به نام خدا

مهندسی ICT

موضوع: نرم افزار snort

درس: امنیت شبکه

تهیه کننده :

یاسمین عبدالله پور

89114314

استاد:آقای دکتر رضا امیرپور

موسسه جهاد دانشگاهی خوزستان

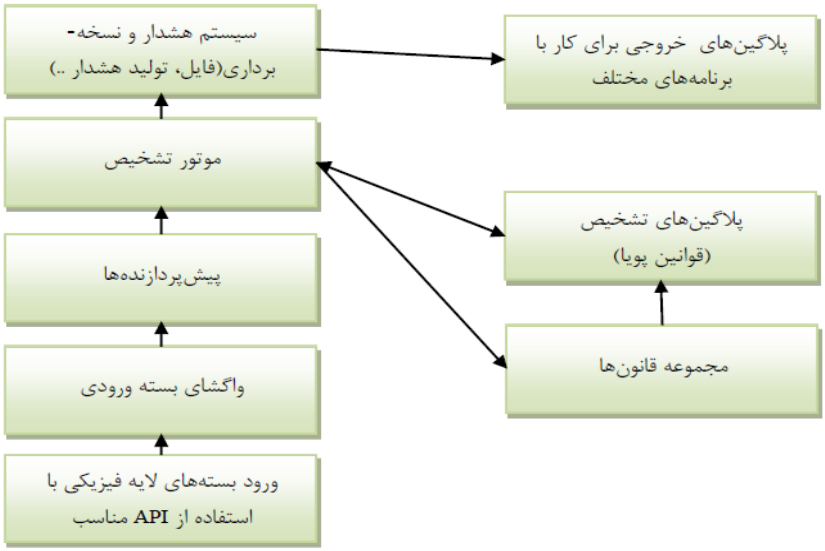
بهار91

معرفی اسنورت و ابزار آن

اسنورت یک ابزار کدباز تشخیص و جلوگیري از نفوذ است .این ابزار در سال هاي 1999-1998 توسط مارتین روزچ موسس شرکت *source file* توسعه داده شد .این شرکت به پشتیباني فني از این ابزارمی پردازد و بر اساس این ابزار براي سازمانهاي مختلف ساختارها و راه حلهاي امنیتي اراده میدهد .اسنورت تا سال ۲۰۱۱ با بیش از ۳ میلیون دانلود، متداولترین فناوري تشخیص نفوذ در دنیا میباشد.اسنورت میتواند در یکي از این سه حالت کاري پیکربندي شود :استراق سمع کننده بسته ها ، نسخه بردار از ترافیک و سیستم تشخیص نفوذ .در حالت اول ابزار تمام ترافیک ورودي به واسط شبکه اي را درکنسول سیستم نمایش میدهد .در حالت نسخه بردار، اسنورت ترافیک ورودي را در روي دیسک ذخیره مي کند .مهمترین کارکرد اسنورت سیستم تشخیص نفوذ است .این ابزار عمدتا با استفاده از قوانین به تشخیص الگوهاي حملات و نیز ناهنجاریهاي موجود در ترافیک شبکه میپردازد. در ادامه به بررسي معماري این ابزار میپردازیم؛ سپس در مورد نجوه عملکرد آن توضیح مي دهیم.همچنین در مورد قوانین اسنورت و ساختار آن بحث میکنیم .در ادامه به بررسي مشخصات موتور تشخیص و ویژگیهاي و قابلیتهاي جدید آن میپردازیم .همچنین به قابلیتهاي این ابزار در کار با پایگاههاي داده ودیگر برنامههاي مدیریتي اشاره میکنیم.

اجزاي تشکیل دهنده اسنورت

ابزار اسنورت از نظر منطقي به بخشهاي زیر تقسیم میشود .شکل زیر جریان داده را در اسنورت نشان مي دهد ورودي این سیستم بسته لایه فیزیکي است؛ که توسط *API* مناسب (معمولا *lippack*) از واسط شبکه دریافت شده است .در این قسمت مراحل عبور بسته را از بخشهاي مختلف اسنورت نشان میدهیم؛ سپس در بخش هاي بعد به بررسي جزئيتر هر یک از بخشهاي سیستم میپردازیم.



معماری اسنورت

مطابق شكل هر كدام از بسته هاي دريافت شده از ورودي شبكه، وارد كدگشاي بسته ها ميشوند؛ سپس درادامه روند يا از آنها صرف نظر ميشود؛ يا رويدارد متناظر با آنها در فابل خروجي ثبت مي شود يا آنكه منجر به توليد هشدار مي‌شوند.

- ✓ واگشايي بسته ورودي

این قسمت بسته ها را از واسطهاي مختلف لايه شبكه دريافت ميكند و آن را به فرمتي مناسب براي مولفه ، موتور تشخيص تبديل ميكند. واسطهاي شبكه‌اي ميتوانند اترنت مانند *slip*،*ppp* و يا ديگر واسطهاي شبكه اي باشند.

- ✓ پيش پردازنده ها

این بخش شامل اجزاي نرم افزاري يا *plug-in هایی* می‌شود كه براي منظم كردن و سامان دهي به بسته هاي دريافتي قبل از ورود به موتور رويداد استفاده ميشوند .بعضي پيش پردازنده ها همچنين با يافتن ناهنجاريها در سرآيند بسته ها، اقدامات مخرب را كشف ميكنند.در بسياري از موارد حمله كننده ها براي فريب سيستم تشخيص نفوذ در شناخت امضاها اقدام به تغييرحروف بزرگ به كوچک، يا اضافه كردن فاصله بين حروف ميكنند .در اين صورت اگر ابزار تشخيص فقط بر اساس مقايسه رشته‌ها عمل كند در شناخت دچار اشكال ميشود . پيش پردازنده ها اين رشته هاي موجود در بسته ها را به نحوي تغيير ميدهند تا امكان شناسايي آن براي ابزار تشخيص نفوذ باشد.در بسياري موارد حمله كننده براي شناسايي نشدن الگوي حملات از قطعه قطعه سازي بسته ها استفاده ميكند .قطعه قطعه سازي به طور عادي در شبكه زماني كه طول بسته براي گذر از يک لينک به اندازه كافي كوچک نباشد اتفاق ميافتد. در مرحله پيشپردازش بسته هاي قطعه قطعه‌سازي شده دوباره به هم می‌پيوندند. فرامين *http* كدگشايي مي‌شوند و در صورتي كه در آن ها كاراكترهاي غيرمعمول باشد يا طول آنها بيشتر از حد معمول باشند شناسايي ميشوند.

- ✓ موتور تشخيص

این بخش مهم ترين بخش ابزار اسنورت است .اين قسمت به اعمال قوانين موجود در پايگاه داده بر روي بسته ها ميپردازد و مشخص ميكند كه آيا بسته مربوط به يک حمله احتمالي است يا خير .بسته به توانايي بستري كه برنامه اسنورت بر روي آن اجرا ميشود؛ سرعت پاسخدهي به بستهها متفاوت خواهد بود .باري كه بر روي موتور تشخيص ابزار تشخيص نفوذ قرار دارد؛ به اين پارامترها وابسته است:

  - تعداد قوانين موجود براي بررسي
  - قدرت سيستمي كه برنامه اسنورت بر روي آن در حال اجرا مي باشد.
  - ترافيک شبكه

موتور تشخيص براساس شناسه هاي مختلفي كه در هر كدام از بستههاي دريافتي مي باشد؛ اقدامات راانجام ميدهد .اين شناسه هاي مي توانند موارد زير باشند:

- سرآيند لايه شبكه بسته (*IP* فرستنده و گيرنده)
- سرآيند لايه انتقال بسته (*TCP*،*UDP*،*ICMP*)
- سرآيند لايه كاربرد، اين قسمت بر اساس پروتكلهاي مختلف به كار رفته مانند (*HTTP*،*FTP* و ...) مي تواند پيكربندي خاص داشته باشد.

✓ سيستم نسخه برداري و هشدار

بر اساس آنچه كه موتور تشخيص در فايل پيدا مي كند؛ رويداد متناظر با بسته مي تواند به عنوان يک هشدار تلقي و اعلام شود يا آنكه به سادگي ثبت شود.

✓ پلاگين ها ماژول های واسط خروجي

براساس پيكربندي سيستم اين قسمت قادر است نسخه هاي خروجي سيستم را به نحو مناسبي ذخيره كند .از آن جمله ميتوان به اين موارد اشاره كرد:

- ثبت خروجي در يک فايل متني
- ايجاد وقفه هاي *SNMP*
- ثبت رويداد در بانک داده مانند*MySQL*
- توليد خروجي*XML*
- اعمال نتايج بدست آمده بر ديواره آتش و مسيرياب هاي شبكه

بررسي عملكرد اسنورت

در اين بخش ابتدا به در مقدمه اي به قوانين اسنورت و ساختار و قابليت هاي آنها خواهيم پرداخت .اعمال قوانين بر روي بسته ها مبناي اصلي عملكرد اسنورت است .در اين زمينه به معرفي نحوه بارگذاري قوانين درحافظه و نحوه‌ی جستجو در آنها خواهيم پرداخت .در ادامه به بررسي عملكرد دقيق پيش پردازنده ها و موتورتشخيص خواهيم پرداخت .در اين قسمتها در مورد قابليت ها و انتظارات جديد از پيش پردازنده ها بحث خواهد شد و ساختار جديد موتور تشخيص كه از نسخه ۲.۰ به بعد اسنورت معرفي شده است؛ معرفي مي شود .بعد از اين قسمت، به بررسي ماژول هاي و پلاگين هاي خروجي و قابليت هاي آنها خواهيم پرداخت .

قوانين اسنورت

به طور كلي قانون عبارت است از توصيف وضعيت خاص در شبكه و همينطور عملياتي كه در صورت بروز

اين وضعيت بايد صورت پذيرد .ويژگي‌هايي كه براي توصيف اين وضعيت به كار ميروند؛ گزينه هاي انتخابي قانون نام دارند .بر اساس گزينه هاي انتخابي ميتوان تمام ويژگي هاي هر كدام از بسته ها را در نظر گرفت.هركدام از قوانين اسنورت از دو بخش مهم سرآيند و گزينه هاي انتخابي قانون، تشكيل شده است. سرآيند بسته از موارد زير تشكيل شده است:

- عملیات: عملياتي كه در صورت برآورده شدن شرايط قانون بايد صورت پذيرد. مهم ترين اين عمل ها عبارتند از اعلان هشدار، نسخه برداري، صرف نظر كردن از ادامه بررسي از بسته و يا فراخواني قانون ديگر است. در اسنورت حالت برخط، امكان صرف نظر از بسته هم وجود دارد. همچنين امكان فرخواني قوانين پوياي تعريف شده توسط كاربر وجود دارد كه درباره آن بعدا بحث خواهد شد.
- آدرس *IP* منبع و مقصد بسته: اين قسمت ميتواند شامل دسته و يا دامنهاي از آدرسهاي *IP* نيز باشد.
- آدرس پورت منبع و مقصد گيرنده
- جهت حركت بسته
- پروتكل ارتباط: اين قسمت نشاندهنده اين است كه اين قانون بر روي چه بسته هايي بايد اعمال شود مهمترين قوانيني كه در اين قسمت مشخص ميشوند؛ ميتوان *UDP*، *ICMP*، *TCP*، *IP* نام برد. بخش گزينه های انتخابي قانون شامل موارد مختلفي است كه در صورت بروز همه آنها قانون مورد نظراعمال مي شود .از جمله گزينه هاي انتخابي مهم ميتوان به اين موارد اشاره كرد:
- عنوان قانون: اين قسمت شامل مواردي مانند پيغام متناظر اجراي قانون، شناسه امنيتي قانون (*SID*) و شناسه‌ی نوع قانون (*GID*)
- *Content*: با استفاده از اين گزينه امكان يافتن الگوي كاركتري يا بايتي در محتواي بسته وجود دارد. اين گزينه هزينه محاسباتي زيادي براي تطابق الگو دارد و بهتر است در آفست هاي محدود و هدفدار استفاده شود . همچنين گزينه هايي به نام *uricontent* براي جستجو در خروجي نرمال سازي شده پيش پردازنده *HTTP* به كار می‌رود.
- *Flow*: اين گزينه امكان بررسي الگو در جريان ارتباطي خاص را فراهم ميكند .با استفاده از اين گزينه به ارتباطات مشخصي اشاره ميشود كه توسط روند دست دهي *TCP* ايجاد شده اند و در جهت مشخص، به سمت سرور يا كلاينت برقرار شده اند.
- *Classtype*: اين گزينه يک ابزار كلاس‌بندي قوانين است .با استفاده از اين گزينه امكان اولويت‌دهي به قوانين فراهم ميشود .طبيعي است كه برخي آسيب ها از بقيه اهميت بيشتري دارند.
- *Bytejump* و *bytetest*: اين گزينه ها امكان بررسي و چاپ رشته هاي بايتي را بر اساس انديس مشخص، فراهم ميكنند .در مورد *bytejump* اين انديس از خود بسته استخراج ميشود و در مورد *bytetest* اين رشته در خود قانون تعريف می‌شود.
- گزينه‌ی *PCRE*: اين گزينه امكان هم به جستجوي محتواي بسته با يک رشته ميپردازد؛ با اين تفاوت كه رشته مورد نظر ميتواند به صورت يک عبارت منظم تعريف شده باشد .استفاده از اين گزينه باعث سادگي ميشود؛ لكن توان پردازشي بيشتري را لازم دارد.
- گزينه‌ی *flowsbit*: اين گزينه اين امكان را ايجاد ميكند تا بتوان بسته هاي مرتبط با يک ارتباط را به هم ارتباط داد .براي اضافه كردن اطلاعات وضعيت هر ارتباط، يک اشاره گر پوينده به عنوان شاخص براي يک جدول نگاشت بيتي، در ساختار دادهاي شبه بسته، مقداردهي ميشود؛ كه از طريق آن مي توان به بسته‌هاي ديگر آن

جريان دسترسي داشت .اين گزينه اين امكان را ايجاد ميكند تا بتوان قوانين ديگري را صادر نمود. هر كدام از جريان‌های *TCP* و *UDP* از روی آدرس منبع و مقصد شناسایی می‌شوند. مقدار دهی به *flowbits* و پويش جريان هاي مختلف و مقداردهي به فيلد اشاره گر مربوط به پويش جريان؛ توسط پيش پردازنده جريان انجام ميشود .براي اينكه اين قانون كار كند بايستي كه پيش پردازنده جريان پيكربندي شده باشد .برخلاف قوانين ديگر كه روي همه بسته‌ها اعمال مي شوند؛ قانون هاي كه گزينه‌ی انتخابي *flowbits* در آنها فعال شده باشد؛ فقط بر روي بسته هاي يک جريان مشخص اعمال می‌شوند.

- با استفاده از اين گزينه امكان ست كردن بيتهايي به نشانه پرچم وجود دارد كه بر مبناي آن قوانين درمراحل بعد ميتوانند تصميم‌گيري كنند .به اين ترتيب امكان بررسي سلسله رخدادها و حفظ وضعيت در حملات چند مرحله اي وجود دارد.
- حدود آستانه :حدود آستانه داراي كاركرد دوگانهاي هستند؛ نخست آنكه مي توانند وقوع يک رويداد ناشي از يک تطابق الگو يا شرايط ديگر انجام عمليات مربوط به قانون را تنها بعد از چند مرحله تكرار امكانپذير كنندبه اين ترتيب از انجام عمليات ربوط به قانون تا رسيدن به آستانه مورد نظر جلوگيري كنند .به طور مثال تلاش ناموفق براي ورود به سيستم بعد از سه بار، ميتواند يک اقدام مشكوک باشد.كاركرد ديگر حدود آستانه ايجاد محدوديت بر روي دفعات اعمال عمليات مربوط به قانون است .به اين ترتيب از وقوع تعداد نامحدود هشدار و عمليات هاي زمانبر دسترسي به پايگاه داده، جلوگيري ميشود.

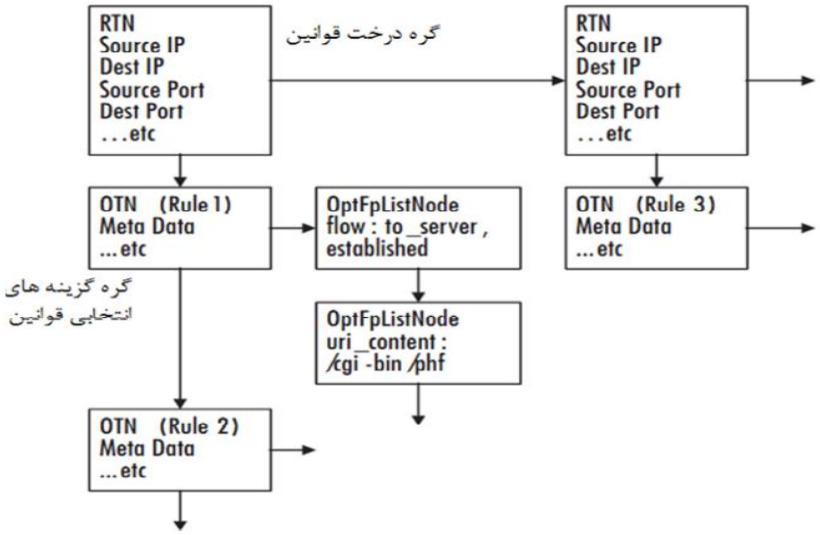
بارگذاري اوليه اسنورت

در موقع بارگذاري اوليه نرم افزار؛ سه فاز اصلي طي مي شود .نخست پارامترهاي ورودي خط دستور تفسيرمي شوند و گزينه ها و پيكربندي مورد نظر براي اجرا مشخص ميشود .در مرحله دوم پيكربندي مورد نظر براي اجرا از روي فايل پيكربندي خوانده ميشود .اين فايل شامل اطلاعات پيكربندي ديگري است كه درخط دستور وجود ندارند از جمله : چندين متغير پرچم براي فعال شدن بخشهاي خاص، مقادير پيكربندي، متغيرهاي پيكربندي پيشپردازشگر، واسطهاي خروجي و همچنين قوانين خاص .منطق تجزيه و تحليل اين فايل براي هر كدام از اجزاي ذكر شده ميتواند خاص باشد . همچنين امكان اعمال پيكربندي دلخواه بر روي اين فايل وجود دارد .مهم ترين بخش در اين مرحله، تجزيه و تحليل منطقي قوانين است.

تجزيه و تحليل منطقي قوانين

هر قانون اسنورت شامل دو بخش است :سرآيند، و يک ليست از گزينه هاي اختياري بخش سرآيند نوع قانون(هشدار، نسخه برداري، گذردهي و...)، پروتكل مربوط به قانون (*IP*، *TCP*، *UDP*و...) *و نيز آدرس منبع و مقصد و شماره پورت هاي مربوطه و همين طور سوي حركت بسته است .هر كدام از اين قوانين مي توانند بر روي بسته هاي ورودي* اعمال شوند.بخش جالب توجه و مهم در مورد قوانين در فاز بارگذاري اسنورت، ساختمان داده درختي است كه از تجزيه و تحليل منطقي قوانين بدست ميآيد .به اين منظور اسنورت از سرآيند هر يک قوانين گره درختي مربوط به

قانون (*RTN*) را می‌سازد شامل آدرس *IP* منبع و مقصد و جهت حرکت بسته‌ها هستند .از روي اطلاعات انتخابي هر كدام از قانون ها گرهي به نام گره درختي گزينه انتخابي يا (*OTN*) ساخته ميشود .يكي از گزينههايي كه در اين گرهها قرار داده ميشود؛ گزينههاي انتخابي قوانين هستند.گره هاي گزينه هاي انتخابي معمولا در غالب گره هاي قانون مربوطه ذخيره ميشوند.به هنگام اجراي برنامه، اسنورت تمام قوانين فعال موجود را خوانده شده و در ساختار دادهاي مناسبي درحافظه قرار ميگيرند .بعد از اين مرحله قوانين براي انجام عملكلاسبندي به ردهبند قوانين سپرده مي شوند؛ تا به زيركلاسهايي طبقهبندي شوند .مرحله شناسايي رده يا كلاس قانون مورد نظر براي هر كدام ازبسته ها يا جريان هاي ورودي نيز صورت مي گيرد؛ و هر بسته براساس پارامترهاي يكتاي آن به رده قوانين مورد نظر انتساب داده ميشود . براي مثال، فرض كنيم كل مجموعه قوانين ۱۵۰۰ عدد باشند .اين قوانين براساس پروتكل هاي لايه انتقال و كاربرد ميتوانند به دسته هاي كوچكتري تقسيم شوند .فرضا ۵۰۰ قانون مربوط به *HTTP* كلاينت به سرور و پنجاه قانون ديگر به *HTTP* سرور به كلاينت اختصاص يابند.(شكل زير ساختار قوانين اسنورت مي‌باشد.)



دريافت سيگنالها و فرامين در حين اجراي برنامه

يكي ديگر از مسائلي كه در زمان اجراي برنامه توسط اسنورت به آن پرداخته مي شود؛ عبارت است ازمديريت دستورات و وقفه هايي كه از طرف كاربر و واسط كاربري برنامه به آن وارد مي‌شود.

مراحل پردازش بر مبناي بسته ها در اسنورت

يكي از مهمترين و جذابترين بخشهاي معماري اسنورت مسائل مربوط به پردازش بستهها ميباشد .نرم افزار اسنورت سيستمي است كه كلا بر مبناي بسته هاي شبكهاي (كه ورودي آن محسوب مي شوند) عمل مي‌كند .به طوركلي نوع داده هاي بسته كه در اسنورت مورد پردازش قرار ميگيرد طي مراحلي از بخش هاي مختلف سيستم عبور ميكند .در ابتدا بسته از لايه شبكه دريافت مي شود .بعد از واگشايي بسته وسازماندهي نوع دادهاي بسته در حافظه، بسته به واحد پيش

پردازنده فرستاده مي شود؛ تا ترافيک نرمال شبکه توليد شود .سپس بسته به موتور تشخيص سپرده ميشود تا با تمام قوانين بارگذاري شده مطابقت داده شود .در نهايت بسته به واحد هشدار و خروجي ارسال ميشود تا در صورت صورت تطابق با قوانين اعلان هشدار داده شود و بسته در خروجي سيستم ثبت شود.نرم افزار اسنورت هر کدام از شبه بسته ها را به عنوان ساختار دادهاي نماينده بستههاي شبکه توليد ميکند.پيش پردازندهی *stream4* بستههاي لايه انتقال را بازسازي ميکند؛ در اين بين بسته هاي تکراري حذف و بسته هايي که به دليل شرايط شبکه خارج از ترتيب رسيده اند يا آنکه قطعه قطعه شده اند به حالت نرمال برميگردند .اين بسته هاي ميتوانند به عنوان يک شيئ با ساختار خاص که بيانگر رخدادهاي خاص درترافيک باشد نيز تعريف شوند و به لايه هاي بعدي نرم افزار ارسال شوند .براي مثال پيش پردازنده تشخيصدهنده حمله پويش پورت مي توانند نشانه هايي را در اين شبه بسته ها قرار دهند که نشان دهنده رخداد حمله پويش پورت باشد.

- دريافت بسته

بعد از آنکه اسنورت مراحل بارگذاري را انجام داد وارد مرحله دريافت بستهها ميشود. دريافت بسته ها از لايه

شبکه از طريق يک ريسمان به نام *Interface* که در *src/snort*.c قرار داد انجام میشود. برای اين کار از واسط برنامههای کاربردی *libpcap* استفاده میشود اين *API* کتابخانه نرم افزاري اين امکان را فراهم مي کند که تمام بسته هاي عبوري از کارت شبکه اي که به عنوان استراق سمع کننده در مسير ترافيک شبکه قرار داده شدهاند دريافت شوند. *Libpcap* براي هر کدام از بسته هاي شبکه اطلاعات زير را فراهم ميکند:

- زمان دقيق دريافت بسته از واسط شبکه بر حسب ميکروثانيه
- طول بسته بر روي رسانه انتقال
- تعداد بايت هاي بسته دريافتي
- نوع لينک(مثلا اترنت يا شبکه بي سيم يا ....)که بسته از آن دريافت شده است.
- يک اشاره گر به داده هاي اصلي هر بسته

نرم افزار اسنورت در حالت بي اثر از واسط برنامه کاربردي *libpcap* براي دريافت بسته ها استفاده مي کند. در حالت درون خط دو *API* توسط اسنورت پشتيباني مي شوند *jpfw* و *ipq* چون اسنورت براساس شبهبستههای *libpcap* نوشته شده است در اين موارد هم شبه بسته ابتدا به فرمت بسته *libpcap* درآمده سپس به لايهی بالاتر ارسال میشود. بعد از مرحله دريافت بسته تابع *process packet* فراخواني مي شود که تمام کارهاي مربوط به پردازش بسته در آن صورت ميگيرد؛ اين کارها که در ادامه به آنها مي پردازيم عبارتند از واگشايي بسته، پيش پردازش بسته و در ادامه بررسي تطابق با قوانين موجود توسط موتور تشخيص و در نهايت ماژول هاخروجي و سامانه نسخه برداري از رويدادها و هشدارهي.

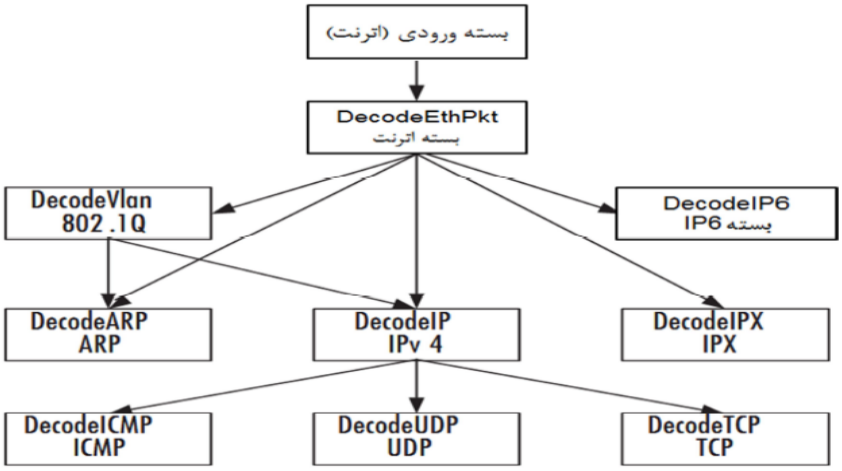
بايد توجه داشت در حالت کلي نرمافزار اسنورت در هر زمان توانايي پردازش يک بسته را داراست؛ به همين جهت؛ در صورتي که پردازش بعضي بستههاي زمان زيادي صرف کند و ترافيک شبکه بالا باشد؛ اين امکان وجود دارد که بعضي

بسته ها از دست بروند.از دست رفتن بسته ها در حالت بي اثر ممكن است موجب غفلت از بروز بعضي حملات شود . همچنين در حالت درون خطي نيز اين مساله به دليل عدم امكان ارسال مجدد به موقع بسته از پورت خروجي، موجب مشكلات شبكه اي مانند ايجاد گلوگاه و تاخير زياد و نهايتا افت كاركرد شبكه شود. از دست رفتن بستهها بر روي بازدهي و عملكرد خود سيستم اسنورت هم اثر دارد .به دليل اينكه پيش پردازنده هاي اسنورت در صدد ايجاد ترافيک نرمال براساس بسته هاي مربوط به هم و متوالي از ورودي هستند؛ لذا در صورتي كه بسته اي خارج از ترتيب برسد يا آنكه در مسير شبكه از دست رفته باشد؛ تا زمان به سر رسيدن زمان زمان سنج ، بسته بافر ميشود و منابع حافظه اي و پردازشي را نگه ميدارد .اين مساله به نوبه خود ميتواند سيستم را بيشتر مشغول كند و باعث شود بسته هاي بيشتري از دست بروند؛ به اين ترتيب اين روند حلقهاي به كاهش كارايي اسنورت منجر شود.به دليلي اين مشكلات شبكهاي لازم است فرآيندهايي براي بازرسي ترافيک شبكه و نرخ بسته هاي ازدست رفته صورت گيرد؛ مضاف بر اينكه اين خاصيت خود ميتواند امكان بروز حملات جلوگيري از سرويس را در سيستم محيا كند؛ به اين ترتيب كه حمله كننده با ارسال بسته هاي خارج از ترتيب و نامنظم عملكرد اسنورت را دچار اشكال كند .در صورت استفاده از اسنورت در حالت درون خطي، اطلاعات مربوط به از دست رفتن بسته ها نميتواند خيلي دقيق باشد زيرا *API* مورد استفاده براي دريافت بسته ها (همان *jpfw* و *jpq*) مانند *libpcap* امكان نگهداري بسته هاي خارج از ترتيب و دستيابي به اطلاعات آنها را ندارد .به اين دليل بايد روش ديگري براي پيدا كردن تعداد بستههاي از دست رفته در نظر گرفته شود.پردازش بسته ها يكي از جاهايي است كه ممكن است بسته ها از دست بروند؛ از جاهاي مهم ديگري كه ممكن است باعث شود بسته ها از دست بروند؛ دريافت بسته ها از واسط شبكه و انتقال آن به اسنورت توسط واسط برنامه‌ی كاربردی *libpcap* است. به اين دليل مطالعات زيادي براي افزايش كارائي اين مرحله صورت پذيرفته است .در ملاحظات اوليه از سازوكار نگاشت حافظهاي براي انتقال بسته ها استفاده شد .در پياده سازي هاي بعدي نشان داده شد كه استفاده از يک سازوكار مبتني بر وقفه ميتواند كارائي را افزايش دهد .در اين حالت *API* براي هر كدام از بستهه اي دريافتي يک وقفه توليد ميكند و به اين ترتيب هسته را از وجود بسته جديد مطلع مي كند .بعد از اين مرحله هم نشان داده شد استفاده از راي گيري بازدهي بهتري دارد .دراين حالت هسته سيستم عامل به طور دوره اي چک ميكند كه آيا بسته جديدي وارد شده است يا نه .بعد ازاين مراحل در سال *2004* نشان داده شد كه استفاده از ساختار يک بافر حلقه‌ای می‌تواند به مراتب كاركرد بهتری داشته باشد.

- واگشايی بسته‌ها

بعد از آنكه بسته به از واسط برنامه كاربردي به اسنورت رسيد؛ وارد بخش واگشايي بسته مي شود. اين كدگشايي در مراحل مختلف بسته به نوع بسته در هر كدام از لايههاي شبكه صورت مي پذيرد. نرم افزار اسنورت تعدادي از پروتكلهاي لايه پيوند دادهها را پشتيباني ميكند؛ از جمله *Token ,Etherne* 802.1 *Ring FDDI, Cisco HDLC, SLIP, PPP*. در ادامه اسنورت مي تواند پروتكل لايه هاي بالاتر را نيز پشتيبانی كند از جمله *ICMP*، *UDP* ، *TCP*، *IP* و .... همين طور براي ديگر پروتكل ها هم كدگشاهايي وجود دارد. براساس هر نوع كدگشايي كه عمل واگشايي بسته انجام شود؛ عمل يكساني بايد انجام شود .اين كار اين است كه اشاره گرهاي مختلفي به بخشهاي مختلف بسته اصلي در ساختار داده اي شبه بسته مقداردهي

شوند .در اين روند صحت و درستي بسته هم چک ميشود؛ و ناهنجاريها و بستههايي با وضعيت نادرست گزارش ميشوند .نتيجه فرآيند کدگشايي يک شيئ از ساختار دادهاي شبهبسته است که تمام اشاره گرهاي آن به بخشهاي مهم بسته مقداردهي شدهاند .اين بسته ميتواند توسط بخش هاي ديگر اسنورت مورداستفاده قرار گيرد. به خاطر آنکه بيشتر شبکههايي که اسنورت روي آنها نصب ميشوند مبتني بر اترنت هستند؛ ما گراف فراخواني توابع مختلف براي واگشايي بسته اترنت را آوردهايم .در هر مرحله براساس پروتکل لايه بعد تابع واگشايي خاص مورد استفاده قرار ميگيرد .بسته دريافتي به تابع *DecodeEthPKt* سپرده ميشوند .سپس با قرار گرفتن ساختار اترنت، مراحل بعدي کدگشايي روي بسته انجام مي شود .اين کدگشايي در مراحل مختلف انجام ميشود .اين ساختار دادهاي بين تمام بخشهاي اسنورت مشترک است و شامل فيلدهاي مختلفي است که با پيشرفت نرم افزار به آن اضافه شده اند .از جمله اين فيلدها مي توان به تعقيب کننده جريان *TCP* تعقيبکننده قطعات جدا شده بسته هاي *IP* و تعقيب کننده جريان داده اشاره کرد.



مراحل واگشايی بستهی اترنت در اسنورت

پيش پردازنده ها

بعد از اينکه بسته واگشايي شد؛ ساختار بسته به پيشپردازنده سپرده ميشود؛ تا طيف متنوعي از عمليات مختلف روي آن انجام شود .از جمله اين عمليات، عمليات نرمالسازي ترافيک (با درست کردن ترتيب بسته هايي که خارج از ترتيب دريافت شدهاند) اعمال روشهاي تشخيص غير مبتني بر قوانين و نيز روش هاي روش هاي تشخيص مبتني بر آمار مي باشد.

اعمال قوانين توسط موتور تشخيص

موتور تشخيص وظيفه تطابق بسته دريافتي را با تمام قوانين موجود به عنوان قوانين ورودي را دارد .اسنورت با شروع از ابتداي درخت قوانين (که در مرحله بارگذاري نرمافزار از روي قوانين ورودي ايجاد شده است) درطول شاخه هاي آن پيش مي رود و بسته را با قوانين مقايسه ميکند و در صورت پيدا کردن قوانين متناسب، آن قانون را روي بسته اعمال ميکند .در

هر گرهي كه مطابقت با گره درخت قانون (*RTN*) احراز شد اسنورت به سراغ گره‌های درختی انتخابی (*OTN*) می‌رود و مطابقت بسته با آن شرايط را هم بررسي مي كند .در اين مورد اسنورت ليست قوانين كشف شده را براي اجرا به صف قوانين اضافه مي كند صف رويدادها منجر به دو قابليت اساسي در اسنورت شده اند :اول اينكه در حالتي كه چندين قانون مطابق بسته يافت شدهاند؛ كدام يک قانون ها اجرا شود؛ دوم اينكه براي هر كدام از بسته ها اين امكان وجود داشته باشد كه چند هشدار متفاوت توليد شود .در نسخههاي قبلي اسنورت، كه صف رويداد وجود نداشت هر كدام از قوانين به محض كشف روي بسته اعمال و هشدار مربوطه توليد مي شد؛ هر چند كه هشداراهميت كمي داشت يا آنكه ترتيب لحاظ شده براي اجراي قانون مناسب نبود .با وجود صف رويداد بعد ازآنكه هر كه قانون توسط موتور تشخيص براي اجرا كشف شد؛ يا آنكه واحدهاي پيش پردازنده يا واگشاي بسته قصد توليد هشداري را داشتند؛ ليست تمام اين هشدارها و رويدادها به صف قوانين اضافه مي شود وهنگامي كه اين صف پر شد يا كار مطابقت بسته با كل قوانين را انجام داد؛ اسنورت بر اساس مكانيزمي بهترين و با اولويت ترين رويداد را انتخاب كرده و هشدار مربوطه را توليد ميكن د يا آن كه آن رويداد را با توجه به بقيه هشدارها ناديده ميگيرد .اينكه كدام يک از قوانين صف زودتر براي اجرا انتخاب شوند؛ قابل پيكربندي است؛ اين قانون ميتواند قانوني با طولانيترين رشته كشف شده؛ و يا قانوني با اولويت بالاتر باشد.همچنين اين كه براي هر بسته بيش از يک هشدار توليد شود قابل پيكربندي است.در مورد قانونهاي پيدا شده توسط موتور تشخيص، بعد از اينكه قانون اعمال شد و قبل از اين كه پلاگين هاي خروجي اعمال شوند؛ دو كار اساسي بايد انجام شوند .يكي تعيين و بروز رساني حدود آستانه و ديگري عمليات برداشت.

- تعيين و بروز رساني حدود آستانه

بعد از اينكه هشداري توليد شد؛ موتور تشخيص وارد مرحله حدود آستانه ميشود .با استفاده از تعيين حدود آستانه تنظيم كننده قوانين ميتواند تعداد رويدادها و هشدارهايي را كه به وسيله قوانين توليد مي شوند محدود نمايند .اين كار به سه طريق ممكن است .نخستين روش محدود كردن است؛ اين روش تعداد رويدادهايي را كه توسط هر يک از قوانين ميتوانند فراخواني شوند را محدود مي كند .با محدودكردن يک قوانيني كه تعداد رخداد آنها در واحد زمان زياد است؛ از بروز حملات جلوگيري از سرويس (*DOS*) جلوگیری می‌شود اين مساله از بروز هشدارها و اعمال قوانين به پلاگين هاي خروجي در مواردي كه ممكن است تعداد هشدارها در يک ساعت، از مقدار معيني بگذرد جلوگيري ميكند .حدود آستانه ازبروز هشدارهاي زياد و جلوگيري كردن از توجه به هشدارهاي مم در ميان آنها جلوگيري مي كند .براي مثال استفاده از اين قانون در فايل پيكربندي تعدار هشدارهاي هر يک از قوانين در هر دقيقه محدود به يک هشدار مي شود:

threshold gen_id 1, sig_id 0, type limit, track by_src, count 1, seconds 60

دومين روش مقدار دادن به حدود آستانه است .در اين روش حد آستانه تعداد هشدارهايي است كه بايد بروز كنند تا يک قانون بتواند فراخواني شود .مقدار آستانه امكان نوشتن قوانيني را ميدهد كه حملات تست همه حالات ممكن را تشخيص مي دهند . اين حملات براي كشف شناسه عبور و دسترسي به منابع حساس صورت ميگيرد و طي آن تعداد زيادي از حالات ممكن شناسه عبور، امتحان شده تا در نهايت شناسه عبور كشف شود .وقتي كه حد آستانه براي تعداد دفعات ناكامي در ورود به

سيستم مقداردرنظر گرفته شود؛ تلاش اول براي وارد كردن شناسه و ورود به سيستم هشداري توليد نمي كنند؛ ولي تلاش هاي بعد از آن توليد هشدار ميكنند كه به نوبه خود ميتواند پاسخ مناسب، مانند ايجاد تاخير دردسترسي كاربر را در پي داشته باشد .اين قانون حد آستانه تلاش براي ورود به سيستم را طوري مقداردهي ميكند كه بعد از تلاش ناموفق، تنها بعد از دقيقه، امكان تلاش مجدد وجود داشته باشد.

threshold:type threshold, track by_dst, count 5, seconds 60;

- بازدارندگي از قوانين خاص

فايل پيكربندي قانون مشخص شده با شناسه *sid* برابر با 1852 در صورتی كه آدرس *IP* مقصد برابر با 10.1.1.1 باشد فراخوانی نمی‌شود.

suppress gen_id 1, sig_id 1852, track by_dst, ip 10.1.1.1

بخش نسخه برداري و ماژولهاي خروجي

بعد از رسيدن بسته به موتور تشخيص و تطابق آن با قوانين موجود، بسته به واحد نسخه برداري و ماژول هاي خروجي مي رسد در نسخه‌ی 1.x نرم افزار اسنورت اولين قانوني كه مطابقتش با بسته ورودي احراز ميشد؛ بلافاصله روي آن بسته اعمال ميشد و هشدار مربوطه نيز توليد ميشد .در نسخههاي امروزي تمام اين قوانين در يک صف به نام صف رويدادها قرار ميگيرند تا اقدام مناسب روي آنها انجام شود و هشدارمناسب آن توليد شود.

ساختار داخلي موتور تشخيص اسنورت

قابليت اصلي اسنورت براي تشخيص حملات با اعمال قوانين توسط موتور تشخيص صورت مي گيرد .اين موتور براساس نوع قوانين تشخيص داده شده نوع حمله را تشخيص ميدهد و هشدارهاي لازمه را توليد ميكند .در اين قسمت ابتدا به گزينه هاي انتخابي قوانين مي پردازيم سپس به مطابقت دهنده الگو مي پردازيم

گزينه هاي انتخابي قوانين خاص مربوط به موتور جستجو

بسياري از قوانين برمبناي مقايسه هاي سادهاي در فيلدهاي ساختار شبه بسته هستند؛ مانند پيدا كردن رشته كاراكتري خاصي در بسته؛ با اين حال بسياري از قوانين داراي گزينه هاي انتخابي به مراتب پيچيده تري هستند .از جمله مهم ترين گزينه هاي انتخابي مورد استفاده در قوانين اسنورت مي توان به *content*، *flowbits* و *PCRE* ، *bytejump*، *bytetest* اشاره كرد كه در قسمت ساختار قوانين اسنورت به توانايي آنها پرداخته شد.

موتور بررسي تطابق الگو

- ساخت و پيكربندي موتور تطابق الگو

پيكربندي موتور تشخيص با ساختن ساختار درختي شامل قوانين آغاز ميشود .هدف از ساخت چنين درختي اين است كه با گروهبندي قوانين، تعداد قوانيني كه در مورد هر بسته مورد مطابقت قرار مي گيرند كاهش يابد .موتور تشخيص با گروه بندي قوانين، برپايه شماره پورت مقصد شروع به كار مي كند.

- بررسي و مقايسه الگوريتم هاي تطابق الگو با يكديگر

موتور تطابق الگوي اسنورت از سه الگوريتم پشتيباني مي كند: *Aho-Corasick* يا *Manaber* modified *Wu-*، *ac* يا *lowmem* يا *low memory key- word trie* و *mwv* البته نتيجه تطابق در همه اين الگوريتمها مشابه است؛ با اين حال از نظر زمان و حافظه مصرف و زمان بارگذاري اوليه قانون ها در حافظه تفاوتها و بده بستان دارند .در زير ما يک نتيجه يک مقايسه بين اين الگوريتمها را آورده‌ايم .دراين مقايسه از يک ترافيک نمونه ضبط شده، به حجم ۵.۱ گيگابايت استفاده شده است .در اين مقايسه مقدار حافظه اي كه درست بعد از بارگذاري اوليه مصرف شده است، زماني كه صرف پردازش تمام فايل *pcap* شده است، و نيز زماني كه صرف بارگذاري اوليه اسنورت شده است؛ آورده شده است .اين شبيه‌سازی با استفاده از سري قوانين منتشر شده به همراه نسخه نرم افزار 2.6 كه در آگوست 2006 منتشر شده است انجام شده است .در حالت اول با ۴۹۵۵ قانون كه به طور پيش فرض فعالند و در حالت دوم

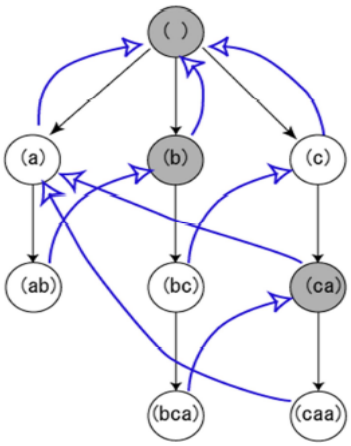
تمام ۶۵۹۲ قانون فعالند .از يک سيستم كامپيوتر پنتيوم ۳ با كلاک پردازنده ۵۵۰ مگاهرتز و ۱ گيگابايت حافظه اصلي براي اين شبيه سازي انجام شده است.

مقايسه انواع الگوريتم‌های تطبيق الگوی موجود در اسنورت

	برای اجرا روی ۴۹۹۵ قانون			اجرا روی ۶۵۹۲ قانون		
الگوريتم	مصرف حافظه (MB)	زمان بارگذاری قوانين(ثانيه)	زمان پردازش کل بسته‌ها	مصرف حافظه (MB)	زمان بارگذاری قوانين(ثانيه)	زمان پردازش کل بسته‌ها
ac	۱۴۱	۲۲.۲	۴۰۰.۸	۴۹۳	۱۷۶.۴	۷۶۶.۶
acs	۴۹	۲۰.۲	۴۹۰.۴	۱۸۳	۱۷۹.۰	۹۳۶.۳
ac-std	۲۴۳	۷.۷	۳۹۹.۶	۸۳۶	۳۴.۰	۸۴۱.۴
ac-banded	۷۶	۲۰.۰	۴۰۸.۹	۳۲۳	۱۷۸.۶	۷۸۱.۷
ac-sparsebands	۵۳	۱۹.۷	۴۳۵.۹	۲۳۶	۱۷۸.۵	۷۸۷.۲
mwm	۶۰	۴.۴	۴۲۱.۰	۱۰۲	۶.۲	۷۹۵.۶
lowmem	۳۵	۳.۹	۴۵۸.۹	۵۷	۵.۱	۸۳۴.۴

Dictionary {a, ab, bc, bca, c, caa}

Path	In Dictionary	Suffix Link	Dict Suffix Link
()	-		
(a)	+	()	
(ab)	+	(b)	
(b)	-	()	
(bc)	+	(c)	(c)
(bca)	+	(ca)	(a)
(c)	+	()	
(ca)	-	(a)	(a)
(caa)	+	(a)	(a)



یک نمونه ماشین حالت تولید شده از روی الگوریتم Aho–Corasick

موتور تشخیص الگوریتم پویا

این ویژگي جدید از نسخه ۶.۲ به اسنورت اضافه شده است؛ این موتور قابلیت این را ایجاد میکند تا بتوان نوشت و از آن در برنامه استفاده کرد .این قوانین که قوانین شیئ مشترک نیزقوانین پویاي که به زبان C نامیده مي شوند؛ دو قابلیت مهم را ایجاد میکنند؛ نخست آنکه قابلیت هاي کاربردي پیچیده تري را نسبت به امکانات بیشتري قوانین متني ایجاد میکنند؛ به این ترتیب با استفاده از قابلیت قوي برنامه نویسي زبان C براي توصیف حالاتي که به سادگي با قوانین معمولي ممکن نیست فراهم میشود.دومین قابلت مهم نیز این است قابلیت ارائه قوانین به صورت جعبه سیاه را مي دهند .این قضیه از نظرملاحظات امنیتي براي سازمانها اهمیت فوقالعاده دارد؛ زیرا سازوکار کشف قوانین شیئ مشترک کامپایل شده به سادگي ممکن نیست. در مرحله نخست قبل از آنکه قوانین پویا اضافه شوند؛ باید که در هنگام راه اندازي برنامه اسنورت گزینه ی فعالسازی- enable-dynamicplugin را در خط فرمان اضافه کرد .همین طور براي استفاده از این قوانین باید با استفاده از قوانین تنه آنها را فعال کرد .قوانین تنه امکان فعال و یا غیر فعال کردن قوانین پویا را فراهم مي کنند. واسط برنامه کاربري مربوط به تشخیص پویا امکان تعریف قوانین پویاي اسنورت را با استفاده از ساختمان دادهی زبان C را فراهم میکند.

در زیر نمونهاي از ساختار منتسب به قانون را در واسط برنامه کاربري پویاي استاندارد اسنورت مي بینیم.

در این نمونه مشخصات عمومي قانون از جمله شناسه امنیتي قانون (sid) پروتکل مربوط به قانون و آدرسهای IP و پورت مربوط به منبع و مقصد و همچنین تابع ارزیابي مربوط به قانون را مي بینیم.

```
Rule sid1000000 =
{
/* IPInfo */
{
        IP_PROTO_TCP,           /* Protocol */
        "$EXTERNAL_NET",        /* Source IP */
        "any",                  /* Source port */
        RULE_DIRECTIONAL,       /* or RULE_BIDIRECTIONAL */
        "$HOME_NET",            /* Destination IP */
        "any"                   /* Destination port */
},
        /* RuleInformation */
{
        3,                      /* GID */
        1000000,                /* SID */
        1,                      /* Revision */
        "misc-activity",        /* Classification */
        0,                      /* Priority */
        "Example dynamic rule 1",  /* Message */
        NULL                    /* References */
},

NULL,   /* Rule options */
NULL,   /* eval function */
0,      /* Internal use */
0,      /* Internal use */
0,      /* Internal use */
NULL    /* Internal use */
```

ساختار پيش پردازنده هاي اسنورت

در نسخه هاي ابتدايي اسنورت كار پيشپردازندهها به نرمال كردن ترافيک محدود مي شد؛ ولي امروزه كاربردهاي گسترده تري براي آن وجود دارد؛ از جمله ميتوان به نرمال سازي پروتكل ها و عمليات تشخيص ناهنجاري ها اشاره كرد .به اين ترتيب پيش پردازنده ها هم هشدارهاي مخصوص خود را توليد ميكنند .از اين نظر پيشپردازنده ها مانند موتورهاي تشخيصي هستند كه قابليت تشخيص ناهنجاري و توليد هشدار را قبل از رسيدن بسته به موتور تطابق الگو دارا هستند پيش پردازنده ها مانند موتور تشخيص يا به عبارتي موتور قانون مبتني بر قوانين نيستند؛ آنها به طور مستقل كار ميكنند و هدف ايشان اين است كه موتورتشخيص ساده ترين ترافيک را دريافت كند .در شكل زير جايگاه پيش پردازنده ها و انواع آنها و عملي كه هر كدام از آنها انجام ميدهند؛ آورده شده است.

پیش‌پردازنده‌های مهم اسنورت

براي رسيدن به ترافيک نرمال نياز به بازسازي بسته ها داريم .پروتکل TCP/IP پروتکلی است پايدار و قابل اطمينان براي ارسال مطمئن بسته ها .با استفاده از اين پروتکل امکان دريافت بسته هاي خارج از ترتيب وتکه تکه شده با اندازه‌هاي مختلف، وجود دارد .اين زمينه راهکار خوبي براي حملهکننده است .اسنورت سه نوع پيش پردازنده مهم دارد .پيش پردازنده هاي مهم اسنورت براي بازسازي بسته ها و جريان ها عبارتند از: frag2،frag3و stream4 اسنورت براي نرمال سازي پروتکل هاي لايه کاربرد هم پيش پردازنده هاي مخصوصي را ارائه ميدهد .همچنين پيش پردازنده هايي براي کشف ناهنجاري ها و تشخيص مبتني برناهنجاري وجود دارد .گروه ديگري از پيش پردازنده ها هم براي پويش عملکرد و بهره وري خود پردازه اسنورت در حال فعاليت هستند .در ذيل به نمونه هاي مهم آنها و اهميت و کاربردي که دارند و همين طورامکانات و گزينه هاي پيکربندي آنها مي پردازيم.

پیش پردازنده frag2

اين پيش پردازنده براي تشخيص حملات Dos مبتني بر قطعه قطعه سازي بسته ها مفيد است .در اين حملات معمولا مهاجم با ارسال بسته هاي قطعه قطعه شده متعدد؛ در صدد است از آسيب پذيري پشته پروتکلی IP استفاده نمايد .بسياري از سيستم ها با دريافت بسته هاي قطعه اي که آفست آن ها نادرست مقداردهي شده و يا باعث مي شود داده بسته قبلي دوباره نويسي شود؛ دچار اشکال ميشوند؛ و ميتوان از اين راه حتي آنها را ريست کرد يا باعث آسيبپذيري ديگري در آنها شد .به طور معمول بستههاي قطعه قطعه شده نبايد با هم اشتراکي داشته باشند.

پیش‌پردازه‌ی frag3

اين پيش پردازنده، نوعي پيش پردازنده مبتني بر هدف ، براي مديريت بسته هاي قطعه قطعه سازي شده است؛كه به دو منظور اساسي زير طراحي شده است:

- اجراي سريع و بدون عمليات پيچيده مديريت داده
- مدل كردن مقصد بسته به صورت هدفگرا براي مقابله با روش هاي فرار حمله كننده از طريق جابجايی بسته‌های دريافتی

اين پيش پردازنده از ساختارهاي دادهاي جدول درهم سازي sfxhash و ليست پيوندي، براي مديريت داخلي داده ها استفاده ميكند؛ اين امر باعث ميشود بازدهي و عملكرد قطعي و قابل تعيين frag3 در محيطهای غير قطعي و وجود قطعه قطعه سازي زياد بين بسته ها مي شود.روش هاي تشخيص مبتني بر هدف، راهكار جديدي در پردازش بسته ها در سيستمهاي تخصيص نفوذ شبكه اي ميباشد .در اين روش به جاي تعقيب روند مراوده پروتكلي و جستجو به دنبال مدلي براي كشف حملات از روي آنها؛ به دنبال مدل كردن اهداف اساسي در اهداف واقعي است. وقتي پشته هاي پروتكلي مبتني بر IP بر روی سيستم عامل ها و يا برنامه هاي كاربردي پياده سازي مي‌شود شوند؛ اين پياده سازي ها توسط برنامه نويس هايي صورت مي گيرند كه مستندات RFC مربوط به هر پروتكل را مطالعه ميكنند .معمولا در مستندات در مورد مسائل و شرايط مرزي ابهام وجود دارد و تمام جزئيات و حالات را نمي توان پيش بيني كرد .به اين ترتيب در پياده سازي هاي مختلفي كه از پشته ها صورت مي‌گيرد گاهي جنبههاي اساسي به صورت هاي مختلفي پيادهسازي ميشوند و اين براي IDS فاجعه است . در حالت طبيعي بسته ممكن است از چند مسير متفاوت برسد كه در يكي قطعه قطعه شده و ديگري نشده است . همچنين بسته هاي قطعه اي دريافتي ممكن به خاطر مسيرهاي مختلف و MTU متفاوت در مسيرهای گوناگون با آفست هاي مختلفي قطعه قطعه شده باشند .در بسياري موارد ميتوان كاري كرد كه براساس پيادهسازی‌های مختلف كه در لايه‌ی IP وجود دارد، بسته هاي قطعه اي با ترتيب مختلفي كنار هم قرار گيرند. در مورد انتخاب بسته قطعه اي مناسب و نحوه همبندي آن در پياده سازي هاي مختلف مانند مايكروسافت ويندوز ، لينوكس و ... اختلاف سياست وجود دارد .حتي امكان دارد مهاجم، اعداد اين آفستها را طوري قرار دهد كه آفست بسته قطعهاي كه رسيده باعث شود در ادامه بسته قطعهاي نادرستي قرار گيرد.زمان انتظار براي كامل شدن بسته قطعه اي ميتواند در اين امر موثر باشد؛ به طوري كه مثلا زمان انتظاردر اسنورت به پايان برسد ولي در مايكروسافت ويندوز يا سيستم عامل ديگر اين بستهها سرهمبندي شوند؛ در اين صورت اسنورت از اين بسته قافل شده است.براي حل اين مشكل نياز است تا انواع پيادهسازي انواع سياستهاي سرهم بندي بسته در ميزبان هاي مختلف پياده سازي شود .بعد از تحقيقاتي كه توسط تيم تحقيقاتي شركت سورس فاير در مرجع [] انجام شده است. در مورد پيادهسازي پشته پروتكلي IP بر روي سكوهاي سخت افزاري (مسيرياب هاي شبكه و ديواره آتش ساخت شركت هاي مختلف مانند سيسكو و...) و همچنين سيستم عامل هاي مختلف دسته بندي هايي صورت گرفته است كه نتيجه آن تعريف هفت حالت مختلف پياده سازي پشته پروتكلي IP و نيز نگاشت آن با سكوهاي سختافزاري و سيستم عامل هاي مختلف است اين هفت حالت عبارتند از: BSD، Solarisو Windows ، Last، First، Linux، BSD-Right نگاشت اين روش ها با بعضي توليدات تجاري بر طبق مرجع []در زير آورده شده است.

سکوی نرم‌افزاری یا سخت‌افزاری	روش پیاده‌سازی پشته پروتکلی TCP/IP
Cisco IOS	Last
HP JetDirect (printer)	BSD-right
HP-UX 11.00	First
IRIX 4.0.5F	BSD

Linux 2.2.10	linux
Linux 2.4 (RedHat 7.1-7.3)	linux
OpenBSD (version unknown)	linux

چند نمونه از روش پیادهسازي پروتکل TCP/IP بر روی سکوهای مختلف

البته اطلاعات اين جدول در معرض تغيير است .براي انجام تنظيمات و پيکربندي مناسب اسنورت بايد به راهنما و جداول و مستندات اسنورت که با هر نسخه از آن عرضه ميشود؛ توجه کرد تا سيستم کاملا بروزپيکربندي شود. در سال ۲۰۰۶ ، در مورد استفاده از ضعف پيش پردازنده frag2 در تشخيص حملات، طي آزمايشي نشان داده شد که بسته هاي قطعه اي دريافتي که در اسنورت دريافت شده و به خاطر طي سقف زماني درست سرهمبندي نمي شدند؛ باعث بروز حمله ميشوند .البته آن زمان frag3 هنوز عرضه نشده بود و اين تست بر روي آن انجام نشده بود .با انجام تمهيدات لازمه در frag3 از جمله تشخيص مبتنی برهدف در نظر گرفتن ttl سقف زماني سرهم نهي بسته ها و بررسي اشتراک اطلاعات بسته هاي سرهمبندي شده؛ frag3 که بعدا عرضه شد دچار اين مشکل نبود. در پيکربندي اين پيش پردازنده اطلاعات دامنه IP مشمول اين پيش پردازش (شامل يک يا دسته اي ازIPها) نوع سياست برهم بندي بسته هاي قطعه اي، حدود مربوط به اختلاف ttl بسته هاي قطعه اي دريافتي مربوط به يک بسته و... آورده شده است .در مورد عملکرد اين پيش پردازنده هم بايد گفت که همانند frag2 ترافيک قطعهاي دو بار بررسي ميشود؛ يک بار به محض ورود بسته، که بلافاصله به موتور تشخيص تحويل مي شود؛ بک بار هم بعد از اينکه کل قسمت هاي مربوط به بسته کامل شد.

پیش پردازنده تشخیص جریان

در اين پيشپردازنده به عنوان يک پيش پردازنده سبک، هدف اين است که مشخص شود کدام يک از عوامل با کدام عامل ديگر در حال ارتباط است؟ اين ارتباط بر روي کدام پورت در حال انجام است؟ کدام سرور وکدام کلاينت است؟از جمله پيکربنديهايي که در مورد اين پيشپردازنده قابل انجام است ميتوان به حداکثر مقدار حافظه مورد استفاده (memcap) حداکثر تعداد رديف هاي موجود در جدول جريان هاي موجود و روش درهم سازي اطلاعات جدول(با استفاده از کليدهاي بايتي يا عدد صحيح) می‌باشد.

پیش پردازنده stream4

اين پيش پردازنده به منظور توانا ساختن اسنورت در مديريت مراودات پروتكلي، به طوراختصاصی TCP/IP ايجاد شده است. در واقع هدف اين است كه هر يک از ارتباطات TCP/IP و جريان داده مربوطه مديريت شوند دو هدف اصلي كه اين پيش پردازنده طراحي شده است عبارتند از :نگهداري وضعيت ارتباط TCP و بازسازی جريان ارتباطی TCP

نگهداري وضعيت ارتباط TCP

پروتكل TCP يک پروتكل مبتني بر ارتباط است .بنابراين در طي مراحل تقاضاي ارتباط، ارتباط انجام شده و يا تقاضاي خاتمه ارتباط صورت ميپذيرد .در مقابل اينترنت يک شبكه ديتاگرام است؛ مسيريابها و ديواره هاي آتش موجود بدون حفظ وضعيت مربوط به هر ارتباط و نيز بستههاي پيشين ردوبدل شده، فقط براساس پرچم هاي بسته فعلي، وضعيت ارتباط را تعيين ميكنند .بسياري از ابزارهاي پويش پورت با استفاده از اين ضعف با ارسال بسته هاي probe كه در آنها فقط برچمهای ack ست شده است به دنبال اين هستند تا پورتهای باز مربوط به ارتباطهای TCP/IP را كشف كنند. اين ابزارها به اين ترتيب بر خلاف معمول، بدون طی مرحلهی ايجاد TCP كه با ست كردن پرچم هاي fin و ack صورت ميپذيرد؛ مي توانند از ضعف ديوارههاي آتش كه وضيت ارتباط را حفظ نميكنند استفاده كرده و در ارتباطاتي كه قبلا از طرف ديواره آتش مجاز دانسته شدهاند؛ دخالت نمايند.خصوصيت بي حافظه بودن و نگه نداشتن وضعيت ارتباطات در جريان، يک ضعف بزرگ براي ابزارهاي تشخيص نفوذ است .در گذشته اين ابزارها قبل از بروز هشدار هيچ گونه سابقه و نشانهاي از بروز حمله را نگه نميداشتند و تنها به اعمال قانون در مورد بسته هاي فعلي ميپرداختند .ممكن است آن ها زمان زيادي راصرف پرداختن به اعمال قوانين در مورد بسته هايي كنند كه در واقع مربوط به جريان نباشند .در سال 2001 شخصی به نام جيووانی ابزاری به نام stick را ارائه كرد. اين ابزار حمله تلاش دارد ابزار تشخيص نفوذ را دچار انبوهي از هشدارها كند.

اسنورت در ابتدا ابزاري مبتني بر بسته بود كه تنها قوانين را بر روي بسته ها اعمال مي كرد و وضعيت ارتباطات را نگه نميداشت .اين اقدام باعث شد تا در همان سال پيش پردازنده stream4 برای اولين بار به اسنورت اضافه شد. اين پيش پردازنده باعث ميشود تا اسنورت بسته هاي نامربوط به ره جريان را كشف كند و آن را به عنوان نشانه حمله پويش پورت اعلام كند .از نسخه 1.9 اسنورت به بعد با استفاده از كلمه كليدی flow در هر يک از قوانين TCP ميتوان به وضعيت و جهت آن ارتباط اشاره كرد. اين پيشپردازنده براي حفظ حالت ارتباط TCP داراي گزينه هاي پيكربندي متنوعي است از جمله:

detect_scans: براي فعال كردن هشدار در مورد حملات پويش پورت، كه در موردشان توضيح داده شد.
detect_state_problems: با فعال كردن اين گزينه، اسنورت انواع ناهنجاريهاي ار تباط TCP و بستههای probe نامربوط را كشف و اعلان میكند.
disable_evasion_alerts: ست كردن اين گزينه كه به طور پيش فرض غير فعال است؛ باعث ميشود تا قابليت كشف حملات مبتني بر اضافه كردن بسته در جريانها غيرفعال شود .در اين حملات، براساس ارسال مجدد بسته، بسته دوباره فرستاده ميشود؛ به اميد اينكه بسته اول ناديده گرفته شود و بسته دوم در نظر گرفته شود .گاهي نيز ممكن است حملهكننده با

ارسال بسته ريست، اميد داشته باشد كه ميزبان آن بسته راناديده بگيرد ولي IDS از پويش ارتباط دست بردارد .مهاجم مي تواند در بسته SYN داده مورد نظر خود را ارسال كند به اين اميد كه ابزار تشخيص نفوذ از اين داده صرف نظر كند. همچنين پيكربندي هاي ديگري هم در مورد اين پيش پردازنده موجود هستند؛ از جمله :روش ضبط وضعيت هر كدام از ارتباطات (بايتي يا كاركتري) محدود كردن پويش ارتباطات TCP به پورت‌های خاص، حداكثر زمان پويش ارتباط، بر اساس زمان آخرين بسته منتقل شده، حداكثر تعداد ارتباطات در حال پويش و نيز حداكثر حافظه اختصاص داده شده به اين منظور .همچنين در حالت بكارگيري اسنورت برخط، امكان حذف بستههايي كه مربوط به ارتباط خاصي نيستند؛ وجود دارد.

بازسازي جريان ارتباطيTCP

در حالت كلي حملات چند مرحلهاي با فاصله زماني، بستر مناسبي براي آسيبپذيري سيستمها و گذر از سد IDS هستند. نگه داشتن بسته هاي قبلي مربوط به يک ارتباط TCP به اسنورت امكان كشف حملاتي را ميدهد كه حمله كننده طي ارسال چند بسته مختلف قصد نفوذ به سيستم را دارد .بسياري از حملات وجود دارند كه براي تشخيص الگوي آنها بايد اطلاعات چندين بسته TCP را كنار هم بررسي كنيم .با پيدايش اين پيش پردازنده؛ اسنورت اين قابليت را دارد تا بسته هاي مختلف مربوط به يک ارتباط را براي كشف الگوي حملات، با هم بررسي كنيم. براي اين استفاده هم پيش پردازنده stream5 داراي گزينه هاي پيكربندي متنوعي است از جمله :انتخاب جهت جريان مورد بررسي، مثلا فقط سرور به كلاينت يا برعكس؛ تعيين پورتهايي كه با اين پيش پردازنده پويش ميشوند نگه داشتن بسته هاي قبلي مربوط به ارتباطهاي مختلف نياز به منابع حافظهاي زيادي دارد؛ به طور پيش فرض بعضي پورت ها از جمله پورتهاي مربوط به پروتكل هاي مهم از جمله HTTP و FTP و... برای پويش انتخاب می شوند.

پيش پردازنده stream5 بسته هاي مربوط به يک ارتباط پويش شونده را با پايان ارتباط به صورت يک شبه بسته است كه داده هاي آن به صورت يكجا به موتور تشخيص داده ميشود .در ضمن بايد توجه كرد كه هر كدام از بسته ها يک به محض رسيدن و پيش پردازش، به لايه بالاتر داده ميشوند؛ و يک بار هم بعد ازكامل شدن و نهايي شدن ارتباط .بعد از اين مرحله حافظه بازپس گرفته ميشود .البته در صورتي كه تعداد ارتباطات پويش شونده و يا حافظه در دسترس به حد آستانه برسد؛ به صورت تصادفي يكي از جريان ها انتخاب شده و اطلاعات آن به موتور تشخيص تحويل، و حافظه آن آزاد مي شود.

پيش پردازنده هاي مربوط به واگشايي و نرمال سازي پروتكل هاي مختلف

تا اين مرحله كار نرمال سازي ترافيک و ارائه آن به ترتيب صحيح انجام شده است .تشخيص مبتني بر قوانين معمولا در مورد پروتكل هايي كه داده ميتواند به صورتهاي مختلف بيان شود كارايي زيادي ندارد .براي مثال سرورهاي وب به صورتهاي مختلفي URL را ميپذيرند؛ براي مثال وب سروريis به جای كاراكتر (/) كاراكتر (|) را می‌پذيرد .

برای پروتكل TCP پيش پردازنده مخصوصي طراحي شده است كه ميتواند بسته ها را باز و ديكد نمايد؛ فيلدهاي پروتكل HTTP را پيدا نموده و آنها را نرمال ميكند .اين پيش پردازنده در حالات كاري مختلفي براي كار با انواع سرورهاي HTTP قابل پيكربندی است.

پيش پردازنده هاي مربوط به تشخيص مبتني بر ناهنجاري

- پيش پردازنده پويش پورت

يكي از مهمترين اين پيش پردازنده ها پيش پردازندهی sfportscan است براي كاركرد اين پيش پردازنده حتما بايد پيش پردازنده جريان فعال باشد؛ تا معلوم شود چه منبعي با چه مقصدي در حال ارتباط است.انواع حملات پويش پورت قابل شناسايي كه با اين پيشپردازنده قابل پيكربندي هستند؛ عبارتند از:

Portscan

در اين حملات تعداد محدودي از ميزبانها يک ميزبان را براي تعداد زيادي از پورتها پويش ميكنند؛ تا به اين ترتيب پورتهاي باز آن ميزبان را پيدا كنند.

Portsweep

در اين حمله تعداد محدودي از ميزبانها تعداد زيادي از ميزبانها را براي پيدا كردن تعداد محدودي ازپورتهاي باز پويش ميكنند .مثلا حملهكننده قصد دارد تمام شبكه را براي پيدا كردن پورت هاي ۸۰ باز،

پويش كند.

decoy_portscan

در اين حمله تعداد زيادي از ميزبان ها تعداد محدودي از ميزبان هاي ديگر را براي پيدا كردن تعداد محدودی از شماره پورتها، به منظور كشف پورتهاي باز پويش ميكنند .اين مورد شبيه portsweep است با اين تفاوت كه احتمالا حمله كننده تعداد زياد IP را جعل نموده است.

distributed_portscan

اين نوع حمله بسيار شبيه decoy_portscan است با اين تفاوت كه تعداد زيادی از پورت‌های يک ميزبان خاص توسط تعداد زيادي ميزبان (احتمالا جعلی) براي پيدا كردن پورت هاي باز پويش ميشود. اين پيش پردازنده گزينه هاي پيكربندي ديگري هم براي تعيين نوع پروتكل بررسي شده براي حمله پويش پورت (TCP،UDP،...) تعيين سطح حساسيت (برای كاهش هشدارهای نادرست) تعيين آدرس‌های IP برای بررسی (مثلا ميزبان هاي داخلي شبكه ويا ميزبان هاي خارجي) صرف نظر از آدرس هاي IP مبدا خاص(مثلا به دليل استفاده از ابزارهاي امنيتي پويش شبكه مانند Nessus) و نيز صرف نظر كردن از آدرس‌های IP مقصد خاص (مثلا به خاطر وجود سرور به وب)

پيش پردازنده هاي مربوط به پويش عملكرد اسنورت

در حالت كلي ارزيابي عملكرد اسنورت و مشاهده اثر بخشهاي مختلف بر روي عملكرد سيستم و اعمال پيكربندي مناسب در دورههاي مختلف روي آن عمل بسيار با اهميتي محسوب ميشود. پيش پردازندهاي به نام perfmonitor در دوره هاي زماني مشخص ميتواند اطلاعات مختلفي را درباره عملكرد اسنورت فراهم ميكند . اين پيش پردازنده اطلاعات مربوط به تعداد بسته هاي ردوبدل شده، اطلاعات مربوط به تعداد ارتباطات در جريان و پروتكل مربوط به هر كدام و تعداد ارتباطات صورت گرفته مربوط به هر كدام از پورتها و ... ميباشد .اين پيش

پردازنده قابليت پيكربندي براي ارائه خروجي در فايل يا كنسول،تنظيم بازه زماني بين نمونه گيري از اطلاعات بهره وري و نيز ارائه تعداد رخدادهاي كشف شده (شامل تعدادالگوهاي مقايسه شده و تعداد تطابقهاي رخ داده را دارا ميباشد.) در ادامه به بعضي قابليتهاي مهم پيش پردازه‌ی perfmonitor و ديگر پيش‌پردازه‌های مهم خواهيم پرداخت.

گزينه‌ی max

با فعالسازي اين گزينه، بهره وري بهينه سيستم بر اساس شرايط موجود و پارامترهاي اندازه گيري شده محاسبه ميشود؛ سپس اين مقدار با بهره وري واقعي اندازه‌گيري شده سيستم در حال جاري، مقايسه مي شود .اين گزينه نقش محوري در بررسي عملكرد اسنورت و پيكربندي مجدد آن در شرايط گوناگون دارد.

پيش پردازنده پويش بهره وري اعمال قوانين

اين پيش پردازنده نسخهبردار قوانين ۱، قادر است تا اطلاعات بهره وري مربوط به اجراي هر كدام ازقوانين رامحاسبه و ثبت كند .عملكرد اين قسمت شامل دو جنبه است .نخست آنكه هر يک از قوانين در چند درصدمواقع بر روي بسته ها اعمال ميشوند؛ اهميت اين ساله از آنجاست كه قرار بر اين است كه هر كدام از قوانين به نحوي نوشته شوند تا در كمترين دفعات ممكن و با كمترين تعداد هشدارهاي نادرست اعمال شوند.نكته دوم تعداد بسته هايي است كه در هر دوره زماني با قوانين مختلف مطابقت داده مي شوند؛ در اين صورت ميتوان فهميد چه مقدار از زمان اسنورت صرف تطابق بسته ها با قوانين مي شود .با انجام اين

محاسبات معلوم ميشود گلوگاه بهره وري كجاست و براي بازدهي بهتر كدام دسته از قوانين بهتر است مورد بازبيني قرار گيرند. براي پيكربندي اين پيش پردازنده پروفايل گيري از قوانين، تنظيمات متعددي وجود دارد؛ از آن جمله ميتوان به اين موارد اشاره كرد :فعال كردن نسخهبردار قوانين، مرتب سازي قوانين در خروجي (برحسب تعداد دفعات استفاده از قانون؛ تطابقهاي رخ داده با قانون مورد نظر؛ تعداد دفعاتي كه به طور متوسط، هرقانون در مورد هر بسته ها اعمال ميشود؛ تعداد دفعاتي كه به طور متوسط، هر كدام از قوانين در مورد هركدام از بسته ها مورد بررسي قرار ميگيرد و تطابق رخ مي دهد)

```
Rule Profile Statistics (all rules)
==========================================================
    Num      SID GID    Checks   Matches  Alerts      Microsecs  Avg/Check  Avg/Match Avg/Nonmatch
    ===      === ===    ======   =======  ======      =========  =========  ========= ============
      1     1054   1        10         0       0           2246      224.7        0.0        224.7
      2     2589   1         5         0       0            993      198.7        0.0        198.7
      3     3465   1        38         0       0           3706       97.5        0.0         97.5
      4     3045   1        14         0       0           1341       95.8        0.0         95.8
      5      939   1         2         0       0            172       86.2        0.0         86.2
      6     3486   1        50         0       0           4178       83.6        0.0         83.6
```

نمونه‌ای از خروجی پيش‌پردازنده پويش بهره‌وری اعمال قوانين اسنورت

پیش پردازنده پویش بهره وري عملکرد پیشپردازنده ها

بهره وري اسنورت علاوه بر زمان صرف شده براي تطابق و اعمال قوانين، به پيش پردازنده ها نيز بستگي دارد.اين پيش پردازنده بعد از پيكربندي گزينه هاي پيكربندي مختلفي را فراهم ميكند .از آن جمله مي توان به اين موارد اشاره كرد :انتخاب پيش پردازنده هاي مختلف براي عمل پروفايل گيري، روش مرتب كردن خروجي (بر حسب تعداد دفعات كه يک پيش پردازنده در دوره زماني مشخص به كار گرفته شده است؛ تعداد دفعاتي كه هر پيش پردازنده براي بسته هاي مختلف به كار گرفته شده است؛ و نيز زماني كه به طور سرجمع هركدام از پيش پردازنده ها به كار گرفته شده است.)

پيش پردازنده هاي پويا

پيش پردازند ههاي متعددي ميتوانند به صورت پويا نوشته شوند و به عنوان اجزاي پويا به هنگام اجراي اسنورت بارگذاري شوند .واسط برنامه كاربري مربوط به اجزاي پويا، وظيفه بارگذاري اين پيش پردازنده ها و ايجاد امكان ارتباط اين بخشها با برنامه اسنورت و فراخواني توابع بخشهاي ديگر اسنورت را بر عهده دارد. از جمله مهم ترين پيش پردازنده هاي پوياي اضافه شده به اسنورت ميتوان به پويش گر پروتكل SMTP وFTP_TALENT اشاره كرد.

پيش پردازنده arpspoof

اين پيشپردازنده براي مقابله با حفره امنيتي پروتكل ARP ايجاد شده است .اين وضيت به اين صورت است كه ميزبان بدخواه، با پاسخ به بسته هاي ARP مبني بر درخواست آدرس فيزيكي ميزبان خاص، آدرس خاص خود را در بسته ARP براي آن ارسال ميكند .به اين ترتيب پيام ها براي ميزبان بدخواه ارسال مي شود و همينطور امكان دارد اين ميزبان بدخواه بستهها را براي ميزبان مقصد اصلي هم بفرستد و به اين ترتيب
خود را در ميان ارتباط قرار دهد و تمام بسته هاي ردوبدل شده ميان اين ميزبان ها را بدست آورد. اسنورت اين فعاليتهاي مخرب را با استفاده از بررسي بسته هاي ARP پيدا مىكند به اين صورت كه آدرس IP و آدرس فيزيكى را در بستههاى ARP با جدولي كه در فايل پيكربندي براي آن مشخص شده است؛ مطابقت ميدهد و فعاليت اين چنيني را كشف ميكند.

مديريت حسگرهاي اسنورت و فرآيند تحليل دادههاي بدست آمده از حسگرها

ابزار اسنورت به عنوان يک سيستم تشخيص نفوذ قادر است نشانهها و الگوهاي حملات مختلف را كشف نمايد و نتيجه را به عنوان هشدار در خروجي ثبت نمايد .اسنورت از هشدار به عنوان راهي براي اعلان نتايج در خروجي استفاده مينمايد .هشدار در واقع پيغامي است كه توسط يكي از سازوكارهاي تشخيص دهنده ايجاد شده است (پيشپردازنده يا موتور تشخيص) .در اين مرحله ما با تعداد زيادي از هشدارها مواجه مي شويم؛ بنابراين نياز به ابزارهايي داريم تا بتوانيم اين داده ها را به نحوي مديريت و تحليل كنيم.

فرآيند تحليل داده

بعد از جمعآوري دادهها و هشدارهاي مختلف، از منابع مختلف، مانند خروجي ابزارهاي تشخيص نفوذ

مختلف به صورت هشدارها، نسخههاي خروجي ديوارههاي آتش، وارد مرحله تحليل دادهها ميشويم .به روند تحليل دادهها و هشدارهاي مختلف و ايجاد هموندي ميان آنها تحليل داده گفته ميشود .در اين فرآيند از ابزارهاي خاصي استفاده ميشود كه به دادههاي جمع آوري شده از منابع مختلف دسترسي پيدا كرده؛ و با

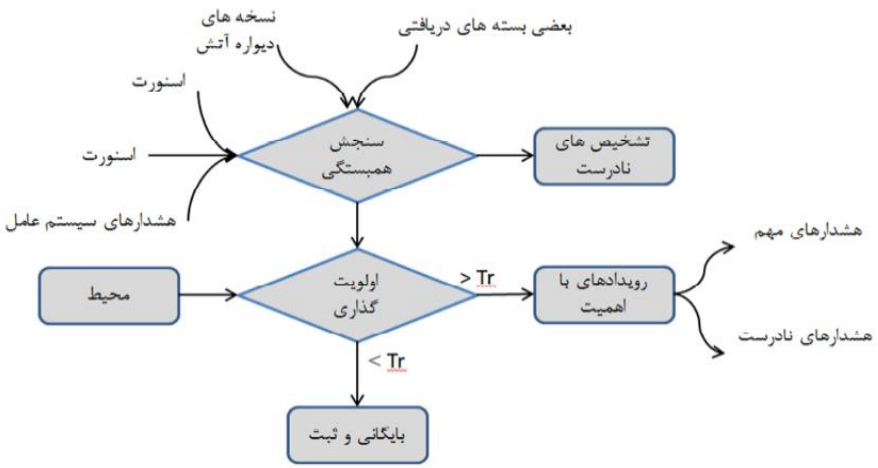
استفاده از روش هاي مختلف ميان آنها ارتباط ايجاد ميكند و آنها را گروهبندي نموده و نتيجه را به صورت مفهوم تري بيان ميكند .اهداف اصلي اين فرآيند به قرار زيرند:

- ايجاد هشدار بهنگام در مورد حملات كشف شده
- تعيين و تاييد بروز حملات، به اين معني كه در از روي نشانه هاي حمله و مراحل و رخدادهاي مربوط به آن، به طور دقيق نوع حمله تعيين شود؛ و هشدارهاي نادرست و اطلاعات نامربوط تميزداده شود.
- تعيين اينكه حمله مود نظر چه زماني اتفاق افتاده و شامل چه مراحلي بوده. چه سيستمهايي درگير آن بوده اند و چه تاثيري بر روي آنها داشته است. همينطور اينكه منشا اين حمله از كجا بوده وسابقه اين منشا و اقدامات سابق آن چه بوده اند.
- گزارش دهي كامل رويدادهاي مربوطه؛ به همراه سيستم هاي مورد تهاجم و هشدارهاي مربوطه.

براي اين منظور نياز به روشي داريم تا بين رويدادها و هشدارهاي موجود ايجاد اولويت كنيم؛ اين تعيين اولويت تابع عوامل مختلفي است؛ از جمله اينكه:

- آيا از طرف مهاجم؛ قبلا حمله اي صورت گرفته است؟
- ماشين يا سرور مورد حمله قرار گرفته چقدر اهميت دارد؟
- ميزان آسيب پذيري نسبت به حمله مورد نظر براي سرويس مورد حمله قرار گرفته در چقدر است؟
- آيا از حمله كشف شده نمونه هاي مشابه وجود داشته است؟
- با توجه به نظر مدير سيستم؛ آسيب پذيريهاي مهم در مورد كدام اجزاي سيستم تعريف شده است؟

به اين ترتيب روند پردازش داده به صورت زير مي باشد:



شکل (۲-۱۱) روند پردازش داده؛ بهره‌گیری از منابع داده‌ای به منظور شناخت و آشکارسازی رویدادها

هشدارهاي اسنورت در واقع توسط بستهها توليد ميشوند؛ هر بسته شامل اطلاعات محدودي است بنابراي امكان هشدار نادرست بسيار زياد است .براي بدست آوردن ديد سطح بالاتر در مورد رخدادهاي مختلف بايد بين رويدادها ارتباط ايجاد كنيم .ايجاد ارتباط بين هشدارهايي كه به رخدادهاي مربوط به يک منبع و يا يک ويژگي مشترک ديگر هستند؛ كمک شاياني در حذف هشدارهاي نادرست و پيدا كردن حملات احتمالي ميكند.منابع داده هاي مورد استفاده براي پردازش داده ها شامل انواع هشدارهاي توليد شده توسط ابزار تشخيص نفوذ و نسخه هاي گزارش ديواره آتش ميباشد .اين هشدارها ميتوانند از خروجي خود ابزار تشخيص نفوذ ناشي شوند يا آنكه در پايگاه داده مشترک حسگرهاي سيستم تشخيص نفوذ قرار داشته باشند .همچنيناين دادهها ميتوانند اطلاعات در مورد آسيب پذيريها باشند .به طور مثال با اعلان يک نفوذ در سيستم براي كشف آسيب پذيري مربوطه و با استفاده از ابزارهاي تست نفوذ اين آسيب پذيري تست ميشود .بعد از اين تست معلوم ميشود كه آيا آسيب پذيري گزارش شده واقعي بوده و يا يک اعلان نادرست بوده است. پايگاه داده مركزي جايي است كه تمام حسگرها ميتوانند هشدارهاي مورد نظر خود را در آن ثبت كنند تا به عنوان سابقه نگهداري شود .ابزارهاي مختلفي از واسط اين پايگاه داده استفاده مي كنند و با استفاده از اطلاعات آن، به دادهها دسترسي پيدا كرده و با استفاده از معيارهاي اولويت ذكر شده در مورد رويدادها،

رويدادهاي با اهميت را كشف ميكنند .بعد از كشف رويدادهاي با اهميت در مرحله تحليل داده ها، امكان گروه بندي رويدادها و نمايش گرافيكي آنها ميسر ميشود .با استفاده از نمايش گرافيكي و خوشه بندي رويدادهاي مشابه در يک دسته با استفاده از روشهاي هوش مصنوعي؛ بسياري از ناهنجاري ها قابل تميز هستند.

بعد از كشف رويدادهاي مهم ايد گزارش كاملي از آن رويداد ارائه شود:

- دقيقا چه اتفاقي رخ داده است؟
- رخداد مورد نظر در چه زماني رخ داده است؟ در نظر گرفتن رخدادهاي مختلف و تقدم و تاخر آنها بر يكديگر در كشف حملات چند مرحله اي و سابقه نفوذ در سيستم مهم است.

- این رخداد در کدام قسمت شبکه و روي چه سيستمي رخ داده است؟ نقش و اهميت اين سيستم چه بوده است.
- مبدا حمله چه بوده است؟ اين مهاجم در چند مرحله به اقدامات مخرب دست بزند يا با استفاده از همکار اين کار را انجام دهد؛ مهم است.
- به چه دليل اين اقدامات صورت گرفته است؟ اين پرسش و پاسخ مربوط به آن کليد اصلي کشف نقشه مهاجم است. اينکه مهاجم طي چه مراحلي قصد دارد به مقصود خود برسد و يا با استفاده از هر کدام از اقدامات به چه چيزي نايل ميشود؛ نکته مهمي است که مقصود اصلي مهاجم را مشخص مي کند و به طبع آن راه مناسب پاسخ به آن را معلوم ميکند.

ابزارهاي تحليل داده و مديريت حسگرهاي اسنورت

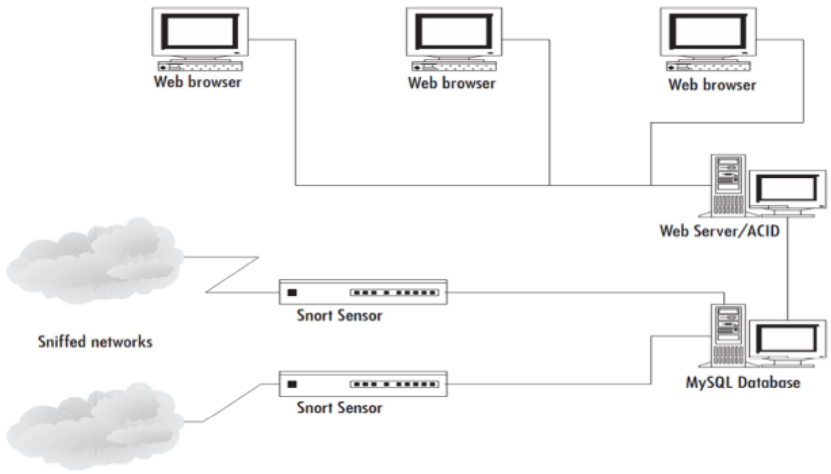
ابزارهاي تحليل داده بر اساس کارکردي که دارند به چند دسته تقسيم ميشوند .اين دسته ها عبارتند از:ابزارهاي مبتني بر پايگاه داده، نسخه هاي پردازش داده، ابزارهاي نمايشي و ابزارهاي هشداردهي به هنگام در زير کاربرد هر کدام از اين دسته ابزارها را بيان نموده و نمونههايي از ابزارهاي کدباز موجود را معرفي خواهيم نمود.

ابزارهای مبتنی بر پایگاه داده

اين ابزارها واسط گرافيکي را فراهم ميکنند که به کمک آنها ميتوان فرآيند تحليل داده هاي ثبت شده درپايگاه داده را مديريت کرد و سرعت بخشيد .مهمترين ابزارهايي که امروزه از اين دسته کاربرد دارند عبارتند: BASE و SGUIL

BASE

این ابزار مانند ACID مبتنی بر PHP است. که به منظور مديريت رخدادهاي ثبت شده در پايگاه داده به کار می‌رود. اين ابزار نتايج را از سرور پايگاه داده دريافت کرده و تحليل ميکند .رخدادهاي پايگاه داده ميتواند از ابزارهاي تشخيص نفوذ متفاوتي آمده باشند؛ همينطور ديوارههاي آتش و ديگر ابزارهاي مديريت شبکه .براي کار با اين ابزار بايد بر روی سرور مربوطه یک سرور APPACHI نصب شده باشد.



نمبندی اجزای IDS با استفاده از حسگرهای اسنورت و BASE .

SGUIL

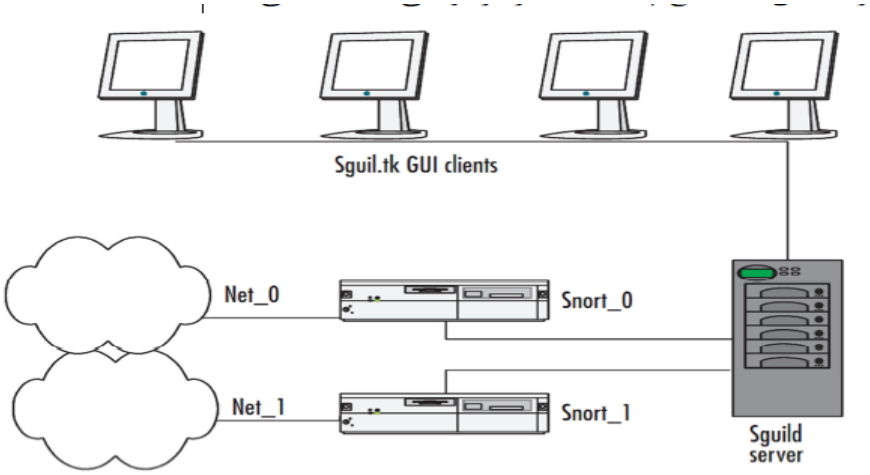
اين ابزار مثل ACID و BASE داراي يک کنسول براي مديريت تحليل هشدارهاي اسنورت ميباشد .اين ابزار به منظور مديريت هشدارهاي پايگاه دادهاي مربوط اسنورت ايجاد شده است .اين ابزار از سه بخش مهم تشکيل شده است:

چندين سري از نسخههاي اجرايي براي اجراي حسگرهاي اسنورت موجود در شبکه

يک برنامه سرور با امکان ارائه واسط گرافيکي کاربر

چندين برنامه کلاينت براي ارتباط با سرور

البته همه اين بخشها ميتوانند بر روي يک سرور قرار داشته باشند؛ ولي استفاده از کلاينتهاي مختلف امکان دسترسي از مناطق به منابع پايگاه داده را از نواحي مختلف ميدهد.



یک پیاده‌سازی نمونه از برنامه SQUIL به همراه دو حسگر اسنورت

كلاينت هاي اين ابزار امكان مشاهده رخدادها را به صورت دسته بندي شده و سازمان دهي شده در فاصله زماني تقريبا بلادرنگ فراهم ميكنند .همچنين امكان ايجاد جستجو در پايگاه داده و دريافت رويدادهاي داراى ويژگی‌های خاص وجود دارد.

نسخه هاي پردازش داده

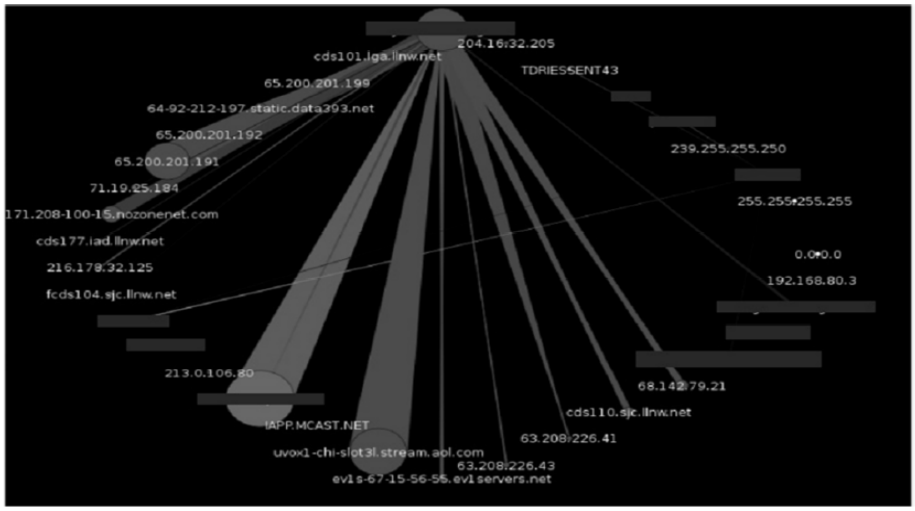
اين نسخه ها امكان اجراي سريع و بدست آوردن ديد مناسب از هشدارها را دارند .اين دسته از نسخه ها با كاربري ساده و خيلي سريع بدون نياز به راه اندازي سيستم بزرگ با سرورهاي مديريت، امكان ارتباط با اسنورت و مشاهده رويدادها و دسته بندي آنها را مي دهند.

ابزارهاي نمايشي

اين ابزارها هشدارها را به صورت دستهبندي شده و خوشهاي و در قالب گرافيكي نمايش مي دهند .به اين ترتيب امكان كشف ناهنجاري ها بيشتر ميشود .در زير به چند نمونه از آنها اشاره ميكنيم.

Etherape

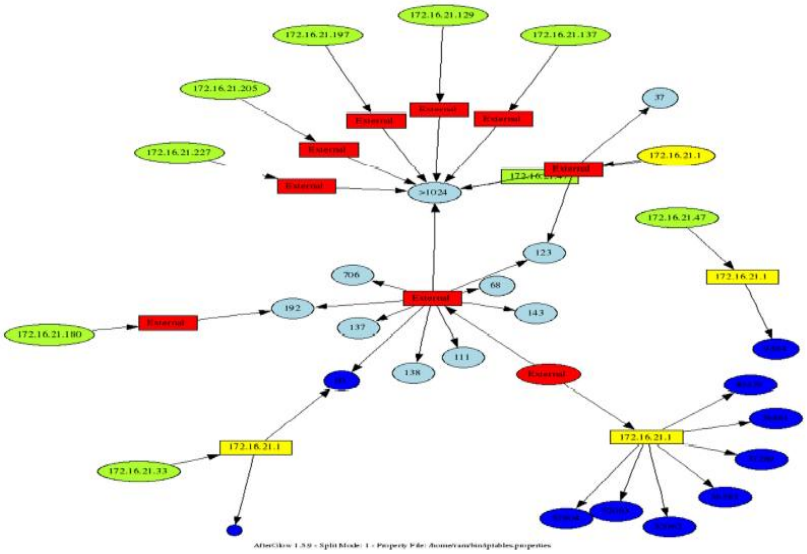
در شكل زير نمونه‌ای از خروجی اين ابزار را می‌بينيد. در اين شكل انواع ارتباطات بين سيستم هاي مختلف و محدوده آدرس IP مختلف آورده شده همچنين پروتكل‌های مختلف ميتوانند با رنگهاي مختل آورده شوند .به اين ترتيب ترافيک شبكه و پروتكل هاي مورد استفاده به همراه سيستمهايي كه از اين پروتكل ها استفاده ميكنند مي توانند مشخص شوند و داده هاي آماري نيز از توزيع اين ترافيک ها توليد شود.



صورت نمایش گرافیکی ترافیک شبکه با استفاده از EtherApe

AfterGlow

این ابزار شامل مجموعهاي از برنامه هاي نسخه اي است که قابليت نمايش گرافهاي ارتباطي را دارد. نمونه‌ای از خروجی آن را در شکل زیر می‌بینید.



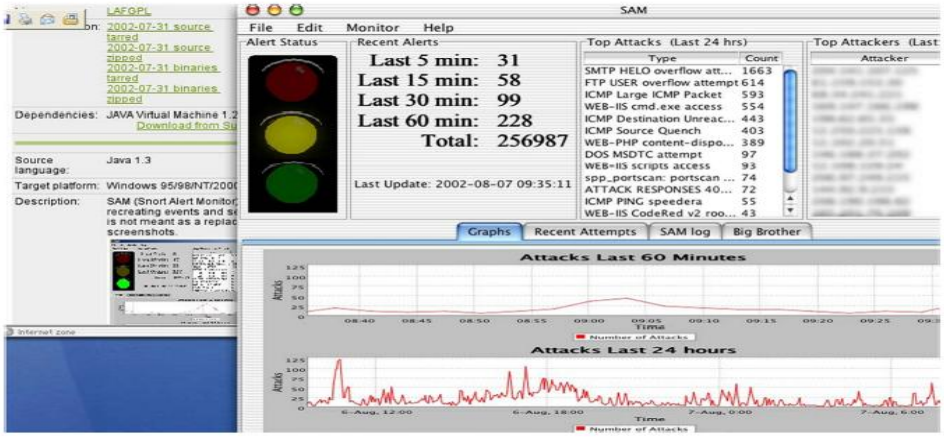
نمونه‌ای از خروجی ابزار AfterGlow

ابزارهاي هشداردهي بهنگام

اين ابزارها هشدارهاي اسنورت را پويش ميكنند و با توجه به هشدارهايي كه مشاهده ميكنند اقدام مناسب را صورت ميدهند . اين اقدام ميتوان به نوبه‌ی خود هشدار به مدير سيستم باشد .همچنين امكان اجراي نسخه‌های shell وجود دارد. كه به كمك آنها مي توان پاسخ مناسب را در مقابل حمله ايجاد كرد. به طور مثال، فراخواني يک برنامه براي ارسال پيام پيكربندي به ديواره آتش باشد تا به كمک آن راه نفوذ به شبكه از سوي مهاجم بسته شود .در زير به معرفي چند ابزار از اين دست مي پردازيم:

SAM

اين ابزار يک كنسول مبتني بر جاوا است كه هشدارهاي اسنورت را دريافت نموده و نمودار آن را رسم ومشخصات حملات را دسته بندي ميكند .همچنين اين ابزار قادر به ايجاد هشدارهاي صوتي براي حملات خاص است .در زير نمونه اي از خروجي اين نرمافزار را در زمان مي بينيم.



نمونه‌ای از خروجی SAM و رويدارهای مشخص شده بر اساس زمان و حمله‌كننده و مقصد حمله و نوع آن

ساختار جديد موتور تشخيص اسنورت

از نسخه ۰.۲ به بعد اسنورت موتور تشخيص تغييرات ساختاري عمدهاي كرده است و موتور تشخيص آن مورد بازبيني مهندسي قرار گرفته است .براساس اين تغييرات اسنورت داراي يک موتور تشخيص با قابليت اعمال چند قانون بر روي بسته ها است؛ در نسخه هاي قبلي از بين تمام قوانيني كه الگوي آن ها در بسته شناسايي ميشد؛ تنها يكي قابل اجرا بود .اين موتور رويداد قادر به شناسايي همخواني هر كدام از رويدادهاي كشف شده؛ با مجموعه قوانين در دسترس است .اين موتور قادر است با انجام جستجو بر روي قوانين؛ با پيدا كردن مجموعه قوانين همخوان؛ براي هر كدام از رويدادها يک صف از قوانين تشكيل دهد كه به ترتيب بروري آن اعمال گردند.در نسخه جديد موتور تشخيص؛ اين موتور از سه بخش مجزا تشكيل شده است .اين بخش ها عبارتند ازبهينه ساز قوانين ، جستجوگر قوانين چندگانه و انتخابگر رويداد.