

باسمه تعالی
درس: آزمایشگاه امنیت شبکه



نام و نام خانوادگی: تینا توکلی

شماره دانشجویی: 9922762220

شماره تمرین: 07

1. معماری و عملکرد کلی DPI را شرح دهید.

DPI (بازرسی بسته عمیق) امنیت شبکه را با بازرسی محتوای بسته های داده ای که در یک شبکه حرکت می کنند، بهبود می بخشد. تجزیه روند:

1. Mirrored Traffic: ترافیک شبکه از یک سوئیچ یا دستگاه شبکه دیگر به یک دستگاه DPI منعکس می شود. این ترافیک آینه ای شامل کپی هایی از تمام بسته های داده ای است که در شبکه حرکت می کنند.
2. Preprocessing: دستگاه DPI داده های گرفته شده را برای تجزیه و تحلیل آماده می کند. این ممکن است شامل فیلتر کردن نویز یا داده های نامربوط باشد.
3. Analyze/vAnalyze: موتور DPI بسته های داده را بازرسی می کند. می تواند به دنبال محتوای مخرب باشد، برنامه های مورد استفاده را شناسایی کند یا مطابقت با ختمشی های شرکت را بررسی کند.
4. Threat Detection/DPV: موتور DPI از امضاها یا سایر روش های تشخیص برای شناسایی تهدیدها استفاده می کند. این تهدیدها می تواند بدافزار، نفوذ یا استخراج غیرمجاز داده باشد.
5. مانیتورینگ دستگاه DPI می تواند یافته های خود را به یک GUI یا سیستم نظارتی گزارش دهد. این به مدیران شبکه اجازه می دهد تا ببینند در شبکه آنها چه اتفاقی می افتد و اقدامات لازم را انجام دهند.

2. ابزارها و برنامه های DPI برای امنیت شبکه را لیست کرده، عملکرد، مزایا و معایب آنها را با هم مقایسه کنید

ابزار	مزایا	معایب	عملکرد
SolarWinds Network Performance Monitor (NPM)	- نظارت جامع بر شبکه فراتر از DPI. - تجزیه و تحلیل گسترده برنامه و پروتکل. - کشف و نگاشت خودکار دستگاه های شبکه.	- می تواند برای استقرارهای کوچکتر گران باشد.	- تعادل خوبی بین امنیت و عملکرد ارائه می دهد. - بر بینش های بلندرنج تمرکز دارد.
ManageEngine DataSecurity Plus	- مجموعه بزرگی از ویژگی های محافظت از داده ها از جمله DPI. - نظارت بر فعالیت کاربر و جلوگیری از از دست دادن داده ها.	- تنظیم و مدیریت پیچیده برای برخی از ویژگی ها.	- عملکرد می تواند بسته به ویژگی های انتخابی متفاوت باشد.
ManageEngine NetFlow Analyzer	- گزینه مقرون به صرفه با قابلیت های DPI داخلی (Network Packet Sensor). - تجزیه و تحلیل مبتنی بر جریان برای نظارت کارآمد ترافیک شبکه.	- شناسایی برنامه محدود در مقایسه با برخی از ابزارها.	- برای عملکرد با تجزیه و تحلیل جریان بهینه شده است.
Site24x7 Network Monitoring	- راه حل مبتنی بر ابر با قابلیت های DPI داخلی. - آسان برای استقرار و مدیریت.	- گزینه های سفارشی سازی محدود در مقایسه با ابزارهای محلی.	- معماری مبتنی بر ابر می تواند بر عملکرد شبکه های بزرگ تأثیر بگذارد.
nDPI	- منبع باز و رایگان برای استفاده. - بسیار قابل تنظیم برای نیازهای خاص.	- نیاز به تخصص فنی برای راه اندازی و مدیریت دارد. - پشتیبانی محدود در مقایسه با ابزارهای تجاری.	- عملکرد به سخت افزار و پیکربندی بستگی دارد.

نکته:

Paessler PRTG: ارائه اسکن بسته با قابلیت های DPI، مناسب برای نیازهای اولیه.

Netifyd: راه حل DPI مبتنی بر ابر با تمرکز بر نظارت بر فعالیت کاربر.

AppNeta: ابزار نظارت بر عملکرد شبکه با تشخیص برنامه پیشرفته با استفاده از DPI.

انتخاب ابزار DPI مناسب به نیازهای خاص است. عواملی مانند:

اندازه و پیچیدگی شبکه: شبکه های بزرگتر ممکن است به ابزارهای قدرتمندتری نیاز داشته باشند.

بودجه: گزینه های منبع باز در دسترس هستند، اما ابزارهای تجاری ویژگی ها و پشتیبانی بیشتری ارائه می دهند.

تخصص فنی: استفاده از برخی از ابزارها آسان تر از سایر ابزارها است.

کارکردهای مورد نظر: اولویت بندی ویژگی هایی مانند شناسایی برنامه، تشخیص تهدید یا جلوگیری از از دست دادن داده ها.

3. معماری و عملکرد ابزار Snort را تحلیل و بررسی کنید.

Snort یک سیستم تشخیص نفوذ شبکه (IDS) رایگان و منبع باز است که می تواند تجزیه و تحلیل ترافیک در زمان واقعی و ثبت بسته ها را در شبکه های IP انجام دهد.

خلاصه ای از نحوه عملکرد سنتی Snort:

گیرنده بسته: Snort از یک کتابخانه گیرنده بسته (مانند libpcap) برای دریافت بسته ها از شبکه استفاده می کند. این کار می تواند در حالت promiscuous انجام شود، جایی که تمام ترافیک شبکه را مشاهده می کند، یا در حالت غیر promiscuous، جایی که فقط ترافیک هدایت شده به یک میزبان یا شبکه خاص را مشاهده می کند.

پیش پردازنده ها: قبل از ورود به موتور تشخیص، بسته های ضبط شده پیش پردازش می شوند. پیش پردازنده ها می توانند داده ها را عادی سازی کنند (به عنوان مثال، تبدیل همه کاراکترها به حروف کوچک)، پروتکل ها را رمزگشایی کنند (به عنوان مثال، اطلاعات را از هدرهای HTTP استخراج کنند) و سایر وظایف را برای آماده سازی داده ها برای تجزیه و تحلیل انجام دهند.

موتور تشخیص: موتور تشخیص بسته های پیش پردازش شده را با مجموعه قوانین مقایسه می کند. این قوانین الگوهای ترافیک مخرب را تعریف می کنند. اگر بسته ای با یک قانون مطابقت داشته باشد، Snort یک هشدار تولید می کند.

تجزیه کننده قوانین:

قوانین Snort را می خواند و تفسیر می کند. نحوه را تأیید می کند، اطلاعات مربوطه را استخراج می کند و قوانینی را برای پردازش کارآمد توسط موتور تشخیص آماده می کند. تجزیه کننده تضمین می کند که قوانین به درستی ساختار یافته و توسط Snort قابل درک هستند.

ثبت و هشدار:

Snort قابلیت های ثبت و هشدار را برای اطلاع مدیران در مورد حوادث احتمالی امنیتی فراهم می کند. هنگامی که موتور تشخیص یک بسته مشکوک یا یک قانون مطابقت را شناسایی می کند، یک هشدار تولید می کند. هشدارها را می توان به مقاصد مختلفی مانند خروجی کنسول، فایل های گزارش، پایگاه های داده یا حتی از طریق ایمیل یا مکانیسم های اعلان دیگر ارسال کرد.

ماژول های خروجی:

ماژول های خروجی مسئول رسیدگی به هشدارها و گزارش های تولید شده توسط Snort هستند. آنها تعیین می کنند که هشدارها باید کجا و در چه قالبی ارسال شوند. Snort از ماژول های خروجی مختلف پشتیبانی می کند و امکان ادغام با سیستم های مختلف، مانند سرورهای syslog، پایگاه های داده، سیستم های SIEM (اطلاعات امنیتی و مدیریت رویداد) و سایر ابزارهای امنیتی شبکه را فراهم می کند.

ذخیره سازی:

Snort می تواند ترافیک شبکه گرفته شده را برای تجزیه و تحلیل بعدی ثبت کند. بسته به پیکربندی می تواند بسته ها یا فقط ابرداشته های آنها را ذخیره کند. این ویژگی به ویژه برای تحقیقات پزشکی قانونی، اهداف انطباق، یا تجزیه و تحلیل پس از حادثه مفید است.

پیکربندی:

Snort یک رابط خط فرمان برای پیکربندی و مدیریت سیستم فراهم می کند. CLI به مدیران اجازه می دهد تا قوانین را تعریف کنند، گزینه های پیش پردازش را مشخص کنند، ماژول های خروجی را پیکربندی کنند، و تنظیمات مختلف را برای انطباق رفتار Snort با محیط شبکه تنظیم کنند.

سیستم ممکن است قادر به انجام موارد زیر باشد:

توانایی های Snort را افزایش دهید: افزونه هوشمند می تواند به طور بالقوه توانایی Snort را برای شناسایی تهدیدات جدید و نوظهوری که لزوماً توسط قوانین موجود پوشش داده نمی شوند، گسترش دهد.

بهبود کارایی: می توان از یادگیری ماشین برای طبقه بندی بسته ها و اولویت بندی مواردی که به احتمال زیاد مخرب هستند استفاده کرد. این می تواند به کاهش تعداد مثبت کاذب تولید شده توسط Snort کمک کند.

البته:

Snort یک ابزار قدرتمند است، اما راه اندازی و نگهداری آن می تواند پیچیده باشد. مجموعه قوانین باید به طور مرتب به روز شود تا با آخرین تهدیدات مطابقت داشته باشد.

Snort می تواند از نظر منابع سنگین باشد. می تواند بر عملکرد شبکه، به ویژه در شبکه های پرسرعت فشار وارد کند.

Snort یک سیستم تشخیص است، نه یک سیستم پیشگیری. می تواند شما را از فعالیت های مشکوک آگاه کند، اما نمی تواند از وقوع حملات جلوگیری کند.

4. Snort و IPTables را جنبه های مختلف مقایسه کنید

IPTables و Snort هر دو ابزار مهم برای امنیت شبکه هستند، اما اهداف متفاوتی دارند.

تفاوت های کلیدی:

عملکرد:

IPTables: فایروال - کنترل دسترسی به شبکه با فیلتر کردن ترافیک ورودی و خروجی بر اساس آدرس های IP، پورت ها، پروتکل ها و سایر معیارها. می تواند ترافیک ناخواسته را مسدود کند یا ترافیک خاصی را مجاز کند.

Snort: سیستم تشخیص نفوذ (IDS) - ترافیک شبکه را برای فعالیت مشکوک بر اساس قوانین و امضاهای از پیش تعریف شده رصد می کند. می تواند در صورت شناسایی تهدیدات بالقوه به مدیران هشدار دهد، اما به طور مستقیم ترافیک را مسدود نمی کند.

کارکرد:

IPTables: ساده تر و پیکربندی آن آسان تر است. فیلتر کردن و کنترل ترافیک اولیه را ارائه می دهد.

Snort: پیچیده تر است و نیاز به دانش مجموعه قوانین و امضاها دارد. بازرسی عمیق تر محتوای ترافیک شبکه و انحرافات پروتکل را ارائه می دهد.

استقرار:

IPTables: به طور معمول در دروازه یا فایروال شبکه اجرا می شود.

Snort: می تواند روی یک سنسور IDS اختصاصی یا در کنار IPTables در همان دستگاه مستقر شود.

عملکرد:

IPTables: سبک وزن است و تأثیر کمی بر عملکرد شبکه دارد.

Snort: می تواند از نظر منابع سنگین باشد، به خصوص با مجموعه قوانین پیچیده، که ممکن است بر عملکرد شبکه تأثیر بگذارد.

هشدارها و اقدامات:

IPTables: عمدتاً برای مسدود کردن ترافیک استفاده می شود. قابلیت های هشدار محدودی دارد.

Snort: برای فعالیت مشکوک هشدار ایجاد می کند و به مدیران اجازه می دهد تا بررسی و اقدام کنند. می توان آن را با سایر ابزارهای امنیتی برای پاسخ های خودکار ادغام کرد.

منحنی یادگیری:

IPTables: یادگیری و مدیریت آن آسان تر است.

Snort: برای پیکربندی قوانین و تفسیر هشدارها به دانش فنی بیشتری نیاز دارد.

به طور کلی:

IPTables: خط مقدم دفاع، کنترل دسترسی به شبکه و جلوگیری از ترافیک غیرمجاز.

Snort: تجزیه و تحلیل عمیق تری را ارائه می دهد و به شناسایی تهدیدات بالقوه ای که ممکن است از فایروال عبور کنند کمک می کند.

چه زمانی از کدام استفاده کنیم:

از **IPTables** برای کنترل دسترسی اولیه و فیلتر کردن ترافیک استفاده کنید.

از **Snort** در کنار **IPTables** برای تشخیص و تجزیه و تحلیل پیشرفته تهدید استفاده کنید.

جدول:

ویژگی	IPTables	Snort
عملکرد	فایروال - کنترل دسترسی به شبکه	IDS - نظارت بر ترافیک برای فعالیت مشکوک
کارکرد	فیلتر کردن و کنترل ترافیک اولیه	بازرسی عمیق بسته ها، تشخیص مبتنی بر امضا
استقرار	دستگاه دروازه/فایروال	سنسور اختصاصی یا در کنار IPTables
عملکرد	سبک وزن، تأثیر کم	منابع سنگین، تأثیر بالقوه بر عملکرد
هشدارها و اقدامات	عمدتاً برای مسدود کردن ترافیک استفاده می شود	هشدار برای بررسی و اقدامات بالقوه ایجاد می کند
منحنی یادگیری	یادگیری و مدیریت آسان تر	نیاز به دانش فنی بیشتر برای پیکربندی قوانین
مورد استفاده	خط مقدم دفاع، کنترل دسترسی	تشخیص و تجزیه و تحلیل پیشرفته تهدید

5. نقش DAQ در فرآیند نصب ابزار Snort چیست؟

Snort برای انجام فرآیند تشخیص نفوذ به کتابخانه ای به نام **DAQ (Data Acquisition)** متکی است. **DAQ** به عنوان یک لایه انتزاع عمل می کند و به **Snort** اجازه می دهد تا با کتابخانه های مختلف گیرنده بسته در سیستم عامل های مختلف ارتباط برقرار کند.

خلاصه ای از نقش **DAQ** در نصب **Snort**:

DAQ: Provides Flexibility: به **Snort** اجازه می دهد تا با کتابخانه های مختلف گیرنده بسته مانند **libpcap** (رایج در لینوکس) یا **WinPcap** (در ویندوز استفاده می شود) کار کند. این فرآیند نصب را ساده می کند زیرا **Snort** نیازی به کامپایل شدن برای هر کتابخانه خاص ندارد.

DAQ: Abstraction Layer: به عنوان واسطه ای بین **Snort** و کتابخانه گیرنده بسته زیرین عمل می کند. **Snort** صرف نظر از کتابخانه خاص مورد استفاده، با استفاده از یک **API** سازگار با **DAQ** ارتباط برقرار می کند. این امر کد **Snort** را قابل حمل تر و نگهداری آن را آسان تر می کند.

عملکرد: DAQ فراتر از گیرنده بسته پایه، قابلیت هایی را ارائه می دهد. می تواند وظایفی مانند تنظیم فیلترهای گیرنده، مشخص کردن رابط های شبکه و بافر کردن بسته ها برای پردازش توسط Snort را انجام دهد.

فرض کنید Snort یک نگهبان امنیتی است که باید فعالیت را در یک ساختمان (شبکه) رصد کند. DAQ به عنوان یک سیستم کارت کلید جهانی (لایه انتزاع) عمل می کند که به نگهبان (Snort) اجازه می دهد تا به سیستم های ورود درب مختلف (کتابخانه های گیرنده بسته) در ساختمان های مختلف (سیستم عامل ها) دسترسی داشته باشد. این امر تضمین می کند که نگهبان می تواند فعالیت را به طور کارآمد رصد کند، صرف نظر از سیستم امنیتی خاص موجود در محل.

یعنی به طور خلاصه، DAQ به طور مستقیم در عملکرد اصلی تشخیص نفوذ Snort دخیل نیست. اما با فعال کردن Snort برای تعامل با مکانیسم های گیرنده بسته زیرین در سیستم های مختلف، نقش مهمی ایفا می کند، نصب را ساده می کند و قابلیت حمل را ارتقا می دهد.

بخش عملی:

نصب پکیج ها در ماشین DUT:

```
sudo apt-get update
sudo apt-get upgrade

sudo apt-get install -y build-essential libpcap-dev libpcap3-dev libdumbnet-dev bison flex zlib1g-dev
snort

sudo systemctl enable snort

sudo systemctl start snort

...

sudo systemctl status snort

sudo journalctl -u snort

...
```

برای نصب DAQ:

```
cd /usr/src

sudo wget https://www.snort.org/downloads/snort/daq-2.0.7.tar.gz

sudo tar -xzf daq-2.0.7.tar.gz

cd daq-2.0.7

sudo ./configure

sudo make

sudo make install
```

برای configuration در `sudo nano /etc/snort/snort.conf`:

`ipvar HOME_NET any`

`ipvar EXTERNAL_NET any`

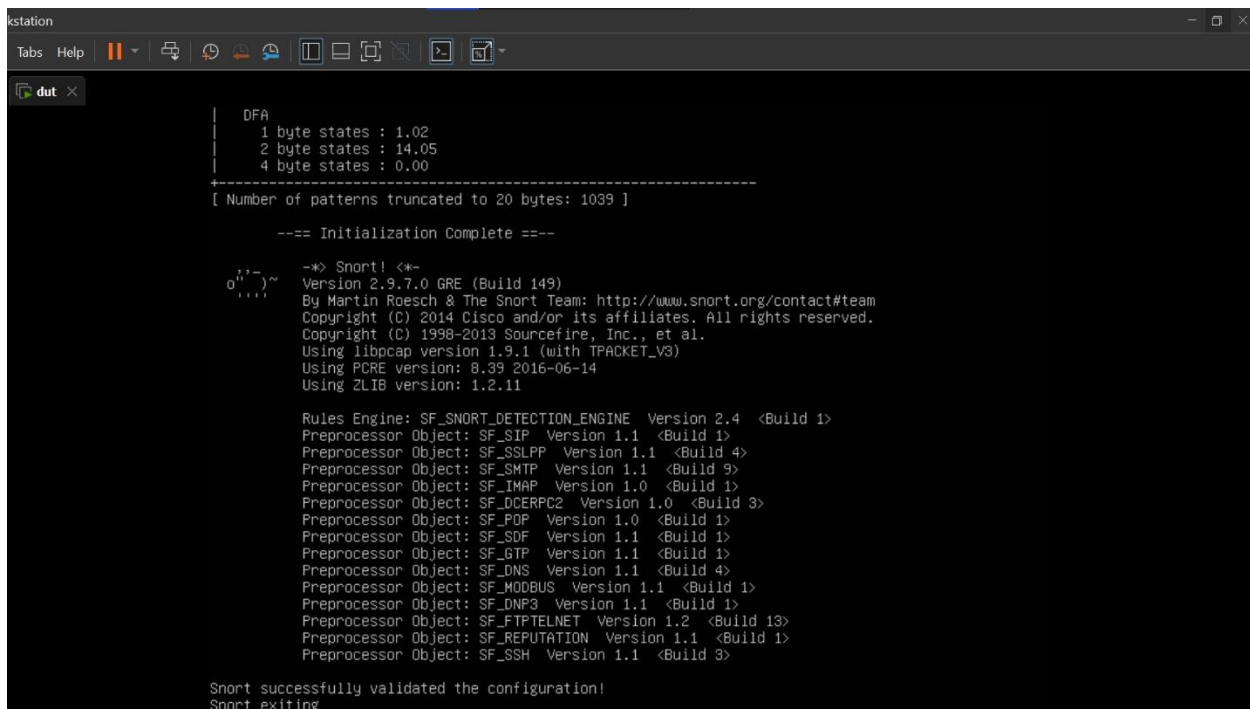
`output alert_fast: /var/log/snort/alerts`

`output alert_syslog: LOG_LOCAL1 LOG_ALERT`

`include /etc/snort/rules/local.rules`

برای اینکه درستی کانفیگ ها را چک کنیم:

`sudo snort -T -c /etc/snort/snort.conf`



```
kstation
DFA
  1 byte states : 1.02
  2 byte states : 14.05
  4 byte states : 0.00
-----
[ Number of patterns truncated to 20 bytes: 1039 ]

---- Initialization Complete ----

--> Snort! <*-
Version 2.9.7.0 GRE (Build 149)
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using libpcap version 1.9.1 (with TPACKET_V3)
Using PCRE version: 8.39 2016-06-14
Using ZLIB version: 1.2.11

Rules Engine: SF_SNORT_DETECTION_ENGINE Version 2.4 <Build 1>
Preprocessor Object: SF_SIP Version 1.1 <Build 1>
Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>
Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>
Preprocessor Object: SF_FOP Version 1.0 <Build 1>
Preprocessor Object: SF_SDF Version 1.1 <Build 1>
Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>

Snort successfully validated the configuration!
Snort exiting
```

و برای اضافه کردن قانون:

`#include $RULE_PATH/local.rules`

`sudo nano /etc/snort/rules/local.rules:`

`alert tcp any any -> any 80 (msg:"TCP traffic detected"; sid:1000001;)`

برای شنود کردن:

`sudo snort -A console -c /etc/snort/snort.conf -i ens37`

```
Reload thread started, thread 0x7f71eb24e700 (8251)
Decoding Ethernet
Set gid to 119
Set uid to 114

--- Initialization Complete ---

-*> Snort! <*-
Version 2.9.7.0 GRE (Build 149)
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using libpcap version 1.9.1 (with TPACKET_V3)
Using PCRE version: 8.39 2016-06-14
Using ZLIB version: 1.2.11

Rules Engine: SF_SNORT_DETECTION_ENGINE Version 2.4 <Build 1>
Preprocessor Object: SF_SIP Version 1.1 <Build 1>
Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>
Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>
Preprocessor Object: SF_POP Version 1.0 <Build 1>
Preprocessor Object: SF_SDF Version 1.1 <Build 1>
Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Commencing packet processing (pid=8242)
07/05-10:08:35.132149 [**] [1:1000001:0] TCP traffic detected [**] [Priority: 0] {TCP} 192.168.100.2:20 -> 192.168.101.2:80
07/05-10:08:35.132149 [**] [1:503:7] MISC Source Port 20 to <1024 [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.100.2:20 -> 192.168.101.2:80
```

Figure 1- برای یک فایل آلوده شنود کرده

```
DUT x IN x Out x
Memory: 142.1M
CGroup: /system.slice/snort.service
└─8476 /usr/sbin/snort -m 027 -D -d -l /var/log/snort -u snort -g snort -c /etc/snort/

Jul 05 10:14:06 dut snort[8476]: Preprocessor Object: SF_POP Version 1.0 <Build 1>
Jul 05 10:14:06 dut snort[8476]: Preprocessor Object: SF_SDF Version 1.1 <Build 1>
Jul 05 10:14:06 dut snort[8476]: Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Jul 05 10:14:06 dut snort[8476]: Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Jul 05 10:14:06 dut snort[8476]: Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Jul 05 10:14:06 dut snort[8476]: Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
Jul 05 10:14:06 dut snort[8476]: Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Jul 05 10:14:06 dut snort[8476]: Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
Jul 05 10:14:06 dut snort[8476]: Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Jul 05 10:14:06 dut snort[8476]: Commencing packet processing (pid=8476)
lines 1-20/20 (END)
root@dut:/home/tina# systemctl status snort
● snort.service - LSB: Lightweight network intrusion detection system
   Loaded: loaded (/etc/init.d/snort; generated)
   Active: active (running) since Fri 2024-07-05 10:14:05 UTC; 19s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 8450 ExecStart=/etc/init.d/snort start (code=exited, status=0/SUCCESS)
    Tasks: 2 (limit: 1010)
   Memory: 137.4M
   CGroup: /system.slice/snort.service
           └─8476 /usr/sbin/snort -m 027 -D -d -l /var/log/snort -u snort -g snort -c /etc/snort/

Jul 05 10:14:06 dut snort[8476]: Preprocessor Object: SF_SDF Version 1.1 <Build 1>
Jul 05 10:14:06 dut snort[8476]: Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Jul 05 10:14:06 dut snort[8476]: Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Jul 05 10:14:06 dut snort[8476]: Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Jul 05 10:14:06 dut snort[8476]: Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
Jul 05 10:14:06 dut snort[8476]: Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Jul 05 10:14:06 dut snort[8476]: Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
Jul 05 10:14:06 dut snort[8476]: Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Jul 05 10:14:06 dut snort[8476]: Commencing packet processing (pid=8476)
Jul 05 10:14:21 dut snort[8476]: [1:1000001:0] TCP traffic detected {TCP} 192.168.100.2:20 -> 192.168.101.2:80
lines 1-20/20 (END)
```

Figure 2- به جای اینکه از دستور بالا استفاده کنم از دستور `systemctl snort start` استفاده میکنم

برای ساخت Directory for Log Files:

```
sudo mkdir /var/log/snort/pcap_analysis
sudo chown snort:snort /var/log/snort/pcap_analysis
```

سپس در sudo nano /etc/rsyslog.conf:

```
local1.* @192.168.101.2:514
```

سپس:

```
sudo systemctl restart rsyslog
```

برای تست کردن :

```
logger -p local1.alert "Test alert from Snort"
```

در ماشینin:

```
sudo apt-get install tcpdump
sudo apt-get update
sudo apt-get install python3 python3-pip
pip3 install scapy
sudo tcpdump -i eth0 /path/to/traffic.pcap
```

dcapآلوده باید ایجاد کنیم:

برای این کار از scapy استفاده میکنیم و با کد پایتون زیر فایل مورد نظر را تولید میکنیم:

```
nano malicious_packet.py
```

```
'''
```

```
from scapy.all import *
```

```
ip = Ether()/IP(dst="192.168.101.2")
```

```
tcp = TCP(dport=80, flags="S")
```

```
payload = "\x41\x42\x43\x44" + "\x90" * 12 +
```

```
"\x31\xc0\xb0\x04\x31\xdb\xb3\x01\x68\x6e\x61\x6d\x65\x68\x75\x73\x65\x72\x89\xe1\x31\xd2\xb2\x0f\xcd\x80\x31\xc0\x31\xdb\x31\xc9\xb0\x3f\xcd\x80\x31\xc0\x31\xdb\xb0\x3f\x41\xcd\x80\x31\xc0\x31\xdb\xb0\x3f\x41\xcd\x80\x31\xc0\x31\xd2\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x31\xd2\xb0\x0b\xcd\x80"
```

```
packet = ip/tcp/Raw(load=payload)
```

```

pack=[packet]*99
filename = "malicious_packet.pcap"
wrpcap(filename, pack)
print(f"Packet saved to {filename}")
send(packet)
'''

```

سپس با دستور زیر فایل pcap آلوده را ران میکنیم:

```
sudo python3 malicious_packet.py
```

```

tcp = TCP(dport=80, flags="S")
payload = "\x41\x42\x43\x44" + "\x90" * 12 + "\x31\xc0\xb0\x04\x31\xdb\xb3\x01\x68\x6e\x61\x6d\x65\x20"

packet = ip/tcp/Raw(load=payload)
filename = "malicious_packet.pcap"
wrpcap(filename, packet)
print(f"Packet saved to {filename}")
send(packet)

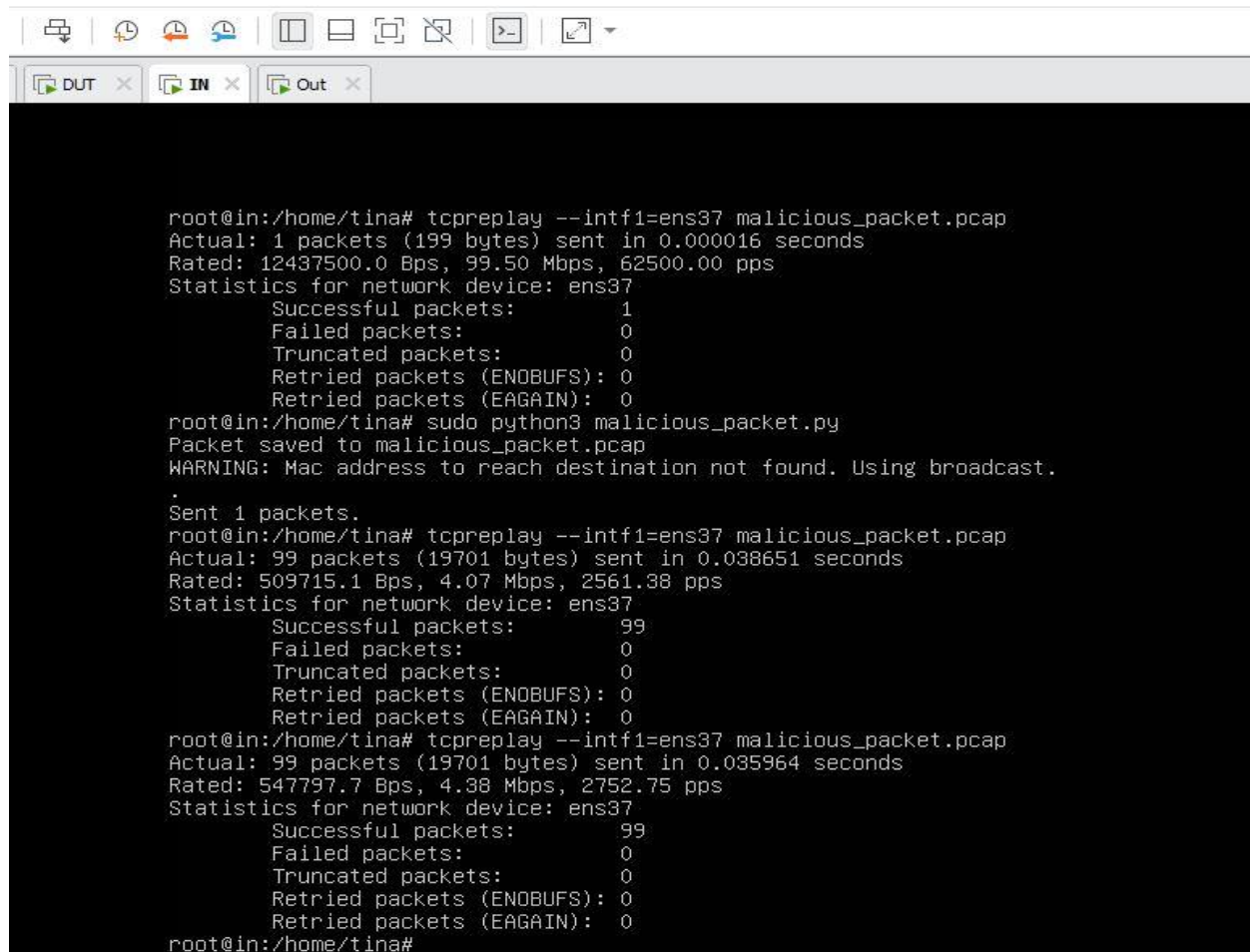
root@in:/home/tina# sudo python3 malicious_packet.py
Packet saved to malicious_packet.pcap
WARNING: Mac address to reach destination not found. Using broadcast.
Sent 1 packets.
root@in:/home/tina#

```

Figure 3-در ابتدا یکی می ساخت ولی بعد برای موارد ارزیابی به 99 مورد افزایش دادم

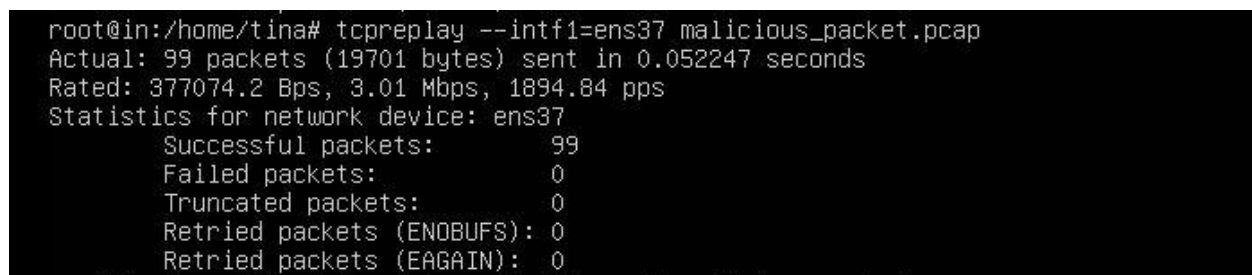
سپس:

```
sudo tcpreplay --intf1=ens37 malicious_packet.pcap
```

A screenshot of a terminal window with a dark background and white text. The window has three tabs at the top: 'DUT', 'IN', and 'Out'. The terminal shows a series of commands and their outputs. The first command is 'tcpdump --intf=ens37 malicious_packet.pcap', which outputs statistics for 1 packet. The second command is 'python3 malicious_packet.py', which outputs a warning about a missing MAC address and confirms 1 packet sent. The third command is 'tcpdump --intf=ens37 malicious_packet.pcap', which outputs statistics for 99 packets. The fourth command is 'tcpdump --intf=ens37 malicious_packet.pcap', which also outputs statistics for 99 packets. The terminal ends with the prompt 'root@in:/home/tina#'.

```
root@in:/home/tina# tcpdump --intf=ens37 malicious_packet.pcap
Actual: 1 packets (199 bytes) sent in 0.000016 seconds
Rated: 12437500.0 Bps, 99.50 Mbps, 62500.00 pps
Statistics for network device: ens37
  Successful packets:      1
  Failed packets:         0
  Truncated packets:      0
  Retried packets (ENOBUFS): 0
  Retried packets (EAGAIN): 0
root@in:/home/tina# sudo python3 malicious_packet.py
Packet saved to malicious_packet.pcap
WARNING: Mac address to reach destination not found. Using broadcast.
.
Sent 1 packets.
root@in:/home/tina# tcpdump --intf=ens37 malicious_packet.pcap
Actual: 99 packets (19701 bytes) sent in 0.038651 seconds
Rated: 509715.1 Bps, 4.07 Mbps, 2561.38 pps
Statistics for network device: ens37
  Successful packets:      99
  Failed packets:         0
  Truncated packets:      0
  Retried packets (ENOBUFS): 0
  Retried packets (EAGAIN): 0
root@in:/home/tina# tcpdump --intf=ens37 malicious_packet.pcap
Actual: 99 packets (19701 bytes) sent in 0.035964 seconds
Rated: 547797.7 Bps, 4.38 Mbps, 2752.75 pps
Statistics for network device: ens37
  Successful packets:      99
  Failed packets:         0
  Truncated packets:      0
  Retried packets (ENOBUFS): 0
  Retried packets (EAGAIN): 0
root@in:/home/tina#
```

Figure 4- قبل از اینکه در rsyslog تغییراتی انجام دهیم

A screenshot of a terminal window with a dark background and white text. The terminal shows the output of the 'tcpdump --intf=ens37 malicious_packet.pcap' command, displaying statistics for 99 packets sent. The output includes the actual number of packets, the rated bandwidth, and a detailed breakdown of packet statistics (successful, failed, truncated, etc.).

```
root@in:/home/tina# tcpdump --intf=ens37 malicious_packet.pcap
Actual: 99 packets (19701 bytes) sent in 0.052247 seconds
Rated: 377074.2 Bps, 3.01 Mbps, 1894.84 pps
Statistics for network device: ens37
  Successful packets:      99
  Failed packets:         0
  Truncated packets:      0
  Retried packets (ENOBUFS): 0
  Retried packets (EAGAIN): 0
```

Figure 5- بعد از اینکه داخل فایل rsyslog تغییرات را انجام دادم

در ماشین out:

در sudo nano /etc/rsyslog.conf:

local1.* /var/log/snort.log

module(load="imudp")

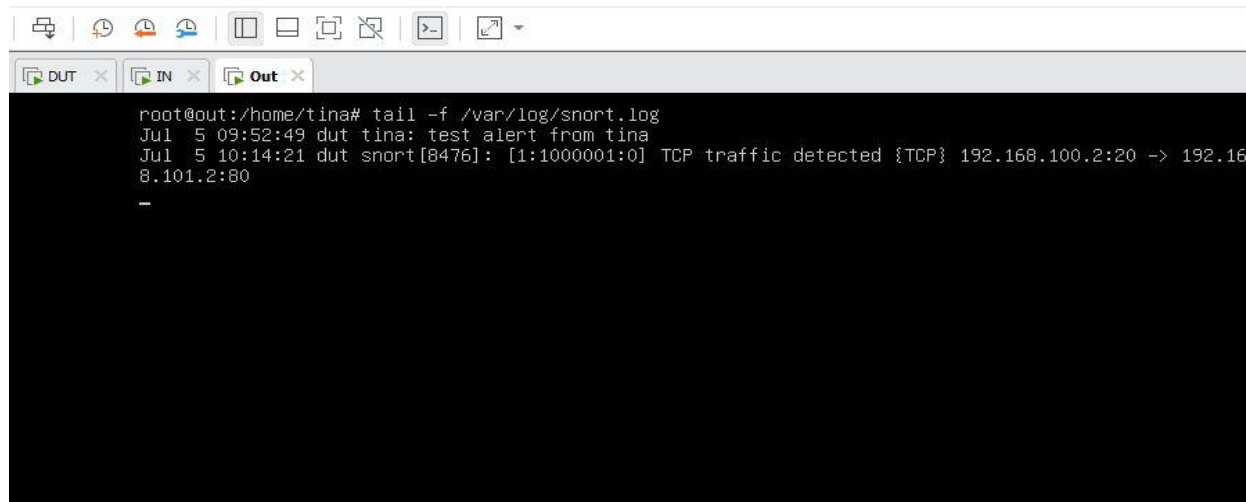
```
input(type="imudp" port="514")
```

و سپس :

```
sudo systemctl restart rsyslog
```

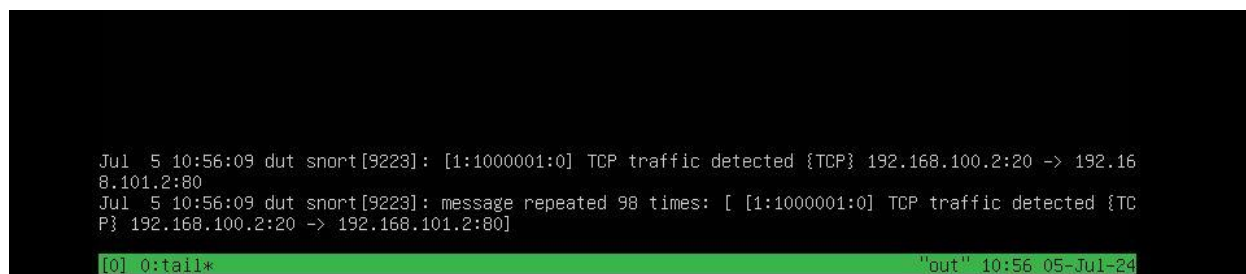
بررسی لاگ‌های دریافتی :

```
Tail -f /var/log/snort.log
```



```
root@out:/home/tina# tail -f /var/log/snort.log
Jul  5 09:52:49 dut tina: test alert from tina
Jul  5 10:14:21 dut snort[8476]: [1:1000001:0] TCP traffic detected {TCP} 192.168.100.2:20 -> 192.168.101.2:80
-
```

Figure 6-نمایش لاگهای فایل آلوده



```
Jul  5 10:56:09 dut snort[9223]: [1:1000001:0] TCP traffic detected {TCP} 192.168.100.2:20 -> 192.168.101.2:80
Jul  5 10:56:09 dut snort[9223]: message repeated 98 times: [ [1:1000001:0] TCP traffic detected {TCP} 192.168.100.2:20 -> 192.168.101.2:80]
[0] 0:tail* "out" 10:56 05-Jul-24
```

Figure 7- ارزیابی دقت -> نمایش لاگهای فایل آلوده (که از 99 فایل فقط 98 تا دریافت کرده)