گزارش هفتم

نظری

معماری و عملکرد کلی DPI

DPI یک روش پیشرفته برای بررسی و مدیریت ترافیک شبکه است. این روش فراتر از فیلترینگ و تحلیل اولیه بسته‌ها عمل می‌کند و محتوای کامل بسته‌ها را مورد بررسی قرار می دهد و به طور گسترده‌ای برای امنیت شبکه، مدیریت ترافیک، استخراج داده‌ها، و اجرای سیاست‌ها استفاده می‌شود. در زیر، شرحی از معماری و عملکرد کلی آن آمده است:

معماری:

- Packet Capture Engine: این قسمت، بسته های داده ای که در شبکه در حال حرکت هستند را برای تحلیل دریافت می کند.
- Pre-Processing Module:

وظیفه: پاکسازی و نرمال‌سازی بسته‌های جمع‌آوری‌شده برای آماده‌سازی آن‌ها برای بازرسی عمیق‌تر.

کارها: حذف داده‌های زائد، تجمیع مجدد بسته‌ها و رمزگشایی.

- Protocol Analysis Engine:

وظیفه: شناسایی و رمزگشایی پروتکل‌های استفاده‌شده توسط بسته‌ها.

کارها: تجزیه هدر پروتکل‌ها، استخراج محتوا و شناسایی ناهنجاری‌ها در استفاده از پروتکل.

- Pattern Matching Engine:

وظیفه: بازرسی محتوا در برابر مجموعه‌ای از الگوها یا امضاهای از پیش تعریف‌شده.

کارها: جستجوی امضاهای حملات شناخته‌شده، بدافزار یا الگوهای خاص داده.

- Behavior Analysis Module:

وظیفه: تحلیل رفتار ترافیک برای شناسایی ناهنجاری‌ها، تهدیدات ناشناخته یا تخلفات سیاستی.

کارها: تحلیل رفتاری، تکنیک‌های اکتشافی و یادگیری ماشینی.

- Decision Module:

وظیفه: تصمیم‌گیری در مورد اقدامی که باید براساس تحلیل انجام شود.

اقدامات: اجازه، مسدود کردن، محدود کردن سرعت یا هدایت ترافیک.

  - **مجاز کردن:** اگر محتوا ایمن تلقی شود، به بسته اجازه داده می شود به مسیر خود ادامه دهد.

- **مسدود کردن:** اگر محتوا مخرب باشد (به عنوان مثال، بدافزار)، بسته مسدود می شود تا از رسیدن به مقصد آن جلوگیری شود.
- **اولویت بندی:** انواع خاصی از ترافیک، مانند کنفرانس ویدیویی، می توانند برای اطمینان از ارائه روان اولویت بندی شوند.

- Policy Enforcement Engine:

وظیفه: اعمال سیاست‌های از پیش تعریف‌شده به ترافیک بررسی‌شده.

کارها: اجرای سیاست‌های امنیتی، کیفیت خدمات (QoS) و قوانین تطابق.

- Logging and Reporting Module:

وظیفه: ثبت رویدادها و تولید گزارش برای تحلیل و حسابرسی.

کارها: ثبت داده‌های ترافیک بررسی‌شده، تهدیدات شناسایی‌شده و اقدامات اجرای سیاست.

- Management and Control Interface:

وظیفه: فراهم کردن یک رابط برای مدیران برای پیکربندی و نظارت بر سیستم.DPI

کارها: داشبوردهای مبتنی بر وب، رابط‌های خط فرمان (CLI) یاAPI ها.

عملکرد کلی:

- طبقه‌بندی ترافیک:
فرآیند: DPI نوع برنامه یا خدمات مرتبط با ترافیک را با بررسی محتوای بسته‌ها شناسایی می‌کند.
موارد استفاده: مدیریت پهنای باند، اولویت‌بندی برنامه‌ها و محدودیت ترافیک.
- اجرای امنیت:
فرآیند: DPI تهدیدات امنیتی را با تحلیل محتوای بسته‌ها برای الگوهای مخرب شناسایی و کاهش می‌دهد.
موارد استفاده: شناسایی و پیشگیری از نفوذ، مسدود کردن بدافزار و جلوگیری از نشت داده‌ها.
- فیلتر محتوا:
فرآیند: DPI محتوای بسته‌ها را بررسی می‌کند تا داده‌های ناخواسته یا مضر را فیلتر کند.
موارد استفاده: مسدود کردن دسترسی به وب‌سایت‌های نامناسب، سانسور محتوا و اجرای تطابق.
- تطابق با سیاست:
فرآیند: DPI اطمینان می‌دهد که ترافیک شبکه با سیاست‌ها و مقررات سازمانی مطابقت دارد.
موارد استفاده: نظارت بر داده‌های حساس، اجرای سیاست‌های استفاده از داده و حسابرسی تطابق.
- کیفیت خدمات (QoS):
فرآیند: DPI می‌تواند ترافیک را براساس برنامه یا کاربر اولویت‌بندی کند تا عملکرد بهینه را تضمین کند.

موارد استفاده: اولویت‌بندی VoIP، استریم ویدئو و برنامه‌های حیاتی کسب و کار.

- استخراج داده:

فرآیند: DPI می‌تواند اطلاعات مفیدی از ترافیک برای تحلیل و هوش تجاری استخراج کند.

موارد استفاده: تحلیل رفتار کاربران، الگوهای استفاده و روندهای بازار.

- مدیریت ترافیک:

فرآیند: DPI به مدیریت منابع شبکه با کنترل جریان ترافیک براساس محتوا و زمینه کمک می‌کند.

موارد استفاده: توزیع بار، کنترل ازدحام و استفاده بهینه از پهنای باند.

ابزارها و برنامه های DPI و مقایسه عملکرد و مزایا و معایب

- **Suricata**

عملکرد:

سوریکاتا یک سیستم شناسایی و پیشگیری از نفوذ (IDS/IPS) با قابلیت DPI است.

این ابزار قابلیت بررسی ترافیک شبکه به صورت real time را با پردازش چند نخی دارد.

پشتیبانی از شناسایی بر اساس امضا، تجزیه و تحلیل پروتکل و استخراج فایل را دارد.

مزایا:

متن باز: توسعه توسط جامعه با حمایت فعال و به‌روزرسانی‌های مستمر.

چند نخی: پردازش بهینه شبکه‌های با سرعت بالا.

قابلیت توسعه: پشتیبانی از قوانین سفارشی و پلاگین‌ها.

معایب:

پیچیدگی: نیاز به تنظیمات و تنظیمات پیشرفته که نیاز به مهارت دارد.

مصرف منابع: استفاده بالای CPU و حافظه در برخی موارد.

نصب و پیکربندی: نیاز به راه‌اندازی و ادغام در ساختار شبکه موجود دارد.

- **Snort**

عملکرد:

دیگر یک IDS/IPS متن‌باز با قابلیت DPI.

استفاده از شناسایی بر اساس امضا، تجزیه و تحلیل پروتکل و جستجوی محتوا.

پشتیبانی از نظارت و مسدود کردن ترافیک بر اساس قوانین قابل تنظیم.

مزایا:

رشد شده: با جامعه کاربری بزرگ و پشتیبانی فعال از توسعه‌دهندگان.

بر اساس قوانین: قابلیت تنظیم قوانین برای نیازهای خاص امنیت شبکه.

ادغام: عملکرد خوب با ابزارها و SIEMs امنیتی.

معایب:

بر اساس امضا: ممکن است به اتهاماتی که بدون تجدید نظر معطوف شود، از دست بروید.

عملکرد: پردازش شبکه با سرعت بالا ممکن است نیازمند سخت‌افزار بهینه باشد.

پیچیدگی: مانند سوریکاتا، پیکربندی و مدیریت پیچیده می‌تواند باشد.

- **Zeek**

عملکرد:

ابزار نظارت امنیت شبکه با قابلیت اسکریپت‌زنی قدرتمند.

تجزیه و تحلیل ترافیک شبکه به صورت زمان‌واقع برای تولید لاگ‌ها و فراداده‌ها.

تمرکز بر روی دید کلی از شبکه، تجزیه و تحلیل پروتکل و تشخیص الگوی ترافیک.

مزایا:

آگاهی از پروتکل: درک عمیق از پروتکل‌های شبکه برای تجزیه و تحلیل دقیق.

قابلیت اسکریپت‌زنی: زبان اسکریپت‌نویسی انعطاف‌پذیر (BroScript) برای سفارشی‌سازی.

مقیاس‌پذیری: مدیریت ترافیک زیاد و مقیاس خوب در محیط‌های توزیع شده.

معایب:

منحنی یادگیری: نیاز به آشنایی با BroScript برای سفارشی‌سازی.

مصرف منابع: مصرف بالای CPU و فضای ذخیره‌سازی به دلیل لاگ‌گذاری دقیق.

نصب و راه‌اندازی: نیاز به ادغام در محیط‌های موجود شبکه.

- **Cisco Firepower**

عملکرد:

راه‌حل تجهیزات امنیتی تجاری با قابلیت DPI.

ارائه امکانات پیشرفته تشخیص و پیشگیری از تهدید.

یکپارچه‌سازی با سیستم اکوسیستم سیسکو برای امنیت جامع شبکه.

مزایا:

مناسب برای اینترپرایز: امکانات امنیتی قوی برای سازمان‌های بزرگ.

مدیریت یکپارچه: مدیریت مرکزی با Cisco Firepower Management Center (FMC).

اتوماسیون: پشتیبانی از اتوماسیون و ارکستراسیون برای عملیات امنیتی.

معایب:

هزینه: هزینه‌های پرداختی و نیازهای سخت‌افزاری گران.

قفل کردن تامین: بهره‌مندی اصولی از مزایای ادغام با محصولات سیسکو.

پیچیدگی: نیاز به راه‌اندازی و پیکربندی برای نصب‌ونصب‌های کوچک‌تر.

معماری و عملکرد کلی Snort

Snort یک سیستم رایگان و متن باز تشخیص نفوذ شبکه (NIDS) و پیشگیری از نفوذ (IPS) است که به طور گسترده برای تجزیه و تحلیل ترافیک در زمان واقعی و امنیت شبکه استفاده می شود. در اینجا توضیحی از معماری آن و نحوه عملکرد آن آمده است:

معماری:

Snort به طور معمول در یک معماری تک لایه یا چند لایه عمل می کند:

تک لایه: همه پردازش ها روی یک ماشین انجام می شود. راه اندازی این روش ساده تر است، اما برای شبکه های پر ترافیک عملکرد ضعیف تری دارد.

چند لایه (مدل حسگر و کارگر): گرفتن ترافیک (حسگر) و تجزیه و تحلیل (کارگر) در ماشین های مختلف جدا شده است. این باعث بهبود مقیاس پذیری و عملکرد برای شبکه های بزرگتر می شود.

در اینجا اجزای کلیدی Snort، صرف نظر از مدل استقرار، آورده شده است:

Packet Capture Engine: این قسمت بسته های داده در حال حرکت در شبکه را دریافت می کند.

Rule Interpreter: بسته های capture شده را با مجموعه قوانین تعریف شده توسط کاربر تجزیه و تحلیل می کند. این قوانین الگوها یا ویژگی های ترافیک مخرب را مشخص می کنند.

Detection Engine: بر اساس تطابق قوانین، موتور تشخیص تعیین می کند که آیا یک بسته مشکوک است یا خیر.

Output Engine: Snort می تواند بر اساس نتایج تشخیص اقدامات مختلفی انجام دهد. این موارد ممکن است شامل موارد زیر باشد:

ثبت وقایع: ضبط اطلاعات مربوط به بسته های مشکوک برای تجزیه و تحلیل بعدی.

هشدار دادن: ارسال اعلان به پرسنل امنیتی در مورد تهدیدات بالقوه.

مسدود کردن بسته (حالت IPS): مسدود کردن فعالانه بسته های مخرب برای جلوگیری از رسیدن آنها به مقصد (نیازمند پیکربندی اضافی).

عملکرد:

دریافت بسته: Snort با استفاده از پیش پردازشگر ترافیک شبکه را دریافت می کند.

تطابق با قوانین: بسته های capture شده توسط مفسر قوانین با مجموعه قوانین تعریف شده توسط کاربر مقایسه می شوند.

تشخیص: موتور تشخیص بسته های مشکوک را بر اساس تطابق قوانین شناسایی می کند.

خروجی: Snort بر اساس نتایج تشخیص، اقداماتی مانند ثبت وقایع، هشدار دادن یا مسدود کردن (حالت IPS) انجام می دهد.

مقایسه Snort و Iptables

Snort و Iptables هر دو ابزار مهم برای امنیت شبکه هستند، اما اهداف متفاوتی دارند:

Snort:

عملکرد: سیستم تشخیص نفوذ (IDS) و سیستم پیشگیری از نفوذ (IPS)

تمرکز: تجزیه و تحلیل ترافیک شبکه برای شناسایی فعالیت های مشکوک و تهدیدات بالقوه.

روش: از یک سیستم مبتنی بر قوانین برای مقایسه الگوهای ترافیک با قوانین از پیش تعریف شده برای حملات شناخته شده یا رفتارهای مخرب استفاده می کند.

قابلیت ها:

طیف وسیعی از تهدیدات، از جمله بدافزار، کرم ها و اسکن های شبکه را تشخیص می دهد.

تجزیه و تحلیل ترافیک شبکه را به صورت real-time ارائه می دهد.

قابل تنظیم برای ثبت فعالیت های مشکوک، ارسال هشدار یا حتی مسدود کردن ترافیک مخرب (حالت IPS) است.

محدودیت ها:

برای اینکه در برابر تهدیدات در حال تحول موثر باشد، نیاز به نگهداری و به روز رسانی مداوم مجموعه قوانین خود دارد.

تجزیه و تحلیل بسته ها می تواند توان پردازش را مصرف کند و بر عملکرد شبکه تأثیر بگذارد.

ممکن است مثبت های کاذب ایجاد کند و ترافیک عادی را به عنوان مشکوک شناسایی کند.

Iptables:

عملکرد: فایروال

تمرکز: کنترل ترافیک ورودی و خروجی شبکه بر اساس قوانین از پیش تعریف شده.

روش: ترافیک را بر اساس معیارهایی مانند آدرس های IP مبدا و مقصد، پورت ها و پروتکل ها فیلتر می کند.

قابلیت ها:

ترافیک ناخواسته را بر اساس قوانین تعریف شده توسط کاربر مسدود می کند.

به انواع خاصی از ترافیک مانند وبگردی یا ایمیل اجازه می دهد.

کنترل دقیق بر دسترسی به شبکه را ارائه می دهد.

محدودیت ها:

عمدتا بر مسدود کردن ترافیک بر اساس قوانین از پیش تعریف شده تمرکز دارد، نه تشخیص تهدیدات در زمان واقعی.

توانایی محدود برای شناسایی و تجزیه و تحلیل محتوای ترافیک شبکه.

قابلیت های پیشرفته تشخیص تهدید مانند Snort را ارائه نمی دهد.

نقش DAQ در فرایند نصب ابزار Snort

در Snort، DAQ مخفف Data Acquisition Library (کتابخانه دستیابی داده) است. DAQ با عمل به عنوان یک لایه انتزاعی برای گرفتن بسته های شبکه، نقش مهمی در نصب و عملکرد Snort ایفا می کند. در اینجا نحوه قرارگیری DAQ در تصویر آمده است:

Snort و گرفتن بسته:

خود Snort مستقیماً بسته های شبکه را دریافت نمی کند. این ابزار برای انجام این کار به کتابخانه های خارجی متکی است.

قبلا Snort از کتابخانه libpcap برای گرفتن بسته استفاده می کرد. این بدان معنا بود که توسعه دهندگان Snort مجبور بودند کدهایی را به طور خاص برای عملکردهای libpcap بنویسند.

DAQ به عنوان یک لایه انتزاعی:

کتابخانه DAQ به عنوان یک لایه میانی بین Snort و کتابخانه های اساسی گرفتن بسته (مانند libpcap) عمل می کند.Snort بدون توجه به کتابخانه خاص گرفتن بسته ای که استفاده می شود، از طریق یک رابط استاندارد با DAQ ارتباط برقرار می کند.

این مزایای متعددی را ارائه می دهد:

انعطاف پذیری: Snort را می توان به راحتی با کتابخانه های مختلف گرفتن بسته بدون تغییر کد اصلی آن ادغام کرد. این به Snort اجازه می دهد تا با فناوری ها و سیستم عامل های جدید سازگار شود.

قابل حمل بودن: Snort با استفاده نکردن از یک کتابخانه خاص برای گرفتن بسته، در پلتفرم های مختلف قابل حمل تر می شود.

قابلیت نگهداری: توسعه دهندگان می توانند بدون نگرانی در مورد پیچیدگی های کتابخانه های مختلف گرفتن بسته، بر روی عملکردهای اصلی Snort (تجزیه و تحلیل ترافیک و تطابق قوانین) تمرکز کنند.

عملی

آدرس IP هر ماشین به صورت زیر می باشد.

```
homa@in:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:76:0a:7d brd ff:ff:ff:ff:ff:ff
    inet 192.168.88.159/24 brd 192.168.88.255 scope global dynamic ens33
       valid_lft 1592sec preferred_lft 1592sec
    inet 192.168.88.141/24 brd 192.168.88.255 scope global secondary dynamic ens33
       valid_lft 1596sec preferred_lft 1596sec
    inet6 fe80::20c:29ff:fe76:a7d/64 scope link
       valid_lft forever preferred_lft forever
3: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:76:0a:87 brd ff:ff:ff:ff:ff:ff
    inet 192.168.101.2/24 brd 192.168.101.255 scope global ens37
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe76:a87/64 scope link
       valid_lft forever preferred_lft forever
homa@in:~$
```

```
homa@dut:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:85:e5:4c brd ff:ff:ff:ff:ff:ff
    inet 192.168.88.133/24 brd 192.168.88.255 scope global dynamic ens33
       valid_lft 1442sec preferred_lft 1442sec
    inet 192.168.88.172/24 brd 192.168.88.255 scope global secondary dynamic ens33
       valid_lft 1443sec preferred_lft 1443sec
    inet6 fe80::20c:29ff:fe85:e54c/64 scope link
       valid_lft forever preferred_lft forever
3: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:85:e5:56 brd ff:ff:ff:ff:ff:ff
    inet 192.168.101.1/24 brd 192.168.101.255 scope global ens37
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe85:e556/64 scope link
       valid_lft forever preferred_lft forever
4: ens38: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:85:e5:60 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.1/24 brd 192.168.100.255 scope global ens38
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe85:e560/64 scope link
       valid_lft forever preferred_lft forever
```

```
homa@out:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:da:fc:03 brd ff:ff:ff:ff:ff:ff
    inet 192.168.88.153/24 brd 192.168.88.255 scope global dynamic ens33
       valid_lft 1750sec preferred_lft 1750sec
    inet 192.168.88.145/24 brd 192.168.88.255 scope global secondary dynamic ens33
       valid_lft 1755sec preferred_lft 1755sec
    inet6 fe80::20c:29ff:feda:fc03/64 scope link
       valid_lft forever preferred_lft forever
3: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:da:fc:0d brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.2/24 brd 192.168.100.255 scope global ens37
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:feda:fc0d/64 scope link
       valid_lft forever preferred_lft forever
```

با استفاده از دستورات زیر مسیریابی انجام شده است.

```
homa@in:~$ sudo ip route add 192.168.100.2 via 192.168.101.1
RTNETLINK answers: File exists
homa@in:~$ ip route show
default via 192.168.88.2 dev ens33
default via 192.168.88.2 dev ens33 proto dhcp src 192.168.88.141 metric 100
192.168.88.0/24 dev ens33 proto kernel scope link src 192.168.88.159
192.168.88.2 dev ens33 proto dhcp scope link src 192.168.88.141 metric 100
192.168.100.2 via 192.168.101.1 dev ens37
192.168.101.0/24 dev ens37 proto kernel scope link src 192.168.101.2
```

```
homa@out:~$ sudo ip route add 192.168.101.2 via 192.168.100.1
RTNETLINK answers: File exists
homa@out:~$ ip route show
default via 192.168.88.2 dev ens33
default via 192.168.88.2 dev ens33 proto dhcp src 192.168.88.145 metric 100
192.168.88.0/24 dev ens33 proto kernel scope link src 192.168.88.153
192.168.88.2 dev ens33 proto dhcp scope link src 192.168.88.145 metric 100
192.168.100.0/24 dev ens37 proto kernel scope link src 192.168.100.2
192.168.101.2 via 192.168.100.1 dev ens37
```

```
homa@dut:~$ sudo sysctl -w net.ipv4.ip_forward=1
[sudo] password for homa:
net.ipv4.ip_forward = 1
homa@dut:~$ sudo sysctl -p
```

و از طریق دستور زیر بررسی درستی این ارتباطات انجام شده است.

```
homa@in:~$ ping 192.168.100.2
PING 192.168.100.2 (192.168.100.2) 56(84) bytes of data.
64 bytes from 192.168.100.2: icmp_seq=1 ttl=63 time=1.37 ms
64 bytes from 192.168.100.2: icmp_seq=2 ttl=63 time=1.12 ms
64 bytes from 192.168.100.2: icmp_seq=3 ttl=63 time=0.952 ms
64 bytes from 192.168.100.2: icmp_seq=4 ttl=63 time=0.824 ms
^C
--- 192.168.100.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 0.824/1.066/1.371/0.204 ms
homa@in:~$
```

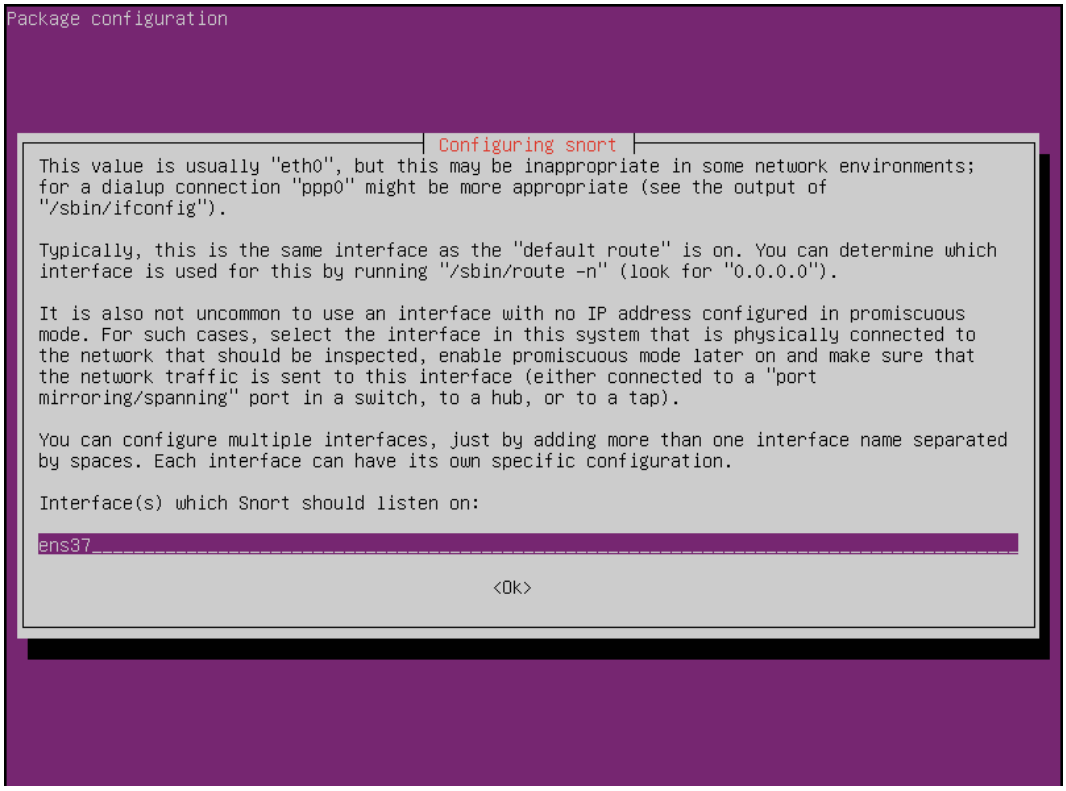
```
homa@out:~$ ping 192.168.101.2
PING 192.168.101.2 (192.168.101.2) 56(84) bytes of data.
64 bytes from 192.168.101.2: icmp_seq=1 ttl=63 time=1.35 ms
64 bytes from 192.168.101.2: icmp_seq=2 ttl=63 time=1.27 ms
64 bytes from 192.168.101.2: icmp_seq=3 ttl=63 time=1.05 ms
64 bytes from 192.168.101.2: icmp_seq=4 ttl=63 time=1.03 ms
^C
--- 192.168.101.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 1.030/1.176/1.353/0.139 ms
homa@out:~$ _
```

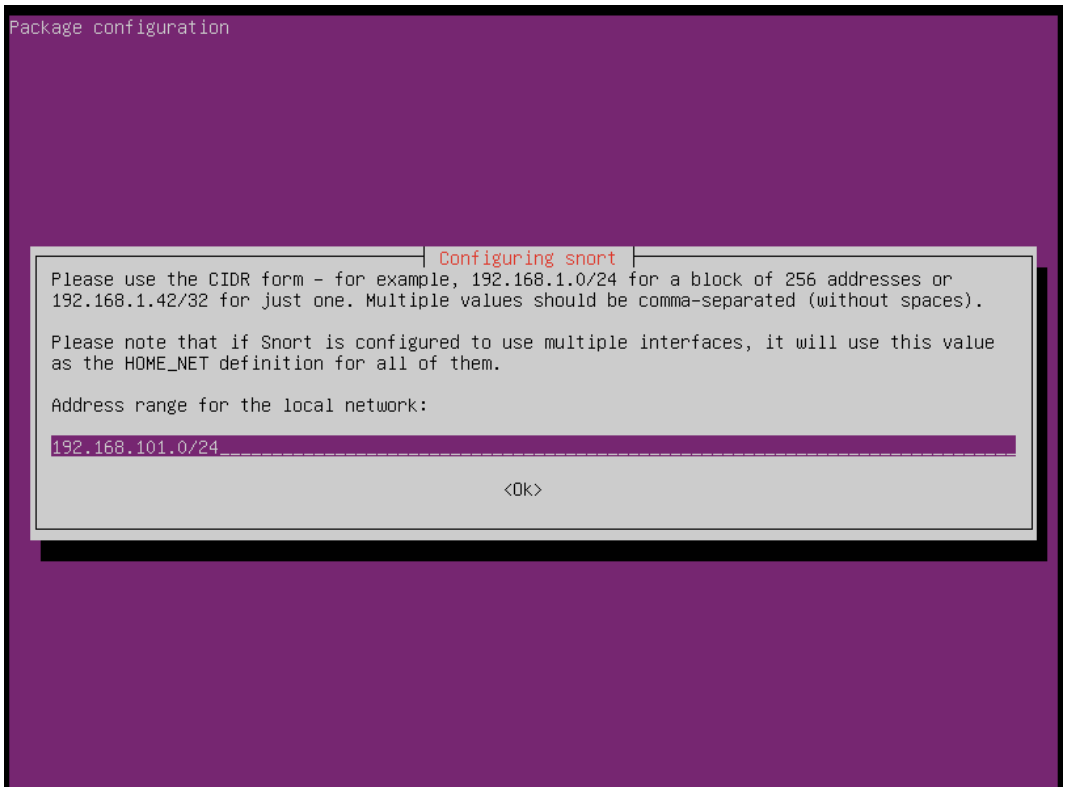
نصب و راه اندازی ابزار Snort در dut

نصب ابزار Snort با اسفاده از دستور زیر انجام می شود.

```
homa@dut:~$ sudo apt-get install snort -y
```

سپس مشخص می کنیم که روی کدام اینترفیس و با چه آدرس شبکه ای باید listen کند. در اینجا اینترفیس متصل به ماشین in داده شده است.


Package configuration
Configuring snort
This value is usually "eth0", but this may be inappropriate in some network environments;
for a dialup connection "ppp0" might be more appropriate (see the output of
"/sbin/ifconfig").
Typically, this is the same interface as the "default route" is on. You can determine which
interface is used for this by running "/sbin/route -n" (look for "0.0.0.0").
It is also not uncommon to use an interface with no IP address configured in promiscuous
mode. For such cases, select the interface in this system that is physically connected to
the network that should be inspected, enable promiscuous mode later on and make sure that
the network traffic is sent to this interface (either connected to a "port
mirroring/spanning" port in a switch, to a hub, or to a tap).
You can configure multiple interfaces, just by adding more than one interface name separated
by spaces. Each interface can have its own specific configuration.
Interface(s) which Snort should listen on:
ens37
<Ok>


Package configuration
Configuring snort
Please use the CIDR form - for example, 192.168.1.0/24 for a block of 256 addresses or
192.168.1.42/32 for just one. Multiple values should be comma-separated (without spaces).
Please note that if Snort is configured to use multiple interfaces, it will use this value
as the HOME_NET definition for all of them.
Address range for the local network:
192.168.101.0/24
<Ok>

برای بررسی درستی فرایند نصب ابزار از دستور زیر استفاده می کنیم.

```
homa@dut:~$ snort --version

   ,,_     -*> Snort! <*-
  o"  )~   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
```

با دستور زیر فایل کانفیگ Snort را باز کرده و تنظیمات لازم را در این فایل انجام می دهیم.

```
homa@dut:~$ sudo nano /etc/snort/snort.conf
```

مطابق تصویر زیر IP شبکه هدف تعیین می شود

```
  GNU nano 4.8                          /etc/snort/snort.conf                          Modified
#  1) Set the network variables.
#  2) Configure the decoder
#  3) Configure the base detection engine
#  4) Configure dynamic loaded libraries
#  5) Configure preprocessors
#  6) Configure output plugins
#  7) Customize your rule set
#  8) Customize preprocessor and decoder rule set
#  9) Customize shared object rule set
##################################################

##################################################
# Step #1: Set the network variables.  For more information, see README.variables
##################################################

# Setup the network addresses you are protecting
#
# Note to Debian users: this value is overriden when starting
# up the Snort daemon through the init.d script by the
# value of DEBIAN_SNORT_HOME_NET s defined in the
# /etc/snort/snort.debian.conf configuration file
#
ipvar HOME_NET 192.168.101.0/24

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET any
# If HOME_NET is defined as something other than "any", alternative, you can
# use this definition if you do not want to detect attacks from your internal
# IP addresses:
#ipvar EXTERNAL_NET !$HOME_NET

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo
```

با دستور زیر درستی عملکرد Snort را با فایل کانفیگ موجود بررسی می کنیم.

```
homa@dut:~$ sudo snort -T -c /etc/snort/snort.conf
```

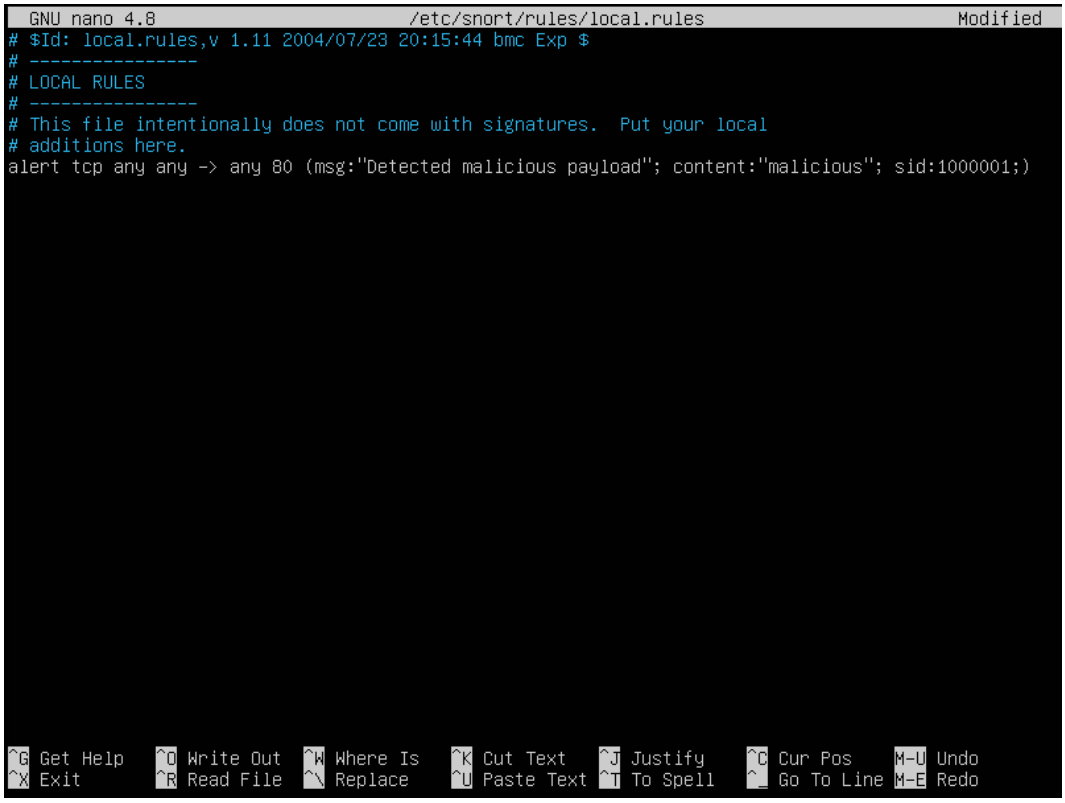
مطابق تصویر زیر این ابزار به درستی اجرا می شود.

```
|   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! <*-
  o"  )~   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_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>

Snort successfully validated the configuration!
Snort exiting
```

با استفاده از دستور زیر به فایل قوانین لوکال Snort دسترسی پیدا می کنیم.

```
homa@dut:~$ sudo nano /etc/snort/rules/local.rules_
```

قانون زیر را به فایل قوانین Snort اضافه می کنیم تا هر پکتی با پروتکل tcp از هر پورت مبدا و با هر آدرس IP ای به هر آدرس IP مقصد با پورت 80 برود و در قسمت payload آن عبارت malicious وجود داشته باشد، پیامی با محتوای زیر را alert دهد.



با استفاده از دستور زیر پکت هایی که روی اینترفیس ens37 ماشین dut دریافت می شوند در فایل malicious_packets.pcap ذخیره می شوند.

```
homa@dut:~$ sudo tcpdump -i ens37 -w malicious_packets.pcap
tcpdump: listening on ens37, link-type EN10MB (Ethernet), capture size 262144 bytes
```

با استفاده از nping که در nmap قرار دارد در ماشین in پکت های آلوده (در قسمت payload آن عبارت malicious وجود دارد) به تعداد 100 تا تولید می شود و روی پورت 80 با پروتکل TCP ارسال می کند.

```
homa@in:~$ sudo nping --tcp -p 80 --data-string "malicious payload" -c 100 192.168.101.1
[sudo] password for homa:

Starting Nping 0.7.80 ( https://nmap.org/nping ) at 2024-06-21 11:27 UTC
SENT (0.1161s) TCP 192.168.101.2:24309 > 192.168.101.1:80 S ttl=64 id=52017 iplen=57  seq=3797973237
 win=1480
RCVD (0.1169s) TCP 192.168.101.1:80 > 192.168.101.2:24309 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (1.1198s) TCP 192.168.101.2:24309 > 192.168.101.1:80 S ttl=64 id=52017 iplen=57  seq=3797973237
 win=1480
RCVD (1.1204s) TCP 192.168.101.1:80 > 192.168.101.2:24309 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (2.1259s) TCP 192.168.101.2:24309 > 192.168.101.1:80 S ttl=64 id=52017 iplen=57  seq=3797973237
 win=1480
RCVD (2.1265s) TCP 192.168.101.1:80 > 192.168.101.2:24309 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (3.1411s) TCP 192.168.101.2:24309 > 192.168.101.1:80 S ttl=64 id=52017 iplen=57  seq=3797973237
 win=1480
RCVD (3.1430s) TCP 192.168.101.1:80 > 192.168.101.2:24309 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (4.1476s) TCP 192.168.101.2:24309 > 192.168.101.1:80 S ttl=64 id=52017 iplen=57  seq=3797973237
 win=1480
RCVD (4.1484s) TCP 192.168.101.1:80 > 192.168.101.2:24309 RA ttl=64 id=0 iplen=40  seq=0 win=0
```

```
7 win=1480
RCVD (89.9634s) TCP 192.168.101.1:80 > 192.168.101.2:24309 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (90.9689s) TCP 192.168.101.2:24309 > 192.168.101.1:80 S ttl=64 id=52017 iplen=57  seq=379797323
7 win=1480
RCVD (90.9706s) TCP 192.168.101.1:80 > 192.168.101.2:24309 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (91.9792s) TCP 192.168.101.2:24309 > 192.168.101.1:80 S ttl=64 id=52017 iplen=57  seq=379797323
7 win=1480
RCVD (91.9840s) TCP 192.168.101.1:80 > 192.168.101.2:24309 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (92.9902s) TCP 192.168.101.2:24309 > 192.168.101.1:80 S ttl=64 id=52017 iplen=57  seq=379797323
7 win=1480
RCVD (92.9935s) TCP 192.168.101.1:80 > 192.168.101.2:24309 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (94.0000s) TCP 192.168.101.2:24309 > 192.168.101.1:80 S ttl=64 id=52017 iplen=57  seq=379797323
7 win=1480
RCVD (94.0021s) TCP 192.168.101.1:80 > 192.168.101.2:24309 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (95.0088s) TCP 192.168.101.2:24309 > 192.168.101.1:80 S ttl=64 id=52017 iplen=57  seq=379797323
7 win=1480
RCVD (95.0103s) TCP 192.168.101.1:80 > 192.168.101.2:24309 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (96.0150s) TCP 192.168.101.2:24309 > 192.168.101.1:80 S ttl=64 id=52017 iplen=57  seq=379797323
7 win=1480
RCVD (96.0178s) TCP 192.168.101.1:80 > 192.168.101.2:24309 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (97.0264s) TCP 192.168.101.2:24309 > 192.168.101.1:80 S ttl=64 id=52017 iplen=57  seq=379797323
7 win=1480
RCVD (97.0290s) TCP 192.168.101.1:80 > 192.168.101.2:24309 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (98.0353s) TCP 192.168.101.2:24309 > 192.168.101.1:80 S ttl=64 id=52017 iplen=57  seq=379797323
7 win=1480
RCVD (98.0376s) TCP 192.168.101.1:80 > 192.168.101.2:24309 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (99.0426s) TCP 192.168.101.2:24309 > 192.168.101.1:80 S ttl=64 id=52017 iplen=57  seq=379797323
7 win=1480
RCVD (99.0445s) TCP 192.168.101.1:80 > 192.168.101.2:24309 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (100.0488s) TCP 192.168.101.2:24309 > 192.168.101.1:80 S ttl=64 id=52017 iplen=57  seq=37979732
37 win=1480
RCVD (100.0524s) TCP 192.168.101.1:80 > 192.168.101.2:24309 RA ttl=64 id=0 iplen=40  seq=0 win=0

Max rtt: 4.558ms | Min rtt: 0.391ms | Avg rtt: 2.397ms
Raw packets sent: 100 (5.700KB) | Rcvd: 100 (4.600KB) | Lost: 0 (0.00%)
Nping done: 1 IP address pinged in 100.13 seconds
homa@in:~$
```

سپس Snort روی فایل malicious_packets.pcap با کانفیگ های فایل snort.conf اجرا شده و نتیجه بر روی کنسول و در فایل لاگی که در آدرس /var/log/snort قرار دارد مشاهده می شود.

```
homa@dut:~$ sudo snort -r malicious_packets.pcap -c /etc/snort/snort.conf -A console -v -l /var/log/
snort
```

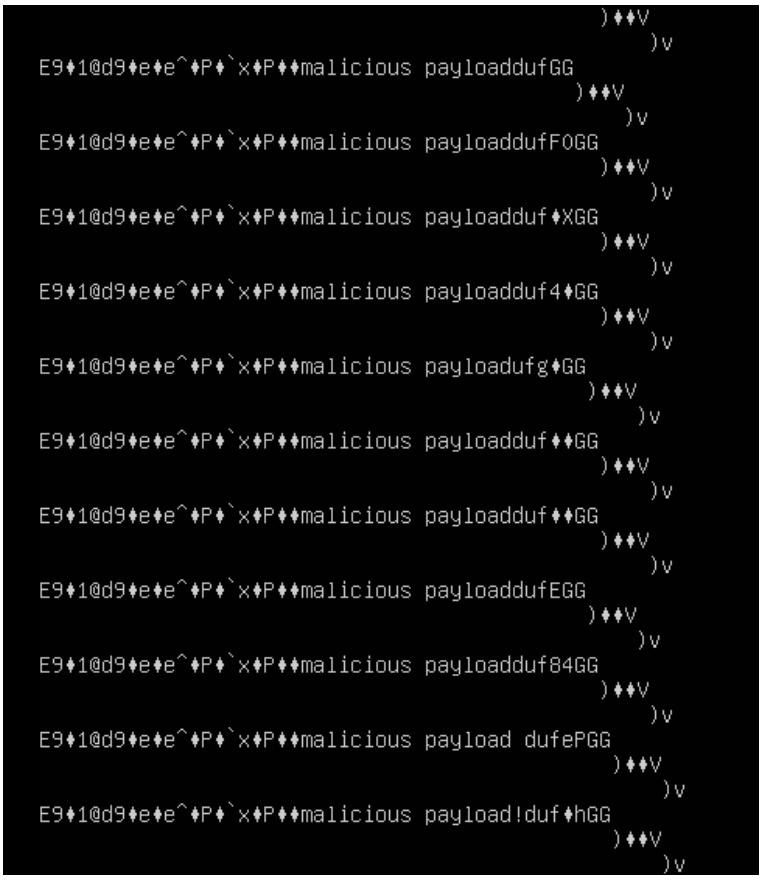
همانطور که مشاهده می شود آنالیز بر روی هر 100 تا پکت انجام می شود.

```
            UDP Port Filter
                  Filtered: 0
                 Inspected: 0
                   Tracked: 0
===============================================================================
HTTP Inspect - encodings (Note: stream-reassembled packets included):
    POST methods:                         0
    GET methods:                          0
    HTTP Request Headers extracted:       0
    HTTP Request Cookies extracted:       0
    Post parameters extracted:            0
    HTTP response Headers extracted:      0
    HTTP Response Cookies extracted:      0
    Unicode:                              0
    Double unicode:                       0
    Non-ASCII representable:              0
    Directory traversals:                 0
    Extra slashes ("//"):                 0
    Self-referencing paths ("./"):        0
    HTTP Response Gzip packets extracted: 0
    Gzip Compressed Data Processed:       n/a
    Gzip Decompressed Data Processed:     n/a
    Total packets processed:              100
===============================================================================
SMTP Preprocessor Statistics
  Total sessions                                : 0
  Max concurrent sessions                       : 0
===============================================================================
dcerpc2 Preprocessor Statistics
  Total sessions: 0
===============================================================================
===============================================================================
SIP Preprocessor Statistics
  Total sessions: 0
===============================================================================
Snort exiting
homa@dut:~$ _
```

در نهایت لاگی که ایجاد می شود از طریق دستور زیر قابل نمایش است.

```
homa@dut:/var/log/snort$ sudo cat snort.log.1718970099
```

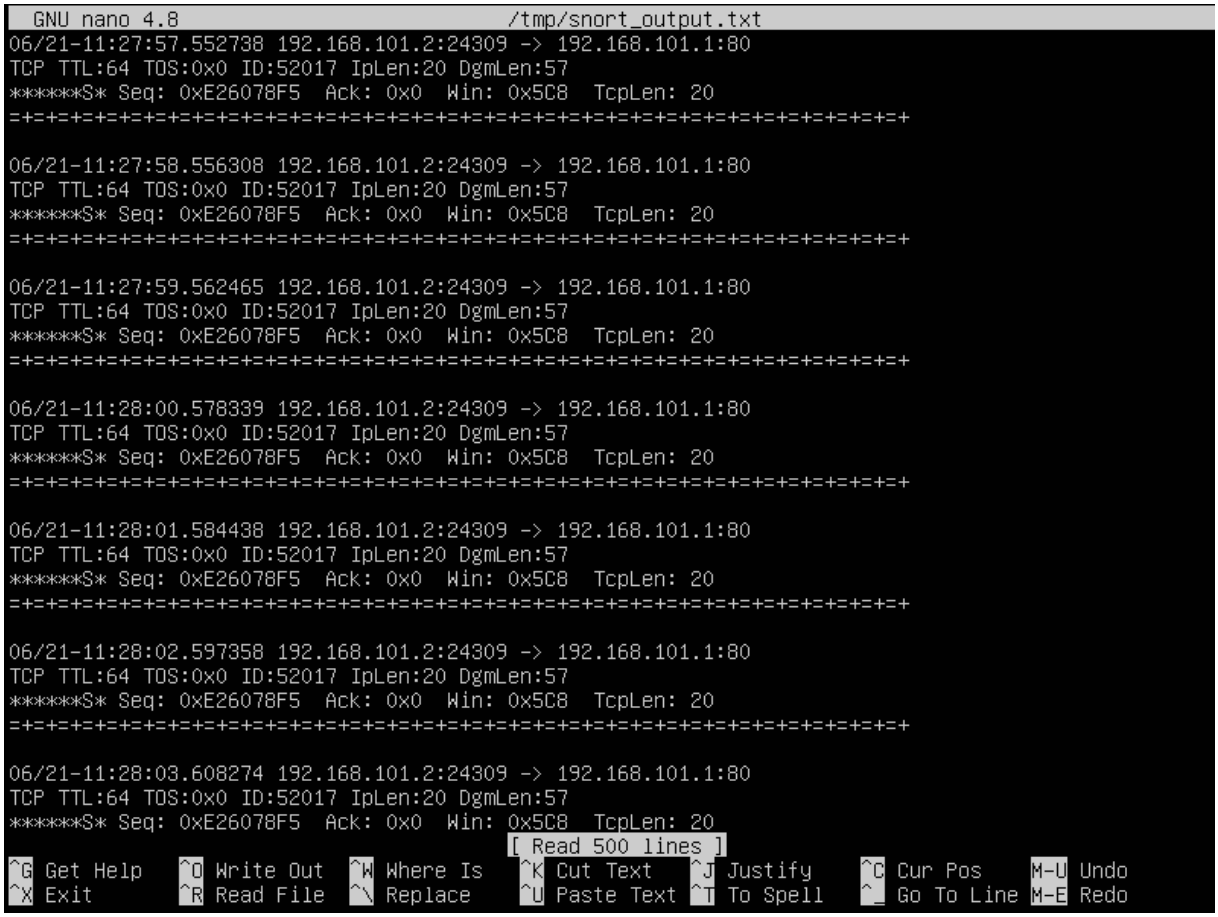
محتوای این فایل به صورت زیر است



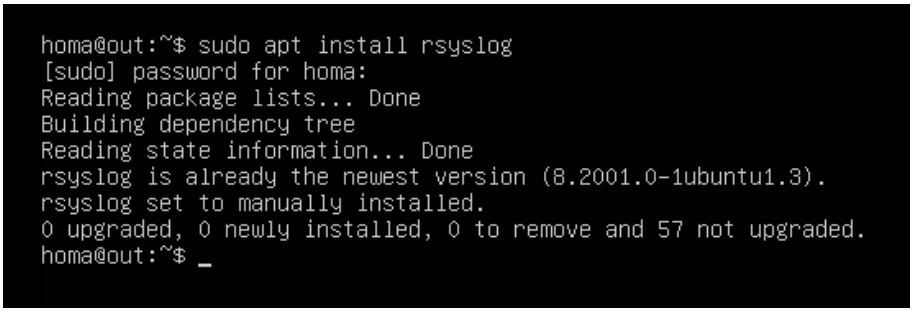
با استفاده از دستور زیر نتایج حاصل از تحلیل Snort که بر روی کنسول نمایش داده می شود را در فایل snort_output.txt ذخیره میکنیم تا بتوانیم آن را مشاهده کنیم.

```
homa@dut:~$ sudo snort -r /var/log/snort/snort.log.1718970099 -A console > /tmp/snort_output.txt
```

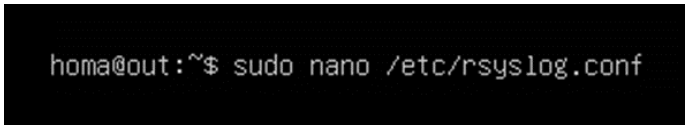
محتوای فایل snort_output.txt به صورت زیر می باشد.



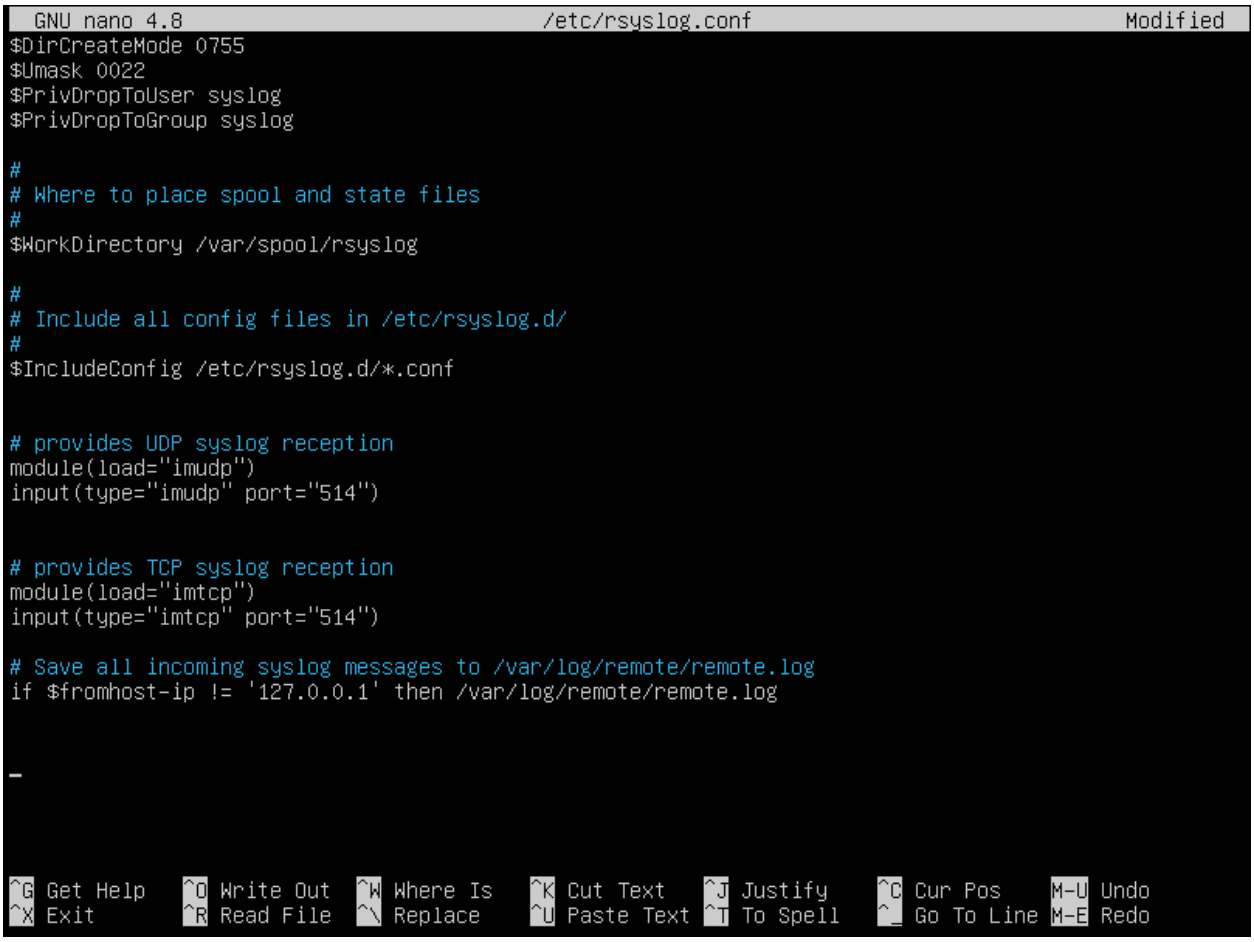
به منظور تنظیم ماشین out برای دریافت لاگ فایل ها از ماشین dut ابتدا rsyslog را نصب می کنیم.



و برای تنظیم آن فایل کانفیگ مربوطه را باز می کنیم.



در این فایل تنظیمات لازم برای دریافت پکت های UDP و TCP را انجام می دهیم و آدرس محل ذخیره لاگ های دریافتی از سایر ماشین ها را نیز تعریف می کنیم.



دایرکتوری زیر را برای نگهداری فایل های دریافتی ایجاد و سطوح دسترسی آن را تنظیم می کنیم.

```
homa@out:~$ sudo mkdir -p /var/log/remote
homa@out:~$ sudo chown syslog:adm /var/log/remote
homa@out:~$ sudo chmod 755 /var/log/remote
```

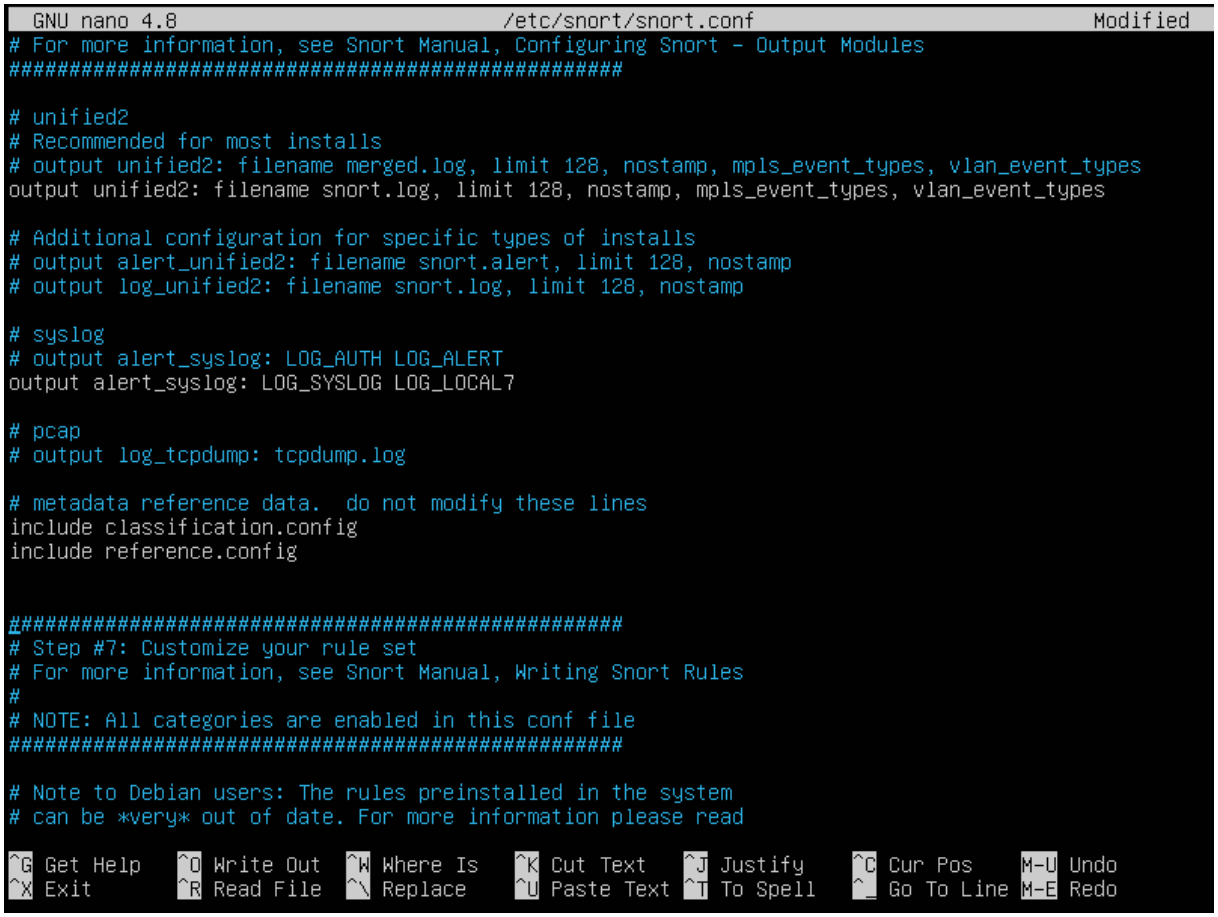
سپس توسط دستور زیر rsyslog را ری استارت می کنیم تا کانفیگ های جدید اعمال شود.

```
homa@out:~$ sudo systemctl restart rsyslog
```

پس از انجام کانفیگ های مربوطه در ماشین out برای دریافت لاگ ها، کانفیگ های لازم در ماشین dut را نیز انجام می دهیم.

```
homa@dut:~$ sudo nano /etc/snort/snort.conf
```

با استفاده از تنظیمات زیر Snort ، لاگ ها را به syslog میفرستد.



سپس Snort را ری استارت می کنیم تا تغییرات اعمال شوند.

```
homa@dut:~$ sudo systemctl restart snort
```

سپس فایل rsyslog را در ماشین dut نیز باز می کنیم تا تغییرات لازم را در این فایل نیز اعمال کنیم.

```
homa@dut:~$ sudo nano /etc/rsyslog.conf
```

با استفاده از 514 :192.168.100.2@ *.* در فایل کانفیگ rsyslog مشخص می کنیم که همه لاگ ها با پروتکل UDP روی پورت 514 به آدرس IP مشخص شده فرستاده شوند.

```
  GNU nano 4.8                        /etc/rsyslog.conf                        Modified
#
# Use traditional timestamp format.
# To enable high precision timestamps, comment out the following line.
#
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

# Filter duplicated messages
$RepeatedMsgReduction on

#
# Set the default permissions for all log files.
#
$FileOwner syslog
$FileGroup adm
$FileCreateMode 0640
$DirCreateMode 0755
$Umask 0022
$PrivDropToUser syslog
$PrivDropToGroup syslog

#
# Where to place spool and state files
#
$WorkDirectory /var/spool/rsyslog

#
# Include all config files in /etc/rsyslog.d/
#
$IncludeConfig /etc/rsyslog.d/*.conf

*.* @192.168.100.2:514


^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos   M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell  ^_ Go To Line M-E Redo
```

سپس دستور زیر rsyslog را ری استارت می کنیم تا کانفیگ های جدید اعمال شود.

```
homa@dut:~$ sudo systemctl restart rsyslog
```

با استفاده از دستور زیر پکت هایی که روی اینترفیس ens37 ماشین dut دریافت می شوند در فایل malicious_packets2.pcap ذخیره می شوند.

```
homa@dut:~$ sudo tcpdump -i ens37 -w malicious_packets2.pcap
tcpdump: listening on ens37, link-type EN10MB (Ethernet), capture size 262144 bytes
```

با استفاده از nping که در nmap قرار دارد در ماشین in پکت های آلوده (در قسمت payload آن عبارت malicious وجود دارد) به تعداد 10 تا تولید می شود و روی پورت 80 با پروتکل TCP ارسال می کند

```
homa@in:~$ sudo nping --tcp -p 80 --data-string "malicious payload" -c 10 192.168.101.1
[sudo] password for homa:
```

```
Starting Nping 0.7.80 ( https://nmap.org/nping ) at 2024-06-21 15:07 UTC
SENT (0.1081s) TCP 192.168.101.2:48421 > 192.168.101.1:80 S ttl=64 id=40285 iplen=57  seq=3941960628
 win=1480
RCVD (0.1091s) TCP 192.168.101.1:80 > 192.168.101.2:48421 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (1.1170s) TCP 192.168.101.2:48421 > 192.168.101.1:80 S ttl=64 id=40285 iplen=57  seq=3941960628
 win=1480
RCVD (1.1177s) TCP 192.168.101.1:80 > 192.168.101.2:48421 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (2.1249s) TCP 192.168.101.2:48421 > 192.168.101.1:80 S ttl=64 id=40285 iplen=57  seq=3941960628
 win=1480
RCVD (2.1257s) TCP 192.168.101.1:80 > 192.168.101.2:48421 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (3.1295s) TCP 192.168.101.2:48421 > 192.168.101.1:80 S ttl=64 id=40285 iplen=57  seq=3941960628
 win=1480
RCVD (3.1302s) TCP 192.168.101.1:80 > 192.168.101.2:48421 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (4.1422s) TCP 192.168.101.2:48421 > 192.168.101.1:80 S ttl=64 id=40285 iplen=57  seq=3941960628
 win=1480
RCVD (4.1429s) TCP 192.168.101.1:80 > 192.168.101.2:48421 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (5.1450s) TCP 192.168.101.2:48421 > 192.168.101.1:80 S ttl=64 id=40285 iplen=57  seq=3941960628
 win=1480
RCVD (5.1455s) TCP 192.168.101.1:80 > 192.168.101.2:48421 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (6.1546s) TCP 192.168.101.2:48421 > 192.168.101.1:80 S ttl=64 id=40285 iplen=57  seq=3941960628
 win=1480
RCVD (6.1551s) TCP 192.168.101.1:80 > 192.168.101.2:48421 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (7.1608s) TCP 192.168.101.2:48421 > 192.168.101.1:80 S ttl=64 id=40285 iplen=57  seq=3941960628
 win=1480
RCVD (7.1612s) TCP 192.168.101.1:80 > 192.168.101.2:48421 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (8.1705s) TCP 192.168.101.2:48421 > 192.168.101.1:80 S ttl=64 id=40285 iplen=57  seq=3941960628
 win=1480
RCVD (8.1710s) TCP 192.168.101.1:80 > 192.168.101.2:48421 RA ttl=64 id=0 iplen=40  seq=0 win=0
SENT (9.1789s) TCP 192.168.101.2:48421 > 192.168.101.1:80 S ttl=64 id=40285 iplen=57  seq=3941960628
 win=1480
RCVD (9.1797s) TCP 192.168.101.1:80 > 192.168.101.2:48421 RA ttl=64 id=0 iplen=40  seq=0 win=0

Max rtt: 0.722ms | Min rtt: 0.226ms | Avg rtt: 0.465ms
Raw packets sent: 10 (570B) | Rcvd: 10 (460B) | Lost: 0 (0.00%)
Nping done: 1 IP address pinged in 9.29 seconds
homa@in:~$
```

سپس Snort روی فایل malicious_packets2.pcap با کانفیگ های فایل snort.conf اجرا شده و نتیجه بر روی کنسول ظاهر شده و لاگ آن برای ماشین out نیز فرستاده می شود.

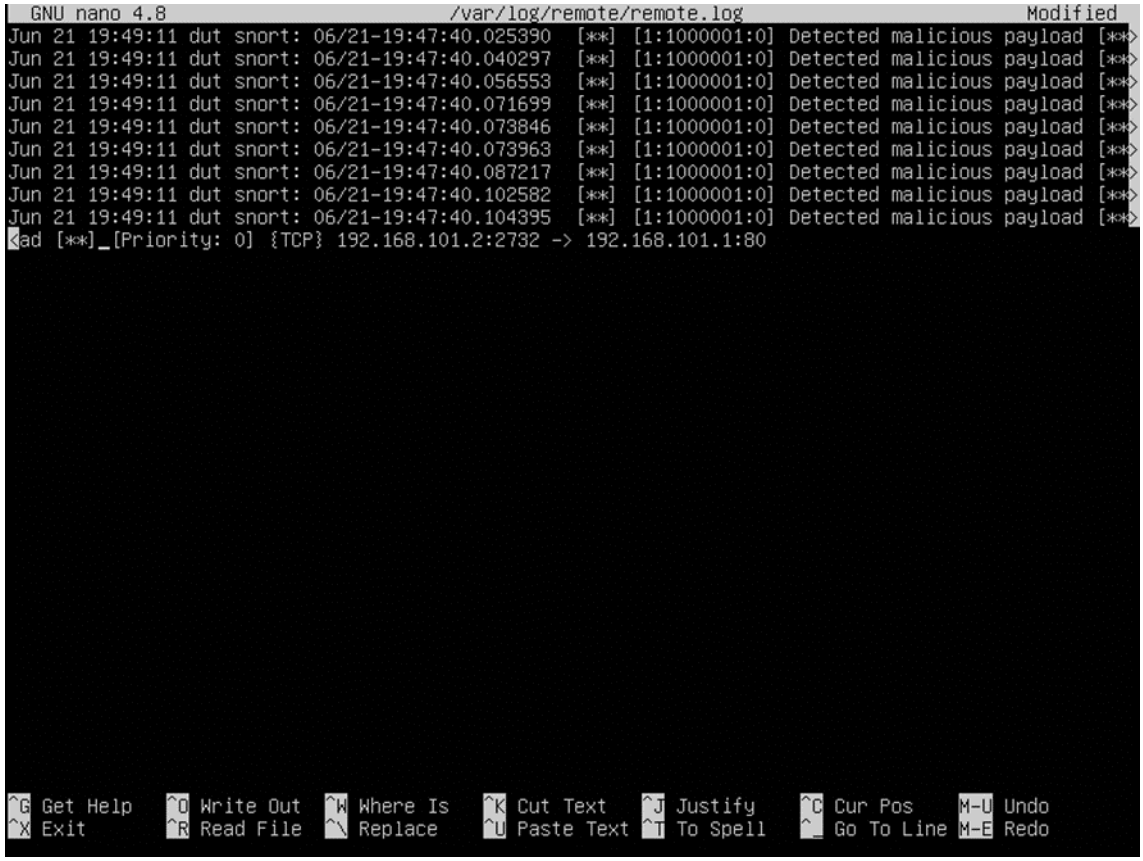
```
homa@dut:~$ sudo snort -r malicious_packets2.pcap -c /etc/snort/snort.conf -A console | logger -t sn
ort -p local7.info_
```

```
            UDP Port Filter
                  Filtered: 0
                 Inspected: 0
                   Tracked: 0
===============================================================================
HTTP Inspect - encodings (Note: stream-reassembled packets included):
    POST methods:                         0
    GET methods:                          0
    HTTP Request Headers extracted:       0
    HTTP Request Cookies extracted:       0
    Post parameters extracted:            0
    HTTP response Headers extracted:      0
    HTTP Response Cookies extracted:      0
    Unicode:                              0
    Double unicode:                       0
    Non-ASCII representable:              0
    Directory traversals:                 0
    Extra slashes ("//"):                 0
    Self-referencing paths ("./"):        0
    HTTP Response Gzip packets extracted: 0
    Gzip Compressed Data Processed:       n/a
    Gzip Decompressed Data Processed:     n/a
    Total packets processed:              10
===============================================================================
SMTP Preprocessor Statistics
  Total sessions                                : 0
  Max concurrent sessions                       : 0
===============================================================================
dcerpc2 Preprocessor Statistics
  Total sessions: 0
===============================================================================
===============================================================================
SIP Preprocessor Statistics
  Total sessions: 0
===============================================================================
Snort exiting
homa@dut:~$
```

سپس محل ذخیره سازی لاگ های ریموت در ماشین out را باز می کنیم.

```
homa@out:~$ sudo nano /var/log/remote/remote.log_
```

همانطور که در این فایل قابل مشاهده است لاگ ها ارسال شده اند و متن alert ، پروتکل پکت و آدرس ip های مبدا و مقصد و پورت های آن قابل مشاهده است.



Snort را بر روی فایل malicious_packets2.pcap اجرا می کنیم و برای اینکه همه قسمت های لاگ قابل خواندن برای انسان باشد، از دستور زیر استفاده می کنیم و لاگ ها را در snort_output.txt ذخیره می کنیم و سپس آن را مشاهده می کنیم.

```
homa@dut:~$ sudo snort -r malicious_packets2.pcap -c /etc/snort/snort.conf -A console > snort_output
.txt_
```

```
            UDP Port Filter
                  Filtered: 0
                 Inspected: 0
                   Tracked: 0
===============================================================================
HTTP Inspect - encodings (Note: stream-reassembled packets included):
    POST methods:                         0
    GET methods:                          0
    HTTP Request Headers extracted:       0
    HTTP Request Cookies extracted:       0
    Post parameters extracted:            0
    HTTP response Headers extracted:      0
    HTTP Response Cookies extracted:      0
    Unicode:                              0
    Double unicode:                       0
    Non-ASCII representable:              0
    Directory traversals:                 0
    Extra slashes ("//"):                 0
    Self-referencing paths ("./"):        0
    HTTP Response Gzip packets extracted: 0
    Gzip Compressed Data Processed:       n/a
    Gzip Decompressed Data Processed:     n/a
    Total packets processed:              10
===============================================================================
SMTP Preprocessor Statistics
  Total sessions                                  : 0
  Max concurrent sessions                         : 0
===============================================================================
dcerpc2 Preprocessor Statistics
  Total sessions: 0
===============================================================================
===============================================================================
SIP Preprocessor Statistics
  Total sessions: 0
===============================================================================
Snort exiting
```

```
homa@dut:~$ cat snort_output.txt
06/21-15:07:56.084659  [**] [1:1000001:0] Detected malicious payload [**] [Priority: 0] {TCP} 192.16
8.101.2:48421 -> 192.168.101.1:80
06/21-15:07:57.093362  [**] [1:1000001:0] Detected malicious payload [**] [Priority: 0] {TCP} 192.16
8.101.2:48421 -> 192.168.101.1:80
06/21-15:07:58.101431  [**] [1:1000001:0] Detected malicious payload [**] [Priority: 0] {TCP} 192.16
8.101.2:48421 -> 192.168.101.1:80
06/21-15:07:59.105953  [**] [1:1000001:0] Detected malicious payload [**] [Priority: 0] {TCP} 192.16
8.101.2:48421 -> 192.168.101.1:80
06/21-15:08:00.118587  [**] [1:1000001:0] Detected malicious payload [**] [Priority: 0] {TCP} 192.16
8.101.2:48421 -> 192.168.101.1:80
06/21-15:08:01.121325  [**] [1:1000001:0] Detected malicious payload [**] [Priority: 0] {TCP} 192.16
8.101.2:48421 -> 192.168.101.1:80
06/21-15:08:02.130968  [**] [1:1000001:0] Detected malicious payload [**] [Priority: 0] {TCP} 192.16
8.101.2:48421 -> 192.168.101.1:80
06/21-15:08:03.137076  [**] [1:1000001:0] Detected malicious payload [**] [Priority: 0] {TCP} 192.16
8.101.2:48421 -> 192.168.101.1:80
06/21-15:08:04.146857  [**] [1:1000001:0] Detected malicious payload [**] [Priority: 0] {TCP} 192.16
8.101.2:48421 -> 192.168.101.1:80
06/21-15:08:05.155381  [**] [1:1000001:0] Detected malicious payload [**] [Priority: 0] {TCP} 192.16
8.101.2:48421 -> 192.168.101.1:80
homa@dut:~$
```

همانطور که مشاهده می شود تعداد پکت های مشکوک شناسایی شده 10 تا می باشد و تعداد پکت های ارسالی نیز 10 تا می باشد بنابراین دقت Snort در اینجا و برای این تعداد پکت 100 درصد بوده است.

به منظور ارزیابی سرعت تحلیل Snort از دستور زیر استفاده می کنیم تا مدت زمان تحلیل برای 10 پکت را ارزیابی کنیم.

```
homa@dut:~$ time sudo snort -r malicious_packets2.pcap -c /etc/snort/snort.conf -A console > /dev/nu
ll
```

همانطور که مشاهده می شود مدت زمان واقعی برای 10 پکت ذخیره شده در فایل pcap، 14.354 ثانیه ارزیابی شده است.

```
===============================================================================
HTTP Inspect - encodings (Note: stream-reassembled packets included):
    POST methods:                         0
    GET methods:                          0
    HTTP Request Headers extracted:       0
    HTTP Request Cookies extracted:       0
    Post parameters extracted:            0
    HTTP response Headers extracted:      0
    HTTP Response Cookies extracted:      0
    Unicode:                              0
    Double unicode:                       0
    Non-ASCII representable:              0
    Directory traversals:                 0
    Extra slashes ("//"):                 0
    Self-referencing paths ("./"):        0
    HTTP Response Gzip packets extracted: 0
    Gzip Compressed Data Processed:       n/a
    Gzip Decompressed Data Processed:     n/a
    Total packets processed:              10
===============================================================================
SMTP Preprocessor Statistics
  Total sessions                                : 0
  Max concurrent sessions                       : 0
===============================================================================
dcerpc2 Preprocessor Statistics
  Total sessions: 0
===============================================================================
===============================================================================
SIP Preprocessor Statistics
  Total sessions: 0
===============================================================================
Snort exiting

real    0m14.354s
user    0m2.278s
sys     0m1.981s
homa@dut:~$
```

برای تحلیل مدت زمان ارزیابی Snort علاوه بر دستور بالا می توان به صورت زیر زمان شروع و پایان ارزیابی را بدست آوریم و سپس از اختلاف این مقادیر duration را محاسبه کنیم.

```
homa@dut:~$ nano snort_analysis.sh_
```

```
 GNU nano 4.8                          snort_analysis.sh                          Modified
#!/bin/bash

# Variables
PCAP_FILE="malicious_packets.pcap"  # Name of the pcap file
SNORT_CONF="/etc/snort/snort.conf"  # Path to Snort configuration file
SNORT_OUTPUT="snort_output.txt"     # File to store Snort output

# Measure processing time with Snort
echo "Analyzing packets with Snort..."
START_TIME=$(date +%s.%N)

sudo snort -r $PCAP_FILE -c $SNORT_CONF -A console > $SNORT_OUTPUT

END_TIME=$(date +%s.%N)
DURATION=$(echo "$END_TIME - $START_TIME" | bc)

# Display results
echo "==========================================="
echo "Snort Analysis Complete"
echo "-------------------------------------------"
echo "Total Processing Time : $DURATION seconds"
echo "==========================================="

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo
```

```
homa@dut:~$ sudo ./snort_analysis.sh
```

```
        HTTP Inspect - encodings (Note: stream-reassembled packets included):
            POST methods:                         0
            GET methods:                          0
            HTTP Request Headers extracted:       0
            HTTP Request Cookies extracted:       0
            Post parameters extracted:            0
            HTTP response Headers extracted:      0
            HTTP Response Cookies extracted:      0
            Unicode:                              0
            Double unicode:                       0
            Non-ASCII representable:              0
            Directory traversals:                 0
            Extra slashes ("//"):                 0
            Self-referencing paths ("./"):        0
            HTTP Response Gzip packets extracted: 0
            Gzip Compressed Data Processed:       n/a
            Gzip Decompressed Data Processed:     n/a
            Total packets processed:              10
        ===============================================================================
        SMTP Preprocessor Statistics
          Total sessions                                   : 0
          Max concurrent sessions                          : 0
        ===============================================================================
        dcerpc2 Preprocessor Statistics
          Total sessions: 0
        ===============================================================================
        ===============================================================================
        SIP Preprocessor Statistics
          Total sessions: 0
        ===============================================================================
        Snort exiting
        ===========================================
        Snort Analysis Complete
        -------------------------------------------
        Total Processing Time : 11.879567389 seconds
        ===========================================
        homa@dut:~$
```