1 مقدمه 5
1-1 انواع حملات 5
1-1-1 حملات رد سرویس 6
1-1-2 حملاتی که به منظور بدست آوردن اطلاعات صورت می‌گیرند 8
1-1-3 حملاتی که سرویسدهی روی شبکه را دچار مشکل می‌کنند 9
1-2 امنیت پروتکلها 9
1-2-1 پیچیدگی سرویس 10
1-2-2 سوءِ استفاده از سرویس 10
1-2-3 اطلاعات ارائه‌شده توسط سرویس 10
1-2-4 میزان دیالوگ با سرویسگیر 11
1-2-5 قابلیت پیکربندی سرویس 11
1-2-6 نوع مکانیزم احراز هویت استفاده شده توسط سرویس 11
2 فایروالهای packet-filter 14
2-1 فیلترهای stateless 14
2-1-1 کنترل بسته‌ها بر اساس نوع پروتکل 14
2-1-2 کنترل بسته‌ها بر اساس آدرس IP 15
2-1-3 کنترل بسته‌ها بر اساس پورتهای TCP/UDP 15
2-1-4 کنترل بسته‌ها از روی سایر اطلاعات موجود در سرآیند 16
2-1-5 مشکلات فیلترهای استاندارد 17
2-1-6 کنترل بسته‌ها توسط سیستم‌عامل 18
2-2 فیلترهای stateful 18

3-2 مشکلات فیلترها 19
3 NAT 20
3-1 انواع ترجمه آدرس در NAT 22
3-1-1 ترجمه پویا 22
3-1-2 ترجمه ایستا 23
3-1-3 توزیع بار 23
3-1-4 افزونگی (Redundancy) 23
3-2 مشکلات NAT 24
4 پراکسی 26
4-1 عملکردهای امنیتی پراکسی 27
4-1-1 پنهان کردن اطلاعات سرویسگیرها 27
4-1-2 بستن یک سری URL 27
4-1-3 کنترل محتویات بسته‌ها 28
4-1-4 اطمینان از سالم بودن بسته‌ها 28
4-1-5 کنترل روی دسترسی‌ها 28
4-2 تاثیر پراکسی در سرعت 29
4-2-1 cache کردن 29
4-2-2 توزیع بار 29
4-3 مشکلات پراکسی 29
5 سیستم‌های تهاجم‌یاب 31
5-1 سیستم‌های تهاجم‌یاب بر مبنای بازرسی 32
5-2 سیستم‌های تهاجم‌یاب طعمه 33

6 IP Filter 34
6-1 نصب IP Filter روی Solaris 35
6-2 پیاده‌سازی یک فیلتر با استفاده از IP Filter 35
7 Snort 41
7-1 مود Sniffer 42
7-2 مود Packet Logger 42
7-3 مود تهاجم‌یاب شبکه 43
7-4 فیلترهای BPF 44
7-5 فایل پیکربندی Snort 46
7-5-1 preprocessorها 47
7-5-2 قوانین تهاجم‌یاب 48
7-5-3 ماجول‌های خروجی 48
8 SAINT 51
8-1 فایل پیکربندی 52
8-2 خط فرمان 57
8-3 فرمت بانک‌اطلاعاتی 61
8-3-1 بانک‌اطلاعاتی facts 61
8-3-2 بانک‌اطلاعاتی all-hosts 62
8-3-3 بانک‌اطلاعاتی todo 63
8-3-4 بانک‌اطلاعاتی CVE 63
8-4 آنالیز خروجی 63

1 مقدمه

این متن به بررسی انواع سیستمهای امنیتی و بررسی نقاط ضعف و قوت هرکدام می‌پردازد. در این بخش مقدماتی در مورد امنیت پروتکلها و انواع حملات بیان می‌شود و بخشهای بعدی به بررسی دقیق انواع فایروال (فیلتر [1]، NAT[2] و پراکسی[3]) و سیستمهای تهاجم‌یاب[4] می‌پردازد. سپس سه نمونه از نرم‌افزارهای مفید امنیتی (IPF، Snort و SAINT) معرفی می‌گردد.

1-1 انواع حملات

در این قسمت یک سری از روشهای متداول برای جمله به شبکه‌های کامپیوتری توضیح داده می‌شود ودر مورد هرکدام مشخصات ونحوه شناسایی آن حمله بیان شده است. این حملات درچهار دسته عمده تقسیم‌بندی شده‌اند:

- حملات رد سرویس یا DoS[5]
- حملات استثماری[6]
- حملاتی که به منظور بدست آوردن اطلاعات صورت می‌گیرند[7]
- حملاتی که سرویسدهی روی شبکه را دچار مشکل می‌کنند[8]

---

[1] Packet Filter

[2] Network Address Translation

[3] Proxy

[4] Inrusion Detection Systems

[5] Denial-of-service attacks

[6] Exploitation attacks

[7] Information gathering attacks

[8] Disinformation attacks

1-1-1 حملات رد سرویس

این نوع حملات با ایجاد یک بار زیاد و غیرعادی روی سرورها باعث از کار افتادن سرویسهای ارائه شده توسط آنها می‌شوند. از آنجا که انجام دادن این نوع حمله ساده است، لذا بیشتر متداول می‌باشد. این قسمت بیشتر این حملات را توضیح میدهد:

Ping of Death

این حمله از طریق بسته‌های ICMP صورت می‌گیرد. حجم بسته‌های ICMP به 64KB محدود می‌شود و بسته‌هایی که در سرآیند آنها حجم بسته بیشتر از این مقدار بیان شده (در حالیکه نیست) ممکن است در سمت گیرنده مشکلاتی ایجاد کنند چون بسیاری از سیستم‌عاملها کنترل دقیقی روی بسته‌های معیوب ندارند. این نوع حمله نسبتا قدیمی است و امروزه تمام سیستم‌عاملها قادر به تشخیص آن می‌باشند.

Teardrop

این حمله از طریق fragmentهای IP صورت می‌گیرد. یک fragment شامل اطلاعاتی است که بیان می‌کند چه قسمتی از بسته داخل آن قرار دارد. بسیاری از سیستمها ممکن است با گرفتن fragmentهایی که متعلق به یک بسته بوده و با هم تناقض دارند (یک قسمت از بسته در دو fragment قرار داشته باشد) دچار مشکل شوند. این نوع حمله نیز قدیمی است.

UDP Flooding

این حمله با استفاده از سرویسهای echo و chargen صورت می‌گیرد. با فرستادن یک درخواست جعلی از طرف یک سرویس echo برای یک سرویس chargen می‌توان به راحتی حجم زیادی از ترافیک را روی شبکه ایجاد کرد.

SYN Flooding

این حمله با فرستادن بسته‌های SYN پروتکل TCP صورت می‌گیرد. برای یک سرور دریافت یک بسته SYN به معنی گرفتن فضایی از حافظه برای آن ارتباط و فرستادن یک بسته ACK در پاسخ می‌باشد. فضای حافظه تخصیص داده شده تا زمان timeout یا بسته شدن ارتباط باقی می‌ماند. اگر تعداد زیادی بسته SYN فرستاده شود موجب اتلاف قسمت عمده‌ای از حافظه می‌شود، هرچند فرستادن بسته‌های ACK نیز زمان و پردازش زیادی لازم دارد. این حمله در نهایت سرور را به وضعیتی می‌کشاند که قادر به قبول ارتباط جدید نمی‌باشد. از آنجا که فرستنده بسته‌های SYN در این حمله منتظر پاسخ نمی‌ماند می‌تواند بسته‌ها را قبل از فرستادن تغییر دهد و هر بار یک آدرس تصادفی بجای آدرس فرستنده آنها قرار دهد. در این صورت تشخیص حمله بسیار مشکل می‌شود.

Land Attack

این حمله شبیه SYN Flooding می‌باشد. در این حمله یک بسته SYN برای سرور ارسال می‌شود که آدرس فرستنده و گیرنده آن هر دو آدرس خود سرور است. سرور پس از دریافت این بسته پاسخ آن را برای خودش می‌فرستد که نتیجه‌ای مشابه SYN Flooding به همراه دارد.

Smurf Attack

این حمله از طریق بسته‌های ICMP صورت می‌گیرد. در این حمله یک بسته ICMP Request داخل شبکه فرستاده می‌شود که آدرس reply آن آدرس broadcast شبکه می‌باشد. چنین بسته‌هایی معمولا ترافیک بالایی داخل شبکه ایجاد می‌کنند.

Fraggle Attack

این حمله شبیه Smurf Attack است ولی بجای بسته‌های ICMP از بسته‌های UDP استفاده می‌کند.

E-mail Bombs

این نوع حمله شامل فرستادن نامه‌های بزرگ بطور مداوم برای یک سیستم است. از آنجا که سمت فرستنده و گیرنده دارای بار نسبتا مساوی هستند از این روش کمتر می‌توان بعنوان یک DoS واقعی استفاده کرد.

Malformed Attacks

بسیاری از سرویسها هنگام دریافت بسته‌هایی که دارای خطا می‌باشند با مشکل مواجه می‌شوند چون کنترل دقیق روی بسته‌های معیوب ندارند و این بسته‌ها باعث ایجاد مشکل در برنامه سرور می‌شوند. یک تقسیم بر صفر یا سرریز بافر می‌تواند سرور را از کار بیندازد یا سبب دسترسی افراد غیرمجاز به آن شود. هر سرویسی ممکن است در معرض این حمله قرار بگیرد چون در هر لحظه امکان پیدا شدن یک bug در برنامه مربوطه وجود دارد. بیشترین مواردی که از این حمله مشاهده شده بر روی سرویسهای وب و پست الکترونیکی بوده است.

حملات استثماری

این نوع حملات بیشتر برای بدست آوردن کنترل مستقیم روی یک ماشین انجام می‌شود. مهمترین این حملات از قرار زیر می‌باشند:

حدس زدن password

بسیاری از سرورها برای ارائه سرویس نیاز به احراز هویت کاربران از طریق password دارند. برنامه‌هایی وجود دارند که یک سری از کلمات (اسامی، کلمات dictionary، اعداد، ...) را بطور اتوماتیک تست می‌کنند تا به یک password معتبر دسترسی پیدا کنند.

Trojan Horse

Trojan Horse به برنامه‌ای گفته می‌شود که اغلب توسط یک مهاجم روی سیستم نصب می‌شود و اطلاعاتی در مورد سیستم به خارج از شبکه می‌فرستد یا راهی برای دسترسی غیرمجاز به سیستم فراهم می‌کند که به آن backdoor می‌گویند. Trojan Horse معمولا برنامه کوچکی است که به سادگی نصب می‌شود و از دید کاربر نیز پنهان می‌ماند.

Buffer Overrun

اکثر سرورها برای رسیدگی به درخواستهایی که از شبکه دریافت می‌کنند فضایی از حافظه را به عنوان بافر اختصاص می‌دهند. اغلب برنامه‌ها حجم این بافر را به یک مقدار ثابت محدود می‌کنند یا به بسته‌های رسیده اطمینان کرده و اندازه بسته‌ها را از روی اطلاعات سرآیند آنها استخراج می‌کنند. این پدیده معمولا زمانی اتفاق می‌افتد که طول یک بسته از مقدار در نظر گرفته شده برای بافر بیشتر باشد یا اطلاعات غلط در مورد طول خود به برنامه بدهد. برای مثال اگر طول یک بسته 256 بایت باشد ولی در اطلاعات سرآیند طول بسته 240 بایت معرفی شده باشد 240 بایت بسته داخل بافر قرار می‌گیرد و 16 بایت اضافی در یک مکان دیگر از حافظه نوشته می‌شود و منجر به از بین رفتن اطلاعات آن قسمت حافظه می‌شود. در این حالت با قرار دادن کد ماشین در 16 بایت آخر بسته ممکن است بتوان این کد را روی سرور اجرا کرده کنترل سرور را بدست آورد.

2-1-1 حملاتی که به منظور بدست آوردن اطلاعات صورت می‌گیرند

این نوع حملات هیچگونه صدمه‌ای به سیستم نمی‌زنند و تنها برای بدست آوردن اطلاعات جهت حملات بعدی مورد استفاده قرار می‌گیرند. مهمترین اطلاعاتی که یک مهاجم می‌تواند بدست آورد در مورد آدرس سیستمهای داخل شبکه ، سیستم‌عامل روی آنها، پورتهای باز این سیستمها و کاربران روی آنها می‌باشد. برای پیدا کردن آدرسهای داخل شبکه از نرم‌افزارهایی استفاده می‌شود که برای یک دسته از آدرسها پیغام TCMP Request می‌فرستد. با دریافت پاسخ این بسته‌ها سیستمهای موجود در داخل شبکه شناسایی می‌شوند و هرکدام از این آدرسها برای حملات بعدی مورد بررسی قرار می‌گیرند. قبل از حمله باید اطلاعات خاصی در مورد هر سیستم بدست آورد که این اطلاعات می‌تواند شامل سیستم‌عامل، پورتهای باز و کاربران معتبر روی آن سیستم باشد. برنامه‌هایی تحت عنوان Port Scanner وجود دارند که می‌توانند با فرستادن بسته‌های خاصی به سیستم اطلاعاتی در مورد پورتهای باز، سرویسهای موجود روی سیستم و سیستم‌عامل آنها بدست بیاورند. Port Scannerها انواع مختلف دارند و بعضا از روشهایی استفاده می‌کنند که به سختی قابل تشخیص می‌باشند. برای تشخیص نام کاربران روی سیستم نیز می‌توان از سرویسهایی نظیر finger استفاده کرد. سرویس finger در سیستم‌عاملهای مبتنی بر Unix اطلاعات مفیدی در مورد کاربران ارائه می‌کند ولی از این سرویس برای پیدا کردن نام کاربران معتبر نیز می‌توان استفاده کرد.

3-1-1 حملاتی که سرویسدهی روی شبکه را دچار مشکل می‌کنند

این نوع حملات بر روی سرورهای شبکه اعمال می‌شود و آنها را وادار می‌کند اطلاعات اشتباه به سرویسگیرها بدهند. این حملات معمولا راه را برای حملات بعدی باز می‌کند. دو نمونه از این حملات عبارتند از:

DNS Cache Pollution

از آنجایی که سرورهای DNS هنگام ردوبدل کردن اطلاعات با سرورهای دیگر از هیچ مکانیزم امنیتی خاصی استفاده نمی‌کنند مهاجمین می‌توانند با دادن اطلاعات غلط به سرور DNS آنها را وادار کنند اطلاعات اشتباه به سرویسگیرها بدهند. سپس سرویسگیرها از همین اطلاعات غلط استفاده می‌کنند. در این حالت بجای وصل شدن به یک سایت خاص ممکن است به سایت مهاجمین وصل شوند.

email جعلی

تولید نامه‌های جعلی از طریق سرور پست الکترونیکی کار بسیار ساده‌ای است چون هیچ مکانیزم امنیتی خاصی برای احراز هویت کاربران استفاده نمی‌شود. این کار به سادگی پیکربندی اشتباه یک سرویسگیر می‌باشد. با ارسال نامه‌های جعلی برای کاربران از طرف اشخاص مورد اطمینان آنها می‌توان باعث نصب یک Trojan Horse روی سیستم آنها، ارسال اطلاعات محرمانه در پاسخ نامه، یا اتصال کاربران به یک سایت خاص شد.

2-1 امنیت پروتکلها

در این قسمت یک سری پروتکلهای متداول که بر پایه IP کار می‌کنند از لحاظ امنیتی مورد بررسی قرار می‌گیرند. از آنجا که هرکدام از این پروتکلها برای ارائه یک سرویس بکار می‌روند، دو اصطلاح پروتکل و سرویس معمولاً بجای یکدیگر بکار می‌روند. میزان آسیب پذیری یک سرویس یا پروتکل با پاسخ دادن به سؤالات زیر مشخص می‌شود :

- سرویس مربوطه چقدر پیچیدگی دارد؟
- این سرویس چگونه می‌تواند مورد سوءِ استفاده قرار بگیرد؟
- چه اطلاعاتی در مورد شبکه توسط سرویس افشا می‌شود؟
- چه مقدار دیالوگ با سرویسگیر انجام می‌شود؟
- سرویس تا چه حد قابلیت پیکربندی و برنامه‌نویسی دارد؟
- چه سرویسهای دیگری بر پایه این سرویس قرار گرفته‌اند؟
- این سرویس چه مکانیزمی برای احراز هویت سرویسگیرها استفاده می‌کند؟

1-2-1 پیچیدگی سرویس

سرویسهای پیچیده خیلی زودتر از سرویسهای ساده مورد تهاجم قرار می‌گیرند. سرویس echo یک سرویس ساده است که تمام کاراکترهای ارسالی از طرف سرویسگیر را دوباره برای وی می‌فرستد. این سرویس بیشتر برای مقاصد تست مورد استفاده قرار می‌گیرد. در مقابل سرویس پست الکترونیکی یک سرویس پیچیده می‌باشد که نامه‌های الکترونیکی را رد وبدل می‌کند. بسیاری از سرویسهای مرتبط با این سرویس مانند POP و IMAP نیاز به احراز هویت کاربر قبل از ارائه سرویس به وی دارند، هرچند در مقابل سرویس SMTP نامه‌ها را بدون توجه به فرستنده آنها (هر کاربری که باشد، حتی یک کاربر قلابی) ارسال می‌کند. اگر اطلاعات مربوط به password کاربران افشا گردد، مکانیزم امنیتی و احراز هویت فریب داده شود، یا حتی خود سرویس به گونه‌ای مورد تهاجم واقع شود که اطلاعات محرمانه شبکه را به بیرون منتشر کند، در هرکدام از این شرایط امنیت شبکه در معرض خطر بزرگی قرار گرفته‌است.

2-2-1 سوء استفاده از سرویس

یک سرویس می‌تواند به خودی خود یک سرویس ساده و بی‌خطر باشد، ولی می‌تواند در مقاصد مخرب نیز مورد استفاده قرار گیرد. سرویس chargen یک سرویس UNIX برای تولید مداوم کاراکترهای ASCII می‌باشد. این سرویس از آنجا که کاراکترهای تصادفی تولید می‌کند برای تست نرم‌افزارهای شبکه یک ابزار قدرتمند می‌باشد. این سرویس می‌تواند به سادگی مورد سوءِاستفاده قرار گیرد. برای مثال فرض کنید که یک بسته SYN با آدرس فرستنده تحریف شده برای این سرویس فرستاده شود. در مقابل سرویس سیل عظیمی از کاراکتر را برای کسی که آدرس وی بجای آدرس فرستنده در بسته قرار دارد فرستاده خواهد شد. در این حالت ایجاد کننده این بار ترافیکی بدون اینکه هزینه‌ای مصرف کرده باشد جریان بسته‌ها را بین دو آدرس دلخواه ایجاد می‌کند.

3-2-1 اطلاعات ارائه‌شده توسط سرویس

بعضی سرویسها در عمل بسیار ساده‌اند ولی می‌توانند برای شبکه خطرناک باشند. سرویس finger برای راحتی کاربران UNIX طراحی شده‌است. این سرویس یک سری اطلاعات در مورد accountهای موجود در سیستم ارائه می‌کند. مهاجمین می‌توانند از این سرویس برای پیدا کردن accountهای فعال سیستم استفاده کنند. پیدا کردن نام یک account معتبر می‌تواند نقطه شروع مناسبی برای حمله به سیستم باشد.

4-2-1 میزان دیالوگ با سرویسگیر

امن کردن یک سرویس با دیالوگ ساده به مراتب راحتتر از امن کردن سرویسی است که نیاز به دیالوگهای پیچیده با سرویسگیر دارد. برای مثال سرویس HTTP (در نسخه‌های اولیه و بدون در نظر گرفتن CGI و ASP و موارد مشابه) یک پروتکل ساده است که در آن سرویسگیر تقاضای یک سری منابع را به سرور می‌دهد و سرور نیز بدون توجه به وضعیت ارتباط موجود در صورت امکان منابع درخواست شده را برای سرویسگیر تهیه می‌کند (این ارتباط بصورت stateless است). امن کردن یک ارتباط stateful به مراتب مشکلتر است، مخصوصا اگر سرویس نیاز به احراز هویت سرویسگیر نیز داشته باشد و درخواستها و پاسخهای بین سرور و سرویسگیر موجب تغییر در وضعیت ارتباط شود.

5-2-1 قابلیت پیکربندی سرویس

هر اندازه سرویس قابل پیکربندی و برنامه‌ریزی باشد امکان بروز اشتباه در این تنظیمات بیشتر می‌شود و در نتیجه امکان پیدا شدن bugهای مختلف در آن بسیار زیاد است. از این رو سرورهایی مانند Exchange Server و Internet Information Server (یا هر وب سروری که امکان اجرا کردن برنامه‌هایی را برای تولید صفحات HTML در آن وجود داشته باشد) ممکن است دارای مشکلات امنیتی باشند که همه آنها در حال حاضر شناخته شده نیستند و به مرور زمان پدید می‌آیند.

6-2-1 نوع مکانیزم احراز هویت استفاده شده توسط سرویس

سرویسهایی که نیاز به احراز هویت سرویسگیر دارند از دو طرف در معرض خطرات امنیتی قرار دارند: اول اینکه خود مکانیزم استفاده شده ممکن است ضعیف باشد و این امر باعث سوءِ استفاده از سرویس می‌شود، دوم اینکه اغلب کاربران از یک password برای سرویسهای مختلف استفاده می‌کنند و در صورت لو رفتن password یک سرویس سایر سرویسها نیز در معرض خطر قرار می‌گیرند. یک نمونه بارز این خطر سرویس POP است. این سرویس اغلب از passwordهای خود سیستم برای احراز هویت کاربران استفاده می‌کند و بسیاری از سرورهای POP امکان رد و بدل کردن passwordها بطور امن را ندارند. در صورت لو رفتن یک password سرویس POP کل سیستم در معرض تهدیدهای امنیتی قرار می‌گیرد.

در انتهای این قسمت یک جدول تهیه شده‌است که میزان فاکتورهای ذکر شده را برای تعدادی از سرویسهای معمول ارائه می‌کند:

نام سرویس	پورت و پروتکل	پیچید گی	میزان سوء استفاده	اطلاعات ارائه شده	دیالوگ با سرویسگیر	قابلیت پیکربندی
DHCP	UDP 68	نسبتاً پیچیده	متوسط	متوسط	کم	متوسط
Chargen	TCP & UDP 19	ساده	زیاد	-	-	-
Daytime	UDP 13	ساده	کم	-	کم	-
Discard	UDP 13	ساده	کم	-	کم	-
DNS	UDP 53	پیچیده	زیاد	کم	کم	زیاد
Echo	UDP 7	ساده	کم	-	کم	-
Finger	TCP 79	ساده	متوسط	زیاد	کم	متوسط
FTP	TCP 20 & 21	پیچیده	زیاد	متوسط	زیاد	زیاد
Gopher	TCP 70	ساده	کم	کم	کم	کم
HTTP	TCP 80	پیچیده	زیاد	متوسط	زیاد	زیاد

IMAP	TCP 143	ساده	کم	متوسط	کم	کم
LDAP	TCP & UDP 389	پیچیده	کم	متوسط	متوسط	متوسط
NetBIOS	TCP 137 - 139	پیچیده	زیاد	زیاد	زیاد	کم
NFS	TCP & UDP 2049	پیچیده	زیاد	زیاد	زیاد	متوسط
POP3	TCP 110	ساده	متوسط	متوسط	کم	کم
Queue	UDP 17	ساده	کم	-	کم	-
RPC(sun)	UDP 111	متوسط	زیاد	زیاد	زیاد	زیاد
RSH	TCP 514	متوسط	زیاد	زیاد	زیاد	کم
SMTP	TCP 25	پیچیده	متوسط	متوسط	متوسط	زیاد
SNMP	UDP 161	متوسط	زیاد	زیاد	کم	متوسط
Telnet	TCP 23	ساده	زیاد	زیاد	کم	-

TFTP	UDP 69	ساده	زیاد	متوسط	کم	کم

2 فایروالهای packet-filter

packet-filterها ابتدایی‌ترین نوع فایروال می‌باشند. اولین اقدام در جهت امنیت پروتکل TCP/IP نیز همین فیلترها می‌باشند که عملکرد خود را از طریق چک کردن سرآیند بسته‌ها انجام می‌دهند. فیلترها به تنهایی نمی‌توانند امنیت کامل برای شبکه برقرار کنند و باید همراه با انواع دیگر فایروال مانند NAT و پراکسی استفاده شوند، همانطور که NAT و پراکسی نیز نمی‌تواند بدون یک فیلتر قوی عملکرد مناسب داشته باشند. فیلترها در دو نوع stateless (استاندارد) و stateful می‌باشند که در قسمتهای بعدی هرکدام از آنها به تفصیل مورد بحث قرار خواهد گرفت.

1-2 فیلترهای stateless

فیلترها معمولاً مسیریابهایی هستند که با توجه به اطلاعات موجود در سرآیند بسته‌ها در مورد رد شدن آن تصمیم می‌گیرند. فیلترها از لحاظ تئوری می‌توانند این کار را بر اساس تمام فیلدهای موجود در سرآیند بسته انجام دهند، ولی عملاً این کار بر اساس فیلدهای زیر انجام می‌شود که بیشتر متداول هستند:

- پروتکل لایه شبکه و جلسه
- آدرس IP
- پورت TCP/UDP
- شماره fragment
- اطلاعات مربوط به source routing

1-1-2 کنترل بسته‌ها بر اساس نوع پروتکل

این نوع فیلترها بسته‌ها را بر اساس فیلد پروتکل موجود در سرآیند کنترل می‌کنند. از فیلد پروتکل می‌توان برای مشخص کردن یک سری از سرویسها مانند UDP، TCP، ICMP و IGMP استفاده کرد. برای مثال اگر یک

سرویس TCP مانند وب توسط یک سرور ارائه می‌شود می‌توان سایر سرویسها مانند UDP را بست. فیلد پروتکل خیلی کلّی است و از روی آن نمی‌توان کنترل مناسبی روی بسته‌ها داشت.

2-1-2 کنترل بسته‌ها بر اساس آدرس IP

این نوع فیلترها می‌توانند برقراری ارتباط به (یا از) یک سری آدرسهای مختلف را محدود کنند. اکثر فیلترها همه آدرسها را باز گذاشته و دسترسی به یک سری خاص آدرسها را می‌بندند یا برعکس این کار را انجام می‌دهند یعنی دسترسی به همه آدرسها را می‌بندند و یک سری آدرسهای خاص را باز می‌گذارند. این دو سیاست به ترتیب سیاست accept و سیاست deny نامیده می‌شوند. سیاست accept معمولاً مزیتی برای شبکه محسوب نمی‌شوند چون در این حالت کنترل لازم روی همه آدرسها وجود ندارد و هیچ تضمینی نیست که مهاجمینی که آدرسهای آنها بسته شده‌است از آدرسهای دیگر استفاده نکنند. سیاست deny خیلی امن‌تر می‌باشد، در این حالت می‌توان مطمئن شد کسانی که اطلاعات کافی در مورد آنها موجود می‌باشد دسترسی به شبکه دارند.

فیلترهای خوب می‌توانند دسترسی‌ها را بر اساس پروتکل کنترل کنند. برای مثال می‌توان دسترسی همه را به سرویس HTTP باز گذاشت ولی فقط به کاربران شبکه داخلی اجازه استفاده از سرویس Telnet را داد. در فیلترهای ساده معمولا فقط می‌توان دسترسی آدرسهای خاصی را به یک سرویس باز گذاشت یا بست و برای یک سرویس خاص نمی‌توان با آدرسهای مختلف رفتار متفاوتی داشت. باید این نکته را در نظر بگیرید که فیلد آدرس فرستنده که در بسته IP وجود دارد آدرسی نیست که بسته از آن آمده‌است و این آدرس قابل جعل کردن است. از این نقطه ضعف می‌توان برای فرستادن بسته‌های غیرمجاز به داخل شبکه استفاده کرد؛ البته پاسخ این بسته‌ها برای آدرس جعل فرستاده می‌شود. راههایی وجود دارد که می‌توان بر این مشکل غلبه کرد؛ برای مثال می‌توان با استفاده از تکنیک source routing آدرس بازگشت بسته را نیز تعیین کرد، هرچند هنوز هم می‌توان از این نقطه ضعف برای فرستادن بسته‌هایی که نیازی به برگشت آنها نیست (مانند حملات DoS) استفاده کرد. مهاجم باید یک آدرس IP را که اجازه عبور از فیلتر را دارد پیدا کند و بسته را از طرف آن بفرستد.

3-1-2 کنترل بسته‌ها بر اساس پورتهای TCP/UDP

فیلدهای مربوط به پورتهای TCP و UDP معمولترین فاکتور برای کنترل بسته‌ها می‌باشند چون این اطلاعات به دقت مشخص می‌کند که یک بسته به چه منظور فرستاده شده‌است. فیلترکردن پورتها همچنین تحت عنوان فیلترکردن پروتکلها نیز شناخته می‌شود چون هرکدام از این پورتها یک پروتکل سطح بالا را مشخص می‌کنند. پروتکلهای متداولی که بر اساس شماره پورتهای TCP و UDP فیلتر می‌شوند عبارتند از:

Daytime	DNS	NetBIOS Session	Echo
HTTP	IMAP	Quote	Gopher

NFS	FTP	POP	Whois
Telnet	SNMP	RSH	SMTP
NNTP	X Windows		

در مورد پورتها نیز مانند آدرسهای IP دو نوع سیاست accept و deny وجود دارد. در مورد پورتها بر خلاف آدرسهای IP سیاست accept نیز می‌تواند مفید باشد چون اکثر حملات از طریق یک سری پورت شناخته شده انجام می‌گیرد. معروفترین این پورتها عبارتند از:

- Telnet : باز گذاشتن این پورت به مهاجمین اجازه اجرای دستور روی سیستم را می‌دهد که بیشترین دسترسی ممکن را برای آنها فراهم می‌آورد.
- NetBIOS Session : باز گذاشتن این پورت روی سیستمهای Windows یا سرورهای SMB به مهاجمین اجازه دسترسی به فایل‌سیستم را می‌دهد.
- POP : در صورت امکان باید این پورت بسته باشد. در این پروتکل password کاربران بصورت رمز نشده روی شبکه فرستاده می‌شود و مهاجمین می‌توانند با آنالیز کردن بسته‌ها password کاربران را به دست بیاورند. در صورتی که سرویس POP ارائه می‌شود یا باید بوسیله SSL یا از طریق VPN امن شود.
- NFS : عملکرد این پورت برای سیستمهای مبتنی بر UNIX دقیقا مانند پورت NetBIOS برای سیستمهای Windows می‌باشد.

پورتهایی که در اینجا ذکر شد از اهمیت زیادی برخوردارند چون در صورتی که مورد حمله قرار بگیرند می‌توانند باعث شوند فرد مهاجم کنترل کامل بر روی سیستم داشته باشد. پورتهای دیگر مانند DNS از اهمیت کمتری برخوردارند، چون در صورت مورد حمله واقع شدن باعث صدمه دیدن یک سری اطلاعات خاصی می‌شوند و فرد مهاجم نمی‌تواند کنترل کامل بر روی سیستم داشته باشد. به همین علت دارای اهمیت کمتری برای مهاجمین می‌باشند.

4-1-2 کنترل بسته‌ها از روی سایر اطلاعات موجود در سرآیند

سرآیند بسته‌های IP علاوه بر فیلدهای استاندارد که ذکر شد شامل اطلاعات دیگری نیز می‌باشند که از روی آنها می‌توان در مورد ردشدن یک بسته تصمیم گرفت. source routing و fragmentation تکنیکهایی هستند که توسط پروتکل IP پشتیبانی می‌شوند و امروزه چندان استفاده‌ای ندارند. مهاجمین از این دو تکنیک برای حمله کردن به شبکه‌ها استفاده می‌کنند.

source routing 1-4-1-2

source routing برای مشخص کردن مسیر دقیقی که پاسخ یک بسته IP برای رسیدن به مقصد باید از آن عبور کند استفاده می‌شود. از این امکان بیشتر برای تست و عیب‌یابی شبکه‌ها استفاده می‌شده‌است ولی امروزه توسط مهاجمین استفاده می‌شود. آنها با استفاده از IP spoofing یک بسته با آدرس فرستنده جعلی ایجاد می‌کنند و سپس با استفاده از source routing کاری می‌کنند که پاسخ آن بجای گیرنده اصلی برای آنها فرستاده شود. دو نوع source routing وجود دارد. در نوع اول آدرس یک یا چند مسیریاب سر راه مشخص می‌شود و در نوع دوم تمامی مسیر تا مقصد مشخص می‌شود که نوع اول بیشتر توسط مهاجمین مورد استفاده قرار می‌گیرد.

fragmentation 2-4-1-2

fragmentation برای انتقال بسته‌های IP بزرگ از مسیریابهایی است که اندازه frame در آنها کوچک است. این مسیریابها بسته‌ها را به تعدادی frame می‌شکنند که از شماره 0 شماره‌گذاری می‌شوند. این frameها در مقصد سرهم شده و بسته اولیه را دوباره می‌سازند. از آنجا که مهمترین اطلاعات لازم برای کنترل بسته‌ها (شماره پورت TCP و UDP) فقط در frame شماره 0 وجود دارد، روی frameهای بعدی هیچگونه کنترلی وجود ندارد و اغلب فیلترها آنها را عبور می‌دهند. بعضی سیستمها کل frameهای دریافتی را حتی بدون frame شماره 0 سرهم می‌کنند و در صورتی که یک بسته IP سالم را تشکیل دهند آن را پردازش می‌کنند. از این رو مهاجمین نیز می‌توانند یک سری frame تولید کنند که از شماره 1 به بعد یک بسته IP سالم را تشکیل بدهند. از آنجا که اکثر فیلترها فقط frame شماره 0 را بررسی می‌کنند و سایر frameها از فیلتر عبور می‌کنند با این روش می‌توان یک بسته کامل را از فیلتر عبور داد. از این رو بسته‌های fragment شده نباید اجازه ورود به شبکه را داشته باشند.

5-1-2 مشکلات فیلترهای استاندارد

این نوع فیلترها دو مشکل عمده دارند که مانع از این می‌شود که بطور کامل موثر واقع شوند:

این فیلترها محتویات بسته را نمی‌توانند چک کنند.

فیلترها برای تعیین سرنوشت بسته‌ها فقط از اطلاعات سرآیند بسته استفاده می‌کنند و محتوای آنها را بررسی نمی‌کنند. در این صورت اگر ترافیک HTTP که وارد شبکه می‌شود یک Trojan Horse باشد فیلتر متوجه آن نخواهد شد؛ و یا اگر یک نامه الکترونیکی که به پورت 25 سرور فرستاده می‌شود دارای اطلاعاتی باشد که منجر به از کار افتادن سرور شود باز فیلتر متوجه آن نخواهد شد. برای بررسی اینگونه موارد نیاز به یک پراکسی در لایه بالا برای هر سرویس می‌باشد که محتویات بسته را با توجه به پروتکل استفاده شده بررسی کند.

این فیلترها وضعیت ارتباطات موجود را نگهداری نمی‌کنند.

فیلترهای stateless هیچ اطلاعاتی در مورد وضعیت ارتباطات فعال نگهداری نمی‌کنند. این فیلترها تصمیم‌گیری در مورد سرنوشت بسته‌ها را بسته به بسته انجام می‌دهند و تنها فاکتوری که در این میان تصمیم‌گیری می‌کند اطلاعات سرآیند هرکدام از بسته‌هاست.

6-1-2 کنترل بسته‌ها توسط سیستم‌عامل

بسیاری از سیستم‌عاملهای جدید امکان کنترل بسته‌ها را بعنوان بخشی از پشته TCP/IP خود دارند. این بدان معناست که روی هر سرور بسته به سرویسی که ارائه می‌شود می‌توان رفتار متفاوتی با بسته‌ها داشت. در این روش بسته‌ها در مقصد آنها یعنی آخرین مسیری که از آن عبور می‌کنند کنترل می‌شوند. ممکن است چنین به نظر بیاید که با وجود اعمال سیاستهای امنیتی در فایروالها و مسیریابهای شبکه ایجاد فیلتر بر روی هر سرور یک کار اضافی است، چون این کار را بر روی فایروالها و مسیریابها نیز می‌توان انجام داد. ولی این نکته را باید در نظر بگیرید که هیچ سیستم امنیتی مرزی قادر به جلوگیری از حملات داخلی یا حملات از طریق کانالهای VPN و اتصالات dial-up نمی‌باشد. بعنوان یک اقدام امنیتی علاوه بر اینکه یک فایروال در نقاط ورود بسته‌ها به شبکه از شبکه محافظت می‌کند بهتر است هر سرور نیز یک فیلتر مخصوص به خود بر سر راه بسته‌ها قرار دهد تا اطمینان حاصل شود که فقط سرویسهایی را که لازم است ارائه می‌کند.

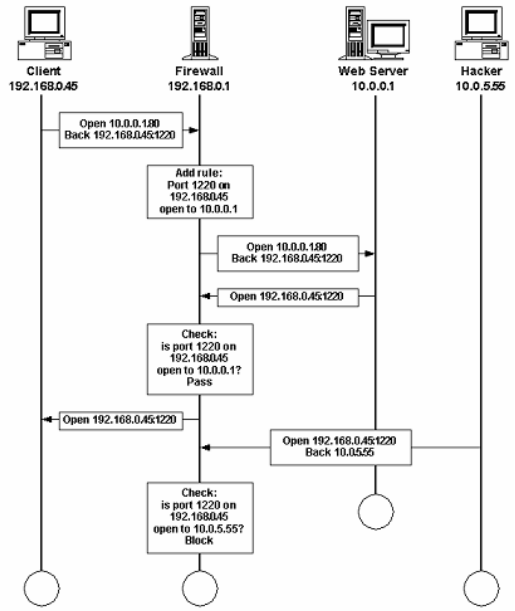
2-2 فیلترهای stateful

فیلترهای استاندارد مشکلات زیادی دارند که همه آنها از این حقیقت ناشی می‌شوند که یک بسته به تنهایی حاوی اطلاعات کافی برای تصمیم‌گیری در مورد سرنوشت آن نمی‌باشد چون این بسته جزئی از یک ارتباط بزرگتر است. فیلترهای stateful این مشکل را با نگهداشتن وضعیت تمام ارتباطات موجود و تصمیم‌گیری از روی این وضعیتها حل کرده‌اند. فیلترهای stateful وضعیت هر ارتباط را در لایه‌های شبکه و جلسه نگهداری می‌کنند، سپس از این اطلاعات برای تشخیص اینکه یک بسته در پاسخ یک درخواست داخلی فرستاده شده یا نه (به احتمال زیاد به قصد خرابکاری ارسال شده) استفاده می‌کنند.

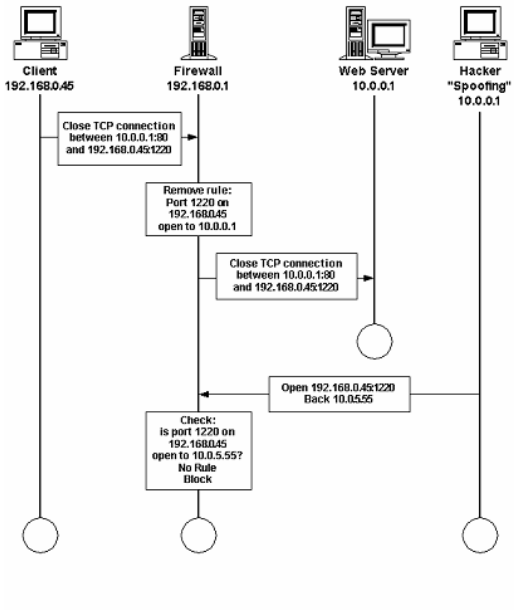
اغلب فیلترهای استاندارد به بسته‌هایی که برای پورتهای بالای 1024 فرستاده شده‌اند اجازه عبور می‌دهند و فرض می‌شود که اینها همه بسته‌هایی هستند که در پاسخ به درخواستهای داخلی فرستاده شده‌اند. این یک سیاست ضعیف است چون ممکن است یک Trojan Horse نیز به یک پورت بالای 1024 گوش کند و بتوان از بیرون از شبکه به آن وصل شده اطلاعات در مورد شبکه بدست آورد یا اقدام به خرابکاری کرد.

وقتی یک سیستم از شبکه داخلی تقاضای برقراری ارتباط با یک سرور خارج از شبکه می‌کند یک بسته TCP SYN ایجاد کرده و به سرور می‌فرستد. قبل از اینکه بسته به سرور برسد فیلتر سر راه اطلاعات بسته را بررسی می‌کند. چون این یک بسته برای ایجاد ارتباط است یک رکورد برای آن ارتباط در جدول ارتباطات موجود ایجاد شده و وضعیت ارتباط نیز ثبت می‌گردد. هنگامی که پاسخ این بسته برمی‌گردد فیلتر با مقایسه آدرس فرستنده و گیرنده

و وضعیت ارتباط مربوطه متوجه می‌شود که این همان پاسخی است که قرار است برسد و بسته را به سمت مقصدش می‌فرستد. اگر بسته‌ای برای شبکه ارسال شود که هیچ رکورد متناظر با آن در جدول وجود نداشته باشد بسته از فیلتر نمی‌تواند عبور کند. این عملکرد در شکل 2-1 نشان داده شده‌است. به محض اینکه بسته‌های مربوط به بستن ارتباط از سوی سرور و سرویسگیر رد و بدل می‌شوند رکورد مربوط به آن ارتباط از جدول حذف می‌شود. بستن ارتباط نیز در شکل 2-2 نشان داده شده‌است.



شکل 2-1



شکل 2-2

3-2 مشکلات فیلترها

فیلترها (استاندارد و stateful) دارای مشکلاتی هستند که مهاجمان از آنها برای دور زدن این فیلترها استفاده می‌کنند. از جمله این ضعفها می‌توان موارد زیر را نام برد:

اطلاعات مربوط به پورتهای TCP و UDP فقط در fragment شماره 0 وجود دارند.

بعضی سیستم‌عاملها کنترل دقیقی روی ترتیب fragmentها ندارند و به محض اینکه fragment آخر را دریافت کنند کل fragmentها را سرهم می‌کنند و در صورتی که یک بسته کامل تشکیل شود آن را به لایه بالاتر منتقل

می‌کنند. بعضی از این ضعف استفاده کرده و یک بسته کامل را fragment کرده ولی آنها را از 1 شماره‌گذاری می‌کنند. در طرف گیرنده اطلاعات مربوط به پروتکل لایه شبکه که در fragment شماره 1 قرار دارد توسط فیلتر نادیده گرفته می‌شود و بنابراین تمام fragmentها از فیلتر عبور می‌کنند و یک بسته کامل IP را تشکیل می‌دهند.

بسیاری از فیلترها بسته‌های ارسالی برای پورتهای بالای 1024 را عبور می‌دهند.

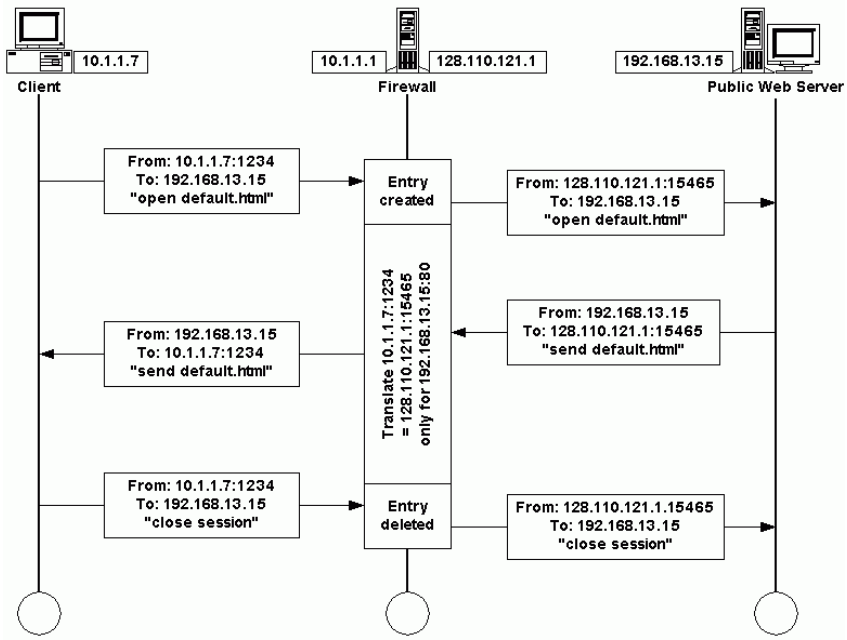
این مشکل فقط مربوط به فیلترهای استاندارد می‌شود که در قسمتهای قبل نیز در مورد آن توضیح داده شده‌است. فیلترهای stateful تمام تقاضاهایی را که برای برقراری ارتباط فرستاده می‌شوند شناسایی کرده و جلوی آنهایی را که مجاز نیستند می‌گیرند.

NAT های داخلی می‌توانند فیلتر را فریب دهند.

اگر کسی در داخل شبکه قادر به راه اندازی یک سرویس NAT بر روی سیستم خود باشد می‌تواند بسته‌هایی را که به یک پورت فیلتر نشده فرستاده می‌شوند ترجمه کرده و به یک پورت فیلتر شده روی یک سرور بفرستد. با استفاده از پراکسی نیز می‌توان این کار را انجام داد. در این مورد در قسمتهای بعد توضیح داده خواهد شد.

3 NAT

سیستم NAT آدرسهای IP شبکه محلی را به آدرسهای یکتا برای استفاده بر روی اینترنت تبدیل می‌کند. هرچند این روش برای ایجاد آدرسهای بیشتر برای استفاده در شبکه داخلی ابداع شده‌است ولی می‌توان از آن برای مخفی کردن اطلاعات مربوط به سیستمهای داخلی نیز استفاده کرد. NAT می‌تواند تمام اطلاعات مربوط به پروتکلهای TCP/IP شبکه داخلی را مخفی کند و از دید خارج چنین به نظر برسد که تمام ترافیک از یک آدرس خاص منتشر می‌شود. NAT همچنین این امکان را فراهم می‌کند که هر محدوده آدرسی را بتوان برای سیستمهای داخلی استفاده کرد، بدون اینکه کوچکترین مشکلی برای شبکه پیش بیاید. فایروال برای این منظور آدرس و شماره پورت مبدا تمام بسته‌هایی را که از شبکه داخلی فرستاده می‌شوند عوض می‌کند. آدرس خود فایروال (یا هر آدرس دیگری که فایروال به آن پاسخ می‌دهد) بجای آدرس مبدا و یک شماره پورت بجای شماره پورت مبدا قرار می‌گیرد. این شماره پورت برای شناسایی آدرس و شماره پورت مبدا بسته بکار می‌رود و اطلاعات مربوطه در یک جدول نگهداری می‌شود. هنگام برگشت پاسخ بسته از اطلاعات این جدول برای پیدا کردن گیرنده آن استفاده می‌شود. در واقع فایروال بین یک سری سوکت داخلی و یک سری سوکت خارجی ارتباط ایجاد می‌کند. شکل 3-1 این عملکرد را با یک مثال توضیح می‌دهد.



0شکل 1-3

در این مثال یک سیستم داخلی با آدرس 10.1.1.7 می‌خواهد ارتباط HTTP با یک سیستم خارجی با آدرس 192.168.13.15 برقرار کنند. یک بسته برای مثال از 10.1.1.7:1234 به 192.168.13.15:80 فرستاده می‌شود. فایروال یا مسیریاب سرراه (با آدرسهای 10.1.1.1 و 128.110.121.1) این بسته را گرفته و این رکوردها را در یک جدول ثبت می‌کند:

- مبدا: 10.1.1.7:1234
- مقصد: 192.168.13.15:80
- مبدا بسته تولیدشده: 128.110.121.1:15465

سپس بسته را با اطلاعات جدید می‌فرستد، یعنی سرور وب یک تقاضا از طرف 128.110.121.1:15465 دریافت می‌کند. هنگام پاسخ دادن نیز سرور بسته‌ها را برای همین آدرس می‌فرستد. فایروال هنگام دریافت بسته جدول داخلی را برای پیدا کردن گیرنده واقعی بسته جستجو می‌کند. اگر اطلاعات رکورد مربوطه با اطلاعات سرآیند بسته مطابقت داشت بسته پس از اعمال تغییرات لازم برای گیرنده واقعی بسته (10.1.1.7:1234) فرستاده می‌شود. اگر رکورد مربوطه پیدا نشد یا اطلاعات سرآیند بسته با رکورد مربوطه مطابقت نداشت بسته اجازه عبور پیدا نمی‌کند.

از آنجا که NAT فقط از یک سری جایگزینی ساده در لایه شبکه استفاده می‌کند و مانند پراکسی‌ها نیازی به اجرای عملیاتهای پیچیده ندارد می‌تواند در بسیاری موارد با سرعتی نزدیک به سرعت routing عمل کند.

1-3 انواع ترجمه آدرس در NAT

خیلی از فایروالها از روشهای مختلفی برای ترجمه آدرس استفاده می‌کنند. چهار عملکرد اصلی NAT به ترتیب میزان استفاده در زیر آمده‌است:

- ترجمه ایستا (port forwarding) : حالتی است که یک سیستم خاص (مثلا یک سرور) همیشه دارای ترجمه آدرس ثابتی است که امکان برقراری ارتباط از طرف سیستمهای خارجی با آن را فراهم می‌کند.
- ترجمه پویا (اتوماتیک) : حالتی است که یک عده از سیستمهای داخلی از یک یا چند آدرس برای ارتباط با شبکه خارجی استفاده می‌کنند. این روش برای مخفی کردن مشخصات سیستمهای داخلی یا گسترش محدوده آدرسهای مورد استفاده در شبکه داخلی استفاده می‌شود.
- توزیع بار : در این حالت یک آدرس ثابت به یک سری آدرس دیگر ترجمه می‌شود که همه سرورهایی هستند که به یک درخواست خاص پاسخ می‌دهند. این روش برای توزیع بار یک سرویس پرترافیک بر روی یک سری سرور استفاده می‌شود.
- افزونگی (Redundancy) : در حالتی که یک شبکه از چند روش برای اتصال به اینترنت استفاده می‌کند از این روش استفاده می‌شود تا در صورت قطع شدن هرکدام از مسیرها از مسیر دیگر استفاده شود.

1-1-3 ترجمه پویا

ترجمه پویا سیستمهای داخلی را از طریق تعویض آدرسهای آنها با یک آدرس دیگر محافظت می‌کند. تا زمانی که یک سیستم داخلی با یک سیستم خارجی ارتباط برقرار نکرده‌است هیچ رکوردی در مورد آن در جدول داخلی فایروال وجود ندارد و در نتیجه هیچ راهی برای برقراری ارتباط با آن وجود ندارد. توجه کنید که فایروال فقط مانع از این می‌شود که سیستمهای خارجی با یک سیستم داخلی ارتباط برقرار کنند. اگر سیستم داخلی به یک سرور خطرناک در بیرون شبکه وصل شود یا یک Trojan Horse روی سیستم باشد که به یک سرور خارجی متصل شود دقیقاً مانند حالتی است که هیچ فایروالی در میان نیست. به همین علت استفاده از NAT به تنهایی برای امنیت شبکه کافی نیست.

فریب زدن سیستم برای وصل شدن به یک سرور کار ساده‌ای است. فرض کنید یک نامه الکترونیکی از طرف رئیس برای یکی از کارمندان فرستاده می‌شود که در آن از وی خواسته شده که به یک سرور وب وصل شود. به احتمال خیلی قوی آن کارمند بدون فکر کردن به آن سرور وصل خواهد شد. جعل کردن نامه الکترونیکی نیز کار خیلی ساده‌ای است.

تعداد کل ارتباطاتی که از طریق یک فایروال NAT می‌تواند ترجمه شود محدود است. از آنجا که در سرآیند IP فقط 16 بیت برای شماره پورت در نظر گرفته شده‌است این تعداد به 65536 محدود می‌شود. البته در بسیاری از سیستم‌عامل‌ها این تعداد به 50000 نیز کاهش می‌یابد، چون بعضی از این پورتها برای مصارف خاصی رزرو شده‌اند.

2-1-3 ترجمه ایستا

ترجمه ایستا (یا port forwarding) در مواردی مورد استفاده قرار می‌گیرد که یک سریس خاص داخل شبکه ارائه می‌شود که باید از بیرون شبکه قابل دسترسی باشد. در بعضی شرایط هم که از پروتکلی استفاده می‌شود که برای درست کار کردن باید آدرس یا شماره پورت خاصی داشته باشد از ترجمه ایستا استفاده می‌شود.

برای قرار دادن یک سرور پست الکترونیکی داخل شبکه باید یک مسیر ثابت از فایروال تا خود سرور ایجاد شود. اگر آدرس سرور 10.1.1.21 و آدرس خارجی فایروال نیز 128.110.121.1 باشد این مسیر می‌تواند از 128.110.121.1:25 به 10.1.1.21:25 باشد. اگر بیش از یک سرور داخل شبکه باشد می‌توان برای هرکدام از آنها یک پورت فایروال را اختصاص داد. تمام این سرورها با یک آدرس خاص مشخص شوند درصورتی‌که هرکدام از آنها را یک سرور خاص مدیریت می‌کند.

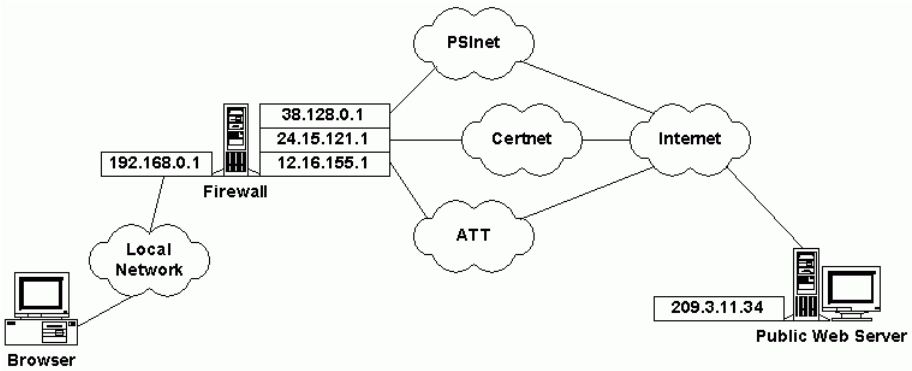
3-1-3 توزیع بار

بعضی از فایروالها می‌توانند با استفاده از NAT کار توزیع بار را روی شبکه انجام دهند. توزیع بار یعنی توزیع ترافیک یک سرویس روی چند سرور، مثلاً تقاضاهایی که برای یک سایت وب خیلی پرطرفدار فرستاده می‌شود روی سرورهای مختلف پخش شود. این توزیع می‌تواند بر اساس میزان بار موجود در هرکدام از سرورها یا از طریق الگوریتم round-robin باشد. برای تصمیم‌گیری از طریق میزان بار موجود، هر سرور باید راهی برای دادن اطلاعات در این مورد به فایروال داشته باشد. از آنجا که هیچ راه استانداردی برای این کار نیست خود فایروال باید یک راه مخصوص برای خود داشته باشد. به همین دلیل هم اغلب فایروالها از روش round-roubin استفاده می‌کنند و فرض می‌کنند که همه سرورها یک میزان بار را دارند. به این نکته توجه کنید که توزیع بار فقط برای پروتکلهایی قابل استفاده است که stateless باشد یا وضعیت ارتباط در سمت سرویسگیر نگهداری شود.

4-1-3 افزونگی (Redundancy)

این نوع ترجمه آدرس ترکیبی از ترجمه پویا و توزیع بار می‌باشد. در این حالت فایروال از طریق چند اتصال مختلف به اینترنت وصل می‌شود و روی هرکدام از آنها یک آدرس مجزا دارد. هنگامی که یک سرویسگیر از داخل شبکه تقاضای برقراری ارتباط با یک سرور خارجی را می‌کند، فایروال با توجه به وضعیت اتصالات یک مسیر (ارزانترین مسیر) را برای برقراری ارتباط با سرور انتخاب می‌کند. سپس با استفاده از ترجمه پویا آدرس سرویسگیر روی آدرس فایروال ترجمه می‌شود و تقاضای موردنظر برای سرور فرستاده می‌شود. در این حالت فایروال بار سرویسگیرهای داخلی را روی چند مسیر مختلف توزیع می‌کند. شکل 2-3 این عملکرد را توضیح می‌دهد. قطع شدن هرکدام از این اتصالات به منزله پر شدن بار آن اتصال در نظر گرفته می‌شود و فایروال هیچ سرویسگیر جدیدی را از آن مسیر عبور نخواهد داد. هرچند پروتکلهای stateful نیاز به برقراری مجدد ارتباط دارند، پروتکلهای

stateless فقط با یک پیغام خطا مواجه می‌شوند و بعد از آن تمام ارتباطات آنها از طریق مسیر جدید برقرار می‌شود بدون اینکه متوجه شوند چه اتفاقی افتاده‌است.



شکل 2-3

2-3 مشکلات NAT

بعضی پروتکلها هستند که از طریق NAT قابل استفاده نمی‌باشند. از این جمله‌اند:

- پروتکلهایی که نیاز به برقراری ارتباط مجدد با سرویسگیر دارند : هیچ مسیر مشخصی به سمت سرویسگیر وجود ندارد. پروتکلهای H.323 (video conferencing)، RSH و IRC از این دسته اند.
- پروتکلهایی که آدرسهای TCP/IP را داخل اطلاعات بسته قرار می‌دهند : اطلاعات داخل بسته با اطلاعات سرآیند یکسان نیست؛ در حقیقت اطلاعات داخل بسته اشتباه می‌باشد. پروتکل FTP از این دسته پروتکلهاست.
- پروتکلهایی که اطلاعات سرآیند TCP/IP را رمز می‌کنند : فایروال قادر به خواندن اطلاعات سرآیند و تغییر آدرسها یا شماره پورتها نیست. پروتکل PPTP از این دسته پروتکلهاست.
- پروتکلهایی که از آدرس فرستنده برای چک کردن مسائل امنیتی استفاده می‌کنند : اطلاعات سرآیند بسته عوض شده و نمی‌توان از آن بعنوان یک معیار استفاده کرد. پروتکل Sqlnet2 از این دسته پروتکلهاست.
- علاوه بر این پروتکل ICMP نیز با NAT مشکل دارد. نرم‌افزار ICMP بعضی وقتها قسمت اول بسته اصلی را (که شامل آدرسهای ترجمه نشده می‌باشد) داخل پیغام ICMP قرار می‌دهد. البته از لحاظ امنیتی هیچ لزومی ندارد که بسته‌های ICMP بتوانند از فایروال عبور کنند.

استفاده از NAT یک سری مشکلات امنیتی نیز دارد که به مواردی از آنها در اینجا اشاره می‌شود:

ترجمه ایستا = عدم امنیت

استفاده از ترجمه ایستا سیستمهای داخلی را محافظت نمی‌کند. استفاده از ترجمه ایستا فقط آدرس و شماره پورت سرویسگیر را بصورت یک به یک عوض می‌کند و هیچ مکانیزم امنیتی روی ارتباط ایجاد شده برقرار نمی‌کند. برای محافظت از یک سرویس داخل شبکه باید از پراکسی استفاده کرد.

یک ارتباط همیشه دوطرفه است.

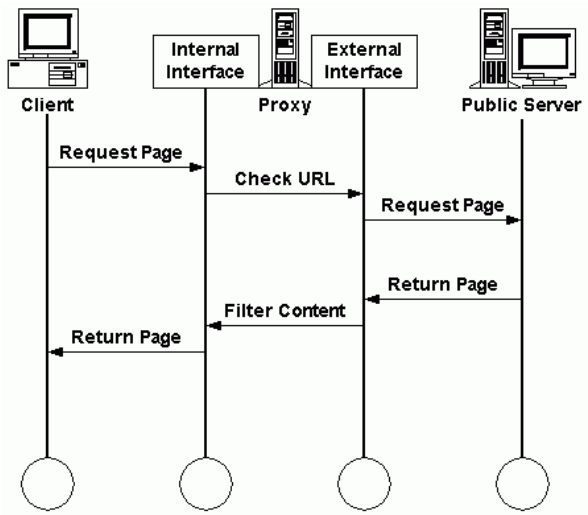
وقتی یک سرویسگیر با یک سرور ارتباط برقرار می‌کند، یک ارتباط از سرور به سمت سرویسگیر نیز ایجاد می‌شود. برقراری ارتباط با بعضی سرورها (مثلاً یک وب سایت) ممکن است منجر به بروز مشکلات امنیتی در شبکه شود. از آنجا که روی تمام ارتباطات ایجاد شده از طرف شبکه داخلی نمی‌توان کنترل داشت بهتر است برای هر سرویس از پراکسی استفاده کرد تا محتویات بسته‌هایی که وارد شبکه می‌شوند کنترل شود.

4 پراکسی

پراکسی‌ها در ابتدا برای cache کردن صفحات وب طراحی شده‌بودند. در روزهای اولیه اینترنت ارتباط برقرارکردن با منابع دیگر نسبتاً کند بود، وب نسبتاً کوچک بود و صفحات وب ثابت (ایستا) بودند. به همین دلیل cache کردن این صفحات کار مفیدی بود. در دنیای امروز سرعت دسترسی به اینترنت خیلی افزایش یافته است، وب به یک دنیای بی‌سروته تبدیل شده و صفحات وب نیز به سرعت به روز درمی‌آیند و محتوای آنها نیز پویا است. از این رو cache کردن صفحات وب چندان کار مفیدی نیست. پراکسی‌ها امروزه برای مقاصد دیگر استفاده می‌شوند. آنها می‌توانند مشخصات کاربران واقعی یک شبکه را پشت یک سیستم واحد مخفی کنند، URLها را فیلتر کنند، محتویات صفحات را کنترل کرده و مانع از ورود موارد مشکوک به شبکه داخلی شوند.

بسیاری از پراکسی‌های حقیقی ابزارهای دیگری مانند NAT و فیلتر را نیز همراه دارند که به آنها کمک می‌کند تا یک ابزار کامل برای امن کردن شبکه باشند. این متن فقط به بررسی پراکسی‌های خالص می‌پردازد. پراکسی‌ها بیشتر برای سرویس وب استفاده می‌شوند چون از ابتدا برای این سرویس طراحی شده‌اند، ولی برای سرویسهای دیگر نیز می‌توان از آنها استفاده نمود و مسلماً عملکرد آنها نیز کاملاً متفاوت خواهد بود. مثالهای این متن در مورد پراکسی سرویس وب است، ولی قابل تعمیم به سایر سرویسها نیز می‌باشند.

یک پراکسی به تمام تقاضاهای برقراری ارتباط که از شبکه داخلی فرستاده می‌شوند گوش می‌دهد و بمحض اینکه یک تقاضا از طرف یک سیستم فرستاده می‌شود درخواست مربوطه را دریافت کرده و آن را از طرف خودش برای سرور مربوطه می‌فرستد و پاسخ سرور را برای سرویسگیر داخلی ارسال می‌کند. شکل 4-1 این فرایند را بیان می‌کند.



شکل 4-1

4-1 عملکردهای امنیتی پراکسی

عمل تبدیل تقاضا از سرویسگیر به پراکسی و قرار دادن پراکسی بین شبکه داخلی و خارجی دارای مزایای امنیتی زیر است:

- پراکسی اطلاعات مربوط به سیستمهای داخل شبکه را پنهان می‌کند.
- پراکسی می‌تواند جلوی یک سری از URL ها را بگیرد.
- پراکسی می‌تواند با چک کردن محتویات صفحات وب جلوی یک سری از اطلاعات خطرناک مانند ویروس یا Trojan Horse را بگیرد.
- پراکسی می‌تواند سالم رسیدن اطلاعات دریافتی را چک کند.
- پراکسی یک نقطه واحد برای کنترل و log کردن فراهم می‌کند.

4-1-1 پنهان کردن اطلاعات سرویسگیرها

مهمترین عملکرد پراکسی پنهان کردن اطلاعات سرویسگیرهاست. درست مانند NAT یک پراکسی طوری رفتار می‌کند که از دید شبکه خارجی کل ترافیک متعلق به یک سیستم واحد به نظر بیاید. تفاوت پراکسی با NAT در این است که پراکسی در لایه transport کار می‌کند و فقط برای یک سرویس خاص این کار را انجام می‌دهد. عملکرد پراکسی چیزی فراتر از عوض کردن یک سری آدرس و شماره پورت می‌باشد و آن نیز به این صورت است: بسته‌ای که از داخل شبکه فرستاده شده توسط پراکسی دریافت می‌شود، درست مانند اینکه مقصد بسته خود پراکسی باشد. این بسته سپس بعنوان یک تقاضا برای سرور مربوطه فرستاده شده و پاسخ آن به سرویسگیر داخلی تحویل داده می‌شوند. از آنجا که کل بسته دوباره ایجاد می‌شود، فقط یک پروتکل خاص (مثلاً HTTP) می‌تواند از پراکسی رد شود نه پروتکلهای دیگر.

یکی از کاربردهای خوب دیگر پراکسی استفاده از یک راه اتصال به اینترنت برای چند سیستم است. با نصب یک پراکسی ساده روی یک سیستم، سیستمهای دیگر می‌توانند از طریق آن به اینترنت وصل شوند.

4-1-2 بستن یک سری URL

با داشتن پراکسی می‌توان جلوی دسترسی سیستمهای داخلی به یک سری منابع (سایت‌های hack، porno، ...) را بست. این کار به سادگی انجام می‌شود: پراکسی می‌تواند قبل از اینکه درخواست سرویسگیر را برای سرور ارسال

کند آدرس سرور را با یک پایگاه داده یا یک لیست چک کند و در صورتی که آدرس سرور غیرمجاز شناخته شود درخواست جدید از طرف پراکسی ایجاد نمی‌شود و عملاً سرویسگیر نمی‌تواند به سرور مربوطه متصل شود.

3-1-4 کنترل محتویات بسته‌ها

از آنجا که پراکسی در لایه transport کار می‌کند و مخصوص یک سرویس خاص می‌باشد می‌تواند محتویات بسته را نیز چک کند، به همین دلیل از پراکسی می‌توان برای کنترل محتویات بسته‌ها نیز استفاده کرد و موارد مشکوک را سر راه حذف کرد. برای مثال می‌توان از پراکسی وب برای حذف کردن برنامه‌های ActiveX و اپلت‌های جاوا، یا از پراکسی SMTP برای چک کردن نامه‌ها (وجود ویروس، نامه‌های SPAM، ...) قبل از دریافت آنها استفاده کرد.

4-1-4 اطمینان از سالم بودن بسته‌ها

بعضی از سرویسها bugهایی دارند که با استفاده از این bugها می‌توان مشکلاتی برای آنها ایجاد کرد. مشهورترین این حملات buffer-overflow است. به نمونه‌های قدیمی این حملات دقت کنید: در سرویس SMTP سیستمهای مبتنی بر UNIX اگر حجم نامه از آنچه در ابتدای آن ذکر شده بیشتر باشد باعث می‌شود برنامه‌ای که در این فضای اضافی قرار دارد اجرا شود. در بعضی سرورهای وب اگر سایز URL بیشتر از 256 کاراکتر باشد موجب اجرای برنامه موجود در فضای اضافی می‌شود. این نوع حملات همه با فرستادن بسته‌های معیوب به سرور انجام می‌شوند و نمونه‌های آنها زیادند که به مرور زمان با پیدا شدن یک bug جدید یک نوع جدید این حملات کشف می‌شود. بهترین راه برای امان ماندن از این حملات چک کردن بسته‌های رسیده از لحاظ سالم و کامل بودن است. از آنجایی که این نوع بسته‌ها فقط با چک کردن اطلاعات داخل بسته بدست می‌آیند، فقط توسط یک پراکسی قابل تشخیص می‌باشند نه در لایه‌های پایینتر.

5-1-4 کنترل روی دسترسی‌ها

پراکسی تنها نقطه‌ای از شبکه است که سیستمهای داخلی از طریق آن می‌توانند به یک سرویس (مثلاً HTTP) روی اینترنت دسترسی داشته باشند و به همین دلیل نقطه خوبی برای اعمال کنترل روی اطلاعات ردوبدل شده می‌باشد. پراکسی می‌تواند دسترسی به سرویس را بر حسب کاربر کنترل کند، میزان استفاده از سرویس را بر حسب کاربر محاسبه کند، لیست سایتهایی را که توسط کاربران دیده‌شده نگهداری کند، و حتی هنگام بروز مشکلات امنیتی خاص alarm ایجاد کند.

2-4 تاثیر پراکسی در سرعت

علاوه بر سرویسهای امنیتی که توسط پراکسی ارائه می‌شود، پراکسی می‌تواند عملکرد قابل توجهی در افزایش سرعت دسترسی به شبکه داشته باشد:

- پراکسی می‌تواند اطلاعاتی را که از طرف شبکه داخلی مرتب درخواست می‌شوند cache کرده و از این طریق دسترسی به شبکه خارجی را تا حد ممکن کاهش دهد.
- پراکسی می‌تواند عمل توزیع بار را روی چند سرور داخلی انجام دهد.

1-2-4 cache کردن

پراکسی‌ها در ابتدا به همین منظور ایجاد شده‌اند. البته با تغییراتی که در دنیای وب صورت گرفته‌است cache کردن صفحات زیاد مفید نیست، مخصوصا صفحاتی که با ابزارهایی مانند ASP نوشته می‌شوند مرتب تغییر محتوا می‌دهند. فقط در برخی موارد که دسترسی به یک یا چند سایت مشخص از طرف کاربران شبکه داخلی زیاد انجام می‌شود cache کردن صفحات اصلی، اشکال و اپلت‌ها می‌تواند مفید باشد.

2-2-4 توزیع بار

اغلب پراکسی‌های جدید می‌توانند در نقش پراکسی معکوس نیز عمل کنند؛ یعنی سرویسگیرهای خارجی را به سرورهای داخلی متصل می‌کنند. در حقیقت کاری که در این میان انجام می‌گیرد توزیع بار سرویسگیرهای خارجی روی سرورهای داخلی می‌باشد. بسیاری از وب سرورهای جدید به علت حجیم بودن عمکردشان نمی‌توانند روی یک سیستم واحد قرار بگیرند. بیشتر برنامه‌نویسی سمت سرور (ASP، JSP، CGI، ...) و تجارت الکترونیکی باعث افزایش بار روی سرور می‌شوند. در این حالت پراکسی تمام درخواستها را از سرویسگیرهای خارجی دریافت کرده و آنها را به سمت سرورهای داخلی هدایت می‌کند. این کار بسیار مشابه توزیع بار از طریق NAT است، با این تفاوت که در NAT این کار در لایه شبکه انجام می‌شد ولی در پراکسی در لایه transport است.

3-4 مشکلات پراکسی

پراکسی دارای یک سری مشکلات می‌باشد که در این قسمت به آنها اشاره می‌شود. پراکسی از کل شبکه داخلی بخوبی محافظت می‌کند، ولی در عوض خودش در معرض خطر قرار دارد. بعضی از مشکلات امنیتی پراکسی از قرار زیر است:

نقطه شکست واحد

شبکه‌ای که دارای یک نقطه اتصال و کنترل است، دارای یک نقطه شکست نیز می‌باشد. با از کار افتادن پراکسی اتصال کل شبکه به اینترنت قطع خواهد شد. پراکسی‌ها، مسیریابها و فایروالها همگی تا حدی دارای این مشکل می‌باشند. این مشکل در مورد مسیریابها با تعریف کردن مسیرهای دیگر به اینترنت حل می‌شود. فایروالها نسبت به پراکسی‌ها امن‌ترند چون فایروالها دارای ابزارهایی مانند فیلتر می‌باشند و قادر به محافظت از خود می‌باشند. پراکسی‌های خالص دارای هیچ مکانیزمی در لایه‌های پایین نمی‌باشند و از این لحاظ خیلی آسیب‌پذیر می‌باشند. برای برقراری امنیت در لایه‌های پایین خود پراکسی باید دارای ابزاری برای فیلترکردن بسته‌ها باشد یا از فیلتر خود سیستم‌عامل برای محافظت استفاده کند.

محدودیت روی سرویسگیرها

اگر یک سرویس قرار باشد از طریق پراکسی قابل دسترسی باشد، سرویسگیرها باید بدانند که چگونه با پراکسی رابطه برقرار کنند (اصلاً با پراکسی ارتباط برقرار کنند). یک سرویسگیر وب برای اینکه با پراکسی کار کند باید طوری تنظیم شود که بسته‌ها را بجای سرور مقصد برای پراکسی بفرستد. اگر یک سرویسگیر قادر به این کار نباشد، نمی‌توان از پراکسی برای آن سرویس استفاده کرد. این مشکل برای سرویسهایی مانند FTP وجود دارد. برای سرویسگیرهای FTP که با اغلب سیستم‌عاملها عرضه می‌شوند دارای این قابلیت نمی‌باشند. در این موارد می‌توان نسخه‌ای از نرم‌افزار را که استفاده از پراکسی را پشتیبانی می‌کند تهیه کرد. استفاده از NAT نیز می‌تواند این مشکل را حل کند چون در NAT می‌توان آدرس فرستنده و گیرنده بسته را تغییر داد. در این حالت بسته‌ها بدون اینکه تنظیم خاصی در سرویسگیرها انجام شود از طرف فایروال برای پراکسی فرستاده می‌شوند. چون پراکسی از دید سرویسگیرها دیده نمی‌شود به آن transparent proxy می‌گویند.

تمام سرویسها باید پراکسی داشته باشند.

برای برقراری امنیت روی تمام سرویسهای شبکه، باید برای تمامی آنها از پراکسی استفاده کرد. تنها پراکسی می‌تواند بسته‌های یک سرویس خاص را بررسی کرده و موارد مشکوک را سر راه حذف کند. در ضمن بعضی سرویسها را (از جمله آنهایی که نیاز به برقراری ارتباط معکوس روی سرویسگیر دارند) نمی‌توان پشت NAT قرار داد و برای آنها باید از پراکسی استفاده کرد. چون پراکسی جدولی از ارتباطات تشکیل شده را نگهداری می‌کند می‌تواند ارتباطات معکوس را در صورت لزوم برقرار کند.

پیکربندی اولیه

معمولا پیکربندی اولیه پراکسی‌ها از لحاظ امنیتی چندان محکم نیست و در صورتی که تصحیح نشود می‌تواند خطر بزرگی برای شبکه باشد. علت آن این است که پراکسی‌ها بیشتر برای به اشتراک گذاشتن ارتباط به اینترنت (و معمولاً برای کاربران غیرتکنیکی) تهیه می‌شوند نه برای مقاصد امنیتی.

علاوه بر موارد ذکرشده در بالا، پراکسی از لحاظ سرعت هم یک مشکل دارد:

ایجاد گلوگاه

پراکسی مانند مسیریاب و فایروال به تنها محل عبور سرویسگیرها تبدیل می‌شود و به همین علت ممکن است بار زیادی روی آن باشد. پراکسی بر خلاف مسیریاب و فایروال در لایه بالاتر کار می‌کند و بسته‌های دریافتی را دوباره تولید می‌کند، به همین علت سرعتش به مراتب کمتر از مسیریاب و فایروال است. برای حل این مشکل باید به مرور زمان و افزایش کاربران تعداد پراکسی‌ها را افزایش داد و هر سری از کاربران را به یکی از آنها متصل کرد.

5 سیستم‌های تهاجم‌یاب

فایروال‌ها نقش بسیار مهمی در تامین امنیت شبکه دارند. آنها تا حد ممکن مانع نفوذ مهاجمین به شبکه می‌شوند، ولی قویترین فایروال‌ها نیز قادر به برقراری امنیت صددرصد نمی‌باشند و بعضی از مهاجمین می‌توانند از آنها عبور کرده دست به اعمال خرابکارانه بزنند. در حالت خیلی خوشبینانه تنها می‌توان در logهای فایروال نشانه‌هایی از حمله را پیدا کرد، آن هم بعد از وقوع آن.

برای روشن شدن بیشتر موضوع بعنوان مثال عملکرد یک کرم اینترنتی را در نظر بگیرید. یک کاربر داخل شبکه یک نامهٔ الکترونیکی از طرف یکی از دوستانش دریافت می‌کند که در آن مثلاً آخرین نسخه از بازی دلخواه وی قرار دارد. با اجرا کردن برنامه یک Trojan Horse برروی سیستم کاربر نصب می‌شود که سر فرصت و هنگامی که کاربر با سیستم کاری ندارد با باز کردن ارتباط به سمت یک سرور شروع به فرستادن اطلاعات مهم داخل شبکه برای آن می‌کند. چنین مکانیزمی توسط فیلترها قابل تشخیص نیست چون معمولاً چنین سرورهایی روی یک پورت مجاز مانند HTTP اجرا می‌شوند. در این حالت فقط پراکسی‌ها می‌توانند با کنترل محتویات بسته‌ها چنین مکانیزم‌هایی را تشخیص دهند. مهاجمین می‌توانند حتی بهتر نیز عمل کنند. آنها می‌توانند سرور خود را روی پورت FTP (یا هر پورت دیگری که اطلاعات باینری از طریق آن مبادله می‌شود) اجرا کنند. سرور و برنامهٔ Trojan Horse طوری تنظیم می‌شوند که قبل از فرستادن اطلاعات اصلی با مبادله یک سری بسته‌ها طوری وانمود کنند که یک ارتباط مجاز را برقرار می‌کنند. چنین حمله‌ای توسط پراکسی نیز قابل تشخیص نمی‌باشد. در این حالت حتی در logهای فایروال نیز نشانه‌ای از حمله دیده نمی‌شود، چون تمام مبادلات انجام شده مجاز می‌باشد. این قسمت با معرفی سیستم‌های تهاجم‌یاب یا IDS به بررسی چگونگی محافظت از شبکه در مقابل حملاتی که فایروال قادر به تشخیص آنها نمی‌باشد می‌پردازد.

سیستم‌های تهاجم‌یاب نرم افزارهایی هستند که با بررسی ترافیک شبکه و از روی یک سری نشانه‌ها این حملات را تشخیص می‌دهند. این نوع سیستم‌ها در کنار فایروال و برای ایجاد امنیت بیشتر بکار می‌روند. یک سیستم تهاجم‌یاب می‌تواند فقط از یک سیستم و یا از تمامی سیستم‌های موجود در شبکه محافظت کند که در حالت دوم NIDS نامیده می‌شود. نحوهٔ برخورد سیستم‌های تهاجم‌یاب در قبال موارد مشکوک و حملات آنها را به دو دستهٔ فعال (active) و انفعالی (passive) تقسیم می‌کند. سیستم‌های تهاجم‌یاب فعال می‌توانند طوری برنامه‌ریزی شوند که به محض بروز یک مورد مشکوک عکس‌العمل مناسب نشان دهند (قطع ارتباط مشکوک، تولید یک هشدار، ...)، ولی

سیستمهمای تهاجم‌یاب انفعالی فقط اتفاقات رخ داده را ثبت می‌کنند که بعداً می‌توانند مورد بررسی قرار گیرند. دو تکنیک مختلف برای طراحی سیستم‌های تهاجم‌یاب وجود دارد که در ادامهٔ به آنها اشاره می‌شود.

1-5 سیستم‌های تهاجم‌یاب بر مبنای بازرسی

این نوع سیستم‌ها متداولترین نوع تهاجم‌یاب می‌باشند. نحوهٔ عملکرد آنها بدین صورت است که فعالیت یک سیستم یا شبکه را زیر نظر می‌گیرند و از طریق یک سری قوانین تعریف شده و یا مقایسه با وضعیت عادی سیستم موارد مشکوک را کشف می‌کنند. تهاجم‌یاب‌ها از فاکتورهای زیر برای کشف موارد مشکوک استفاده می‌کنند:

- نشانه‌هایی که از تحلیل ترافیک شبکه بدست می‌آیند مانند Portscan یا وصل شدن به پورت‌های غیرمجاز
- نحوهٔ بهره‌گیری از منابع سیستم مانند پردازنده و حافظه و یا برقراری ارتباط با شبکه در زمان‌های غیرعادی که نشانهٔ یک حملهٔ اتوماتیک می‌باشد
- نحوهٔ استفاده از فایل‌سیستم مانند وجود فایل‌های تازه ایجاد شده، تغییرات در فایل‌های سیستم عامل یا تغییرات در account کاربران

سیستم‌های تهاجم‌یاب چند نوع مختلف می‌باشند:

- تهاجم‌یاب‌های قانون پذیر: این سیستم‌ها حملات را از روی یک سری قوانین تشخیص می‌دهند. هرکدام از این قوانین یک روش شناخته شده برای حمله را مشخص می‌کنند که یک سری نشانه‌ها یا یک دنباله از فعالیت‌ها را برای کشف حمله بکار می‌برند. وصل شدن به یک پورت خاص یا تغییرات در فایل‌های سیستم عامل از جملهٔ این نشانه‌ها می‌باشند.
- تهاجم‌یاب‌های آماری: این سیستم‌ها با مقایسهٔ وضعیت فعلی سیستم با وضعیت طبیعی آن حملات را کشف می‌کنند. این روش برای کشف حملات جدید و ناشناخته مناسب است ولی در عوض قادر به تشخیص بسیاری حملات بسیار سادهٔ شناخته شده نباشد و آنها را بعنوان رفتار طبیعی سیستم در نظر بگیرد. واضح است که این روش حملات را پس از وقوع آنها می‌تواند کشف کند.
- تهاجم‌یاب‌های هیبرید: این نوع سیستم‌ها ترکیب دو روش قبلی را بکار می‌برند. حملات شناخته شده را بوسیلهٔ قوانین مناسب و حملات ناشناخته را با تحلیل آماری کشف می‌کنند. بسیاری از این نوع سیستم‌ها قادرند پس از کشف یک حملهٔ ناشناخته قوانین مربوط به آن را نیز ایجاد کنند که از آن پس برای تشخیص آن نوع حمله بکار می‌رود.

2-5 سیستم‌های تهاجم‌یاب طعمه

این نوع سیستم‌ها به ظاهر مانند یک سرویس برروی شبکه هستند ولی به‌محض اینکه کسی قصد استفاده از آنها را داشته باشد یک هشدار امنیتی تولید می‌کنند. یک سیستم تهاجم‌یاب طعمه مانند یک سرور به نظر می‌آید که بخوبی از آن محافظت نمی‌شود و به همین علت به سرعت جلب توجه می‌کند. از این طریق توجه مهاجمین جلب سیستمی می‌شود که برای شبکه هیچ اهمیت عملیاتی ندارد و به محض اولین استفادهٔ مهاجم از آن یک هشدار امنیتی تولید می‌شود که مدیر شبکه را از وقوع حمله باخبر می‌کند. تهاجم‌یاب طعمه می‌تواند یک یا چند سرویس روی یک سرور، یک سرور مجزا و یا یک شبکهٔ مجزا (به ندرت) باشد. در حالت اول تهاجم‌یاب چند سرویس TCP/IP برروی یک سیستم ایجاد می‌کند که از آنها در شبکه استفاده نمی‌شود (مانند سرویس echo، chargen یا NFS). در بعضی موارد این سرویس‌ها حتی عملکرد سرویس را نیز ندارند و در پاسخ درخواست‌ها فقط timeout برمی‌گردانند و فقط وصل شدن به پورت را تشخیص می‌دهند. راه حل موثرتر استفاده از یک سیستم کامل بعنوان طعمه است. تنها کاری که باید انجام شود این است که یک نسخه از یک سیستم عامل مورد نظر به همراه تمام سرویس‌های معمول روی سیستم طعمه ایجاد کرد و NAT شبکه را طوری تنظیم نمود که تمام درخواست‌ها را به این سیستم بفرستد، سپس با اضافه کردن قوانین لازم ترافیک سرویس‌های خاصی مانند WWW و SMTP را به سرورهای مربوط فرستاد. کسی که شبکه را scan می‌کند تمام سرویس‌های سیستم بعلاوه سرویس‌های واقعی شبکه را با هم می‌بیند و اینگونه به نظر می‌آید که همه متعلق به یک سیستم واحد می‌باشند. از آنجا که حمله کردن سرویس‌های طعمه (NFS یا NetBIOS) به مراتب ساده‌تر است مهاجمین ترجیح می‌دهند در ابتدا آنها را امتحان کنند و همین باعث تولید هشدار امنیتی می‌شود. در این مرحله می‌توان به مهاجمین اجازه داد به کار خود ادامه دهند و روش‌هایی را که برای حمله به سیستم استفاده می‌کنند شناسایی کرد. مهاجمین پس از مدتی متوجه موضوع می‌شوند که تمهیدات امنیتی مناسب توسط مدیر شبکه برای مقابله با آنان در نظر گرفته شده است. حتی می‌توان یک کپی از سرویس‌های شبکه را نیز برروی این سیستم طعمه قرار داد که مهاجمین دیرتر متوجه موضوع شوند و مدیر شبکه فرصت بیشتری برای امن کردن سرورهای اصلی داشته باشد.

قبل از راه اندازی یک سیستم تهاجم‌یاب در شبکه باید چند نکته را در نظر داشت:

- سیستمی که قرار است تهاجم‌یاب برروی آن نصب شود از امنیت کافی برخوردار باشد.
- زمان سیستم باید صحیح باشد، چون در غیر این صورت اطلاعات ثبت شده و هشدارهای تولید شده توسط سیستم ارزش چندانی نخواهند داشت.
- مکان قرار گرفتن سیستم در شبکه باید مشخص شود. اگر تهاجم‌یاب بین شبکه فایروال و شبکهٔ خارجی قرار گیرد تمامی حملات را شناسایی می‌کند، ولی اگر بین فایروال و شبکهٔ داخلی قرار بگیرد فقط حملاتی را که از فایروال عبور می‌کنند شناسایی می‌کند.

IP Filter یک نرم‌افزار است که می‌توان از آن بعنوان NAT یا فیلتر استفاده کرد. این نرم‌افزار می‌تواند در فرم یک ماجول قابل load یا بعنوان قسمتی از kernel استفاده شود. IP Filter اکنون بعنوان بخشی از سیستم‌عامل FreeBSD عرضه می‌شود و عملکرد آن روی سیستم‌عاملهای زیر نیز کاملاً تست شده‌است:

- Solaris / Solaris-x86 2.3 - 8
- SunOS 4.1.1 - 4.1.4
- NetBSD 1.0 - 1.4
- FreeBSD 2.0.0 - 2.2.8
- BSD / OS-1.1 - 4
- IRIX 6.2
- OpenBSD 2.0 - 2.9
- HP-UX 11.00
- QNX Port

این نرم‌افزار از طریق آدرسهای زیر در دسترس می‌باشد:

- ftp://coombs.anu.edu.au/pub/net/ip-filter/
- http://coombs.anu.edu.au/~avalon/

مستندات تکنیکی IP Filter (HOWTO) نیز از طریق آدرس http://www.obfuscatio.org/ipf/ قابل دسترسی می‌باشد.

خصوصیتهای این نرم‌افزار به طور خلاصه از قرار زیر است:

- می‌تواند بسته‌هایی را که از کارت شبکه‌های مختلف می‌آیند تشخیص دهد.
- می‌تواند بسته‌ها را بر حسب آدرس شبکه یا host فیلتر کند.
- می‌تواند بسته‌ها را بر حسب پروتکل لایه شبکه فیلتر کند.
- می‌تواند بسته‌ها را بر حسب پروتکل IP فیلتر کند.
- می‌تواند جلوی بسته‌های fragment شده را بگیرد.
- می‌تواند جلوی بسته‌هایی که فیلد قسمت IP option آنها پرشده بگیرد که بسته‌های source-route جزء این دسته به شمار می‌آیند.
- می‌تواند در جواب بسته‌هایی که اجازه عبور پیدا نمی‌کنند بسته ICMP error یا TCP reset بفرستد.
- اطلاعات مربوط به وضعیت جریان بسته‌های TCP، UDP و ICMP را نگهداری می‌کند.
- اطلاعات مربوط به fragment های یک بسته را نگهداری می‌کند.
- می‌تواند بعنوان یک NAT کار کند.
- می‌تواند از redirection برای ایجاد ارتباطات یک transparent proxy استفاده کند.
- می‌تواند سرآیند بسته‌ها را به یک برنامه برای احراز هویت پاس دهد.
- می‌تواند یک سری از بسته‌های مشخص را برای log کردن روی یک device بفرستد.

- یک سری آمار از عملکرد نرم‌افزار فراهم می‌کند.

1-6 نصب IP Filter روی Solaris

نرم‌افزار IP Filter برای نصب روی Solaris 7 نیاز به کامپایلر 64 بیتی دارد. برای تبدیل gcc برای تولید باینریهای 64 بیتی باید اعمال زیر را انجام داد:

در مرحله اول باید یک کامپایلر 32 بیتی روی سیستم وجود داشته باشد. برای نصب gcc در این فرم باید مراحل زیر انجام شود:

- gunzip gcc-3.0.tgz
- tar xf gcc-3.0.tar
- mkdir gcc-3.0-sparc-32bit-obj
- cd gcc-3.0-sparc-32bit-obj
- ../gcc-3.0/configure --prefix=/usr/local/gcc-3.0 --enable-languages=c
- make
- make install

در مرحله دوم باید کامپایلر 64 بیتی را توسط کامپایلر موجود کامپایل و نصب کرد. برای این منظور باید مراحل زیر را انجام داد:

- mkdir gcc-3.0-sparc-64bit-obj
- cd gcc-3.0-sparc-64bit-obj
- ../gcc-3.0/configure --prefix=/usr/local/gcc-3.0-v9 --enable-languages=c sparcv9-sun-solaris2
- make
- make install

2-6 پیاده‌سازی یک فیلتر با استفاده از IP Filter

نحوه عملکرد فیلتر توسط یک سری قوانین کنترل می‌شود که در فایل ipf.conf قرار دارند. این فایل شامل یک سری رکورد است که هر رکورد در یک سطر قرار می‌گیرد. یک فایل ipf.conf نمونه شامل این سطرهاست:

```
pass in all
block in all
```

در اغلب فیلترها بررسی قوانین برای یک بسته از بالا به پایین انجام می‌شود و به محض اینکه بسته مورد نظر در یک قانون خاص صدق کرد قوانین بعدی بررسی نمی‌شوند. در IP Filter بر خلاف سایر فیلترها تمام قوانین موجود بررسی می‌شوند و آخرین قانونی که در مورد یک بسته صدق کند برای مشخص کردن وضعیت بسته استفاده می‌شود. در این مثال قانون اول باعث می‌شود تمام بسته‌ها از فیلتر عبور کنند و قانون دوم تمام بسته‌ها را فیلتر

می‌کند. می‌توان با اضافه کردن کلمه کلیدی quick این عملکرد را تغییر داد و بمحض صدق کردن یک قانون برای یک بسته قوانین دیگر بررسی نشوند. با اضافه کردن quick به قانون اول قوانین بالا به فرم زیر تبدیل می‌شوند:

```
pass in quick all
block in all
```

بسته‌ها را می‌توان بر حسب آدرس IP فرستنده و گیرنده آنها کنترل کرد. آدرسهای IP به فرم address/mask نوشته می‌شوند. قرار دادن ! قبل از آدرس نمایانگر تمام آدرسها غیر از آدرس ذکر شده است. کلمه کلیدی any نیز برای مشخص کردن تمام آدرسها (0.0.0.0/0) بکار می‌رود. یک نمونه از این نوع کنترل در زیر آمده است:

```
block in quick from 192.168.0.0/16 to any
block in quick from 172.16.0.0/12 to any
block in quick from 10.0.0.0/8 to any
pass in all
```

در این مثال هیچکدام از بسته‌هایی که آدرس فرستنده آنها یکی از سه دسته آدرس 192.168.0.0/16، 172.16.0.0/12 یا 10.0.0.0/8 باشد اجازه عبور پیدا نمی‌کنند. این آدرسها جزءِ آدرسهایی هستند که برای شبکه‌های محلی از آنها استفاده می‌شوند و روی اینترنت ردوبدل نمی‌شوند، بنابراین نیازی به عبور دادن آنها نمی‌باشد.

بسته‌ها را می‌توان بر حسب کارت شبکه و در دو جهت ورودی و خروجی کنترل کرد. بسته‌هایی که از یک کارت شبکه وارد سیستم می‌شوند با کلمه کلیدی in و بسته‌هایی که از یک کارت شبکه سیستم را ترک می‌کنند با کلمه کلیدی out مشخص می‌شوند. بعنوان یک مثال می‌توان قوانین زیر را برای فیلتر کردن تمام بسته‌های ورودی از کارت شبکه x10 استفاده کرد:

```
block in quick on x10 all
pass in all
```

معمولا از کارت شبکه به تنهایی برای کنترل بسته‌ها استفاده نمی‌شود بلکه اغلب از ترکیب کارت شبکه و آدرس IP استفاده می‌شود. با اطلاع از اینکه چه نوع بسته‌هایی در کدام جهت و از کدام کارت شبکه قرار است عبور کنند می‌توان کنترل دقیق روی بسته‌ها داشت. برای مثال اگر سیستم دارای دو کارت شبکه hme0 و x10 باشد که hme0 به شبکه داخلی با آدرس 20.20.20.0/24 و x10 به شبکه خارجی متصل باشد می‌توان تمام بسته‌هایی را که از کارت شبکه x10 با آدرس فرستنده 20.20.20.0/24 وارد سیستم می‌شوند فیلتر کرد. بعنوان یک مثال می‌توان قوانین زیر را برای محافظت از این شبکه استفاده کرد و آدرسهایی را که در شبکه‌های داخلی از آنها استفاده می‌شود فیلتر کرد:

```
block in quick on x10 from 192.168.0.0/16 to any
block in quick on x10 from 172.16.0.0/12 to any
block in quick on x10 from 10.0.0.0/8 to any
block in quick on x10 from 127.0.0.0/8 to any
block in quick on x10 from 0.0.0.0/8 to any
block in quick on x10 from 169.254.0.0/16 to any
block in quick on x10 from 192.0.2.0/24 to any
block in quick on x10 from 204.152.64.0/23 to any
```

```
block in quick on x10 from 224.0.0.0/3 to any
block in quick on x10 from 20.20.20.0/24 to any
pass in on x10 all
```

برای کنترل بسته‌ها در جهت عکس جای کلمه کلیدی in با out عوض می‌شود. با این کار می‌توان بسته‌هایی را که از کارت شبکه x10 خارج می‌شوند کنترل کرد و بسته‌های مشکوک را قبل از خارج شدن از شبکه فیلتر نمود. همانطور که نمی‌خواهیم بسته‌های غیرمجاز وارد سیستم شوند ممکن است نخواهیم این بسته‌ها از داخل شبکه ما برای دیگران فرستاده شوند. قوانین زیر این کار را انجام می‌دهند:

```
block out quick on x10 from 192.168.0.0/16 to any
block out quick on x10 from 172.16.0.0/12 to any
block out quick on x10 from 10.0.0.0/8 to any
block out quick on x10 from 127.0.0.0/8 to any
block out quick on x10 from 0.0.0.0/8 to any
block out quick on x10 from 169.254.0.0/16 to any
block out quick on x10 from 192.0.2.0/24 to any
block out quick on x10 from 204.152.64.0/23 to any
block out quick on x10 from 224.0.0.0/3 to any
block out quick on x10 from 20.20.20.0/24 to any
pass out on x10 all
```

برای کنترل کردن بسته‌ها براساس پروتکل لایه شبکه از کلمه کلیدی proto استفاده کرد. برای مثال برای فیلتر کردن بسته‌های ICMP می‌توان از دستور زیر استفاده کرد:

```
block in quick on x10 proto icmp from any to any
```

در این حالت تمام بسته‌های ICMP فیلتر می‌شوند و اجازه عبور از شبکه را ندارند. ممکن است لازم باشد فقط یک سری خاص از بسته‌های ICMP فیلتر شوند. برای اینکه ping و traceroute روی شبکه کار کند باید بسته‌های ICMP نوع 0 و 11 اجازه ورود به شبکه را داشته باشند. برای این کار از کلمه کلیدی icmp-type استفاده می‌شود:

```
pass in quick on x10 proto icmp from any to any icmp-type 0
pass in quick on x10 proto icmp from any to any icmp-type 11
block in quick on x10 proto icmp from any to any
```

هنگام اضافه کردن دو دسته مختلف از قوانین فیلتر باید به ترتیب قرار دادن آنها دقت کرد. اگر قوانین مربوطه به بسته‌های ICMP را قبل از قوانین قبلی قرار دهیم آنچه حاصل می‌شود چنین است:

```
pass in quick on tun0 proto icmp from any to 20.20.20.0/24 icmp-type 0
pass in quick on tun0 proto icmp from any to 20.20.20.0/24 icmp-type 11
block in log quick on tun0 proto icmp from any to any
block in quick on tun0 from 192.168.0.0/16 to any
block in quick on tun0 from 172.16.0.0/12 to any
block in quick on tun0 from 10.0.0.0/8 to any
block in quick on tun0 from 127.0.0.0/8 to any
block in quick on tun0 from 0.0.0.0/8 to any
block in quick on tun0 from 169.254.0.0/16 to any
block in quick on tun0 from 192.0.2.0/24 to any
```

```
block in quick on tun0 from 204.152.64.0/23 to any
block in quick on tun0 from 224.0.0.0/3 to any
block in log quick on tun0 from 20.20.20.0/24 to any
block in log quick on tun0 from any to 20.20.20.0/32
block in log quick on tun0 from any to 20.20.20.255/32
pass in all
```

در این حالت تمام بسته‌های ICMP نوع 0 و 11 بدون توجه آدرس فرستنده آنها از فیلتر عبور می‌کنند که بر خلاف انتظار ماست. بنابراین این دسته قوانین باید بعد از قوانین مربوط به آدرس IP قرار داده شوند. قوانین فیلتر پس از اعمال این تغییرات به فرم زیر در می‌آیند:

```
block in quick on tun0 from 192.168.0.0/16 to any
block in quick on tun0 from 172.16.0.0/12 to any
block in quick on tun0 from 10.0.0.0/8 to any
block in quick on tun0 from 127.0.0.0/8 to any
block in quick on tun0 from 0.0.0.0/8 to any
block in quick on tun0 from 169.254.0.0/16 to any
block in quick on tun0 from 192.0.2.0/24 to any
block in quick on tun0 from 204.152.64.0/23 to any
block in quick on tun0 from 224.0.0.0/3 to any
block in log quick on tun0 from 20.20.20.0/24 to any
block in log quick on tun0 from any to 20.20.20.0/32
block in log quick on tun0 from any to 20.20.20.255/32
pass in quick on tun0 proto icmp from any to 20.20.20.0/24 icmp-type 0
pass in quick on tun0 proto icmp from any to 20.20.20.0/24 icmp-type 11
block in log quick on tun0 proto icmp from any to any
pass in all
```

برای کنترل بسته‌ها براساس هرکدام از پروتکلهای IP می‌توان از کلمه کلیدی port استفاده کرد. شماره پورت TCP و UDP نوع پروتکل و سرویس ارائه شده را مشخص می‌کند. برای مثال برای فیلتر کردن سرویسهای rlogin، rsh و telnet باید پورتهای 513، 514 و 23 پروتکل TCP را بست:

```
block in quick on x10 proto tcp from any to 20.20.20.0/24 port = 513
block in quick on x10 proto tcp from any to 20.20.20.0/24 port = 514
block in quick on x10 proto tcp from any to 20.20.20.0/24 port = 23
```

تمام ارتباطات TCP/IP دارای سه قسمت آغازین، میانی و پایانی می‌باشند. یک ارتباط نمی‌تواند بدون داشتن قسمت میانی دارای قسمت پایانی باشد و نمی‌تواند بدون داشتن قسمت آغازین دارای قسمت پایانی باشد. حقیقت این است که هر طور که با قسمت آغازین یک ارتباط رفتار شود با سایر قسمتها هم همان رفتار خواهد بود، به همین دلیل می‌توان وضعیت قسمتهای میانی و پایانی یک ارتباط را از روی قسمت آغازین آن تعیین کرد؛ اگر ارتباط برقرار شود سایر بسته‌های مربوط به آن می‌توانند از فیلتر عبور کنند در غیر اینصورت بقیه بسته‌ها نیز فیلتر می‌شوند. برای محافظت از یک شبکه که فقط سرویس پست الکترونیکی در آن ارائه می‌شود می‌توان از قوانین زیر استفاده کرد:

```
block out quick on x10 all
pass in quick on x10 proto tcp from any to 20.20.20.0/24 port = 25 keep state
```

مشاهده می‌شود که هیچ قانون "pass out" در این دسته قوانین وجود ندارد ولی تمام ارتباطات لازم با سرور برقرار می‌شود چون تمام بسته‌هایی که از شبکه خارج می‌شوند توسط keep state کنترل می‌شود. برای کنترل شبکه طوری که سیستمهای داخلی فقط بتوانند بعنوان سرویسگیر عمل کنند می‌توان از دسته قوانین زیر استفاده کرد:

```
block in quick on x10 all
pass out quick on x10 proto tcp from 20.20.20.0/24 to any keep state
pass out quick on x10 proto udp from 20.20.20.0/24 to any keep state
pass out quick on x10 proto icmp from 20.20.20.0/24 to any keep state
```

در IP Filter راههایی برای کنترل بیشتر بسته‌های TCP وجود دارد. قانون زیر را در نظر بگیرید:

```
pass in quick on x10 proto tcp from any to 20.20.20.0/24 port = 25 keep state
```

در این حالت نه تنها بسته‌های SYN برای پورت 25 از فیلتر عبور می‌کنند، بلکه سایر بسته‌های مربوط به قسمتهای میانی و پایانی ارتباطات قبلی نیز این اجازه را پیدا می‌کنند. چه بسا این بسته‌ها جزئی از یک ارتباط نیز نباشند و به منظور خاصی فرستاده شده باشند. با افزودن کلمه کلیدی flags می‌توان flagهای سرآیند TCP را کنترل کرد. این کلمه کلیدی سه پارامتر مختلف می‌گیرد که عبارتند از S/AUPRFS، S/SA، S/SAFR و S به تنهایی که معادل S/AUPRFS می‌باشد. S/AUPRFS فقط بسته‌های SYN را مشخص می‌کند. S/SA بسته‌هایی را مشخص می‌کند که هرکدام از flagهای SYN، URG، PSH، FIN و RST در آنها تنظیم شده باشد. S/SAFR نیز بسته‌هایی را مشخص می‌کند که هرکدام از flagهای SYN، URG و PSH در آنها تنظیم شده باشد. برای مثال برای اینکه فقط بسته‌های SYN مربوط به ارتباط SMTP روی پورت 25 اجازه ورود به شبکه را داشته باشند می‌توان از قانون زیر استفاده کرد:

```
pass in quick on x10 proto tcp from any to 20.20.20.0/24 port = 25 flags S keep state
```

پس از اعمال قوانین ذکر شده تمام بسته‌هایی که از فیلتر عبور می‌کنند یا مربوط به یک ارتباط می‌باشند که وضعیت آن ثبت شده و یا برای برقراری ارتباط فرستاده شده اند. تنها استثنا بسته‌های fragment شده می‌باشند. frameهای 1 به بعد یک بسته fragment شده بدون اعمال هیچگونه محدودیتی از فیلتر عبور می‌کنند. با اضافه کردن کلمه کلیدی keep flags می‌توان بر روی این نوع بسته‌ها نیز کنترل داشت و سایر frameهای بسته را براساس اولین frame کنترل کرد:

```
pass in quick on x10 proto tcp from any to 20.20.20.0/24 port = 25 flags S keep state keep frags
```

در حالت عادی وقتی یک درخواست TCP برای یک سیستم می‌رسد که سرویس مربوط به آن وجود ندارد یک بسته RST در پاسخ برای سرویسگیر فرستاده می‌شود تا سرویسگیر از عدم وجود سرویس مربوطه بر روی سیستم مطلع شود. در IP Filter نیز می‌توان با فرستادن بسته RST برای پورتهای فیلتر شده سرویس مربوطه را از دید دیگران مخفی نگاه داشت:

```
block return-rst in log proto tcp from any to 20.20.20.0/24 port = 23
```

این روش فقط در مورد ارتباطات TCP می‌باشد. در مورد ارتباطات UDP و ICMP می‌توان با فرستادن بسته ICMP نوع Port Unreachable این کار را انجام داد:

```
block return-icmp(port-unr) in log quick on tun0 proto udp from any to 20.20.20.0/24 port = 111
```

برای اینکه بسته ICMP فرستاده شده با آدرس خود سرور فرستاده شود نه آدرس فایروال باید از return-icmp-as-dest(port-unr) استفاده کرد:

```
block return-icmp-as-dest(port-unr) in log quick on tun0 proto udp from any to 20.20.20.0/24 port = 111
```

نمونه‌ای از یک فایل پیکربندی نمونه فایروال IP Filter در زیر آمده است. این فایروال بر روی یک سرور نصب شده است که سرویسهای DNS، وب و پست الکترونیکی ارائه می‌دهد. آدرس شبکه محلی این سرور 213.29.6.0/24 است و سه دسته آدرس مختلف نیز از طریق مسیریابهای مختلف به این شبکه متصلند که دارای آدرسهای 192.200.9.0/24، 192.200.1.0/24 و 192.200.2.0/24 می‌باشند.

```
block in all
block out all
block return-rst in proto tcp from any to 213.29.6.0/24
block return-icmp(port-unr) in proto udp from any to 213.29.6.0/24
pass out quick proto tcp from 213.29.6.0/24 to any keep state
pass out quick proto udp from 213.29.6.0/24 to any keep state
pass out quick proto icmp from 213.29.6.0/24 to any keep state
pass in quick proto tcp from 192.200.9.0/24 to 213.29.6.0/24 port = 21 flags S keep state
pass in quick proto tcp from 192.200.1.0/24 to 213.29.6.0/24 port = 21 flags S keep state
pass in quick proto tcp from 192.200.2.0/24 to 213.29.6.0/24 port = 21 flags S keep state
pass in quick proto tcp from 213.29.6.0/24 to 213.29.6.0/24 port = 21 flags S keep state
pass in quick proto tcp from any to 213.29.6.0/24 port = 22 flags S keep state
pass in quick proto tcp from 192.200.9.0/24 to 213.29.6.0/24 port = 23 flags S keep state
pass in quick proto tcp from 192.200.1.0/24 to 213.29.6.0/24 port = 23 flags S keep state
pass in quick proto tcp from 192.200.2.0/24 to 213.29.6.0/24 port = 23 flags S keep state
pass in quick proto tcp from 213.29.6.0/24 to 213.29.6.0/24 port = 23 flags S keep state
pass in quick proto tcp from any to 213.29.6.0/24 port = 25 flags S keep state
pass in quick proto tcp from 192.200.9.0/24 to 213.29.6.0/24 port = 53 flags S keep state
pass in quick proto tcp from 192.200.1.0/24 to 213.29.6.0/24 port = 53 flags S keep state
pass in quick proto tcp from 192.200.2.0/24 to 213.29.6.0/24 port = 53 flags S keep state
pass in quick proto tcp from 213.29.6.0/24 to 213.29.6.0/24 port = 53 flags S keep state
pass in quick proto udp from 192.200.9.0/24 to 213.29.6.0/24 port = 53 keep state
pass in quick proto udp from 192.200.1.0/24 to 213.29.6.0/24 port = 53 keep state
pass in quick proto udp from 192.200.2.0/24 to 213.29.6.0/24 port = 53 keep state
pass in quick proto udp from 213.29.6.0/24 to 213.29.6.0/24 port = 53 keep state
pass in quick proto tcp from 192.200.9.0/24 to 213.29.6.0/24 port = 80 flags S keep state
pass in quick proto tcp from 192.200.1.0/24 to 213.29.6.0/24 port = 80 flags S keep state
pass in quick proto tcp from 192.200.2.0/24 to 213.29.6.0/24 port = 80 flags S keep state
pass in quick proto tcp from 213.29.6.0/24 to 213.29.6.0/24 port = 80 flags S keep state
pass in quick proto tcp from any to 213.29.6.0/24 port = 110 flags S keep state
pass in quick proto tcp from any to 213.29.6.0/24 port = 995 flags S keep state
```

Snort 7

Snort یک سیستم تهاجم‌یاب سبک مبتنی بر UNIX می‌باشد که برای استفاده در شبکه‌های کوچک و متوسط مناسب می‌باشد. سورس و مستندات این نرم‌افزار از طریق آدرس زیر قابل دسترسی می‌باشد:

http://www.snort.org/snort-files.htm

Snort می‌تواند بعنوان یک sniffer نیز عمل کرده و ترافیک شبکه را ثبت کند. یک سری از خصوصیات این نرم افزار از قرار زیر می‌باشد:

- § استفاده از کتابخانهٔ libpcap برای دریافت ترافیک شبکه
- § آنالیز کل ترافیک شبکه در مد promiscuous و تو لید هشدارهای امنیتی
- § ثبت ترافیک در فرم‌های مختلف:
  - باینری tcpdump
  - متنی و تفکیک شده براساس آدرس IP
  - ثبت کردن در فرمت XML در یک فایل یا روی شبکه
  - وارد کردن در پایگاه‌دادهٔ SQL
- § امکان مشاهدهٔ محتویات کامل بسته‌ها
- § امکان خواندن ترافیک از فایل‌های باینری به فرمت tcpdump
- § مکانیزم‌های مختلف برای تولید هشدارهای امنیتی:
  - ثبت کردن در یک فایل متنی به‌صورت خلاصه یا کامل
  - ثبت کردن از طریق سرویس syslog
  - فرستادن پیغام‌های "WinPopUp" به یک سیستم Windows
  - فرستادن هشدارها روی یک socket برای یک برنامه
  - ثبت کردن در فرمت XML در یک فایل یا روی شبکه
  - وارد کردن در پایگاه‌دادهٔ SQL به‌صورت خلاصه یا کامل
  - فرستادن SNMP Trap به یک سیستم مانیتورینگ
- § تشخیص حملات مختلف از جمله سرریز بافر، portscan، حملات CGI، SMB probe، OS fingerprinting، ...
- § قوانین انعطاف پذیر برای تشخیص حملات که دائماً به روز درمی‌آیند
- § ساختار ماجولار و قابلیت افزودن ماجول‌ها و pluginها

Snort می‌تواند در سه مود مختلف کار کند: Sniffer، Packet Logger و تهاجم‌یاب شبکه. در مود Sniffer ترافیک شبکه را دریافت کرده و در صفحه نمایش نشان می‌دهد. در مود Packet Logger اطلاعات مربوطه را ثبت می‌کند. در مود تهاجم‌یاب ترافیک شبکه را بررسی می‌کند، آنها را با قوانین تعریف شده توسط کاربر چک می‌کند و برحسب نتیجه در هر مورد اعمالی را که از قبل تعیین شده انجام می‌دهد.

1-7 مود Sniffer

این مود ساده‌ترین حالت کارکرد Snort می‌باشد. با اجرای دستور زیر سرآیند TCP/IP تمام بسته‌هایی که از شبکه عبور می‌کنند روی صفحهٔ نمایش نشان داده می‌شود:

```
snort -v
```

در صورت استفاده از گزینهٔ d- اطلاعات داخل بسته نیز نمایش داده می‌شود:

```
snort -vd
```

برای مشاهدهٔ اطلاعات بیشتر مانند اطلاعات لایهٔ Data Link از گزینهٔ e- استفاده می‌شود:

```
snort -dve
```

2-7 مود Packet Logger

برای ثبت اطلاعاتی که برروی صفحهٔ نمایش ظاهر می‌شوند باید از گزینهٔ l- استفاده کرد. با اضافه کردن این گزینه Snort به‌طور اتوماتیک وارد مود Packet Logger می‌شود. این گزینه دایرکتوری مقصد جهت ثبت فایلهای log را تعیین می‌کند. برای مثال:

```
snort -dev -l /root/log
```

در این حالت تمام بسته‌ها به همراه محتویات آنها و اطلاعات لایهٔ Data Link در دایرکتوری /root/log ذخیره می‌شوند. اطلاعات ترافیک بر حسب آدرس IP فرستنده یا گیرنده در این دایرکتوری قرار می‌گیرند؛ بدین معنی که در این دایرکتوری زیردایرکتوری‌هایی ساخته می‌شود که نام آنها یک آدرس IP می‌باشد و ترافیک هر آدرس در زیردایرکتوری مربوطه ذخیره می‌شود. با کمی دقت هنگام استفاده مشاهده می‌شود که در بعضی مواقع از آدرسهای محلی و در بعضی مواقع دیگر از آدرسهای خارجی برای نام این زیردایرکتوریها استفاده شده است. همین ممکن است بررسی فایل‌ها را دشوار کند. برای رفع مشکل می‌توان از گزینهٔ h- استفاده کرد. این گزینه آدرس شبکهٔ محلی را مشخص می‌کند. دستور زیر شبکهٔ محلی را 192.168.1.0/24 معرفی می‌کند که شامل آدرسهای 192.168.1.0 تا 192.168.1.255 می‌شود:

```
snort -dev -l /root/log -h 192.168.1.0/24
```

هنگام استفاده از این گزینه بسته‌ها برحسب آدرس خارجی (غیر 192.168.1) آنها در دایرکتوری log ذخیره می‌شوند. اگر هر دو آدرس بسته در شبکه داخلی بود بر حسب آدرسی که شمارهٔ پورت متناظرش بالاتر باشد، و درصورتی‌که شمارهٔ پورت‌ها نیز مساوی باشد بر حسب آدرس مبدأ ذخیره می‌شود.

در صورتی که ترافیک شبکه بالا باشد ممکن است نتوان آنها را در فرمت متنی ثبت کرد، چون تبدیل بسته‌ها به فرمت متنی وقت می‌گیرد و همین ممکن است باعث شود تعدادی از بسته‌ها از دست برود. می‌توان کل ترافیک را در یک فایل با فرمت باینری tcpdump ذخیره کرده بعداً آن را بررسی نمود. برای این منظور از گزینهٔ b- استفاده می‌شود:

```
snort -b -l /root/log
```

در این حالت کل ترافیک در یک فایل باینری ذخیره می‌شود، لذا لزومی به استفاده از گزینه‌های دیگر مانند h- نمی‌باشد. به همین علتی که ذکر شد نیازی به گزینه‌های d- و e- نیز نمی‌باشد. نام فایلی که اطلاعات ترافیک در آن ذخیره می‌شود snort.log به‌علاوهٔ یک رشته است که زمان ایجاد فایل را نشان می‌دهد. برای تعیین یک نام دیگر برای فایل از گزینهٔ L- استفاده می‌شود:

```
snort -b -l /root/log -L packet.log
```

پس از اینکه ترافیک شبکه در یک فایل باینری ذخیره شد می‌توان این فایل را توسط tcpdump، Ethereal، ntop، خود Snort و یا بسیاری از نرم‌افزارهای دیگر بررسی نمود. در Snort این کار با استفاده از گزینهٔ r- انجام می‌شود. از اطلاعات داخل فایل باینری می‌توان در هر سه مود Snort استفاده کرد، برای مثال می‌توان آنها را در مود Sniffer در صفحهٔ نمایش مشاهده کرد:

```
snort -dv -r packrt.log
```

برای ثبت ترافیک شبکه روشهای دیگری غیر از دو روش متنی و باینری نیز وجود دارد که تنها از طریق فایل پیکربندی Snort قابل دسترسی می‌باشند.

3-7 مود تهاجم‌یاب شبکه

در مود تهاجم‌یاب یک‌سری قوانین که توسط کاربر تعریف می‌شود برای بررسی ترافیک استفاده می‌شود و با بسته‌ها براساس این قوانین رفتار می‌شود. در این مود فقط بسته‌هایی که در قوانین مشخص شده اند ذخیره می‌شوند نه تمام بسته‌ها. برای استفاده از Snort در مود تهاجم‌یاب باید نام یک فایل که شامل این قوانین تهاجم‌یاب می‌باشد بعنوان یک پارامتر با گزینهٔ c- به آن معرفی شود:

```
snort -dev -l /root/log -h 192.168.1.0/24 -c snort.conf
```

استفاده از گزینهٔ l- در این مود ضرورتی ندارد. در صورتی که دایرکتوری log تعریف نشود Snort از /var/log/snort استفاده خواهد کرد. نکتهٔ قابل توجه این است که نشان دادن اطلاعات ترافیک در صفحهٔ نمایش می‌تواند سرعت عملکرد سیستم را به شدت کاهش دهد و درنتیجه باعث از دست دادن یک‌سری بسته‌ها شود، به همین دلیل برای بالا بردن سرعت باید گزینهٔ v- را حذف کرد. ثبت کردن اطلاعات لایهٔ Data Link نیز اغلب ضروری نیست، بنابراین گزینهٔ e- را نیز می‌توان حذف کرد. دستور زیر Snort را در مود تهاجم‌یاب اجرا می‌کند و بسته‌های مشخص شده را به‌صورت متنی در دایرکتوری /root/log ذخیره می‌کند.

```
snort -d -l /root/log -h 192.168.1.0/24 -c snort.conf
```

برای ذخیره خروجی Snort در مود تهاجم‌یاب راه‌های مختلفی وجود دارد. در حالت پیشفرض ترافیک شبکه به‌صورت متنی و هشدارها به‌صورت کامل ثبت می‌شوند. حالت‌های مختلفی برای ایجاد هشدارهای امنیتی وجود دارد که شش حالت آن از طریق گزینه‌های خط فرمان قابل دسترسی می‌باشند. سایر حالتها فقط از طریق فایل پیکربندی قابل دسترسی می‌باشند.

گزینهٔ A- چهار حالت مختلف برای تولید هشدارهای امنیتی را تعیین می‌کند:

-A fast	در این حالت فقط پیغام هشدار به‌همراه آدرس و شماره پورت‌های داخل سرآیند بسته در یک فایل ثبت می‌شود.
-A full	در این حالت پیغام هشدار به‌همراه سرآیند بسته‌ها در یک ساختار درختی دایرکتوری‌ها ثبت می‌شود.
-A unsock	در این حالت هشدارها روی یک socket فرستاده می‌شوند تا یک برنامهٔ دیگر بتواند آنها را دریافت کند.
-A none	در این حالت هیچ هشدار امنیتی تولید نمی‌شود.

در مود تهاجم‌یاب نیز می‌توان نحوهٔ ثبت کردن ترافیک را تغییر داد. می‌توان با استفاده از گزینهٔ b- بسته‌ها را در فرمت باینری ذخیره کرد. استفاده از گزینهٔ N- باعث می‌شود اطلاعات ترافیک اصلاً ثبت نشود.

با استفاده از گزینهٔ s- می‌توان هشدارها را به سرویس syslog فرستاد تا از آن طریق ثبت شوند. استفاده از گزینهٔ M- باعث می‌شود هشدارهای امنیتی با اجرای سرویسگیر SMB به یک سیستم Windows فرستاده شوند.

4-7 فیلترهای BPF

در همهٔ مودهای کارکرد Snort می‌توان با اعمال یک فیلتر بنام BPF عدهٔ خاصی از بسته‌ها را انتخاب کرده و آنها را برای پردازش‌های بعدی Snort مورد استفاده قرار دهید. فرمت و نحوهٔ استفاده از این فیلتر دقیقاً مانند tcpdump می‌باشد. مثلاً برای مشاهدهٔ بسته‌های ICMP موجود در ترافیک می‌توان از دستور زیر استفاده کرد:

```
snort -dev icmp
```

می‌توان این فیلتر را در یک فایل نوشت و آن فایل را با گزینهٔ F- به Snort معرفی کرد:

```
snort -dev -F bpf-rules-file
```

هر فیلتر از یک‌سری کلمهٔ کلیدی و مقادیر متناظر آنها تشکیل شده است. بیشتر فیلترهای پیچیده با استفاده از عبارات منطقی and، or و not و ترکیب این کلمات کلیدی ساخته می‌شوند. لیست این کلمات کلیدی در زیر آمده است:

کلمهٔ کلیدی	عبارت در صورتی درست است که:
dst host *host*	آدرس IP مقصد بسته *host* باشد، که می‌تواند یک آدرس یا یک نام باشد. این عبارت خلاصه شدهٔ عبارت زیر می‌باشد: ether proto \ip and host *host*
src host *host*	آدرس IP مبدأ بسته *host* باشد.
host *host*	آدرس IP مبدأ یا مقصد بسته *host* باشد.
ether dst *ehost*	آدرس Ethernet مقصد بسته *ehost* باشد، که می‌تواند یک شماره یا یک نام تعریف در /etc/ethers باشد.
ether src *ehost*	آدرس Ethernet مقصد بسته *ehost* باشد.
ether host *ehost*	آدرس Ethernet مقصد بسته *ehost* باشد.
gateway *host*	بسته *host* را به‌عنوان gateway استفاده می‌کند، یعنی مبدأ یا مقصد Ethernet بسته *host* باشد ولی نه مبدأ و نه مقصد IP بسته *host* باشد. *host* باید یک نام تعریف شده در /etc/hosts یا /etc/ethers باشد.
dst net *net*	آدرس شبکهٔ مقصد بسته *net* باشد. *net* می‌تواند یک شماره یا یک نام تعریف شده در /etc/networks باشد.
src net *net*	آدرس شبکهٔ مبدأ بسته *net* باشد.
net *net*	آدرس شبکهٔ مبدأ یا مقصد بسته *net* باشد.
net *net* mask *mask*	آدرس شبکهٔ بسته *net* باشد (با احتساب ماسک *mask*). هرکدام از کلمات src یا dst می‌توانند قبل از آن بیایند.
net *net*/*len*	آدرس شبکهٔ بسته *net* باشد (با احتساب ماسک *len* بیتی). هرکدام از کلمات src یا dst می‌توانند قبل از آن بیایند.
dst port *port*	بسته از نوع IP/TCP یا IP/UDP باشد و شمارهٔ پورت مقصد آن *port* باشد. *port* می‌تواند یک شماره یا یک نام تعریف شده در /etc/services باشد. اگر شمارهٔ پورت ارائه شده دو سرویس مختلف را در TCP و UDP مشخص کند هر دو تست می‌شوند.
src port *port*	بسته از نوع IP/TCP یا IP/UDP باشد و شمارهٔ پورت مبدأ آن *port* باشد.
port *port*	بسته از نوع IP/TCP یا IP/UDP باشد و شمارهٔ پورت مبدأ یا مقصد آن *port* باشد.
less *length*	طول بسته کمتر از *length* باشد.
greater *length*	طول بسته بیشتر از *length* باشد.

ip proto *protocol*	بسته از نوع IP باشد و پروتکل آن *protocol* باشد. *protocol* می‌تواند یکی از مقادیر icmp، igrp، nd، tcp یا udp را داشته باشد. از آنجایی که tcp، udp و tcp خود کلمهٔ کلیدی هستند باید قبل از آنها '\' قرار داد.
ether broadcast	بسته از نوع ethernet broadcast باشد. استفاده از ether در اینجا اختیاری است.
ip broadcast	بسته از نوع IP broadcast باشد.
ether multicast	بسته از نوع ethernet multicast باشد. استفاده از ether در اینجا اختیاری است.
ip multicast	بسته از نوع IP multicast باشد.
ether proto *protocol*	پروتکل Ethernet بسته protocol باشد. protocol می‌تواند ip، arp، rarp یا decnet باشد. تمام این کلمات جزءِ کلمات کلیدی می‌باشند و باید بعد از '\' قرار بگیرند.
ip, arp, rarp, decent	مخفف ether proto *p* می‌باشد که *p* می‌تواند هرکدام از این کلمات باشد.
tcp, udp, icmp	مخفف ip proto *p* می‌باشد که *p* می‌تواند هرکدام از این کلمات باشد.

5-7 فایل پیکربندی Snort

فایل پیکربندی Snort (snort.conf) شامل چند قسمت مختلف می‌باشد:

- تعریف متغیرها، includeها و سایر تنظیمات
- preprocessorها
- قوانین تهاجم‌یاب
- ماجول‌های خروجی

یک نمونه از یک فایل snort.conf در زیر آمده است:

```
#
# Taken and modified from "vision.conf", part of Max Vision's
# ArachNIDs work.  See /usr/doc/snort-1.6/README.snort-stuff for more
# information on how to use this file.

var INTERNAL 192.168.1.0/24
var EXTERNAL 63.87.101.0/24
var DNSSERVERS 63.87.101.90/32 63.87.101.92/32

preprocessor http_decode: 80 443 8080
preprocessor minfrag: 128
preprocessor portscan-ignorehosts: $DNSSERVERS
preprocessor portscan:      $EXTERNAL 3 5  /var/log/snort/portscan.log

# Ruleset, available (updated hourly) from:
#
#   http://dev.whitehats.com/ids/vision.conf
```

```
# Include the latest copy of Max Vision's ruleset
include /etc/snort/vision.conf

#
# Uncomment the next line if you wish to include the latest
# copy of the snort.org ruleset.  Be sure to download the latest
# one from http://www.snort.org/snort-files.htm#Rules
#
# include /etc/snort/06082k.rules

#
# If you wish to monitor multiple INTERNAL networks, you can include
# another variable that defines the additional network, then include
# the snort ruleset again.  Uncomment the two following lines.
#
# var INTERNAL 192.168.2.0/24
# include /etc/snort/vision.conf

# include other rules here if you wish.
```

همانطور که مشاهده می‌شود تعریف متغیرها با استفاده از کلمهٔ کلیدی var و خواندن یک فایل دیگر نیز با استفاده از include و به سادگی انجام می‌گیرد. توضیح کامل در بارهٔ اینها و نیز سایر تنظیمات (که به ندرت انجام می‌گیرد) در مستندات Snort وجود دارد.

1-5-7 preprocessorها

preprocessorها ماجول‌هایی هستند که پس از برداشتن بسته‌ها از روی شبکه و درست قبل از قوانین تهاجم‌یاب روی آنها اعمال می‌شوند و بسته‌ها را تست می‌کنند. prepeocessorها می‌توانند محتوای بسته‌ها را نیز تغییر دهند و حتی هشدار امنیتی تولید کنند. Snort چندین preprocessor همراه خود دارد که عبارتند از: portscan، portscan_ignorehost، frag2 (قبلاً defrag)، stream4 (قبلاً stream)، http_decode، unidecode، rpc_decode، telnet_decode، bo، spade و arpspoof.

کاربرد هرکدام از این preprocessorها در زیر آمده است:

portscan	انواع مختلف portscan را تشخیص می‌دهد، مانند TCP SYN، UDP Scan و TCP Stealth.
portscan_ignorehost	یک یا چند آدرس خاص را از portscan مستثنا می‌کند (مستثنا کردن سرورهای DNS باعث می‌شود حجم هشدارهای اشتباه تولید شده کاهش یابد).
frag2	عمل IP defragmentation را انجام می‌دهد و باعث می‌شود این نوع حملات شناسایی شوند.
stream4	جریان بسته‌های TCP را نگهداری می‌کند و می‌تواند موارد مشکوک را ثبت کند.

http_decode	تمام کاراکترهای XX% در آدرسهای URL را به کاراکترهای متناظر ASCII تبدیل می‌کند که بررسی آن را ساده‌تر می‌کند.
unidecode	مانند http_decode است ولی دربرابر رشته‌های Unicode عمکرد بهتری دارد.
rpc_decode	ترافیک RPC را به فرم استاندارد کدینگ چهاربیتی تبدیل می‌کند.
telnet_decode	ترافیک telnet و ftp را به فرمی که در آن محتویات بسته‌ها قابل خواندن می‌باشند تبدیل می‌کند.
bo	ترافیک Back Orifice را در شبکه شناسایی می‌کند.
spade	the Statistical Packet Anomaly Detection Engine
arpspoof	حملات arp را شناسایی می‌کند.

2-5-7 قوانین تهاجم‌یاب

مهمترین قسمت فایل پیکربندی Snort (snort.conf) معرفی قوانین تهاجم‌یاب آن است و قسمتی است که اغلب نیاز به تغییرات ندارد. این قوانین توسط عدهٔ خاصی نوشته می‌شود و دائماً به روز می‌آیند تا جدیدترین حملات را نیز شناسایی کنند. وظیفهٔ مدیر شبکه گرفتن و استفاده از جدیدترین نسخهٔ این قوانین می‌باشد. معمولاً این قوانین از دو منبع قابل دسترسی می‌باشند:

- سری قوانین Max Vision که هر یک ساعت به روز درمی‌آیند

http://dev.whitehats.com/ids/vision.conf

- سری قوانین Jim Forster

http://www.snort.org/snort-files.htm#Rules

3-5-7 ماجول‌های خروجی

Snort راه‌های مختلفی برای ثبت خروجی وجود دارد. یک‌سری از آنها از طریق خط فرمان و بقیه از طریق فایل snort.conf قابل استفاده می‌باشند. در Snort بیش از یک نوع خروجی قابل استفاده است که در این صورت هرکدام از ماجول‌ها به ترتیب اجرا می‌شوند. نحوهٔ استفاده از این ماجول‌ها به شکل زیر است:

output *name* : *options*

برای مثال:

output alert_syslog: LOG_AUTH LOG_ALERT

در این قسمت به بررسی اجمالی ماجول‌های خروجی Snort می‌پردازیم. اطلاعات بیشتر در مورد نحوهٔ استفاده از این ماجول‌ها و پارامترهای آنها در مستندات مربوط به Snort وجود دارد.

alert_syslog (هشدارها)

output alert_syslog: *facility ptiority options*

این ماجول مانند گزینهٔ s- هشدارها را به سرویس syslog می‌فرستد. این ماجول همچنین این اجازه را به کاربر می‌دهد که پارامترهای syslog را نیز برای ثبت کردن هشدارها تنظیم کند. حالت پیشفرض این پارامترها -LOG AUTH و LOG_ALERT به ترتیب برای facility و priority می‌باشد.

Options:

- LOG_CONS
- LOG_NDELAY
- LOG_PERROR
- LOG_PID

Facilities:

- LOG_AUTH
- LOG_AUTHPRIV
- LOG_DAEMON
- LOG_LOCAL0
- LOG_LOCAL1
- LOG_LOCAL2
- LOG_LOCAL3
- LOG_LOCAL4
- LOG_LOCAL5
- LOG_LOCAL6
- LOG_LOCAL7
- LOG_USER

Priorities:

- LOG_EMERG
- LOG_ALERT
- LOG_CRIT
- LOG_ERR
- LOG_WARNING
- LOG_NOTICE
- LOG_INFO
- LOG_DEBUG

alert_fast (هشدارها)

output alert_fast: *output_filename*

در این حالت هشدارها بصورت خلاصه و یک سطری در یک فایل ذخیره می‌شوند. نسبت به حالت full سریع‌تر است، چون نیازی به ذخیرهٔ کل اطلاعات سرآیند بسته نیست.

alert_full (هشدارها)

output alert_full: *output_filename*

در این حالت پیغام هشدار به‌همراه سرآیند بسته‌ها در یک ساختار درختی دایرکتوری‌ها ثبت می‌شود. در این حالت در ترافیک‌های بالا ممکن است سیستم به‌حدی کند شود که باعث از دست دادن تعدادی از بسته‌ها گردد.

alert_smb (هشدارها)

output alert_smb: *alert_workstation_filename*

این ماجول با استفاده از سرویسگیر SMB موجود بر روی سیستم یک پیغام برای یک یا چند سیستم Windows ارسال می‌کند. فایلی که نام NetBIOS این سیستم‌ها در آن قرار دارد به‌عنوان پارامتر به ماجول پاس داده می‌شود. از آنجا که استفاده از این ماجول یک برنامهٔ خارجی را اجرا می‌کند، ممکن است مشکلات امنیتی پیش بیاورد. به همین علت استفاده از آن توصیه نمی‌شود.

alert_unixsock (هشدارها)

output alert_unixsock

یک socket ایجاد کرده و هشدارهای امنیتی را روی آن برای یک برنامهٔ دیگر می‌فرستد. فعلاً در مرحلهٔ آزمایشی می‌باشد.

log_tcpdump (ترافیک)

output log_tcpdump: *output_filename*

مانند گزینهٔ b- خط فرمان ترافیک را در فرمت باینری tcpdump ذخیره می‌کند.

XML (هشدارها و ترافیک)

output xml: [log | alert], *parameter_list*

با استفاده از این ماجول می‌توان بسته‌ها را به فرمت SNML[9] در یک فایل بر روی شبکه ثبت کرد.

database (هشدارها و ترافیک)

output database: [log | alert], *database_type*, *parameter_list*

این ماجول اطلاعات ترافیک یا هشدارها را در یک پایگاه‌دادهٔ SQL ثبت می‌کند. پایگاه‌داده‌هایی که Snort می‌تواند با آنها کار کند MySQL، PostgreSQL، Oracle و پایگاه‌داده‌های سازگار با unixODBC می‌باشند.

[9] Simple Network Markup Language

SAINT یا Security Administraor's Integrated Network Tool ابزاری است که به منظور بررسی و آنالیز امنیت شبکه استفاده می‌شود. این ابزار اطلاعات مورد نیاز در مورد hostهای راه‌دور و شبکه را بر اساس سرویس‌هایی نظیر finger، NFS، NIS، FTP، TFTP، statd و ... جمع‌آوری می‌کند که در پایگاه‌داده مشخصی ذخیره می‌شود. مطالب جمع‌آوری شده در برگیرندهٔ اطلاعاتی در مورد سرویس‌های موجود، اشکالات و bugهای معروف، ضعف سیاست‌های قرار داده شده و ... می‌باشد. بعد از جمع‌آوری اطلاعات، آنالیز بر روی آنها صورت می‌گیرد که بر اساس آن گزارشی آماده می‌شود.

صفحهٔ نمایشی که SAINT به منظور نمایش اطلاعات استفاده می‌کند به فرم HTML است که برای دیدن این صفحات می‌توان از مرورگرهایی مانند Mosaic، Netscape و Lynx سود جست.

روش عمل SAINT به این صورت است که در ابتدای کار با استفاده از fping تشخیص می‌دهد که چه نودهایی در شبکه قرار دارند (درصورتی‌که ماشینی که بر روی آن SAINT نصب است در پشت فایروال قرار داشته باشد از tcp_scan استفاده می‌کند تا تشخیص دهد که چه host ای موجود است). بعد از این عمل لیست hostهای موجود جمع‌آوری شده و در فایلی ذخیره می‌شود که این لیست به موتور برنامه -که عمل بدست آوردن اطلاعات در مورد هر host را بر عهده دارد- داده می‌شود. موتور برنامه بعد از دریافت لیست hostها بررسی می‌کند که آیا host ای چک شده است یا نه که در صورت چک نشدن دنباله‌ای از test/probeها بر روی آن اعمال می‌شود که در نتیجه یک خروجی برای آن ایجاد می‌شود که می‌توان آنرا در محیط مرورگر دید.

وقتی که نرم‌افزار SAINT نصب شود فایل‌ها و دایرکتوری‌هایی را ایجاد می‌کند که در ادامه توضیح هریک می‌آید:

- *bin/: SAINT از برنامه‌هایی که در این دایرکتوری قرار دارد به منظور اعمال توابع بر روی داده‌ها استفاده می‌کند.
- *config/: فایل‌هایی که در این دایرکتوری قرار دارد تعیین می‌کند که SAINT بر چه اساسی اطلاعات را بگیرد.
- *html/: برنامه‌هایی که در این دایرکتوری قرار دارند فایل‌های perl و یا html هستند که در مرورگر نمایش داده می‌شود.
- *perl/: در این دایرکتدری ماجول‌های perl وجود دارد.
- results/<database name>: در این دایرکتوری بانک‌های اطلاعاتی وجود دارد که عبارتند از:
  - all-hosts: این فایل در برگیرندهٔ IP مربوط به host هایی است که قرار است بررسی شوند.
  - facts: این فایل شامل خروجی‌هایی است که توسط ابزارهای saint.* (که در دایرکتوری rules قرار دارند) ایجاد می‌شوند.
  - todo: این فایل شامل اعمالی است که قرار است بر روی host ها اجرا شود.
  - CVE: این فایل در برگیرندهٔ آسیب‌پذیری‌هایی است که می‌تواند امنیت شبکه را از بین ببرد.

- rules/*: فایل‌هایی که در این دایرکتوری قرار دارند توسط SAINT استفاده می‌شوند تا بر اساس آنها probe مورد نظر بر روی host ها اعمال شود.
- src/*: این دایرکتوری در برگیرندهٔ source code مربوط به SAINT می‌باشد.

1-8 فایل پیکربندی

فایل config/saint.cf مهمترین فایلی است که کاربران با آن کار می‌کنند. در این فایل کاربر مشخص می‌کند که SAINT به چه صورت عمل کند. مواردی که در این فایل مشخص می‌شود عبارتند از:

- Attack level
- Dangerous checks
- Which probes correspond to the attack level
- Custom attack level
- Password guessing
- The target file
- The what's and where's of current probe
- Timeouts
- Timeout signals
- Multitasking
- Proximity variablesTrusted or untrusted
- Target netmask
- Targeting exceptions
- Workarounds: DNS, ICMP
- Firewall variables

Attack level

اولین مطلب در فایل config مربوط است به attack level که سطح بررسی یک نود را تعیین می‌کند. مقادیری که این پارامتر می‌تواند بگیرد برابر است با:

0=light
1=normal
2=heavy
3=heavy+
4=top10
5=custom

مثال:

$attack_level=3;

Dangerous checks

این مطلب پارامتر دیگری است که در میزان attack مربوط به SAINT تاثیر می‌گذارد. این متغیر تعیین می‌کند که attack از نوع خطرناک صورت بگیرد یا نه.

0=no
1=yes

مثال:

$exterme=0;

در صورت فعال بودن این گزینه موارد زیر بررسی می‌شود:

Buffer overflow in IIS 5 for windows 2000
iPlanet Web Publisher buffer overflow
iPlanet HTTP method buffer overflow

Which probes correspond to the attack level

هر سطحی از attack که انتخاب شود تعیین می‌کند که از چه probeهایی استفاده شود (probeهای موجود در دایرکتوری rules قرار دارد).

مثال:

```
@light = (
                'dns.saint',
                'ostype.saint',
                'rpc.saint''
                'showmount.saint?'
              );

@normal = (
                  @light,
                  …
            );
```

Custom attack level

توسط این مطلب تعیین می‌شود در صورت استفاده از custom attack از چه probeهایی استفاده شود.

مثال:

```
$attack_level = 5;
@http = (
            'tcpscan.saint 80',
            'http.saint?'
          );
@custom_level = "http";
```

Password guessing

SAINT سعی می‌کند با استفاده از finger و یا rusers بر روی hostهای مختلف password کاربران را حدث بزند. تعداد دفعاتی که می‌توان password ای را حدث زد توسط این پارامتر تعیین می‌شود.

مثال:

$password_guesses = 2;

The target file

SAINT برای شروع کار خود لازم است که مقصد مورد نظر به نوعی تعیین شود، یک روش استفاده از یک فایل به عنوان ورودی می‌باشد که در فایل آدرس مقصدهای مورد نظر قرار دارد.

مثال:

$use_targer_file = 1; (0=no, 1=yes)
$target_file = "target_file";

Status file

SAINT نام هر probe ای را که انجام می‌دهد در فایلی قرار می‌دهد که توسط این پارامتر تعیین می‌شود.

مثال:

$status_file = "status_file";

Timeouts

هر probe ای که SAINT استفاده می‌کند بعد از مدت زمانی از کار می‌افتد. در این رابطه SAINT سه زمان برای کار خود در نظر گرفته است.

$long_timeout = …;
$med_timeout = …;
$short_timeout = …;

در مقابل هر کدام از پارامترها مدت زمان بر حسب ثانیه می‌آید. متغیر $timeout تعیین می‌کند که مدت زمان اجرای هر probe را چه مقدار باشد.

مثال:

$timeout = 1; (0=short, 1=med, 2=long)

درصورتی‌که برای هر probe مدت زمان متفاوتی در نظر گرفته شود لازم است که timeout هر کدام جداگانه تنظیم شود که این عمل توسط پارامتر $*_timeout$ صورت می‌گیرد.

مثال:

در بعضی موارد اگر hostهای مورد نظر به طور مناسب تعیین نشوند ممکن است که hostهای دیگری نیز در هجوم SAINT به منظور بررسی قرار گیرند. برای جلوگیری از این قضیه باید مقصد را به طور دقیق مشخص کرد.

مثال:

$only_attack_these = "podunk.edu, 192.9.9";

$dont_attack_these = "gov, mil";

Workarounds: DNS, ICMP

برای آنکه SAINT بتواند آدرس‌ها را resolve کند باید استفاده از DNS را در آن فعال کرد.

مثال:

$dont_use_nslookup = 0; (0=use nslookup, 1=don't use nslookup)

SAINT قبل از شروع به تست یک host ابتدا آنرا ping می‌کند تا ببیند که host موجود است یا نه. درصورتی‌که خواسته شود که این عمل صورت نگیرد باید در فایل config این قضیه مشخص شود.

مثال:

$dont_use_ping = 0; (0=use ping, 1=don't use ping)

Firewall variables

درصورتی‌که SAINT از پشت فایروال به بررسی شبکه بپردازد لازم است تا این مساله برای آن مشخص شود.

مثال:

$firewall_flag = 0; (No firewall environment (0) or expect a firewall (1))

در رابطه با فایروال پارامترهای دیگری نیز وجود دارد که به منظور بررسی استفاده می‌شود. fw_timeout که حداکثر زمانی که connection قبل از قطع شدن می‌تواند وجود داشته باشد را تعیین می‌کند، fw_loadlimit برای نشان دادن حداکثر ارتباطات همزمان استفاده می‌شود و fw_tcp_scan که تقریباً برابر است با fw_loadlimit(10000*fw_timeout) به جای $tcpscan_timout می‌تواند استفاده شود.

2-8 خط فرمان

از خط فرمان زمانی استفاده می‌شود که به مرورگر دسترسی وجود نداشته باشد. در این حالت برای اجرای دستورات می‌توان scriptهایی نوشت و آنها را در crontab قرار داد. فرمت دستور saint به صورت زیر می‌باشد:

saint [options] [target1] [target2] …

منظور از target، hostهایی است که قرار است بررسی شود. target می‌تواند به دو صورت آدرس IP باشد و یا به صورت فایلی که شامل لیستی از IPها باشد.

در ادامه لیست پارامترهای مربوطه می‌آید:

-a level

برای تعیین سطح attack استفاده می‌شود.

0=light, 1=normal, 2=heavy, 3=heavy+, 4=top10, 5=custom

متغیر مورد استفاده: $attack_level

-A proximity

به منظور تعیین proximity descent استفاده می‌شود.

متغیر مورد استفاده: $proximity_descent

-c 'name=value; name=value…'

برای تغییر مقدار پارامترهای saint استفاده می‌شود. این دستور دارای متغیر نمی‌باشد.

-C custom level

توسط این دستور می‌توان سطح حملهٔ مورد نظر را ایجاد کرد.

متغیر مورد استفاده: $custom_level

-d directory

برای تعیین اسم database ای که saint اطلاعات خود را از آنجا بخواند و در آنجا بنویسد.

متغیر مورد استفاده: $saint_data

-f

به منظور فعال کردن قابلیت آنالیز firewall.

متغیر مورد استفاده: $firewall_flag

-F filename

درصورتی‌که خواسته شده باشد که hostهای مشخصی مورد بررسی قرار گیرد، می‌توان آدرس این hostها را در فایلی قرار داد. توسط این دستور saint آدرس‌های مورد نظر را از فایل تعیین شده برمی‌دارد.

متغیر مورد استفاده: $target_file

-g guesses

توسط این دستور تعداد دفعاتی که saint سعی می‌کند تا با حدث زدن کلمهٔ عبور account ای را پیدا کند مشخص می‌گردد.

متغیر مورد استفاده: $password_guesses

-h "host1 host2 …"

با این دستور آدرس IPهایی که اجازهٔ کنترل saint را به صورت راه‌دور دارند تعیین می‌شود.

متغیر مورد استفاده: $allow_hosts

-i

از اطلاعات جمع‌آوری شده تا به حال صرف‌نظر می‌کند.

-k

برای kill کردن saint استفاده می‌شود.

-l proximity

حداکثر سطح proximity را تعیین می‌کند.

متغیر مورد استفاده: $max_proximity_level

-m threads

تعداد تست‌ها و حملاتی که به صورت همزمان بر hostها اعمال می‌شود را مشخص می‌کند.

متغیر مورد استفاده: $maximum_threads

-n netmask

درصورتی‌که یک شبکه مورد بررسی قرار گیرد آدرس شبکه به عنوان پارامتر این دستور می‌آید.

متغیر مورد استفاده: $target_netmask

-o list

فقط hostها، domainها و شبکه‌هایی که در list آمده بررسی می‌شوند.

متغیر مورد استفاده: $only_attack_these

-O list

hostها، domainها و شبکه‌هایی که در list آمده مورد بررسی قرار نمی‌گیرد.

متغیر مورد استفاده: $dont_attack_these

-p port

پورت‌های tcp که به آنها گوش داده می‌شود.

متغیر مورد استفاده: $server_port

-q

خارج شدن بدون نمایش اطلاعات.

-r

حالت راه‌دور.

متغیر مورد استفاده: $remote_mode

-s

به منظور فعال کردن subnet expansion استفاده می‌شود.

متغیر مورد استفاده: $attack_proximate_subnets

-S status_file

برای مشخص کردن فایل status استفاده می‌شود.

متغیر مورد استفاده: $status_file

-t level

مدت زمان timeout را تعیین می‌کند.

0=short, 1=medium , 2=long

متغیر مورد استفاده: $timeout

-u

اجرا از روی یک untrusted host.

متغیر مورد استفاده: $untrusted_host=1

-U

اجرا از روی یک trusted host.

متغیر مورد استفاده: $untrusted_host=0

-v

به منظور فعال کردن debugging بر روی خروجی است.

متغیر مورد استفاده: $debug

-V

برای نمایش version و اتمام استفاده می‌شود.

-x

برای فعال کردن انجام تست‌های خطرناک بر روی hostها استفاده می‌شود.

متغیر مورد استفاده: $extreme=1

-X

برای غیرفعال کردن تست‌های خطرناک به‌کار می‌رود.

متغیر مورد استفاده: $extreme=0

-z

-Z

3-8 فرمت بانک‌اطلاعاتی

در SAINT چهار پایگاه‌داده مورد استفاده قرار می‌گیرد که عبارتند از:

- facts : دربر گیرندهٔ اطلاعات گرفته شده است.
- all-hosts : تمام host هایی که saint توانسته است که ببیند در این فایل قرار می‌گیرند.
- todo : اعمالی که بر روی host ها انجام شده است در این فایل مشخص می‌شود.
- CVE : اطلاعات مربوط به CVE و Top-10 در این فایل نگهداری می‌شود.

فرمت پایگاه‌داده‌ها به صورت text است که هر سطر دربر گیرندهٔ اطلاعات مشخصی است. بخش‌های مختلف اطلاعات هر سطر توسط | از یکدیگر جدا می‌شوند.

1-3-8 بانک‌اطلاعاتی facts

در این پایگاه‌داده‌ها آسیب‌پذیری‌های موجود، سرویس‌های پیشنهادی و سایر اطلاعات کشف شده قرار می‌گیرد. اطاعات هر سطر شامل موارد زیر می‌باشد:

- Target
- Service
- Status
- Severity
- Trusted
- Trustee
- Canonical service output
- Text

Target مشخص کنندهٔ IP مربوط به host ای است که SAINT عمل خود را بر روی آن انجام داده است.Service فیلدی است که مشخص می‌کند که چه تستی بر روی این host صورت گرفته است.Status وضعیت تست انجام شده را مشخص می‌کند.

a: available
u: unavailable (e.g. timeout)
n: network (e.g. network or broadcast address)
b: bad (e.g. unable to resolve)
x: look into further?

Severity مشخص می‌کند که آسیب‌پذیری موجود تا چه اندازه خطرناک است. سطح این خطر توسط یک کد نمایش داده می‌شود.

Critical Problems (Red)

rs **:** administrator or root shell access
us **:** user shell access
ns **:** unprivileged (nobody) shell access
ur **:** user file read access
uw **:** user file write access
nr **:** unprivileged file read access
nw **:** unprivileged file write access
ht **:** evidence of a hacker track
bo **:** buffer overflow
nfs **:** access to NFS filesystems
dos **:** denial of service

Areas of Concern (Yellow)

yus **:** user shell through X
yi **:** information gathering
ype **:** privilege elevation

Potential Problems (Brown)

zcio**:** check it out for possible vulnerabilies
zwoi**:** do you want this acccessible on the Internet

Others

g**:** Services (green)
i**:** Information

Trusted و Trustee نشان‌دهندهٔ hostهایی است که تحت عنوان trusted و trustee شناخته می‌شوند. hostهایی که به صورت trusted معرفی می‌شوند می‌توانند به قسمت‌هایی که به صورت trustee هستند دسترسی داشته باشند. هر فیلد trusted و trustee توسط @ به دو قسمت تقسیم می‌شود که سمت چپ @ نشان دهندهٔ کاربر و یا object است و سمت راست نشان دهندهٔ host می‌باشد. برای مثال اگر یک فیلد trustee به صورت /home@target.com باشد مشخص می‌کند که /home در host ای با آدرس target.com به صورت Trustee است و قابل دسترسی از طرف کاربران trustred می‌باشد. در مورد فیلد trusted همین تعریف استفاده می‌شود. Canonical service output در مواردی که یک آسیب‌پذیری پیدا شده باشد توضیحی در مورد آن می‌باشد. Text متنی است که در گزارش نهایی دیده می‌شود.

2-3-8 **بانک‌اطلاعاتی** all-hosts

SAINT برای شروع کار خود ابتدا hostهایی را که می‌بیند را مشخص می‌کند و لیست آنها را در فایلی قرار می‌دهد. هر سطر این فایل شامل موارد زیر است:

- Hostname
- IP address
- Proximity level

- Attack level
- Subnet expansion
- Time

3-3-8 **بانک‌اطلاعاتی** todo

در این فایل اعمالی که SAINT برای بررسی نودها انجام داده است ذخیره شده است. هر سطر این فایل دربر گیرندهٔ موارد زیر می‌باشد:

- Hostname: The hostname of the targeted host
- Probe name: The name of the probe which was run against the host
- Arguments: The arguments with which the probe was

4-3-8 **بانک‌اطلاعاتی** CVE

درصورتی‌که آسیبی پیدا شود که در لیست SANS Top 10 Internet Security Threads و یا CVE باشد را مشخص می‌کند. هر سطر این فایل شامل موارد زیر می‌باشد:

- Top 10 flag: Whether or not the vulnerability is on the Top 10 list ("yes" or "no")
- CVE name(s): The CVE name or names corresponding to the vulnerability, if any
- Vulnerability Text: Corresponds to the text field in the facts database

4-8 آنالیز خروجی

در گزارش و نتیجهٔ نهایی سه دسته‌بندی وجود دارد. اطلاعات موجود در این دسته‌ها می‌تواند مشترک باشد. تفاوتی که در این بین وجود دارد تاکید هر دسته‌بندی بر روی یک مساله می‌باشد. این سه دسته‌بندی عبارتند از:

- Vulnerablities
- Host information
- Trust

در دسته‌بندی اول مشخص می‌شود که در کجاها ضعف وجود دارد و کجاها در معرض آسیب‌پذیری قرار دارد. دسته‌بندی دوم یا Host information دارای اهمیت زیادی است و نشان می‌دهد که server ها در کجای شبکه قرار دارند، host های مهم شبکه کدام‌ها هستند و همچنین شبکه را به subnet هایی می‌شکند. در دسته‌بندی سوم مشخص می‌شود که رابطهٔ بین host های trusted و trustee به چه صورت است.

در کنار هر اطلاعات دایره‌ای وجود دارد که وضعیت آن اطلاعات را مشخص می‌کند. دایره‌ای که در کنار اطلاعات دستهٔ vulnerability قرار دارد اهمیت و مقدار خطری که در رابطه با آسیب موجود وجود دارد را تعیین می‌کند.

دایره‌ای که در کنار اطلاعات دستهٔ host قرار دارد نشان‌دهندهٔ میزان حد آسیب‌پذیری host مورد نظر است. دایره‌ها دارای رنگ‌های مشخصی می‌باشند که عبارتند از:

- Critial problem (قرمز): سرویس‌هایی که مورد هجوم و آسیب قرار دارند.
- Areas of concern (زرد): سرویس‌هایی که بطور مستقیم و یا غیرمستقیم در معرض آسیب قرار دارند، آسیب‌هایی نظیر کشف کلمهٔ عبور و یا اطلاعات دیگر.
- Potential problems (قهوه‌ای): سرویس‌هایی که می‌تواند آسیب‌پذیر باشند و یا نباشند که این مساله بستگی به پیکربندی صورت گرفته در مورد آن سرویس دارد.
- Services (سبز): سرویس‌هایی که در معرض خطر نیستند.
- Other information (سیاه): سرویسی فعال نیست و یا آنکه اطلاعاتی پیدا نشده است.

علاوه بر این دایره‌ها در بعضی جاها پیکان‌هایی وجود دارد که برای نشان دادن آسیب‌هایی از نوع Top-10 استفاده می‌شود.

در ادامه دسته‌بندی‌های موجود در مورد تفکیک اطلاعات توضیح داده می‌شود.

Vulnerabilities

سه روش برای دیدن اطلاعات مربوط به Vulnerabilities وجود دارد:

- Approximate Danger Level: تمام تست‌هایی که بر روی host ها صورت می‌گیرد سطحی از خطر موجود را مشخص می‌کند. در این روش host ها بر اساس میزان خطری که آنها را تهدید می‌کنند مرتب می‌شوند.
- Type of Vulnerability: در این مدل host ها بر اساس نوع خطرها دسته‌بندی می‌شوند. در این مدل لیست خطرهای و آسیب‌های موجود می‌آید و در ادامهٔ هر کدام host هایی که در معرض این خطر هستند مشخص می‌شوند.
- Vulnerability Count: در این روش host ها بر اساس اینکه کدام یک در معرض آسیب‌های بیشتری هستند دسته‌بندی می‌شوند.

Host Information

در این دسته‌بندی اطلاعات هر host به صورت کامل مشخص می‌شود. روش‌های متفاوتی برای جداسازی hostها از یکدیگر در این روش وجود دارد که در ادامه می‌آید:

- Class of Service در این روش host بر اساس سرویس‌هایی که بر روی آنها فعال است از یکدیگر تفکیک می‌شوند. سرویس‌هایی نظیر FTP، WWW و ...
- System Type در این روش host ها بر اساس سخت‌افزاری که استفاده می‌کنند از یکدیگر جدا می‌شوند.
- Internet Domain در این روش host بر اساس domain هایی که دارند دسته‌بندی می‌شوند.
- Subnet
- Hostname

Trust

این دسته‌بندی به این منظور انجام می‌گیرد تا مشخص شود که hostهای با اهمیت در شبکه کدام‌ها هستند. مرتب‌سازی hostها در این روش بر اساس میزان در دسترس بودن صورت می‌گیرد.