تکنیک ها و روش هایی پیشرفته در کشف و تشخیص[1] میزبان

تکنیک هایی جهت تعیین اعتبار در ارتباط یک میزبان

پیشگفتار

در مباحث شبکه، مفاهیم و روش هایی وجود دارند که به ندرت بین افراد مبادله می شوند و همگان سعی در مخفی نگاه داشتن آنها دارند. از جمله این روش ها، مفاهیم پویش یا scanning و روش های تشخیص میزبان، پورت های باز، سرویس های در حال اجرا و ... می باشد (همان طور که در scanner ها می بینیم که اکثرا closed-source هستند). به هر حال، در این مقاله سعی بر انتقال تجربیات و تحقیقاتی که در این زمینه انجام دادم کرده ام و در آینده به صورت مفصل تری حول این مفاهیم، بحث خواهیم کرد.

چکیده

مهندسان امنیت تلاش بسیاری برای بلاک و فیلتر کردن بسته های غیرمتعارف در یک محیط اتصالی شبکه میکنند. عملیات پیشرفته نگاشت (mapping) میزبان، بسیاری از IDS ها، فیلترها و مسیریاب ها را دور زده و الزاما یک نفوذگر را قادر می سازند که میزبان های ناشناخته و دارای دیوارآتش قبلی را map و discover کنند (کشف کنند).

مقدمه

این مقاله سعی خواهد کرد که تکنیک های مورد استفاده جهت کشف (discover) میزبان های دارای دیوار آتش[2] و فیلترشده[3] که به پاسخ های استاندارد PING جواب نمی دهند را توصیف خواهد کرد. در مقاله های قبلی و همچنین فاروم سایت توضیحات بسیاری حول پروتکل های اصلی اینترنت (از جمله TCP، IP، UDP و ICMP) داده شد و لذا در این مقاله از ذکر آنها خودداری کرده و فرض می کنیم که خواننده این مقاله اطلاعات و آگاهی مناسبی از پروتکل های مهم اینترنتی (که در بالا ذکر شد) دارد. بسیاری از دیگر پروتکل ها مورد بحث قرار نخواهند گرفت، اما تکنیک های توصیف و تشریح شده در این مقاله را می توان در بسیاری از پروتکل ها اعمال کرد.

روش های تشخیص میزبان

امروزه، تعداد میزبان های (هاست) فیلترشده و دارای دیوار آتش در اینترنت در حال افزایش می باشد. میزبان هایی که دارای دیوارآتش بوده و بد پیکربندی شده اند، اغلب بسته های پاسخی را که اتصال شبکه ای (یا اینترنتی) آنها را تعیین می کند بلاک می کنند. یک مثال اولیه از این وضعیت، ابزار استاندارد PING (یا Packet Internet Groper) می باشد. PING یک جوابِ ICMP نوع ۳[4] (echo request) را به یک میزبان دلخواه ارسال کرده تا وضعیت اتصال آنلاین او را بررسی و تست کند. بهرحال، چون تعداد بسیاری از این سرورها، انواع بسیاری از انواع کد ICMP (ICMP code types) را بلاک می کنند، لذا جواب سرور معمولا بلاک یا drop شده و بنابراین به دست ما نخواهد رسید (تحویل یا delivery با موفقیت

---

[1] detect
[2] Firewalled host
[3] Heavily filtered
[4] ICMP type 3

انجام نخواهد شد). متاسفانه، در این زمان، یک کلاینت ممکن است گمان کند که شبکه یا میزبان down بوده یا بسختی با دیوارآتش محافظت می شود.

اما واقعا، چگونه شخص عمدا می تواند وضعیت online یک میزبان را تشخیص دهد؟

درک راه هایی که می توانند بر سطوح مشخصی از مجموعه قوانین دیوارآتش فائق آیند، سرانجام به یک کلاینت اجازه می دهد تا تعیین کند آیا یک میزبان به شبکه متصل[5] می باشد و/یا در پشت یک محیط فیلتر شده[6] قرار دارد. این تکنیک تحت عنوان تشخیص میزبان یا Host Detection شناخته می شود.

تشخیص میزبان مشابه پویش یا scanning به طرق مختلف می باشد، اگرچه تشخیص میزبان توجهی فقدان بسته ها به پورت ها یا تغییرات وابسته به هدرهای پروتکل ها، یعنی تنظیم بسته های جواب فلگ دار[7]، ندارد، بلکه هر گونه علامت ارسالی مبنی بر پاسخ از طرف میزبان ریموت را بررسی می کند. در این جنبه، تشخیص میزبان حالتی از پویش PING می باشد که هر فرم و گونه ای از پاسخ و عکس العمل (جواب) را تشخیص می دهد و به این ترتیب دلیل واضحی بر وضعیت ارتباطی سرور خواهد بود.

این مقاله دو تکنیک پهناور در تشخیص میزبان 'PING sweep' را آنالیز می کند که می توان آنها را در یک محیط شبکه ای (یا اینترنتی) برای نگاشت پیشرفته میزبان (host mapping) مورد استفاده قرار داد.

- استخراج جواب های معتبر پروتکل
- تولید جواب های غیرمعتبر در طرف سرور

اولین روش شامل، استخراج جواب های معتبر از پروتکل های مورد پشتیبانی روی یک میزبان می باشد. هر درخواست معتبر ارسالی از یک کلاینت به یک سرور از طریق TCP/IP/UDP/ICMP که جوابی (پاسخ) را بخواهد، به زبان یک درخواست پاسخ داده شده، در این دسته جای می گیرد. چنین روش هایی شامل موارد زیر می باشند:

- **UDP Echo**
- **TCP Echo**
- **UDP Closed Ports**
- **TCP ACK**
- **TCP SYN**
- **TCP SYN|ACK**
- **TCP FIN**
- **TCP NULL FLAGS**
- **TCP XMAS**
- **ICMP Echo Request (Type 8)**
- **ICMP Broadcast**
- **ICMP Router Solicitation (Type 10)**
- **ICMP Timestamp Request (Type 13)**
- **ICMP Information Request (Type 15)**
- **ICMP Address Mask Request (Type 17)**

خلاف این جواب های مطیع بر دستورهای RFC[8]، روش هایی وجود دارند که جواب های غیرمعتبری را از میزبان هدف تولید می کنند تا به این ترتیب حضور پنهانی و نامحسوس آن میزبان را تعیین کرده و تشخیص دهند. البته دریافت یک

---

[5] اصطلاحا به این میزبان های، **Network-Connected** می گوییم.

[6] Filtered Environment

[7] flagged packet replies

[8] به این جواب ها، جواب های RFC-compliant می گوییم.

جواب از هر یک از این روش ها به ما اجازه خواهد داد تا آگاهانه و دانسته، آنلاین بودن و/یا دارای دیوارآتش بودن یک میزبان را تشخیص دهیم. این متدها شامل موارد زیر می باشند:

- قطعه سازی بسته timeout شده (Timedout Packet Fragmentation)
- طول غیرمعتبر برای هدر IP (Invalid IP Header Length)
- مقادیر غیرمعتبر برای فیلد IP (Invalid IP Field Values)

استخراج درخواست های معتبر از پروتکل

اولین دسته صریح و قطعی در تشخیص میزبان، به فرم استخراج گزارش های معتبر از پروتکل جای می گیرند. چندین متد از این قبیل وجود دارند که شامل استفاده از درخواست های معتبر بسته می شوند.

- **Echo port method**
- **UDP method**
- **TCP FLAG method**
- **ICMP request method**

تمام دسته های بالا، روش های ممکنی هستند که به هر کلاینت دلخواه اجازه می دهند یک بسته جواب برگشت داده شده[9] را به منظور تعیین وضعیت اتصال درخواست کنند. به این ترتیب، بسته های برگشت داده شده و منتقل شده، پاسخ های معتبر پروتکلی هستند و لذا از تولید جواب های غیرمعتبر متمایز می باشند، چرا که هر درخواست، از پروتکل های TCP/IP/UDP/ICMP، بدرستی و بدون تغییر دادن یا گستتن هر یک از فیلدهای موجود استفاده می کنند.

روش ECHO port (پورت Echo)

البته دوران طلایی استفاده از این تکنیک به سر آمده، چرا که این تکنیک قدیمی و منسوخ بوده و همگان در همه جا حول آن مطالبی نوشته اند. اما به هر حال، این تکنیک برای تعیین پاسخگویی یک میزبان در سطحی پایه ای استفاده می شد و هنوز می توان آنرا در میزبان های UNIX که بدرستی پیکربندی نشده اند یا دارای مشکلات خاصی هستند استفاده کرد. اغلب اوقات، یک مدیر امنیتی باهوش، ترافیک را به پورت TCP 7 و UDP 7 بلاک کرده یا اجرای این سرویس را از inetd غیر فعال می کند.

پورت TCP/Echo

این روش ساده، از یک دست تکانی سه مرحله ای استاندارد استفاده می کند که به برقراری یک ارتباط با پورت echo (TCP 7) می انجامد. اگر یک ارتباط (تابع connect())برقرار باشد، آنگاه فرض خواهد شد که میزبان آنلاین بوده و بنابراین توالی تشخیص میزبان در این سطح بسیار ابتدایی بوقوع می پیوندد. به این ترتیب، چون دست تکانی سه مرحله ای با بالقوه باز بودن پورت echo همراه بوده و حتی امکان فایروال بودن میزبان نیز وجود دارد، لذا این روش بسیار محدود و مشکل دار می باشد.

اگرچه بسیاری از توزیع های یونیکس/لینوکس به صورت پیش فرض پورت echo را غیرفعال ساخته اند، اما هنوز هم بسیاری از سیستم ها از آنها استفاده می کنند. اهداف اشکال زدایی که ابتدای به ساکن این سرور برای جمع آوری آنها

[9] returned packet reply

تنظیم شده بود، بوسیله مشکلات امنیتی سنگین شده و در نتیجه باعث افزایش ترافیک روی یک کلاینت متصل می گشتند (که در نتیجه پهنای باند و کارآیی پردازش سیستم را کاهش می دهد).

چون دست تکانی سه مرحله ای با یک بسته فلگ SYN اولیه آغاز گشته و یک SYN/ACK را در جواب دریافت می کند، لذا یک کلاینت احتیاجی ندارد تا الگو یا پارادایم دست تکانی را به منظور تعیین پاسخ گویی میزبان ها ادامه دهد. نه تنها امکان لاگ شدن آن وجود دارد، بلکه شانس و امکان بسیاری در اخطار داده شدن آن و متوالیا بلاک شدن آن توسط میزبان دلخواه وجود دارد. به این ترتیب، پیکربندی ساده یا اشتباه از یک دیوارآتش می تواند اجازه عبور به بسته های دارای فلگ SYN را بدهد. به این دلیل که روش TCP echo port بایستی تنها به عنوان یک رفت و امد نهایی مورد استفاده قرار گیرد – و حتی آنوقت هم ایده عاقلانه ای نخواهد بود، مگر اینکه هدف شما، راه انداختن بسیاری از فرم های IDS و زنگ های احتیاطی مدیران سیستم ها باشد! مثالی از کاربرد این روش در یک محیط شبکه ای، می تواند از طریق telnet انجام گردد.

```
cefix@term:~ $ telnet XXX.XXX.XXX.XXX 7
Trying XXX.XXX.XXX.XXX...
Connected to XXX.XXX.XXX.XXX
Escape character is '^]'.
Hello.
Hello.
```

همان طور که در بالا نشان داده شده است، میزبان ریموت به 'Hello' اولیه شما با جواب 'Hello' خودش پاسخ می دهد و در این حالت، بدیهی است که سرور، واکنش انجام داده است (اصطلاحا می گوییم، سرور Responsive یا پاسخگو می باشد). ایجاد یک پویشگر برای ارسال هر داده در هم ریخته ای به پورت ضروری نیست، تمام چیزی که مورد نیاز است، یک ارتباط در حال برقرار می باشد!

توجه: روش هایی دیگری را که اجرای این سرویس را روی یک میزبان ریموت تعیین می کنند و از دست تکانی سه مرحله ای TCP اجتناب می ورزند، می توان برای چنین اهدافی جهت از بین بردن و مغلوب ساختن logger های بسته ها (logger ها، سرویس ها، پروسه ها یا ابزاری هستند که وظیفه لاگ کردن یا ثبت رخداد را بر عهده دارند و لاگرهای بسته نیز وظیفه ثبت عبور و مرور بسته ها و خطاهای موجود در انتقال، تصدیق، دریافت (و غیره) بسته ها را بر عهده دارند) بکار برد (در آینده تکنیک های پیشرفته تری حول پویش پورت را مورد بررسی قرار می دهیم).

پورت UDP/Echo

مشابه روش TCP echo port، پورت UDP 7، به دیتاگرام یک کلاینت با دیتاگرام UDP خود جواب می دهد. چون بلوک بسته ای که در ابتدا ارسال شده بود، با یک جواب (پاسخ) از میزبان راه دور جواب داده شده است، ما می فهمیم که میزبان زنده بوده یا اصطلاحا Alive یا Active می باشد. با استفاده از hping [10] به عنوان تولید کننده بسته، ما موارد زیر را ارسال می کنیم:

```
cefix@term:~ # hping XXX.XXX.XXX.XXX -c 1 -2 -p 7
eth0 default routing interface selected (according to /proc)
HPING XXX.XXX.XXX.XXX (eth0 XXX.XXX.XXX.XXX): udp mode set, 28 headers + 0
data bytes
50 bytes from XXX.XXX.XXX.XXX: seq=0 ttl=64 id=1255 rtt=0.9 ms
```

روش UDP

این تکنیک، پروتکل User Datagram Protocol و پاسخ گویی های شناخته شده اش به پورت های بسته را درگیر کار می کند. این بسته مخفی جواب از متد UDP port scan گرفته شده است. منطق درگیر با کار، با ارسال یک

[10] به آدرس http://www.kyuzz.org/antirez/hping2.html مراجعه کنید.

دیتاگرام UDP به یک پورت بسته یا NON-LISTENING نمایان خواهد شد، به این ترتیب که میزبان دلخواه بایستی با یک پیام خطای ICMP_PORT_UNREACH به ما پاسخ دهد (مبنی بر اینکه پورت قابل دسترسی نیست). چون این میزبان چنین پاسخی را برگشت می دهد، آنگاه می توانیم وضعیت اتصال آنرا تعیین و مشخص گردانیم – و بنابراین خواهیم فهمید که این میزبان زنده می باشد. چرخه ای که در این روش استفاده می شود به صورت زیر می باشد:

- **client -> UDP (to closed port)**
- **server -> ICMP_PORT_UNREACH**

```
cefix@term:~ # hping XXX.XXX.XXX.XXX -c 1 -2 -p 65
eth0 default routing interface selected (according to /proc)
HPING XXX.XXX.XXX.XXX (eth0 XXX.XXX.XXX.XXX): udp mode set, 28 headers + 0
data bytes
ICMP Port Unreachable from XXX.XXX.XXX.XXX  (XXX.XXX.XXX.XXX)
```

پیش بینی یک پورت بسته بسیار ساده است. سعی کنید یک شماره پورت بالا (بزرگتر از ۱۰۲۴ و کوچکتر از ۶۵۵۳۵) انتخاب کنید. البته اگر یک جواب به یک دیتاگرام UDP به کلاینت پس فرستاده نشود، گواه بر این خواهد بود که این بسته توسط هسته drop می شود. به این ترتیب در drop شدن آن، یا پورت مقصد بایستی *باز* باشد (بنابود پورت انتخابی ما بسته باشد) یا اینکه یک سیستم فیلترکننده بسته را بلاک کرده است. به صورت طبیعی، اگر این سناریو رخ داد، پورت دیگری را برای بررسی پاسخگویی یک میزبان انتخاب کنید (پورتی انتخاب کنید که بسته باشد).

بهرحال در استفاده از بسته های UDP باید احتیاط کرد. چون UDP معمولا در خلال حمل و نقل و انتقالات drop شده و/یا توسط دیوارهای آتش بلاک می شود، لذا جواب ICMP_PORT_UNREACH، هرگز به کلاینت نخواهد رسید. بنابراین، در این وضعیت جهت حصول اطمینان، بایستی یک انتقال مجدد (retransmission) انجام داد.

روش های TCP flag (فلگ های TCP)

ساطع و جاری کردن بسته های فلگ دار مختلفی روی یک شبکه، احتمالا موثرترین روش برای تعیین اتصال یک میزبان می باشد. چون این بسته ها به زبان حمل و نقل فراری (گریزان) هستند و خودشان را به عنوان ترافیک معمول روز به روز نشان می دهند، لذا تمایز قائل شدن بین بسته های جمع آوری کننده اطلاعات (بسته های فضول و سرزده) و ترافیک ورودی بی ضرر کار بسیار سختی است. تمام چیزی که جهت موفقیت این تکنیک تشخیص میزبان نیاز است، یک بسته فلگ دار واحد است که بر خلاف روش های فوق الذکر در TCP/UDP echo port، با مراحل دست تکانی سه مرحله ای کار می کند.

خطی مشیء TCP SYN

این روش فلگ SYN، یک پیاده سازی تقریبا موفق از PING sweep می باشد. چون یک پاسخ به هر بسته SYN روی یک پورت بسته یا باز داده می شود، لذا به کلاینت اجازه می دهد تا در تعیین وضعیت آنلاین بودن یک ماشین میزبان مطمئن عمل کند.

دستکاری هدر بسته جهت شامل داشتن فلگ SYN و انتقال آن به یک پورت باز، بسته ای را با بیت های SYN\ACK باز می گرداند. اگر هرگز بسته ای برگشت داده نشد، آنگاه یک کلاینت ممکن است تصور کند که میزبان دارای دیوارآتش بوده یا پورت فیلتر شده است.

```
cefix@term:~ # hping XXX.XXX.XXX.XXX -c 1 -S -p 23
eth0 default routing interface selected (according to /proc)
HPING XXX.XXX.XXX.XXX (eth0 XXX.XXX.XXX.XXX): S set, 40 headers + 0 data bytes
50 bytes from 192.168.1.1: flags=SA seq=0 ttl=64 id=1252 win=32696 rtt=0.9 ms
```

همان طور که نشان داده شده است، SA (SYN\ACK) در بسته برگشتی تنظیم شده است. به هر حال، در این روش از تشخیص میزبان، کلاینت نمی تواند کاملا بداند که آیا یک پورت باز است یا بسته است. چون هم پورت های باز/بسته به بسته SYN جواب می دهند، لذا لزومی به ارسال بسته اولیه جهت آگاهی یافتن از وضعیت باز یا بسته بودن یک پورت وجود ندارد. مثالی از ارسال یک SYN به یک پورت بسته در زیر نشان داده شده است:

```
cefix@term:~ # hping XXX.XXX.XXX.XXX -c 1 -S -p 2
eth0 default routing interface selected (according to /proc)
HPING atlanta (eth0 XXX.XXX.XXX.XXX): S set, 40 headers + 0 data bytes
50 bytes from XXX.XXX.XXX.XXX: flags=RA seq=0 ttl=255 id=1254 win=0 rtt=0.7 ms
```

فلگ های RA (RST\ACK) برگشت داده شده در این بسته، دلالت بر جواب پورت های بسته می کند. چون ما یک بسته برگشتی را دریافت کردیم، می دانیم که میزبان زنده است.

خطی مشیء TCP ACK

این متد، به صورت استدلالی، موثرترین خط مشی برای پویش PING روی یک میزبان ریموت می باشد. قرار دادن فلگ ACK در هدر TCP و انتقال بسته به یک پورت باز یا بسته، بایستی بسته ای را با فلگ RST (reset) برگرداند. از قرار معلوم، این روش به صورت یک ترافیک معمولی تغییر قیافه می دهد، ولی این روش در جاهایی که یک پورت بسته/باز نتیجه نهایی را باز نمی دارد، قابل انعطاف خواهد بود. وضعیت پورت برای این روش محدود نیست و یک کلاینت را قادر می سازد تا یک هدف را روی هر پورت معین (۱ تا ۶۵۵۳۶) گزارش گیری کند و تقریبا دریافت یک جواب تضمین شده می باشد.

البته، میزبان ها ممکن است با قوانین اساسی و موثری روی مسیریاب ها و دیوارهای آتش، این بسته ها را نادیده بگیرند. شانس بسیاری در تعیین محافظت شدن یک میزبان توسط گونه ای از سیستم های فیلترینگ، بواسطه استفاده از ترکیبی از تکنیک های مطرح شده در این مقاله وجود دارد. مثلا، اگر ماشینی پیدا شد که ارتباط TCP echo port را بلاک می کند و هیچ بسته جوابی دریافت نمی شود، آنگاه استفاده از خط مشی TCP ACK به عنوان یک فرض و گمان اشکازدایی که متوجه پورت 7 (echo) می باشد، کلاینت را در پیشگویی و کشف کردن مجموعه قوانین کمک خواهد کرد. سرور ممکن است با بیت RST جواب دهد و این مسئله به این معنی است که:

- پورت 7 TCP (echo) توسط مجموعه قانون دلخواهی فیلترمی شود.
- بسته های دارای فلگ SYN در پورت ۷ بلاک می شوند.
- به بسته های TCP ACK اجازه ورود داده می شود.

تمامی موارد بالا بظاهر آشکار و معلوم هستند، اما احتمالا فرضی که راجع به SYN انجام شد، غلط به نظر می آید. به هر حال، از طریق فرآیند استخراج فهمیدیم که پورت echo فیلتر می شود و نیز فهمیدیم که برای برقراری یک ارتباط روی این پورت، نیاز به استفاده از مذاکره دست تکانی سه مرحله ای داریم که جواب های زیر را شامل کار می کند:

SYN → SYN|ACK → ACK

اکنون همچنین می دانیم که بسته های **ACK**، به کلاینت بسته ای را با بیت **RST** به عنوان جواب بر می گردانند. با استخراج فلگ **ACK** از معادله فوق، به **SYN** و **SYN/ACK** می رسیم. چون فلگ **SYN** ابتدا از کلاینت به هدف انتقال یافته است و یک جواب **SYN/ACK** از مقصد برگشت داده نشده است، ما می توانیم از بیت **SYN/ACK** دست کشید هو آنرا لغو کنیم (چون بسته **SYN** عملا دریافت نشده است، با استفاده از یک دیوارآتش، انتقال آن بلاک می شود). بنابراین، با

استفاده از یک مجموعه قانون یا دیواآتش، بسته هایی که دارای فلگ SYN باشند، روی گره های دریافت کننده drop می شوند. این تکنیک تحت عنوان Firewalking شناخته می شود که شالوده کار، آنالیز کردن انواع بسته ای است که به آنها اجازه عبور داده می شود و نمی شود؛ به این ترتیب می توانیم انواع مجموعه قوانینی که روی یک میزبان دلخواه پیاده سازی شده اند (و نیز آنهایی که پیاده سازی نشده اند) را فهمیده و اصطلاحا MAP کنیم (نگاشت کنیم (!) یا بنگاریم).

با استفاده از HPING جهت ارسال یک بسته ACK به یک پورت بسته و سپس به یک پورت باز، خروجی های زیر را دریافت کردیم:

```
cefix@term:~ # hping XXX.XXX.XXX.XXX -A -c 1 -p 2
eth0 default routing interface selected (according to /proc)
HPING XXX.XXX.XXX.XXX (eth0 XXX.XXX.XXX.XXX): A set, 40 headers + 0 data bytes
50 bytes from XXX.XXX.XXX.XXX: flags=R seq=0 ttl=255 id=1048 win=0 rtt=0.5 ms
```

آرگومان p- پورتی را که بسته باید به آن ارسال گردد مشخص می کند. در این وضعیت، انتقال بسته مستقیما به پورت ۲ (یک پورت NON-LISTENING) انجام شده است.

```
cefix@term:~ # hping XXX.XXX.XXX.XXX -A -c 1 -p 23
eth0 default routing interface selected (according to /proc)
HPING XXX.XXX.XXX.XXX (eth0 XXX.XXX.XXX.XXX): A set, 40 headers + 0 data bytes
50 bytes from XXX.XXX.XXX.XXX: flags=R seq=0 ttl=255 id=1052 win=0 rtt=0.5 ms
```

که پورت ۲۳ باز بود و در نتیجه یک پورت در حال شنود یا یک پورت LISTENING می باشد. همان طور که نشان داده شده است، هر دو جواب از میزبان XXX.XXX.XXX.XXX با فلگ RST روی پورت های بسته و باز پاسخ داده شده است. بنابراین، طبق بررسی فهمیدیم که این میزبان موجودیت داشته یا زنده است (آنلاین است).

خط مشیء TCP SYN\ACK

این متد به زبان سازگاری، قابل انعطاف ترین نیست. کد شبکه بندی BSD [11]،هیچ بسته فلگ داری را به یک بسته گزارشی SYN/ACK پس نمی فرستد، لذا وابسته به معماری شبکه بندی می باشد. بهرحال، تشخیص ویندوز/لینوکس (و دیگران) را می توان با این تکنیک فراهم کرد. یک بسته SYN/ACK در ابتدا به یک پورت دلخواه ارسال می شود (وضعیت باز یا بسته بودن این پورت مهم نیست). بسته برگشت داده شده بایستی دارای بیت RST باشد. چون وضعیت پورت هیچ نقشی را در این سناریو بازی نمی کند، لذا می توان هر پورت تصادفی را بعنوان پورت استفاده کرد.

مثال نشان داده شده در زیر، یک بسته SYN/ACK ارسالی به یک ماشین Win95 در یک پورت غیرشنودی (non-listening) با شماره ۲۳ می باشد. نتیجه یک بسته با فلگ RST می باشد.

```
cefix@term:~ # hping XXX.XXX.XXX.XXX -c 1 -S -A -p 23
eth0 default routing interface selected (according to /proc)
HPING XXX.XXX.XXX.XXX (eth0 XXX.XXX.XXX.XXX): SA set, 40 headers + 0 data bytes
50 bytes from XXX.XXX.XXX.XXX: flags=R seq=0 ttl=128 id=31029 win=0 rtt=0.5 ms
```

همچنین، همان بسته به یک ماشین لینوکس فرستاده شد، با این تفاوت که پورت ۲۳ روی آن در حال شنود بود. مجددا نتیجه، یک بسته با فلگ RST می باشد.

```
cefix@term:~ # hping XXX.XXX.XXX.XXX -c 1 -S -A -p 23
eth0 default routing interface selected (according to /proc)
HPING XXX.XXX.XXX.XXX (eth0 XXX.XXX.XXX.XXX): SA set, 40 headers + 0 data bytes
50 bytes from XXX.XXX.XXX.XXX: flags=R seq=0 ttl=255 id=1258 win=0 rtt=0.5 ms
```

[11] BSD Networking code

خط مشیء TCP FIN

تکنیک مخفیانه تر، تکنیک پویش میزبان TCP FIN می باشد. ارسال یک بسته با این مجموعه فلگ به یک پورت بسته، یک بسته RST/ACK از میزبان ریموت برگشت داده خواهد شد. همچنین، یک پورت باز بسته را دور انداخته و بنابراین برای ما به عنوان روش قابل اعتمادی برای تشخیص میزبان بی فایده خواهد بود.

تعیین یک پورت بسته کاملا راحت است، یک شماره پورت تصادفی، بالاتر از سرویس های رزرو شده (یعنی از ۱ تا ۱۰۲۴) حدس بزنید که فراخوانی پورت های بدون سرویس در بالای این بازه بیشتر است.

```
cefix@term:~ # hping XXX.XXX.XXX.XXX -c 1 -F -p 2
eth0 default routing interface selected (according to /proc)
HPING XXX.XXX.XXX.XXX (eth0 XXX.XXX.XXX.XXX): F set, 40 headers + 0 data bytes
50 bytes from XXX.XXX.XXX.XXX: flags=RA seq=0 ttl=255 id=1260 win=0 rtt=0.5 ms
```

بسته خروجی به یک پورت بسته فرستاده شد و بیت RST\ACK به عنوان جواب دریافت شد. لذا، بسته گزارشی ما توسط سرور جواب داده شد و در نتیجه ما می فهمیم که میزبان زنده است. همچنین، بسته گزارشی جوابی را از میزبان ریموت طلب نکرد. هر یک از مفاهیم و نظریه های زیر دلیل آن را به ما می گوید.

- بسته های **FIN** ورودی توسط دیوارآتش/مسیریاب/**ACL** ها بلاک شده اند.
- ترافیک ورودی روی آن پورت فیلتر شده است.
- پورت گزارش گیری شده باز بوده است (لذا پورت دیگری را امتحان کنید).
- میزبان اصطلاحا **down** می باشد (اتصال آن با شبکه/اینترنت قطع شده است).

خط مشیء TCP NULL

این روش، فرآیند برداشتن (اصطلاحا unset کردن) تمام فلگ ها در هدر TCP و ارسال بسته به یک پورت بسته (NON-LISTENING) را درگیر کار می کند. جواب بایستی بسته ای با بیت های RST/ACK باشد. یک پورت باز به این بسته جواب نخواهد داد (بسته دور انداخته می شود)[۱۲]، لذا باید پورتی را انتخاب کنیم که بدانیم بصورت پیش فرض هیچ سرویسی روی آن در حال شنود نیست.

مثالی از این حالت از پورت بسته با هیچ مجموعه فلگی، در زیر نشان داده شده است:

```
cefix@term:~ # hping XXX.XXX.XXX.XXX -c 1 -p 2
eth0 default routing interface selected (according to /proc)
HPING XXX.XXX.XXX.XXX (eth0 XXX.XXX.XXX.XXX): NO FLAGS are set, 40 headers + 0
data bytes
50 bytes from XXX.XXX.XXX.XXX: flags=RA seq=0 ttl=255 id=1267 win=0 rtt=0.5 ms
```

تعریف یک پورت جهت استفاده به عنوان پورت مورد آزمایش برای این روش از تشخیص میزبان، عمل تقریبا ساده ای است. نتایج این پویش، اغلب به کلاینت تحویل داده نمی شوند (یا اصطلاحا نتایج undelivered باقی می مانند)، چرا که ACL ها و مجموعه قوانین، خصوصا مواظب به گزارش های بسته ای بدون فلگ هستند. لذا این روش به مانند تکنیک های ACK و SYN که در قبل ذکر شدند موثر نخواهد بود، اما البته اگر میزبان به درخواست های استاندارد ICMP echo type 8 جواب ندهد روش مفیدی خواهد بود.

[12] به دور انداخته شدن بسته، **discarding** می گوییم.

خط مشیء TCP XMAS

مشابه روش تشخیص میزبانِ هدر بدون فلگ (هدری را که در آن هیچ فلگی نباشد، هدر بدون فلگ یا هدر فلگ پوچ می گویند) [13]، پویش XMAS، یک جواب از پورت های بسته را با یک بسته تست می کند با این تفاوت که تمام بیت های فلگی روی آن فعال باشند، یعنی روی آن تمام بیت ها برای فلگ های SYN، ACK، FIN، RST، URG، PSH برابر با ۱ باشند (دو بیت رزرو شده نتیجه را تغییر نمی دهند). این روش مبتنی بر پیاده سازی پشته های TCP/IP در یونیکس/لینوکس و BSD بوده و همیشه روی سیستم عامل های ویندوز موفق نخواهد بود.

```
cefix@term:~ # hping XXX.XXX.XXX.XXX -c 1 -p 2 -F -S -R -P -A -U -X -Y
eth0 default routing interface selected (according to /proc)
HPING XXX.XXX.XXX.XXX (eth0 XXX.XXX.XXX.XXX): RSAFPUXY set, 40 headers + 0
data bytes
50 bytes from XXX.XXX.XXX.XXX: flags=RA seq=0 ttl=255 id=1380 win=0 rtt=0.6 ms
```

بیت های RST/ACK دلیل بر این هستند که میزبان جواب ما را دریافت و آن را با بسته انتقالی خودش تایید کرده است. بنابراین، کلاینت این وضعیت را به این گونه تفسیر خواهد کرد که، میزبان دلخواه، زنده بوده و متصل به شبکه می باشد. چون پورت های باز به این درخواست بسته ناهنجار جواب نمی دهند، لذا استفاده از یک پورت باز برای این روش از تشخیص میزبان، کاری عبث و بیهوده خواهد بود.

توضیحات

از اطلاعات بالا، مدارک تضمینی اشاره می کنند که اگر یک میزبان دلخواه در حال اجرای توزیعی از لینوکس/یونیکس/BSD باشد، عملیات تشخیص میزبان روی آن میزبان راحت تر خواهد بود، چرا که این سیستم های عامل به درخواست های بسته ای ناهنجار بسیاری جواب می دهند. اما سیستم های عامل مبتنی بر ویندوز، قابلیتی را در drop کردن بسیاری از ترافیک ناهنجار دارند که نهایتا از تشخیص موفقیت آمیز این میزبان ها در یک محیط شبکه ای (یا اینترنتی) جلوگیری می کند (اما نه در برابر همه روش های پویشی مطرح شده در بالا).

روش های ICMP

پروتکل کنترل پیام یا Internet Control Message Protocol (ICMP) برای گزارش خطاها در پردازش دیتاگرام استفاده می شود و یک بخش جانبی از پروتکل اینترنت یا IP می باشد. تا چند وقت پیش، درباره روش های تشخیص میزبان از طریق ICMP صحبتی به میان نیامده بود تا اینکه مقاله ای از Ofir Akfin، راه هایی که می توان از ICMP برای تشخیص میزبان استفاده کرد را توضیح داد که شامل fingerprinting و inverse mapping (نگاشت معکوس) می باشد.

با اطلاعات امنیتی و افزایش اهمیت آن، آنالیزرهای سیستمی، ACL ها و قوانینی طراحی و پیاده سازی می کنند که تمامی گونه ها و فرم های ICMP را فیلتر کند. اگرچه، تمام فرم های آن مخرب بحساب نمی آیند (انتشارهای smurf، جواب های Excessive Unreachable Error) و بسیاری از گونه های ICMP، ارتباط سرور را در یک محیط شبکه ای یاری می بخشند (برای مثال timestamping).

تشخیص میزبان، نوعی از ICMP که فیلتر می شود را نادیده گرفته و متوجه هرگونه علامت حیاتی که بواسطه چندین ICMP type datagram دلخواه استخراج شده اند می باشد. امروزه، بسیاری از میزبان ها، نوعی از فیلترینگ را در برابر ICMP type 8 دارند (echo request)، اما به دیگر انواع آن کاری ندارند و آنها را فیلترنمی کنند.

ICMP Echo Request (Type 8)

[13] NULL flag header

سرانجام، به روش استاندارد و الزامی مورد استفاده برای تشخیص میزبان رسیدیم. ابزار اشکال زدایی شبکه ای 'PING'، دیتاگرام های ICMP echo request را استخراج می کند تا بدین طریق وضعیت اتصال شبکه را آنالیز کند. در صورتی که میزبان زنده باشد، یک دیتاگرام ICMP از نوع Echo-Reply type 0 برگشت داده می شود.

چون این روش بعنوان روش استاندارد برای تشخیص میزبان طراحی شده است، دیوارهای آتش، مسیریاب ها، ACL ها، مجموعه قوانین خود را حول این امر ساخته و در نتیجه تمامی ترافیک ورودی شبکه از فرم های ICMP type 8 را بلاک می کنند و این دلیلی برای تمامی تکنیک توصیف شده در بالا بوده که از جوابهای استاندارد echo، سرباز زده و طفره می روند (و با این کار معمولا در کار خود موفق می گردند). با تمرکز بیشتر روی بسته ICMP Type 8 نتایج زیر را خواهیم یافت:

1 0123456789	2 0123456789	3 0123456789
Type	Code	Checksum
Identifier	Sequence Number	
Data		

یک جواب نوعی به صورت زیر می باشد:

```
cefix@term:~ # hping -1 XXX.XXX.XXX.XXX -c 1 -C 8
eth0 default routing interface selected (according to /proc)
HPING XXX.XXX.XXX.XXX (eth0 XXX.XXX.XXX.XXX): icmp mode set, 28 headers + 0
data bytes
50 bytes from XXX.XXX.XXX.XXX: icmp_seq=0 ttl=255 id=1273 rtt=0.4 ms
```

همان طور که نشان داده شده است، ۵۰ بایت از ICMP echo reply از میزبان هدف برگردانده شده است. مشابها، مانند دیگر متدهای تشخیص میزبان، بسته ای را دریافت کردیم و در نتیجه پی به زنده بودن میزبان می بریم.

انتشار ICMP (ICMP Broadcast)

عمل انتشار (یا broadcasting)، راهی برای انتقال بسته ها به تمام میزبان ها متصل به یک شبکه می باشد و این کار با ارسال یک echo request (Type 8) به شبکه یا آدرس انتشاری[14] آن انجام می شود. نتیجه اینکه، با اولین بسته اولیه که هر میزبان شبکه دریافت می دارد، یک جواب به کلاینت ارسال خواهد کرد.

در حقیقت، روش انتشار ICMP یا ICMP Broadcasting، روش فوق العاده مفیدی برای نگاشت کامپیوترهای یک شبکه اتصالی و دلخواه می باشد. چون هر گزارش echo جواب داده می شود، این روش تشخیص میزبان، بسادگی به ما اجازه خواهد داد که تمام کامپیوترهای شبکه و در نتیجه کل شبکه را کشف کنیم. به هر حال، یک مانع در کار وجود دارد. طبق پیشفرض، کامپیوترهای ویندوز (به غیر از NT.4 با سرویس پک کمتر از ۴) به بسته های ICMP Type 8 یا Echo Request که مستقیما به آدرس انتشاری یا آدرس شبکه ارسال شده اند جواب نمی دهند، اما در عوض کاملا پنهان و بیصدا، چنین بسته هایی را به دور می اندازند (و ما از همین عمل دور انداختن می توانیم به زنده بودن میزبان پی ببریم). مجددا معلوم

[14] broadcasting address

می شود که سیستم های ویندوز در روش های تشخیص میزبان به صورت ریموت، گریزان تر و بهتر عمل می کنند! در زیر، مثالی از یک بسته ICMP Echo Request که به آدرس شبکه ی یک سرور ارسال شده است را می بینید.

توجه: آدرس IP برابر با XXX.XXX.XXX.4، هیچ جوابی برگشت نداد، چرا که این IP یک Win95 بود.

```
cefix@term:~ # hping -1 XXX.XXX.XXX.0 -c 2
eth0 default routing interface selected (according to /proc)
HPING XXX.XXX.XXX.0 (eth0 XXX.XXX.XXX.0): icmp mode set, 28 headers + 0
data bytes
28 bytes from XXX.XXX.XXX.3: icmp_seq=0 ttl=255 id=13013 rtt=0.4 ms
50 bytes from XXX.XXX.XXX.1: icmp_seq=0 ttl=255 id=426 rtt=0.6 ms
50 bytes from XXX.XXX.XXX.2: icmp_seq=0 ttl=255 id=15319 rtt=0.8 ms

--- XXX.XXX.XXX.0 hping statistic ---
1 packets tramitted, 3 packets received, -100% packet loss
round-trip min/avg/max = 0.4/0.6/0.8 ms
```

واضح است که بسته انتقال یافته واحد، هنگامی که به آدرس شبکه هدایت شد سه پاسخ را دریافت کرد. به این ترتیب، یک بسته ارسالی به آدرس انتشاری، به همین ترتیب، سه دیتاگرام جواب تولید خواهد کرد.

```
cefix@term:~ # hping -1 XXX.XXX.XXX.255 -c 2
eth0 default routing interface selected (according to /proc)
HPING XXX.XXX.XXX.255 (eth0 XXX.XXX.XXX.255): icmp mode set, 28 headers + 0
data bytes
28 bytes from XXX.XXX.XXX.3: icmp_seq=0 ttl=255 id=13098 rtt=0.4 ms
50 bytes from XXX.XXX.XXX.1: icmp_seq=0 ttl=255 id=730 rtt=0.7 ms
50 bytes from XXX.XXX.XXX.2: icmp_seq=0 ttl=255 id=15327 rtt=0.8 ms

--- XXX.XXX.XXX.255 hping statistic ---
1 packets tramitted, 3 packets received, -100% packet loss
round-trip min/avg/max = 0.4/0.7/0.8 ms
```

مجددا می بینید که، این تکنیک روش موفقی را در استفاده از آدرس انتشاری به عنوان مکانیزم نگاشت میزبان های شبکه ای برای ما به ارمغان می آورد!

ICMP Router Solicitation (Type 10)

درخواست های ICMP router discovery (درخواست های ICMP که وظیفه کشف و بررسی وضعیت مسیریاب را به عهده دارند)، تحت عنوان Router Solicitation (تقاضای مسیریاب) نامیده می شوند. هر مسیریاب، بصورت دوره ای یک اعلان مسیریابی[15] (ICMP type 9) را از هر یک از واسط های چند-طرحی[16] خود انتشار می دهد (multicast می کند) که آدرس های IP آن رابط را اعلان می دارد.

این تکنیک برای کشف سیستمی که به عنوان مسیریاب عمل می کند مفید خواهد بود. واضح است که ICMP router solicitation، یک قالب پیامی اختیاری روی یک میزبان استاندارد (میزبانی که به خودی خود فقط یک کامپیوتر معمول است و نه مسیریاب یا ...) می باشد. به هر حال، برای یک مسیریاب، این اجبار وجود دارد که پیاده سازی ICMP router solicitation را بصورت فعال داشته باشد. بنابراین، اگر سرورها در جواب به یک ICMP Type 10، با یک ICMP

[15] Router Advertisement
[16] Multi-Cast

Type 9 جواب دادند، شخص می تواند مطمئن باشد که سرور یک مسیریاب یا وسیله شبکه ای می باشد. نیازی نیست که بگوییم، میزبانی که یک ICMP type 10 را دریافت می کند ولی برای انتقال این پیام ها پیکربندی نشده است، نمی تواند جوابی را پس بفرستد. قالب بسته برای پیام های کشف مسیریاب[17] به صورت زیر می باشند:

0 0 2 4 6 8	1 10 12 14	2 3 16 18 20 22 24 26 28 30 31
Type	Code	Checksum
Reserved		

متاسفانه، hping، قالب های پیامی ICMP type 10,13,15,17 را پیاده سازی نکرده است. در عوض، ما icmpush را به عنوان ابزار تولید/آنالیز بسته انتخاب کرده ایم[18].

```
cefix@term:~ # ./icmpush -vv -rts XXX.XXX.XXX.XXX
 -> Outgoing interface = XXX.XXX.XXX.XXX
 -> ICMP total size = 20 bytes
 -> Outgoing interface = XXX.XXX.XXX.XXX
 -> MTU = 1500 bytes
 -> Total packet size (ICMP + IP) = 40 bytes
ICMP Router Solicitation packet sent to XXX.XXX.XXX.XXX (XXX.XXX.XXX.XXX)

Receiving ICMP replies ...
XXX.XXX.XXX.XXX -> Router Advertisement (XXX.XXX.XXX.XXX)
./icmpush: Program finished OK
```

یک موفقیت دیگر! ما اکنون فهمیده ایم که یک مسیریاب را روی این شبکه پیدا کردیم. ممکن است این مسیریاب، دیگر انواع ICMP را فیلتر یا دیگر پورت ها را بلاک کند، اما بهرحال، این مقاله روش های مختلفی را برای دور زدن این گونه از ACL ها توضیح داده است (بهرحال ...).

ICMP Timestamp Request (Type 13)

جواب (ICMP type 14) داخل یک درخواست timestamp، داده درخواستی اولیه است. بدیهی است که، درخواست های timestamp به منظور گزارش گیری زمان جاری از یک سرور ساخته شده اند.

اغلب، قطع سازگاری پلاتفرم، هنگام درخواست یک جواب timestamp، دستی را به ما قرض خواهد داد!! Win95 و WinNT به گزارش های ارسالی پاسخ نمی دادند و یونیکس/لینوکس/BSD با داده های صحیح و درست جواب می دادند. نگاه به خود بسته ICMP، موارد زیر را برای ما آشکار می سازد:

[17] Router Discovery message

[18] این برنامه از آدرس http://hispahack.ccc.de قابل دریافت است.

0 1	2	1
0123456789	0123456789	012345678901
Type	Code	Checksum
Identifier		Sequence Number
Originate Timestamp		
Receive Timestamp		
Transmit Timestamp		

اولین نمونه نشان داده شده در فوق، یک درخواست ICMP timestamp را به یک سرور لینوکس انتقال می دهد و نتیجه 03:50:32 بود که درون فیلد data کپسوله سازی شد.

```
cefix@term:~ # ./icmpush -vv -tstamp XXX.XXX.XXX.XXX
 -> Outgoing interface = XXX.XXX.XXX.XXX
 -> ICMP total size = 20 bytes
 -> Outgoing interface = XXX.XXX.XXX.XXX
 -> MTU = 1500 bytes
 -> Total packet size (ICMP + IP) = 40 bytes
ICMP Timestamp Request packet sent to XXX.XXX.XXX.XXX (XXX.XXX.XXX.XXX)

Receiving ICMP replies ...
XXX.XXX.XXX.XXX -> Timestamp Reply transmited at 03:50:32
./icmpush: Program finished OK
```

مثال بعدی همان بسته را اما این بار به یک کامپیوتر Win95 ارسال می دارد و می بینیم که هیچ بسته برگشتی دریافت نمیشود:

```
cefix@term:~ # ./icmpush -vv -tstamp XXX.XXX.XXX.XXX
 -> Outgoing interface = XXX.XXX.XXX.XXX
 -> ICMP total size = 20 bytes
 -> Outgoing interface = XXX.XXX.XXX.XXX
 -> MTU = 1500 bytes
 -> Total packet size (ICMP + IP) = 40 bytes
ICMP Timestamp Request packet sent to XXX.XXX.XXX.XXX (XXX.XXX.XXX.XXX)

Receiving ICMP replies ...
./icmpush: Program finished OK
```

درخواست ICMP timestamp، از یک روش مناسب، جهت استفاده برای تشخیص میزبان (مخصوصا سرور های *NIX) استفاده می کند.

ICMP Information Request (Type 15)

این پیام جهت گزارش گیری از یک میزبان، برای کشف آدرس شبکه آن استفاده می شود، هرچند، RFC ها اظهار می دارند که ICMP type 15 (Information Request) و ICMP type 16 (Information Reply) منسوخ هستند،

اما دلیلی نمی شود که بگوییم این پیام ها در هیچ جا استفاده نمی شوند. یک قسمت جانبی از ایـن ICMP type، نتـایج زیـر را بازگو می کند:

1 0123456789	2 0123456789	3 0123456789
Type	Code	Checksum
Identifier	Sequence Number	

یک بسته با ICMP type 15 به یک میزبـان ارسـال شـد و آن میزبـان ایـن درخواسـت را جـواب داد و در نتیجـه موجودیت او روی یک میزبان دلخواه برای ما روشن شد.

```
cefix@term:~ # ./icmpush -vv -info XXX.XXX.XXX.XXX
 -> Outgoing interface = XXX.XXX.XXX.XXX
 -> ICMP total size = 8 bytes
 -> Outgoing interface = XXX.XXX.XXX.XXX
 -> MTU = 1500 bytes
 -> Total packet size (ICMP + IP) = 28 bytes
ICMP Info Request packet sent to XXX.XXX.XXX.XXX (XXX.XXX.XXX.XXX)

Receiving ICMP replies ...
XXX.XXX.XXX.XXX -> Info Reply (XXX.XXX.XXX.XXX)
./icmpush: Program finished OK
```

مجددا می بینیم که نتوانستیم این نتایج را در یک سیستم Win 95/NT بدست آوریم، اما چندین توزیع از NIX*، با موفقیت به این درخواست پاسخ دادند.

ICMP Address Mask Request (Type 17)

درخواست های Address Mask جهت بدست آوردن آدرس subnet mask روی شبکه محلی تولیـد مـی شـوند. جواب به این بـسته گزارشـی اولیـه، یـک ICMP type 18 (Address Mask Reply) مـی باشـد کـه بایـستی حـاوی آدرس subnet باشد. نگاه دقیق تری به ICMP Address Mask نتایج زیر را به بار خواهد آورد:

1 0123456789	2 0123456789	3 0123456789
Type	Code	Checksum
Identifier	Sequence Number	
Subnet Mask Address		

جالب است بدانید که ارسال این ICMP type به توزیع های مختلف لینوکس، هیچ جوابی به صورت ICMP type 18 را در برگشت نداشت، اما سیستم های ویندوز به این ICMP type، جواب دادند. مثال زیر نتیجه ارسـال بـسته Address Mask Request را به یک سیستم ویندوز نشان می دهد:

```
cefix@term:~ # ./icmpush -vv -mask XXX.XXX.XXX.XXX
 -> Outgoing interface = XXX.XXX.XXX.XXX
 -> ICMP total size = 12 bytes
 -> Outgoing interface = XXX.XXX.XXX.XXX
 -> MTU = 1500 bytes
 -> Total packet size (ICMP + IP) = 32 bytes
ICMP Address Mask Request packet sent to XXX.XXX.XXX.XXX (XXX.XXX.XXX.XXX)

Receiving ICMP replies ...
XXX.XXX.XXX.XXX -> Address Mask Reply (255.255.255.0)
./icmpush: Program finished OK
```

لذا subnet mask در طرف ما دریافت شد. هنوز روش تشخیص میزبان دیگری در مقابل سیستم های ویندوز ممکن است تا از آن برای نگاشت میزبان در یک محیط جانبی از پلاتفرم شبکه ای که درخواست ICMP echo را بلاک می کند استفاده کنیم.

توضیحات

بعضی خوانندگان ممکن است به این فکر فرو روند که چرا ICMP reply های میزبان ها را استخراج نکرده و عکس العمل (پاسخ) آنها را برای تشخیص میزبان آنالیز نکنیم؟ RFC 1122 موارد زیر را توضیح می دهد:

یک پیام خطای ICMP، نباید به عنوان نتیجه ی دریافت، ارسال شود:

- یک پیام خطای ICMP، یا
- یک دیتاگرام عازم به یک آدرس IP انتشاری یا آدرس IP چند طرحی، یا
- یک دیتاگرام بعنوان یک انتشار link-layer، یا
- یک قطعه بدون ابتدا، یا
- یک دیتاگرام که آدرس منبع آن یک میزبان واحد را تعریف نمی کند – مثلا، یک آدرس صفر، یک آدرس loopback، یک آدرس انتشاری، یک آدرس چندطرحی- یا یک کلاس آدرس E.

اولین نکته واضح گویای این است که یک میزبان جوابی را به یک دیتاگرام ICMP reply ارسال نمی دارد، لذا توجه ما برای تشخیص میزبان، به استفاده از دیتاگرام های reply جلب می شود.

تولید جواب های پروتکلی غیرمعتبر

این قسمت روش هایی را بازگو می کند که از Internet Protocol (IP) برای فاش کردن پیام های خطا به منظور کشف میزبا های دلخواه استفاده می کند. پروتکل متناظر کپسوله شده (TCP/UDP/ICMP) هیچ تاثیری روی نتایج استفاده از این متد هنگامی بسته بندی دیتاگرام ندارند.

این اصل برای آنالیز کردن وضعیت اتصال یک میزبان، احتیاج به ACL های موثر و فیلترکننده های بسته های خروجی را تاکید می کند. موارد پایه برای این تکنیک بر تولید دیتاگرام های غیرمعتبر و کشف جواب های خارجی مبتنی هستند که یک بسته ناهنجار، به عنوان یک نتیجه نابهنجار تولید می شود.

خطا مشیء IP Header (هدرهای IP)

تولید هدر های IP ناهنجار در فرآیند انتقال (و حمل و نقل بسته ها)، شانس تشخیص میزبان های دارای دیوارآتش و فیلترشده یا افزایش می دهد. بسیاری از فرم های IDS ها و مسیریاب ها، بسته هایی که حاوی هدرهای ناهنجاری چون مقادیر غیرمعتبر برای فیلد ها هستند را drop می کنند. تکنیک های زیر چنین مواردی را نشان می دهند:

- **Timedout Packet Fragmentation**
- **Invalid Header Length**
- **Invalid Field Values**

تکه سازی بسته timeout شده (Timedout Packet Fragmentation)

روش دیگری که در تشخیص میزبان پیشرفته مورد استفاده قرار می گیرد، ارسال نداشتن قطعه سازی بسته می باشد. این مورد ابتدای به ساکن برای ساخت بسته ای با یک Fragment Offset و ارسال آن به یک میزبان ضروری است. بجای سرهم سازی دیتاگرام تکه شده دیگری جهت کامل کردن بسته، کلاینت اجازه می دهد که دیتاگرام تکه شده ی اولیه timeout شود و در نتیجه باعث می شود سرور منتظر بسته بعدی در این توالی (از بسته های تکه شده) شود. تاثیر این مورد، یک ICMP type 11 با Code 1 بوده که برابر با Time Exceeded Fragment Reassembly است و توسط سرور تولید می شود. یک مثال:

```
cefix@term:~ # hping -c 1 -x -y XXX.XXX.XXX.XXX
eth0 default routing interface selected (according to /proc)
HPING dev (eth0 XXX.XXX.XXX.XXX): NO FLAGS are set, 40 headers + 0 data bytes

--- dev hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

توجه: اگر بررسی در مورد دیتاگرام ICMP که میزبان ریموت تولید کرده است انجام نگردد، hping از دست رفتن صددرصدی بسته ها را نشان می دهد. tcpdump اطلاعات زیر را نشان می دهد:

```
20:41:09.309085 YYY.YYY.YYY.YYY > XXX.XXX.XXX.XXX: icmp: ip reassembly time
exceeded [tos 0xc0]  (ttl 255, id 3375)
```

مجددا می بینیم که با استفاده از ارتباط پروتکلی غیرمعتبر، سبب تولید یک جواب از سرور شدیم.

طول غیرمتعبر برای هدر

تعیین یک طول غیرمعتبر برای هدر درون هدر IP، باعث خواهد شد که در میزبان ریموت، یک پیام خطای ICMP type 12 تحت عنوان Parameter Problem تولید شود. نوع کد این پیام موجود در این دیتاگرام ICMP، ممکن است معادل یکی از موارد زیر باشد:

- مقدار «0»: اشاره گر به خطا اشاره می کند.
- مقدار «2»: طول نامناسب

یک کد معادل با 0، مقدار بایت دقیقی که خطای کپسوله شده درون فیلد pointer را باعث می شود برمی گرداند. به این ترتیب، یک کد معادل با 2، دال بر این خواهد شد که کل بسته حاوی خطا می باشد. در هر یک از این دو مورد، میزبان موجود در طرف گیرنده ی این بسته، ICMP type 12 با کد (صفر یا دو) را در برگشت ارسال خواهد کرد و به این ترتیب به

فرستنده می گوید که بسته دور انداخته شد یا drop گشت.

در زیر از ISIC (IP Stack Integrity Checker) – بررسی کننده بی نقص بودن پشته ی IP – برای سرهم سازی یک بسته با طول هدر IP غیرمعتبری برابر با ۶۶ بایت، استفاده شده است:

```
cefix@term:~ # ./isic -s YYY.YYY.YYY.YYY -d XXX.XXX.XXX.XXX -p 1 -V 0 -F 0 -I 66 -D
Compiled against Libnet 1.0.1b
Installing Signal Handlers.
Seeding with 5099
No Maximum traffic limiter
Bad IP Version  = 0%   Odd IP Header Length   = 100%   Frag'd Pcnt   = 0%
YYY.YYY.YYY.YYY -> XXX.XXX.XXX.XXX tos[137] id[0] ver[4] frag[0]

 Wrote 1 packets in 0.00s @ 5649.72 pkts/s
```

ردیابی tcpdump نتایج زیر را نشان داد:

```
21:39:03.755839 XXX.XXX.XXX.XXX > YYY.YYY.YYY.YYY: icmp: parameter problem -
octet 20 [tos 0xd0]  (ttl 255, id 21508)
```

همان طور که مورد نظر بود، یک طول هدر ناهنجار، منجر به تحمیل در برگشت یک جواب دلخواه از سرور شد. نهایتا می توان این روش را برای دور زدن انواع بسیاری از ACL ها و سیستم های فیلترینگ (اگر بدرستی پیکربندی نشده باشند) بکار برد.

مقادیر غیرمعتبر برای فیلدها

در سطح کلی تری باید گفت که، تعیین مقادیر غیرمعتبر درون هر یک از فیلدهای هدرIP، پیام های خطای ICMP را روی میزبان هدف تولید خواهد کرد. در این چنین موردی، ما با فیلد IP PROTO کار می کنیم که در جمع، اندازه ای برابر با ۸ دارد. بنابراین با محاسبه کوچکی بر طبق مباحث ریاضیات گسسته، خواهیم فهمید که کلا امکان وجود، ۲۵۶ ترکیب ممکن در این فیلد است (دو به توان هشت). حقه و نیرنگی (!) که در این وضعیت بکار می بریم، انتخاب یک مقدار پروتکلی است که دال بر یک مقدار پروتکلی قانونی (و مشروع و صحیح) روی آن میزبان نباشد.

خوشبختانه، یک کلاینت قادر است که تعیین کند آیا یک میزبان، یک پروتکل خاص را پشتیبانی می کند یا خیر، به این صورت که سرور در صورت عدم پشتیبانی از آن پروتکل، پیام ICMP type 3 را با کد ۳ که برابر با Destination Protocol Unreachable است تولید می کند. اگر جوابی پس فرستادن نشود، کلاینت فرض می کند که پروتکل تعیین شده، روی آن میزبان پشتیبانی می گردد. برای مثال، apsend [19] برای تولید بسته مورد استفاده قرار گرفته است:

```
cefix@term:~ # perl apsend -s YYY.YYY.YYY.YYY -d XXX.XXX.XXX.XXX -b 8 -p 8
--protocol 0
Packet: 1 from YYY.YYY.YYY.YYY(port: 8) to XXX.XXX.XXX.XXX(port: 8).
Protocol: 0  Type of Service(ToS): 16  ID: 0
```

در مثال بالا، دیتاگرام با پروتکلی برابر با 0 ارسال شد و بنابراین بایستی همیشه یک خطای ICMP را برگشت دهد. یک ردیابی tcpdump نتایج زیر را بازگو کرد:

```
21:58:21.128201 YYY.YYY.YYY.YYY > XXX.XXX.XXX.XXX: icmp: dev.synnergy.net
protocol 0 unreachable [tos 0xd0]  (ttl 255, id 24133)
```

[19] به آدرس http://www.elxsi.de مراجعه کنید.

همان طور که مورد انتظار بود، XXX.XXX.XXX.XXX با یک دیتاگرام ICMP type 3 با کد ۳ برگشت داده شد و به ترتیب فهمیدیم که میزبان زنده است و لذا روش تشخیص میزبان موفق دیگری را کشف کردیم!

توضیحات و نکات نهایی

با استفاده از مقادیر غیرمعتبر برای فیلد IP، می توان بسته های ناهنجار دیگری را برای تولید جواب های دلخواه روی یک میزبان استفاده کرد، بهرحال، در این مقاله راه اصلی و اهم و مهم های آن نشان داده شد و پیمودن باقی مسیر (برای رسیدن به درک کاملی از این موضوع، یعنی تشخیص و کشف میزبان) به عهده خود خواننده خواهد بود که مسلما در این موارد، با در دست داشتن یک راه کلی، عمل کردن و به نتیجه رسیدن (آزمون و خطا) شیرین تر از خواندنِ تنها خواهد بود!! به هرحال، بسیاری از این روش ها را می توان روی بسیاری از پروتکل های IGMP و ARP به عنوان مکانیزم موثری برای تشخیص میزبان های دارای دیوارآتش بکار برد (که در صورت نیاز، آماده به ارائه جزئیات کلیدی در این موارد هستم).

پیاده سازی فیلترهای ترافیک ورودی و خروجی، یک الزام برای هر شبکه است که می خواهد از بسیاری از فرم های تشخیص میزبان به صورت ریموت در امان بماند. یک مجموعه قانون و شالوده قانونی مناسب و صحیح و یک ACL مناسب را نیز بایستی مد نظر داشت و به عنوان یک ابزار استاندارد برای تمرین و اعمال مباحث امنیتی روی شبکه بررسی و آزمایش کرد.

در این لحظه و این نقطه از مقاله، خواننده بایستی به اندازه کافی با اطلاعات مجهز شده باشد تا بدقت و بدور از هرگونه خطا چنین پروتکل هایی را برای تولید جواب های سرور به عنوان یک وسیله برای تشخیص پیشرفته میزبان استنطاق و بازجویی کند.

ترجمه: سعید بیکی (cephexin@secumania.net)

Secumania Security & Vulnerability Research Lab
www.secumania.net