دانشگاه قم
دانشکده فنی و مهندسی
پروژه پایانی کارشناسی رشته مهندسی کامپیوتر

عنوان:

کاربرد احتمال در پیش بینی حملات سایبری و تحلیل رفتار ترافیک شبکه

استاد راهنما:
دکتر خیرالله رهسپارفرد

نگارنده:
امید شعبانعلی

اسفند 1404


بسم الله الرحمن الرحيم

تقدیر:

از زحمات استاد محترم راهنما و همه اساتید عزیز تشکر و قدردانی می نمایم

چکیده:

با افزایش حملات هکری، تشخیص به‌موقع فعالیت‌های مشکوک در شبکه اهمیت زیادی یافته است. روش‌های سنتی که تنها حملات شناخته شده را شناسایی می‌کنند، در برابر روش‌های جدید کارایی لازم را ندارند. در این پروژه، یک سیستم تشخیص نفوذ طراحی شده است که به‌جای لیست کردن حملات، «رفتار غیرعادی» شبکه را تشخیص می‌دهد. برای این کار از مدل ریاضی **رگرسیون خطی** استفاده شده است.

در روند انجام این پروژه، ابتدا ترافیک معمولی شبکه ضبط گردید تا الگوی رفتار سیستم در حالت عادی استخراج شود؛ سپس با استفاده از **قاعده ۳-سیگما (3-Sigma Rule)** و محاسبه انحراف معیار، یک «آستانه مجاز» برای تعداد بسته‌های شبکه (PPS) به دست آمد. برای تست سیستم، با ابزار Zenmap به شبکه حمله شد. نتایج نشان داد که با شروع حمله، ترافیک از خط رگرسیون فاصله گرفته و سیستم سریعاً این ناهنجاری را تشخیص می‌دهد. در نهایت، سیستم به‌گونه‌ای تنظیم شد که به محض تشخیص حمله، یک پیام هشدار خودکار به اپلیکیشن **ایتا** ارسال کند تا کاربر در لحظه مطلع شود. این پروژه نشان می‌دهد که با بهره‌گیری از ریاضیات و آمار و مفاهیم احتمال می‌توان یک سیستم امنیتی کاربردی و هوشمند ساخت.

واژه‌های کلیدی: امنیت شبکه، تشخیص نفوذ، رگرسیون خطی، ناهنجاری ترافیک، n8n، پیام‌رسان ایتا.

فهرست مطالب

فصل 1: مقدمه و نمای کلی سیستم **1**
1-1- مقدمه و اهمیت مانیتورینگ شبکه 2
1-2- تشریح مسئله : چرا تشخیص نا هنجاری (**AnomalyDetection**)؟ 2
1-3- اهداف و سوالات اصلی پروژه 3
1-3-1- اهداف اصلی پروژه 3
1-3-2- سوالات کلیدی 3
1-4- ضرورت و نوآوری پروژه 3
1-5- معرفی ساختار گزارش 4

فصل 2: مفاهیم پایه و مبانی آماری پروژه **5**
2-1- واحد اندازه‌گیری ترافیک(**PPS**) 6
2-2- رگرسیون خطی به زبان ساده 6
2-3- تحلیل باقی‌مانده‌ها(**Residuals**) 6
2-4- محاسبه آستانه بحرانی(**The 3-Sigma Rule**) 6
2-5- ابزار پویش **Nmap** و تاثیر آن بر آمار 7

فصل 3: معرفی ابزارها و زیرساخت اجرایی سیستم **8**
3-1- معرفی نرم‌افزار**Wireshark** 9
3-2- کتابخانه **Scapy** در پایتون 10
3-3- ابزار پویش شبکه(**Zenmap (Nmap**) 10
3-4- پلتفرم اتوماسیون**n8n** 10
3-5- پیام‌رسان ایتا و **API** ایتایار 11

فصل 4: پیاده‌سازی فنی و تشریح سیستم **12**
4-1- مقدمه فنی 13
4-2- پیش‌پردازش و آماده‌سازی داده‌ها(**Preprocessing**) 13
4-3- پیاده‌سازی مدل رگرسیون خطی 13
4-4- محاسبه آستانه تشخیص ناهنجاری(**Threshold**) 14
4-5- مکانیزم مانیتورینگ زنده و شنود پکت‌ها 15
4-6- اتصال به شبکه و اتوماسیون هشدار(**Integration**) 16

فصل 5: آزمایش، ارزیابی و نتیجه‌گیری **17**
5-1- مقدمه 18

5-2- سناریوی آزمایش 18
5-3- مشاهده خروجی در حالت عادی 18
5-4- اجرای حمله و واکنش سیستم 18
5-5- دریافت هشدار در پیام‌رسان ایتا 19
5-6- نتیجه‌گیری نهایی 20
5-7- پیشنهادات برای آینده 20

منابع **21**
منابع 22

پیوست‌ها **23**

فصل 1:

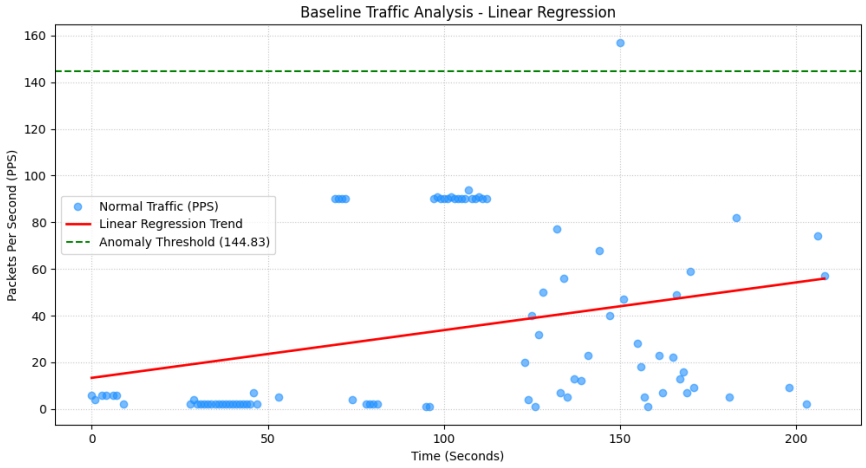
مقدمه و نمای کلی سیستم

۱-۱- مقدمه و اهمیت مانیتورینگ شبکه

امروزه شبکه‌های کامپیوتری ستون فقرات هر مجموعه‌ای هستند و پایداری آن‌ها مستقیماً بر عملکرد سیستم‌ها تاثیر می‌گذارد. با پیچیده‌تر شدن ابزارهای نفوذ، دیگر نمی‌توان تنها به فایروال‌های ساده تکیه کرد. مانیتورینگ یا پایش مداوم ترافیک، یکی از کلیدی‌ترین کارها برای حفظ امنیت است. اگر ما بدانیم در هر لحظه چه اتفاقی در شبکه می‌افتد، می‌توانیم قبل از اینکه یک نفوذ به فاجعه تبدیل شود، جلوی آن را بگیریم. در واقع امنیت شبکه و روش‌های مقابله با نفوذگران، نیازمند یک نگاه دقیق به آمارهای لحظه‌ای شبکه است [1]

۱-۲- تشریح مسئله : چرا تشخیص نا هنجاری (AnomalyDetection)؟

اکثر سیستم‌های امنیتی قدیمی (مثل آنتی‌ویروس‌ها) بر اساس "امضا" کار می‌کنند؛ یعنی فقط حملاتی را می‌شناسند که قبلاً تعریف شده باشند. اما مشکل اینجاست که هکرها مدام از روش‌های جدید استفاده می‌کنند .در این پروژه، مسئله اصلی این است که ما به جای گشتن دنبال یک حمله خاص، به دنبال "رفتار غیرعادی" هستیم. ترافیک شبکه در حالت عادی یک نظم مشخص دارد، اما وقتی حمله‌ای مثل اسکن پورت‌ها (Port Scanning) اتفاق می‌افتد، این نظم به هم می‌خورد. هدف ما این است که با استفاده از مدل ریاضی رگرسیون خطی، این بی‌نظمی را شناسایی کنیم [2]



عکس ۱-۱: تفاوت ترافیک منظم شبکه در حالت عادی و جهش ناگهانی پکت‌ها در زمان حمله

۱-۳- اهداف و سوالات اصلی پروژه

1-3-1- اهداف اصلی پروژه

- **ثبت رفتار عادی (Baseline):** بررسی ترافیک واقعی شبکه (روی اینترفیس 4VMnet) برای فهمیدن اینکه در حالت معمولی چند پکت در ثانیه جابجا می‌شود.
- **محاسبه آستانه مجاز (Threshold):** پیدا کردن یک عدد مشخص به عنوان "مرز خطر" با استفاده از فرمول‌های ریاضی، به طوری که نوسانات کوچک شبکه باعث پیام اشتباه نشود.
- **اتوماسیون اطلاع‌رسانی:** ایجاد یک سیستم هوشمند که به محض دیدن ترافیک غیرعادی، خودش پیام هشدار را به اپلیکیشن ایتا بفرستد.

1-3-2- سوالات کلیدی

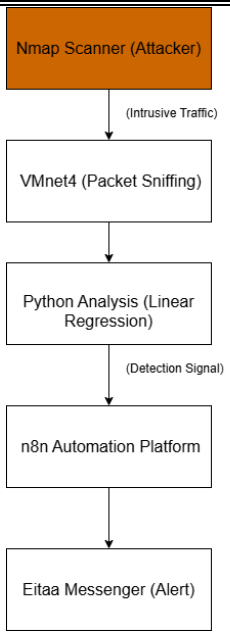
۱. آیا یک فرمول ریاضی ساده (رگرسیون) می‌تواند جایگزین سیستم‌های سنگین و گران‌قیمت امنیتی برای تشخیص اسکن‌های شبکه شود؟

۲. چطور می‌توان ترافیک‌های زمینه‌ای ویندوز (مثل SSDP) را از حملات واقعی تفکیک کرد؟

۳. سرعت پاسخگویی سیستم در زمان استفاده از ابزارهای اتوماسیون (n8n) چقدر است؟

۱-۴- ضرورت و نوآوری پروژه

بسیاری از سیستم‌های هوش مصنوعی فعلی برای اجرا نیاز به سخت‌افزارهای قوی و سنگین دارند. نوآوری این پروژه در "سبک بودن" آن است. ما از مدلی استفاده کردیم که با کمترین فشار به CPU، بیشترین دقت را در تشخیص اسکن‌های پرسرعت دارد. همچنین، استفاده از پلتفرم n8n برای ارسال پیام به ایتا، این پروژه را به یک ابزار کاملاً کاربردی و مدرن تبدیل کرده است.



عکس ۱-۲: نمودار کلی گردش کار سیستم (از ضبط پکت تا دریافت پیام در موبایل)

۱-۵- معرفی ساختار گزارش

این گزارش برای درک بهتر مراحل کار، به ۵ بخش تقسیم شده است:

- **فصل اول :**همین مقدمه و اهدافی که بررسی کردیم.
- **فصل دوم :**بررسی مبانی آماری و فرمول‌هایی که برای تحلیل ترافیک استفاده کردیم.
- **فصل سوم :**معرفی نرم‌افزارها و ابزارهایی مثل Scapy ، Nmapو n8n.
- **فصل چهارم :**جزئیات پیاده‌سازی، کدهای پایتون و نحوه تنظیم وب‌هوک.
- **فصل پنجم :**تست عملی سیستم، مشاهده نتایج در ایتا و نتیجه‌گیری نهایی.

فصل 2:

مفاهیم پایه و مبانی آماری پروژه

۲-۱- واحد اندازه‌گیری ترافیک(PPS)

در این پروژه، به جای تمرکز بر حجم داده‌ها (بایت)، از واحد PPS یا Packet Per Second استفاده کردیم. علت این انتخاب این است که در حملات پویش شبکه (Scanning)، مهاجم تعداد زیادی پکت کوچک برای بررسی پورت‌ها می‌فرستد. بنابراین، شمارش تعداد بسته‌ها در هر ثانیه، معیار بسیار دقیق‌تری برای تشخیص ناهنجاری نسبت به حجم ترافیک است.

۲-۲- رگرسیون خطی به زبان ساده

رگرسیون خطی یکی از ساده‌ترین و در عین حال کاربردی‌ترین روش‌های آماری است. هدف این مدل، پیدا کردن رابطه‌ای بین زمان (محور افقی یا X) و تعداد بسته‌ها (محور عمودی یاY) است. در واقع این مدل سعی می‌کند خطی را رسم کند که از نزدیک‌ترین فاصله ممکن نسبت به تمام نقاط ترافیک عادی عبور کند. معادله این خط به صورت $y = mx + b$ تعریف می‌شود که در آن m نشان‌دهنده شیب خط و b نشان‌دهنده عرض از مبدأ است.

۲-۳- تحلیل باقی‌مانده‌ها(Residuals)

باقی‌مانده یا خطا، در واقع فاصله هر نقطه واقعی ترافیک تا خط رگرسیون است. در یک شبکه پایدار، این فاصله‌ها باید کم و ناچیز باشند. اما زمانی که حمله اتفاق می‌افتد، نقاط جدید ترافیک فاصله بسیار زیادی از خط پیدا می‌کنند. سیستم ما با محاسبه این فاصله‌ها متوجه می‌شود که رفتار شبکه از حالت پیش‌بینی شده خارج شده است [3]

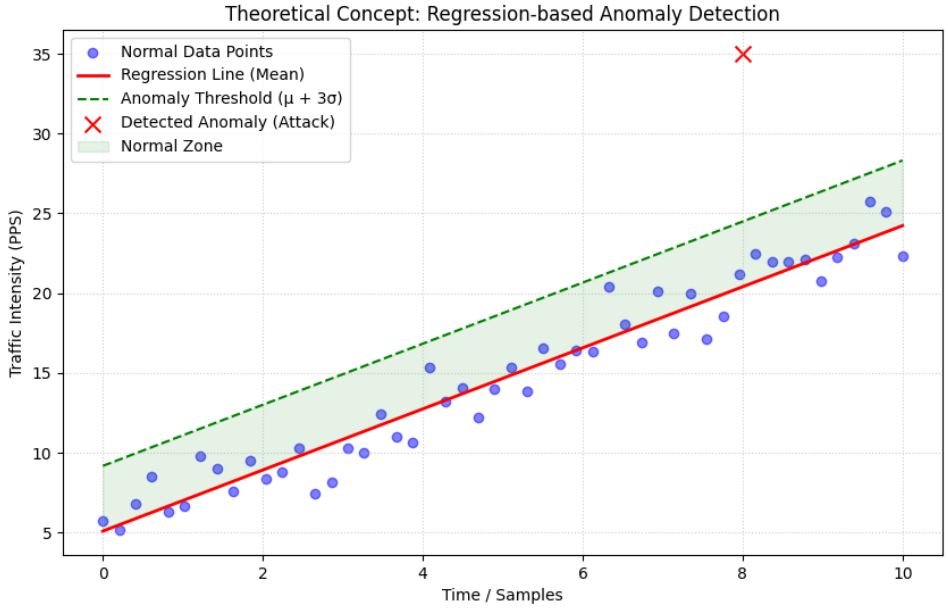
۲-۴- محاسبه آستانه بحرانی(The ۳-Sigma Rule)

یکی از بخش‌های مهم پروژه ما، تعیین عددی است که بالاتر از آن را به عنوان "حمله" در نظر بگیریم. برای این کار از روش خطای استاندارد (Standard Error) استفاده کردیم. طبق کدی که پیاده‌سازی شده، مراحل زیر طی شد :ابتدا میانگین ترافیک (Average PPS) به دست آمد. سپس میزان انحراف استاندارد خطا یا همان Se محاسبه شد. در نهایت، آستانه نهایی از فرمول زیر به دست آمد :

*Threshold = Average + (3 * Se)*

دلیل ضرب کردن عدد ۳ در خطای استاندارد این است که طبق اصول آمار، ۹۹.۷ درصد داده‌های

عادی باید در این محدوده قرار بگیرند. بنابراین اگر ترافیکی از این مرز رد شود، با احتمال نزدیک به ۱۰۰ درصد یک حمله یا ناهنجاری جدی در شبکه رخ داده است.



عکس 2-1: نمایش هندسی قانون ۳-سیگما و نحوه تعیین آستانه ناهنجاری در اطراف خط رگرسیون.

۵-۲- ابزار پویش Nmap و تاثیر آن بر آمار

ابزار Nmap در حالت‌های تهاجمی (مثل سوییچ T4-) بسته‌ها را با سرعت بسیار بالا و بدون وقفه در شبکه می‌فرستد. این کار باعث می‌شود مقدار ترافیک در مدل رگرسیون ما به ناگهان جهش کند و از آستانه یا همان Threshold که در بخش قبل حساب کردیم، عبور کند. سیستم در این لحظه با مقایسه ترافیک لحظه‌ای و آستانه تعیین شده، وضعیت حمله را شناسایی کرده و هشدار صادر می‌کند.

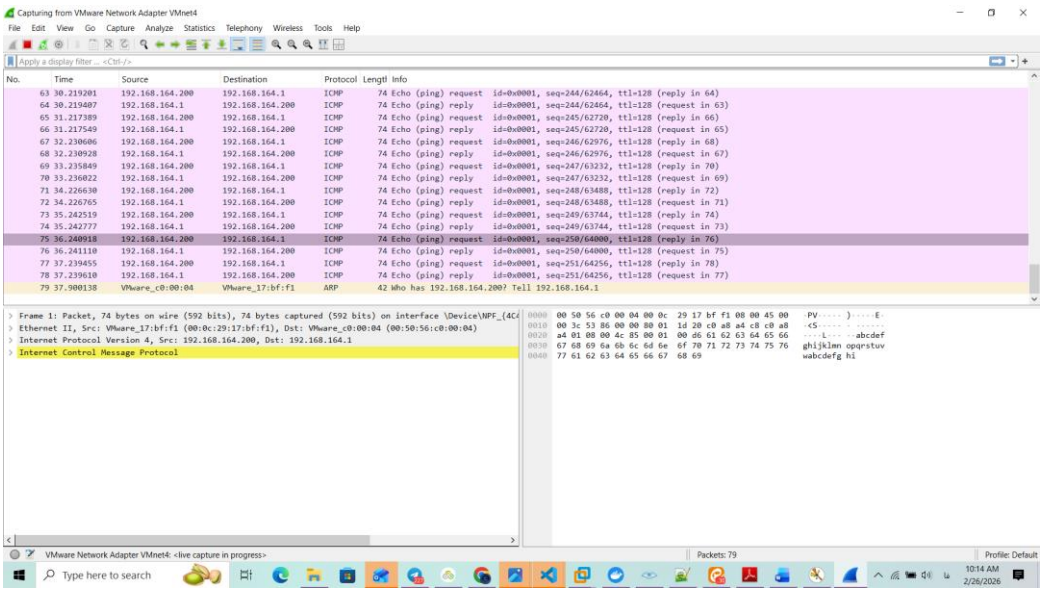
فصل 3:

معرفی ابزارها و زیرساخت اجرایی

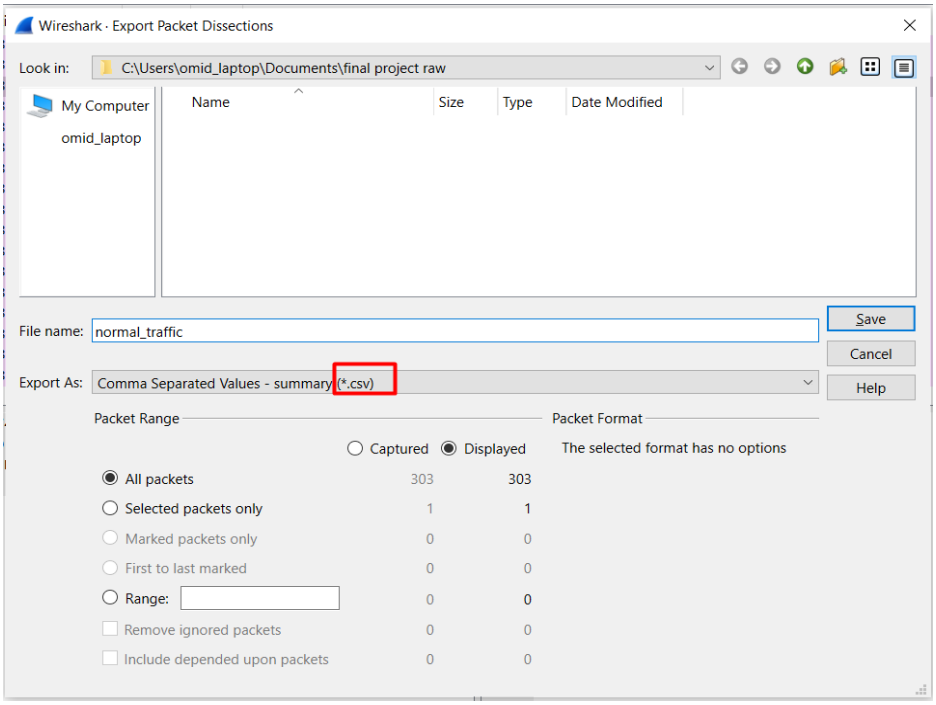
سیستم

۱-۳- معرفی نرم‌افزار Wireshark

ما در این پروژه برای جمع‌آوری داده‌های اولیه (Baseline) از وایرشارک استفاده کردیم. این ابزار به ما اجازه داد تا بفهمیم در شبکه مجازی ما (VMnet4) چه پکت‌هایی در حال رد و بدل شدن است. خروجی این ابزار به صورت فایل CSV استخراج شد تا برای پایتون قابل فهم باشد.



عکس ۳-۱: محیط نرم‌افزار Wireshark در حال پایش و ثبت ترافیک اولیه (Baseline) شبکه در اینترفیس VMnet4



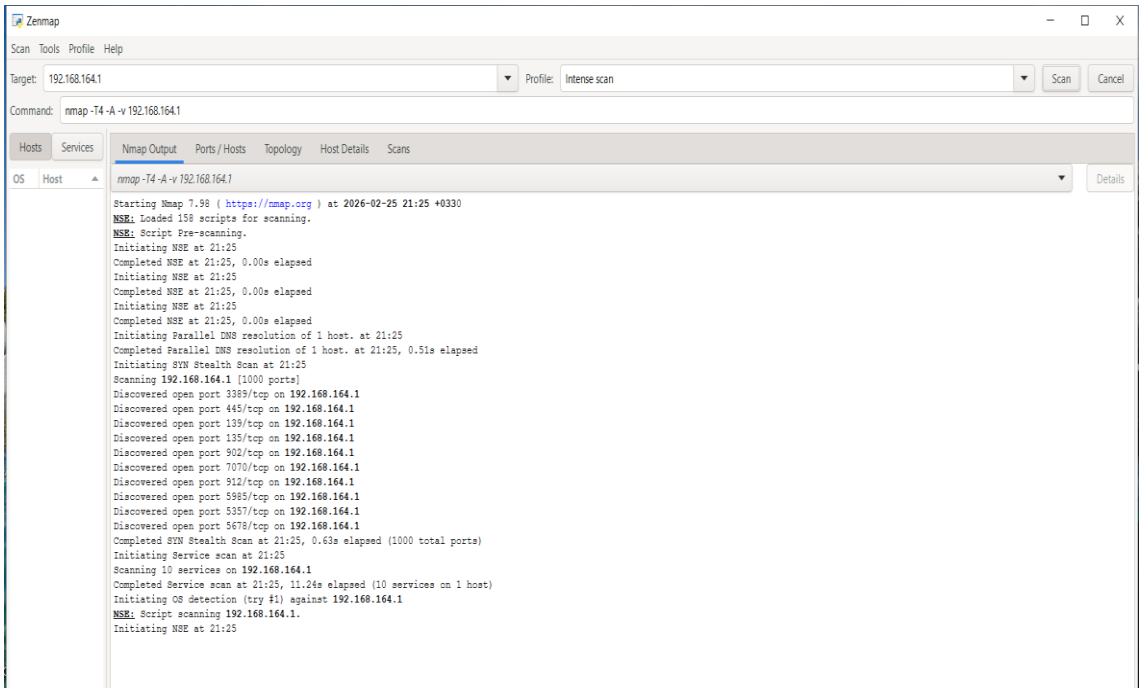
عکس 3-2: فرآیند شنود داده‌های شبکه و آماده‌سازی فایل خروجی CSV برای پردازش در مدل رگرسیون

۳-۲- کتابخانه Scapy در پایتون

قلب بخش مانیتورینگ زنده ما، کتابخانه Scapy است. برخلاف کتابخانه‌های معمولی، Scapyبه ما اجازه می‌دهد مستقیماً به کارت شبکه وصل شویم و پکت‌ها را در لحظه "شنود (Sniff) "کنیم. ما از این ابزار برای شمارش پکت‌ها در بازه‌های ۱ ثانیه‌ای استفاده کردیم.

۳-۳- ابزار پویش شبکه(Nmap) Zenmap

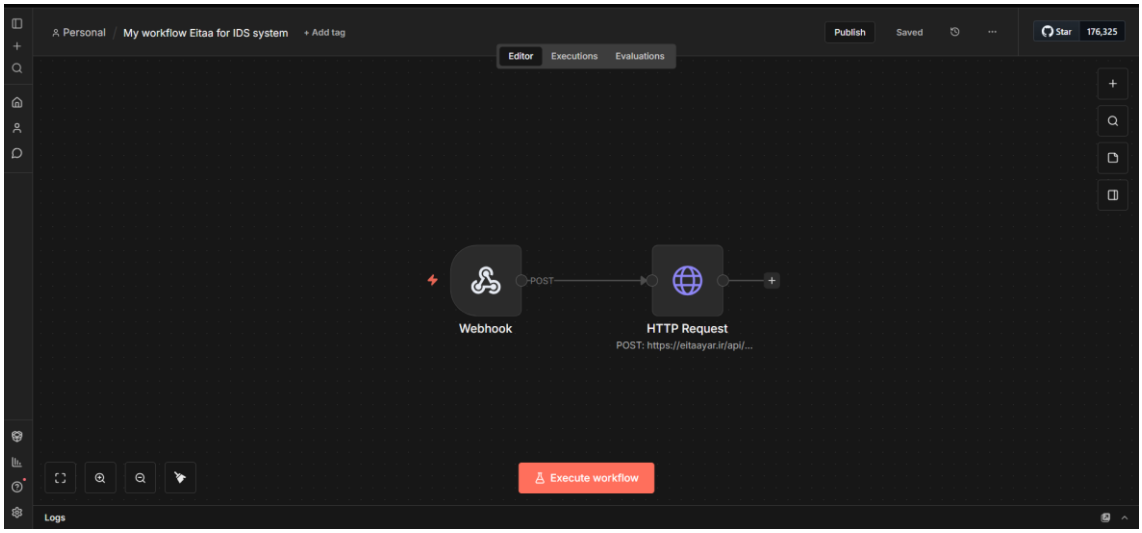
برای اینکه سیستم را تست کنیم، نیاز به یک "مهاجم" داشتیم Zenmap .نسخه گرافیکی Nmap است که با تنظیم آن روی حالت **Intense Scan**، ترافیک سنگینی ایجاد کردیم تا ببینیم آیا مدل رگرسیون ما متوجه این تغییر ناگهانی می‌شود یا خیر.



عکس 3-3: محیط نرم‌افزار Zenmap در حال اجرای اسکن روی آی‌پی هدف

۳-۴- پلتفرم اتوماسیونn۸n

یکی از بخش‌های مدرن این پروژه استفاده از n8n است. این ابزار به عنوان یک "واسط" عمل می‌کند. وقتی پایتون حمله را تشخیص می‌دهد، یک سیگنال به n8n می‌فرستد و n8n وظیفه دارد این سیگنال را به پیام قابل فهم برای "بات ایتا" تبدیل کند. این کار باعث می‌شود سیستم ما بدون نیاز به مانیتور کردن مداومِ کنسول پایتون، هشدارها را به موبایل ما بفرستد.



عکس 3-4: نمای محیط کاربری n8n و گره‌های (Nodes) متصل شده به هم

۳-۵- پیام‌رسان ایتا و API ایتایار

برای بخش اطلاع‌رسانی، از پیام‌رسان بومی ایتا استفاده شد. با استفاده از قابلیت Webhook و اتصال آن به n8n، توانستیم یک کانال ارتباطی امن برای دریافت هشدارهای لحظه‌ای ایجاد کنیم.

فصل 4:

پیاده‌سازی فنی و تشریح سیستم

۴-۱- مقدمه فنی

در این فصل به بررسی جزئیات پیاده‌سازی کدها، نحوه پردازش داده‌های خام و اتصال بخش‌های مختلف سیستم به یکدیگر می‌پردازیم. پیاده‌سازی این پروژه شامل سه بخش اصلی است: استخراج ویژگی‌ها از ترافیک، مدل‌سازی ریاضی و در نهایت مانیتورینگ زنده.

۴-۲- پیش‌پردازش و آماده‌سازی داده‌ها(Preprocessing)

داده‌های خام که از نرم‌افزار وایرشارک به صورت فایل CSV استخراج شده بودند، توسط کتابخانه pandas در پایتون فراخوانی شدند. از آنجایی که ترافیک شبکه در لحظات مختلف نوسان دارد، ما داده‌ها را بر اساس ستون زمان (Time) گروه‌بندی کردیم تا تعداد بسته‌ها در هر ثانیه یا همان PPS مشخص شود. در این مرحله، نوسانات ناگهانی ترافیک (Burst Traffic) نیز به عنوان بخشی از رفتار طبیعی شبکه در داده‌های آموزشی حفظ شدند تا مدل نسبت به فعالیت‌های عادی پرحجم حساسیت کاذب نشان نداده و آستانه تشخیص (Threshold) با دقت بیشتری تعیین گردد. این کار به ما کمک کرد تا یک دید کلی از وضعیت ترافیک نرمال شبکه به دست آوریم.

```
# Preprocessing: Round time to nearest second
# This allows us to count packets per second (PPS)
df['Time'] = df['Time'].astype(float).round(0)

# Group by 'Time' to get the count of packets in each second
pps_data = df.groupby('Time').size().reset_index(name='Packet_Count')
```

شرح برنامه 4-1 : در این قطعه کد، از کتابخانه pandas برای گروه‌بندی داده‌های خام و استخراج نرخ پکت در ثانیه استفاده شده است .

۴-۳- پیاده‌سازی مدل رگرسیون خطی

برای مدل‌سازی ترافیک، از کلاس LinearRegression در کتابخانه scikit-learn استفاده شد. مدل سعی می‌کند رابطه‌ای بین زمان (محور x) و تعداد بسته‌ها (محور y) پیدا کند. این رابطه در ساده‌ترین حالت به صورت فرمول $y = wx + b$ تعریف می‌شود.

پس از آموزش دیدن مدل روی داده‌های عادی، "باقی‌مانده‌ها" یا همان Residuals محاسبه شدند. باقی‌مانده در واقع تفاضل بین مقدار واقعی مشاهده شده در شبکه و مقداری است که مدل ما پیش‌بینی

کرده است. فرمول آن به زبان ساده به این صورت است:

مقدار پیش‌بینی شده - مقدار واقعی = باقی‌مانده

مدل رگرسیون مورد استفاده، با کمترین میزان خطای میانگین مربعات (MSE) بر روی داده‌های آموزشی برازش شد تا پیش‌بینی ترافیک نرمال با دقت حداکثری انجام پذیرد و انحرافات جزئی نیز قابل شناسایی باشند.

```
# Initialize and fit the Linear Regression model
model = LinearRegression()
model.fit(X, y)

# Predict values to calculate residuals
y_pred = model.predict(X)
```

شرح برنامه 4-2: برازش مدل رگرسیون خطی بر روی داده‌های آموزشی جهت استخراج خط روند ترافیک نرمال.

۴-۴- محاسبه آستانه تشخیص ناهنجاری(Threshold)

مهم‌ترین بخش آماری این پروژه، محاسبه "خطای استاندارد برآورد" یا همان Se است. این عدد به ما می‌گوید که داده‌های عادی شبکه چقدر از خط اصلی رگرسیون فاصله دارند. برای تعیین آستانه نهایی و مرز تشخیص حمله، از میانگین ترافیک به علاوه ۳ برابر خطای استاندارد استفاده کردیم. این کار باعث می‌شود دقت سیستم بالا برود و نوسانات معمولی شبکه به اشتباه "حمله" در نظر گرفته نشوند (کاهش هشدارهای اشتباه). فرمول نهایی ما به این صورت بود:

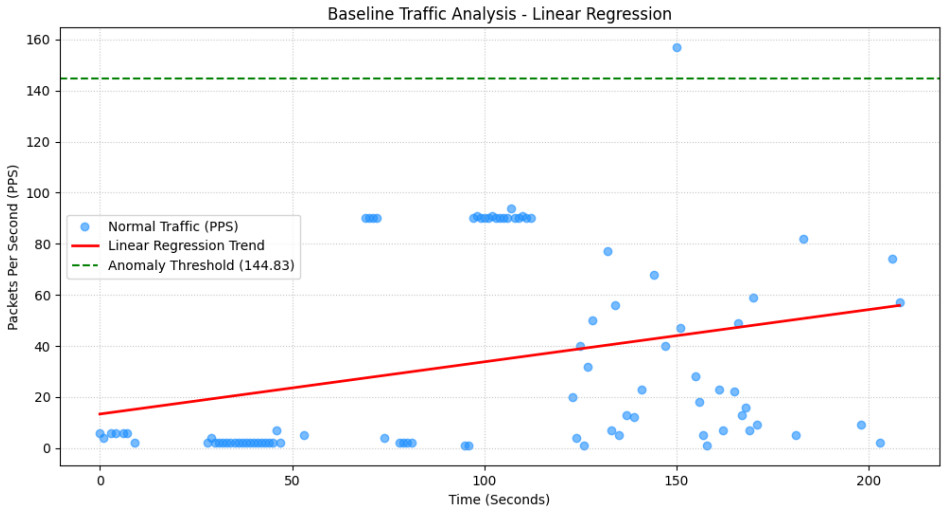
Threshold = Average + (3* Se)

```
# Calculate Standard Error of the Estimate (Se)
# Se = sqrt( sum( (y_actual - y_pred)^2 ) / (n - 2) )
residuals = y - y_pred
se = np.sqrt(np.sum(residuals**2) / (len(y) - 2))

# Calculate Threshold (3-sigma rule: 3 * Se above the regression line)
# We use this as our anomaly detection boundary
avg_pps = np.mean(y)
threshold = avg_pps + (3 * se)
```

شرح برنامه 4-3: محاسبه خطای استاندارد برآورد و تعیین آستانه نهایی تشخیص حمله با استفاده از قانون ۳-سیگما.

طبق محاسبات انجام شده روی داده‌های واقعی این پروژه، عدد 144.83 به عنوان مرز بحرانی یا آستانه نهایی به دست آمد.



شکل 4-1: نمودار تحلیل رگرسیون ترافیک واقعی و تعیین آستانه بحرانی (Threshold) بر اساس داده‌های شبکه.

توضیحات شکل *:در این نمودار، **نقاط آبی** نشان‌دهنده تعداد بسته‌ها در ثانیه (PPS) در طول زمان هستند .**خط قرمز**، روند کلی ترافیک نرمال را مدل‌سازی کرده و **خط‌چین سبز**، آستانه بحرانی (144.83) را نشان می‌دهد که با فرمول Average 3S_e +به دست آمده است. عبور **نقطه ناهنجار** از مرز مشخص شده، لحظه دقیق شناسایی حمله Nmap را تایید می‌کند .*

۴-۵- مکانیزم مانیتورینگ زنده و شنود پکت‌ها

در این بخش از کتابخانه قدرتمند Scapy استفاده شده است. اسکریپت اصلی پروژه با استفاده از تابع sniff روی کارت شبکه مجازی، بسته‌ها را در بازه‌های یک ثانیه‌ای می‌شمارد. منطق تشخیص حمله در کد بسیار ساده و سریع طراحی شده است؛ به این صورت که اگر تعداد بسته‌های شمارش شده در یک ثانیه (PPS لحظه‌ای) از عدد آستانه (144.83) بیشتر شود، سیستم وضعیت را "بحرانی" تشخیص داده و بلافاصله تابع ارسال هشدار را اجرا می‌کند.

```
# Sniff packets for 1 second
scapy.sniff(iface=INTERFACE, prn=process_packet, timeout=1, store=0)

if current_pps > THRESHOLD:
    print(f"\n[!!!] ANOMALY DETECTED! Traffic: {current_pps} PPS")
    send_alert(current_pps)
```

شرح برنامه ۴-۴: مانیتورینگ بلادرنگ اینترفیس شبکه و مقایسه PPS لحظه‌ای با آستانه بحرانی جهت تشخیص ناهنجاری.

```
(.venv) D:\Omid\پروژه دانشگاه سورد 7\Final Project\project\src>py detect_attack.py
[*] Monitoring started on \Device\NPF_{4C486550-C532-4E6F-B54F-88C9D735F769}...
[*] Threshold is set to: 144.83 PPS
Current Traffic: 1986 PPS
[!!!] ANOMALY DETECTED! Traffic: 1986 PPS
[+] Alert sent! Traffic: 1986 PPS
Current Traffic: 0 PPSS
```

عکس 4-2: اسکرین‌شات از محیط خط فرمان (cmd) در لحظه شناسایی ناهنجاری و صدور هشدار سیستمی.

۴-۶- اتصال به شبکه و اتوماسیون هشدار(Integration)

زمانی که حمله تشخیص داده می‌شود، سیستم یک درخواست به سمت سرور n8n ارسال می‌کند. این داده‌ها شامل اطلاعاتی مثل زمان دقیق حمله و تعداد پکت‌ها است. پلتفرم n8n با دریافت این اطلاعات، بلافاصله از طریق یک رابط (Webhook) پیام را به بات طراحی شده در پیام‌رسان ایتا منتقل می‌کند. این زنجیره باعث می‌شود که مدیر شبکه بدون نیاز به چک کردن مداوم سیستم، از طریق موبایل خود در جریان حملات قرار بگیرد.

```python
payload = {
    "method": "Linear Regression Model",
    "traffic": pps,
    "threshold": THRESHOLD,
    "status": "Attack Detected",
    "message": "Intrusion detected by Linear Regression Model",
    "student": "Omid",
    "network": "VMware VMnet4"
}
response = requests.post(N8N_WEBHOOK_URL, json=payload)
```

شرح برنامه ۴-۵: متد ارسال داده‌های ناهنجاری به وب‌هوک n8n جهت اطلاع‌رسانی خودکار در پیام‌رسان ایتا

فصل 5:

آزمایش، ارزیابی و نتیجه‌گیری

۵-۱- مقدمه

در این فصل، کارایی سیستم طراحی شده را در مواجهه با شرایط واقعی بررسی می‌کنیم. هدف این است که ببینیم آیا مدل رگرسیونی که در فصل‌های قبل توضیح دادیم، واقعاً می‌تواند ترافیک عادی را از حمله تشخیص دهد و آیا فرآیند اطلاع‌رسانی به درستی انجام می‌شود یا خیر.

۵-۲- سناریوی آزمایش

برای تست سیستم، یک محیط آزمایشگاهی مجازی شامل دو بخش زیر ایجاد شد :

۱ .**سیستم هدف :(Target)** یک ماشین مجازی که کارت شبکه آن (VMnet4) توسط اسکریپت پایتون ما مانیتور می‌شود.

۲ .**سیستم مهاجم :(Attacker)** استفاده از ابزار Zenmap برای ارسال پکت‌های پویش شبکه (Intense Scan) به سمت سیستم هدف.

۵-۳- مشاهده خروجی در حالت عادی

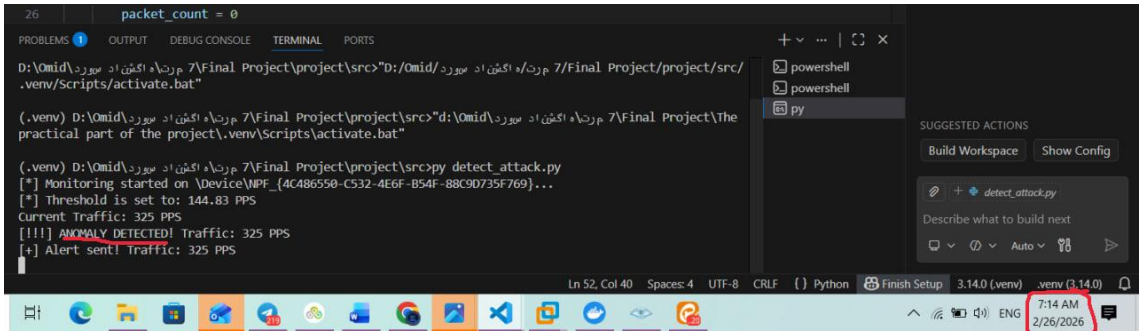
در ابتدای کار، سیستم در شرایطی قرار گرفت که فقط فعالیت‌های معمولی شبکه (مثل پکت‌های شناسایی ویندوز) در جریان بود. در این حالت، تعداد بسته‌ها در هر ثانیه (PPS) بسیار پایین‌تر از آستانه **144.83**بود. نمودار خروجی پایتون نشان داد که تمام نقاط ترافیکی در اطراف خط رگرسیون قرار دارند و هیچ هشدار اشتباهی صادر نشد.

۵-۴- اجرای حمله و واکنش سیستم

به محض شروع اسکن توسط Zenmap، تعداد بسته‌های ارسالی در ثانیه به شدت افزایش یافت و به عددی بالای ۶۰۰ پکت در ثانیه رسید. در این لحظه:

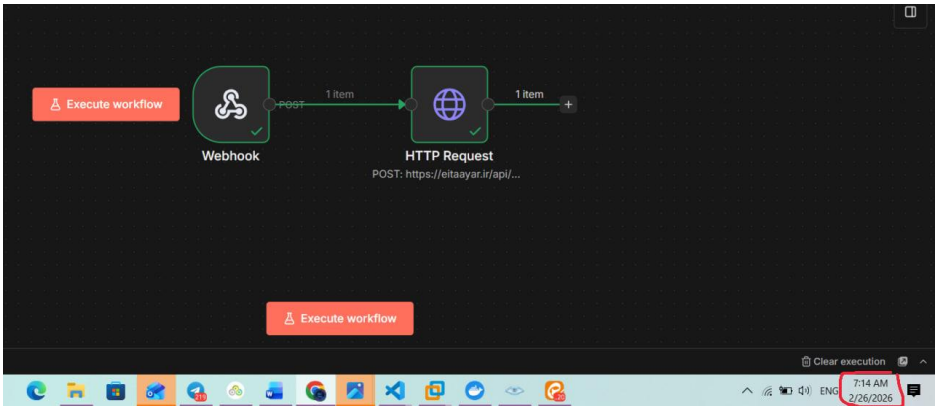
۱. سیستم پایتون بلافاصله جهش ترافیکی را تشخیص داد (چون مقدار PPS از آستانه 144.83 عبور کرد).

۲. در کنسول پایتون پیام "ANOMALY DETECTED" به همراه مقدار دقیق ترافیک چاپ شد.



شکل 5-1: تشخیص ناهنجاری در محیط خط فرمان پایتون بلافاصله پس از شروع حمله.

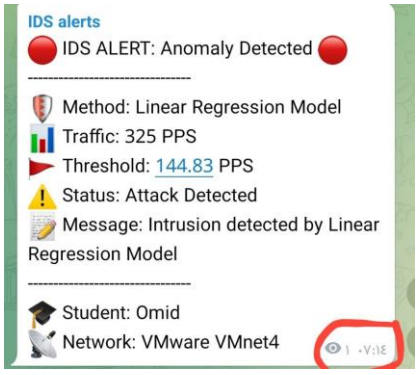
۳. درخواست وب‌هوک به سمت n8n ارسال گردید. با دریافت این درخواست، گره‌های (Nodes) مربوطه در جریان کاری فعال شدند.



شکل 5-2: وضعیت فعال جریان کاری (Workflow) در پلتفرم n8n در لحظه پردازش هشدار.

۵-۵- دریافت هشدار در پیام‌رسان ایتا

پس از تشخیص حمله، پلتفرم n8n طبق جریان کاری (Workflow) تعریف شده، اطلاعات را پردازش کرده و یک پیام شامل زمان حمله و شدت آن را به بات اختصاصی در پیام‌رسان ایتا ارسال کرد. تصویر زیر نمونه‌ای از هشدار دریافت شده روی تلفن همراه را نشان می‌دهد که سرعت بالای پاسخگویی سیستم (کمتر از ۲ ثانیه) را تایید می‌کند.



شکل 5-3: تاییدیه نهایی و دریافت هشدار هوشمند بر روی اپلیکیشن ایتا.

۵-۶- نتیجه‌گیری نهایی

این پروژه نشان داد که می‌توان با استفاده از ابزارهای ساده ریاضی مانند رگرسیون خطی، یک سیستم تشخیص نفوذ (IDS) سبک و کارآمد ساخت. مزیت اصلی این روش، بی‌نیاز بودن از سخت‌افزارهای گران‌قیمت و توانایی تشخیص حملات جدید بدون داشتن الگوی قبلی است. همچنین ترکیب پایتون با ابزارهای اتوماسیون مدرن مثل n8n، امکان مانیتورینگ از راه دور را برای مدیران شبکه به راحتی فراهم می‌کند.

۵-۷- پیشنهادات برای آینده

برای ارتقای این پروژه در آینده می‌توان موارد زیر را مد نظر قرار داد:

- استفاده از مدل‌های رگرسیون پیچیده‌تر برای شبکه‌هایی که نوسانات بسیار زیادی دارند.
- اضافه کردن قابلیت "مسدودسازی خودکار" آی‌پی مهاجم به محض تشخیص حمله در فایروال.
- گسترش سیستم برای تشخیص انواع دیگر حملات مانند حملات منع سرویس (DoS).
- ایجاد یک داشبورد گرافیکی با استفاده از Grafana برای مشاهده لحظه‌ای نمودار رگرسیون و ترافیک شبکه

منابع

منابع

[1] ا. ملکیان، نفوذگری در شبکه و روش‌های مقابله، تهران: انتشارات نص، 1402.

[2] G. Lyon ،Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning ،Insecure.Com LLC ،2009.

[3] D. C. Montgomery ،E. A. Peck و G. G. Vining ،Introduction to Linear Regression Analysis6 ،th Hoboken: John Wiley & Sons ،2021.

پیوست‌ها

در این بخش، مستندات فنی و جزئیات اجرایی پروژه جهت فراهم کردن امکان بازتولید آزمایش‌ها و بررسی دقیق‌تر ساختار سیستم ارائه شده است.

پیوست الف: کدهای کامل اسکریپت‌های پایتون

در ادامه، کدهای کامل نوشته شده برای دو مرحله اصلی پروژه (آموزش مدل و مانیتورینگ زنده) آورده شده است.

۱. فایل train_model.py (تحلیل آفلاین و تعیین آستانه):

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import os

# Configuration - Paths based on your project structure
DATA_PATH = '../data/raw/normal_traffic.csv'
REPORT_PATH = '../reports/baseline_regression.png'

def train_anomaly_detector():
    print(f"Loading data from: {DATA_PATH}...")

    # Load the CSV file provided
    try:
        df = pd.read_csv(DATA_PATH)
    except FileNotFoundError:
        print("Error: normal_traffic.csv not found in data/raw/")
        return

    # Preprocessing: Round time to nearest second
    # This allows us to count packets per second (PPS)
    df['Time'] = df['Time'].astype(float).round(0)

    # Group by 'Time' to get the count of packets in each second
    pps_data = df.groupby('Time').size().reset_index(name='Packet_Count')

    # After pps_data calculation is completed:
```

```
output_path = '../data/processed/pps_data.csv'
pps_data.to_csv(output_path, index=False)
print(f"File successfully saved to: {output_path}")

# Prepare X (Time) and y (Packet Count) for Linear Regression
X = pps_data['Time'].values.reshape(-1, 1)
y = pps_data['Packet_Count'].values

# Initialize and fit the Linear Regression model
model = LinearRegression()
model.fit(X, y)

# Predict values to calculate residuals
y_pred = model.predict(X)

# Calculate Standard Error of the Estimate (Se)
# Se = sqrt( sum( (y_actual - y_pred)^2 ) / (n - 2) )
residuals = y - y_pred
se = np.sqrt(np.sum(residuals**2) / (len(y) - 2))

# Calculate Threshold (3-sigma rule: 3 * Se above the regression line)
# We use this as our anomaly detection boundary
avg_pps = np.mean(y)
threshold = avg_pps + (3 * se)

print("\n" + "="*30)
print(" REGRESSION ANALYSIS RESULTS")
print("="*30)
print(f"Average PPS: {avg_pps:.2f}")
print(f"Standard Error (Se): {se:.4f}")
print(f"Calculated Threshold: {threshold:.2f}")
print("="*30 + "\n")

# Visualization
plt.figure(figsize=(12, 6))
plt.scatter(X, y, color='dodgerblue', alpha=0.6, label='Normal Traffic (PPS)')
plt.plot(X, y_pred, color='red', linewidth=2, label='Linear Regression Trend')
plt.axhline(y=threshold, color='green', linestyle='--', label=f'Anomaly Threshold
({threshold:.2f})')
```

```
        plt.title('Baseline Traffic Analysis - Linear Regression')
        plt.xlabel('Time (Seconds)')
        plt.ylabel('Packets Per Second (PPS)')
        plt.legend()
        plt.grid(True, linestyle=':', alpha=0.7)

        # Save the report
        os.makedirs(os.path.dirname(REPORT_PATH), exist_ok=True)
        plt.savefig(REPORT_PATH)
        print(f"Analysis plot saved to: {REPORT_PATH}")

    if __name__ == "__main__":
        train_anomaly_detector()
```

۲. فایل detect_attack.py (شنود زنده و تشخیص نفوذ):

```
import scapy.all as scapy
from scapy.all import dev_from_index
import time
import requests # For sending alerts to n8n

# --- Configuration ---
# The threshold we calculated in Phase 2
THRESHOLD = 144.83
# Interface name (You might need to check the exact name in Scapy)
INTERFACE = dev_from_index(14)
# n8n Webhook URL (We will fill this in Phase 4)
N8N_WEBHOOK_URL = "http://localhost:5678/webhook-test/62246be6-37d8-
466e-a23e-33c11d8970c9"

packet_count = 0

def process_packet(packet):
    global packet_count
    packet_count += 1

def start_monitoring():
    global packet_count
    print(f"[*] Monitoring started on {INTERFACE}...")
    print(f"[*] Threshold is set to: {THRESHOLD} PPS")
```

```
    while True:
        packet_count = 0
        # Sniff packets for 1 second
        scapy.sniff(iface=INTERFACE, prn=process_packet, timeout=1, store=0)

        current_pps = packet_count
        print(f"Current Traffic: {current_pps} PPS", end='\r')

        if current_pps > THRESHOLD:
            print(f"\n[!!!] ANOMALY DETECTED! Traffic: {current_pps} PPS")
            send_alert(current_pps)
            # Small cooldown to avoid spamming alerts
            time.sleep(5)

def send_alert(pps):
    # Prepare the payload with the exact fields you want in Eitaa
    payload = {
        "method": "Linear Regression Model",
        "traffic": pps,
        "threshold": THRESHOLD,
        "status": "Attack Detected",
        "message": "Intrusion detected by Linear Regression Model",
        "student": "Omid",
        "network": "VMware VMnet4"
    }

    try:
        response = requests.post(N8N_WEBHOOK_URL, json=payload)
        if response.status_code == 200:
            print(f"[+] Alert sent! Traffic: {pps} PPS")
        else:
            print(f"[!] n8n Error: {response.status_code}")
    except Exception as e:
        print(f"[-] Connection failed: {e}")

if __name__ == "__main__":
    start_monitoring()
```

مخزن پروژه در گیت‌هاب : جهت مشاهده تاریخچه توسعه اسکریپت‌ها، دسترسی به فایل‌های داده (Dataset) و مستندات تکمیلی، می‌توانید به مخزن این پروژه در گیت‌هاب مراجعه فرمایید:
https://github.com/Omid-Programmer2/Network-IDS-Regression-n8n

پیوست ب: ساختار جریان کاری در n8n (JSON Export)

برای بازسازی جریان کاری (Workflow) در پلتفرم n8n، می‌توان از قطعه کد JSON زیر استفاده کرد. با کپی کردن این متن و انتخاب گزینه **Import from JSON** در محیط n8n، تمام گره‌ها و ارتباطات وب‌هوک و بات ایتا به طور خودکار پیکربندی می‌شوند:


```
{
 "name": "My workflow Eitaa for IDS system",
 "nodes": [
  {
   "parameters": {
    "httpMethod": "POST",
    "path": "62246be6-37d8-466e-a23e-33c11d8970c9",
    "options": {}
   },
   "type": "n8n-nodes-base.webhook",
   "typeVersion": 2.1,
   "position": [
    -912,
    48
   ],
   "id": "6774cf0d-4597-44fd-8bff-dd623607ad2e",
   "name": "Webhook",
   "webhookId": "62246be6-37d8-466e-a23e-33c11d8970c9"
  },
  {
   "parameters": {
    "method": "POST",
    "url": "https://eitaayar.ir/api/bot430632:cc6f2cbd-580d-42c2-acbc-
44eb9f5a696e/sendMessage",
    "sendBody": true,
    "contentType": "multipart-form-data",
    "bodyParameters": {
```

```
          "parameters": [
            {
              "name": "chat_id",
              "value": "11000415"
            },
            {
              "name": "text",
              "value": "=🔴 IDS ALERT: Attack Detected 🔴 ---------------------------
Status: {{ $json.body.status }} Confidence: {{ $json.body.probability }} Method: {{
$json.body.alert_type }} Student: {{ $json.body.student }} Supervisor: {{
$json.body.professor }} ---------------------------- Target: {{ $json.body.target_os }}
Time: {{ $json.body.timestamp }}"
            }
          ]
        },
        "options": {}
      },
      "type": "n8n-nodes-base.httpRequest",
      "typeVersion": 4.3,
      "position": [
        -640,
        48
      ],
      "id": "a7ce5b06-9086-4d76-9723-af1fc06556fc",
      "name": "HTTP Request"
    }
  ],
  "pinData": {},
  "connections": {
    "Webhook": {
      "main": [
        [
          {
            "node": "HTTP Request",
            "type": "main",
            "index": 0
          }
        ]
      ]
    }
```

```
    },
    "active": false,
    "settings": {
      "executionOrder": "v1",
      "availableInMCP": false
    },
    "versionId": "d1a7d15f-6c8b-4546-8ede-cf851f719c44",
    "meta": {
      "templateCredsSetupCompleted": true,
      "instanceId":
"e48b868895bd0148547c406cff5145792841e60d27055cef4137b5ce120a64d4"
    },
    "id": "48kmV3mliAxYkpwM",
    "tags": []
}
```


پیوست ج: نیازمندی‌های سیستم و مشخصات محیط آزمایشگاهی

جهت اجرای صحیح پروژه، مشخصات نرم‌افزاری و سخت‌افزاری زیر در محیط آزمایشگاهی پیاده‌سازی شده است:

زبان برنامه‌نویسی: Python نسخه 3.10 یا بالاتر.

کتابخانه‌های پایتون:

- scapy: جهت شنود و تحلیل بسته‌های شبکه.
- pandas و numpy: جهت پردازش داده‌ها و محاسبات آماری.
- scikit-learn: جهت پیاده‌سازی مدل رگرسیون خطی.
- requests: جهت برقراری ارتباط با API وب‌هوک.

زیرساخت مجازی‌سازی: VMware Workstation نسخه 17.

سیستم‌عامل‌های مورد استفاده:

- سیستم مانیتورینگ و مهاجم: ویندوز 10 (میزبان اسکریپت پایتون و ابزار Zenmap).
- سیستم هدف :(Target) ویندوز 7 مجازی (مستقر درVMnet4). شایان ذکر است که معماری اسکریپت پایتون به‌گونه‌ای طراحی شده که قابلیت اجرا و پایش ترافیک در سایر سیستم‌عامل‌ها (مانند توزیع‌های مختلف لینوکس) را نیز دارا می‌باشد

ابزار تولید ترافیک ناهنجار: Zenmap (نسخه ویندوزی Nmap) جهت اجرای اسکن‌های تهاجمی (Intense Scan).

پیکربندی شبکه: استفاده از سوییچ مجازی VMnet4 جهت برقراری ارتباط ایزوله بین سیستم مهاجم و هدف و شنود ترافیک توسط Scapy.

پیوست د: فرمول‌های تکمیلی آماری

در این بخش، فرمول محاسبه خطای استاندارد برآورد (S_e) که مبنای تعیین آستانه بحرانی بوده، به تفصیل آورده شده است:

$$S_e = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n-2}}$$

که در آن :

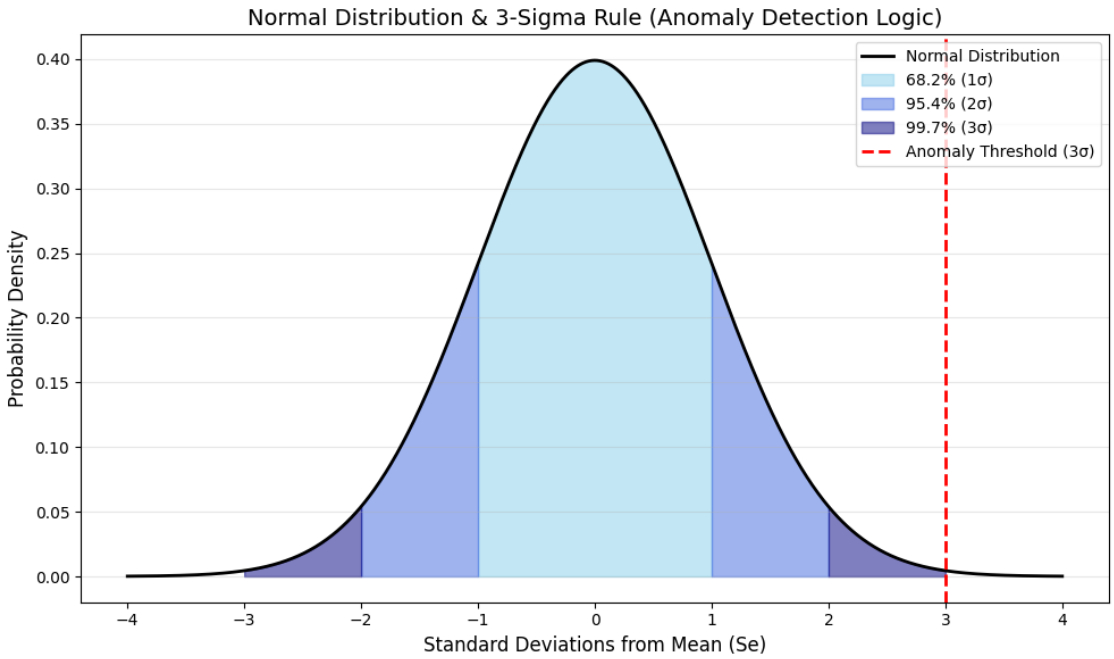
y_i : تعداد واقعی بسته‌های مشاهده شده در هر ثانیه.

$\hat{y}_i$: تعداد بسته‌های پیش‌بینی شده توسط مدل رگرسیون.

n: تعداد کل نمونه‌های زمانی در ترافیک *Baseline*.

$n-2$: درجه آزادی برای مدل رگرسیون خطی ساده.

تحلیل آماری آستانه :آستانه $3S_e$ بر اساس قانون تجربی توزیع نرمال انتخاب شده است تا با پوشش دادن ٪۹۹.۷ از نوسانات تصادفی ترافیک عادی، احتمال هشدار کاذب (False Positive) به کمتر از ٪۰.۳ برسد.



عکس د-۱: مدل توزیع احتمال خطا و مرز تشخیص نفوذ بر اساس قاعده ۳-سیگما