='UTF8') as f:
writer = csv.writer(f)
writer.writerow(['srcIP' , 'dstIP'])
for c in caps:
row = []
row.append(c['ip'].src)
row.append(c['ip'].dst)
writer.writerow(row)
```

پس از طی فرآیند بالا در فایل Part4 تمامی جفت آدرس های مبدا و مقصد ذخیره شده است و پس از آن تعداد تکرار هر خط را با استفاده از خط زیر میشماریم :

```
with open('Part4.csv') as f:
   next(f) # Skip the first line
   print("most 7 pair ip in network ==> " , collections.Counter(line
for line in f).most_commo
```

پس از آن مشخص شد که ۷ تا از پر تکرار ترین مبدا ها و مقاصد بسته ها به شکل زیر است :

```
most 7 pair ip in network ==>
[('208.68.223.27,150.34.246.246\n', 694),
('193.88.30.17,145.237.245.90\n', 541),
('208.42.226.231,144.95.135.177\n', 476),
('150.49.78.128,24.14.181.92\n', 471),
('95.81.244.30,192.79.173.71\n', 422),
('150.49.103.221,28.82.9.142\n', 350),
('157.6.183.152,150.49.103.161\n', 350)]
```

همانطور که مشخص است بیشترین بسته ها بین دو جفت آدرس ۲۰۸.۶۸.۲۲۳.۲۷ و ۱۵۰.۳۴.۲۴۶.۲۴۶ جابه جا شده است .

خروجی تمام جفت آی پی ها در فایل Part4 قابل مشاهده است .

**قسمت ۵ :**

در این سوال برای رسم چارت مورد نظر از کتابخانه ی `matplotlib.pyplot` استفاده شده است .

برای رسم نمودار به این شکل عمل شده است که در ابتدا ۲۰۰۰ تا از جفت آدرس های پرتکراری که در قسمت قبل کلیه ی آن ها را در یک فایل ذخیره کرده بودیم را استخراج میکنیم و آنها را به ترتیب پرتکرار بودن بررسی میکنیم ( از آنجایی که گفته شده در سوال که فقط خط بین آدرس هایی که با هم در ارتباط بودند را رسم کنیم ) پس از این مقدار نگاشت افقی و عمودی هر آدرس را بدست آورده و در آرایه های جداگانه ذخیره کرده و سپس آدرس های پرتکرار را به صورت ۱۵ تا ۱۵ تا بررسی میکنیم ( برای نمونه گیری و جلوگیری از شلوغ شدن نمودار ) .

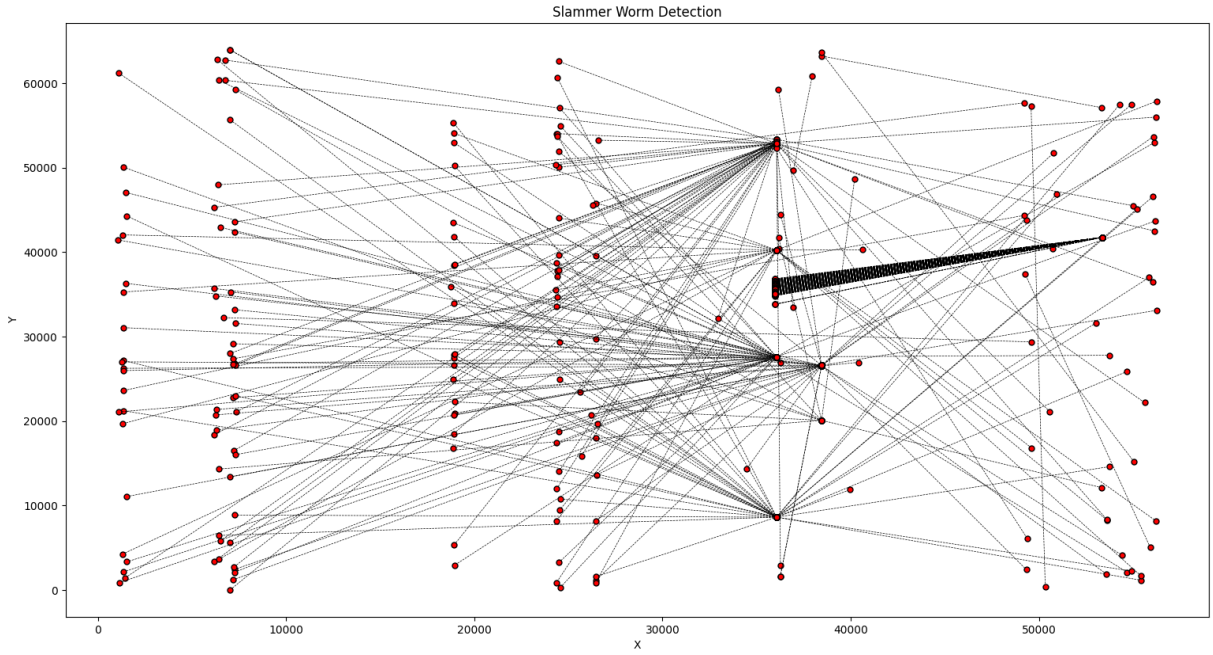
```
array_ips = list(dict.fromkeys(array_ips))
x_array = []
y_array = []
for ip in array_ips:
    a = int(ip.split('.')[0])
    b = int(ip.split('.')[1])
    c = int(ip.split('.')[2])
    d = int(ip.split('.')[3])
    x = a*256 + b
    y = c*256 + d
    x_array.append(x)
    y_array.append(y)

for tpl in range(0 , len(pair_ips) , 20):
    # print(pair_ips[tpl])
    # print()

    src_index = array_ips.index(pair_ips[tpl][0].split(',')[0])
    dst_index = array_ips.index(pair_ips[tpl][0].split(',')[1].split('"\\"')[0])
    x = []
    y = []
    x.append(x_array[src_index])
    x.append(x_array[dst_index])
    y.append(y_array[src_index])
    y.append(y_array[dst_index])
    plt.plot(x, y, '-ok')
```

همانطور که در شکل بالا مشاهده میشود برای نگاشت آدرس ها به نمودار از فرمول های گفته شده در صورت سوال استفاده شده است . این فرمول ها در واقع به شکل درستی طبقه بندی سازمانی را با استفاده از آدرس های IP نگاشت میکند

در نهایت پس از استخراج داده های مورد نیاز برای رسم نمودار در یک حلقه اطلاعات لازم برای رسم را به plt میدهیم نمودار را رسم میکنیم :




همانطور که مشاهده می شود با نرخ نمونه برداری ۲۰ از جفت های پرتکرار مشکوک نمودار داده های منتقل شده توسط کرم های احتمالی به شکل بالا در آمده است .

**قسمت ۶ :**

به نظر می رسد که رفتار بدافزار کرم مورد بررسی در سوال به این شکل است که خود را درون تعدادی از سیستم های سازمان ( که میتواند روتر های سازمان هم باشد ) جا داده و شروع به ارسال خود برای آدرس های تصادفی داخل سازمان میکند . در داخل شکل نیز این رفتار مشخص است . کرم خود را به برخی از سیستم های داخل شبکه رسانده ( مرکز ) و از آنجا شروع به پخش شدن در دیگر سیستم ها شده است تا شبکه را پویش کند و ترافیک عبوری را افزایش دهد ( پیش زمینه ی منع سرویس )

لازم به ذکر است طبق تحقیقات جست و جو ( ویکی پدیا ) در زمینه ی کرم SQL Slammer این کرم زمانی که به یک سیستم می رسد اگر برنامه ی SQL Microsoft Service روی آن در حال اجرا بر روی پورت ۱۴۳۴ باشد آنگاه کرم میتواند توسط این پورت پخش شود . بنابراین بهتر است سیستم هایی مورد هدف ابتدایی باشند که این ویژگی را دارا هستند ( پس از آن کرم خود آدرس های تصادفی تولید میکند و پخش میشود ) هر ۴ نود پرتکرار در شکل بالا مشخص هست که دچار کرم شده اند .