IP ID-сканирование

December 27, 2016

В этой статье описан канонический вариант IP ID-сканирования и его модификации, как уже известные, так и новые (2^ сканирование, скрытное наблюдение за IP ID). IP ID (dumb host) сканирование — относительно новая и очень мощная технология, которая позволяет:

- провести максимально скрытное сканирование. На атакуемом хосте нет даже принципиальной возможности узнать адрес взломщика, а трафик взломщика с релеем (dumb хостом) может быть замаскирован под обычный трафик;
- сканировать системы, доступ к которым закрыт для атакующего файрволом;
- частично исследовать правила файрволов.

Исторические предпосылки

Предпосылка 1.

Для того, чтобы при передаче большого IP-пакета между сетями с разными MTU его можно было фрагентировать, RFC 791 требует от хоста, отправляющего IP-пакет, чтобы совокупность полей Identification (16 бит), Source address, Destination address и Protocol была уникальной в течении времени, пока пакет может быть 'живым' в Сети. Так же допускается, что некоторые хосты могут просто использовать уникальные значения для ID-поля каждого исходящего пакета вне зависимости от source/destination адресов. В большинстве случаев разработчики операционных систем пошли по наиболее простому пути, присваивая каждой последующей исходящей датаграмме значение поля ID на единицу большее, чем в предыдущей.

Предпосылка 2.

TCP-протокол (RFC 793) требует, чтобы хост отсылал RST-пакет в случае, если он получил любой пакет, кроме RST, не относящийся к существующим соединениям. Кроме того, процедура three-way handshake'а требует, чтобы по приходу SYN'а на открытый порт хост ответил ACK и SYN'ом (обычно это происходит в одном пакете, хотя, насколько я понимаю, должна быть возможность выполнить этот этап хэндшейка и двумя пакетами с SYN и ACK-флагами по одному в пакете).

Описание базового метода IP ID-сканирования

18 декабря 1998 года Salvatore Sanfilippo aka antirez описал в своем постинге в bugtraq новый метод сканирования с использованием dumb хостов. Техника сканирования хорошо описана в письме, так что я лишь в общих чертах здесь расскажу о ней.

В оригинале атака позволяет просканировать TCP-порты машины *victim* с использованием '*немой*' (*dumb*) системы. В качестве *dumb* может выступать любая машина, которая бы увеличивала IP ID на единицу с каждым отосланным пакетом (что делают очень многие ОС) и не имела трафика во время сканирования, чтобы изменения IP ID контролировались только атакующим. Так же *dumb* должна иметь возможность выполнить портскан *victim*. (Обратите внимание — *dumb*, не атакующий! — его машина как раз может и не иметь такой возможности, например, трафик с нее может быть специально блокирован файрволом, но это не является препятствием).

Атакующий, зная адрес любой *dumb*-системы, посылает SYN-пакеты на сканируемые порты исследуемой машины (*victim*), заменяя в отправляемых пакетах свой адрес на адрес *dumb*. При этом он создает поток TCP-пакетов на адрес *dumb* для отслеживания изменения поля ID в приходящих к нему с *dumb* пакетах. При 'сетевом штиле' на *dumb*, каждый ответный пакет с него будет идти с полем ID на единицу большим, чем ранее.

Сканируемый хост (*victim*), в зависимости от того, открыт указанный в пакете порт назначения или нет, отвечает на адрес 'отправителя' (то есть на адрес *dumb*) либо RST-пакетом (если порт открыт), либо SYN|ACK-пакетом (если порт открыт).

Получив ответ от *victim*, *dumb* либо игнорирует его (если это RST-пакет), либо (если это SYN|ACK) отвечает сам RST-пакетом, так как пришедший пакет не относится ни к одному из соединений. В первом случае значение поля IP ID в исходящих с *dumb*-пакетах по-прежнему монотонно увеличивается на 1, и атакующий, не обнаружив никакого изменения, делает вывод, что порт закрыт. Во втором случае, отправив RST-пакет, *dumb* увеличил на единицу свой счетчик, и атакующий, заметив этот скачок, делает вывод, что порт открыт.

В чем заключаются отличительные особенности этого типа уязвимости? Сканируемый хост не знает, кто на самом деле атакующий.

- Даже полная запись всего трафика на *victim* и последующий анализ не позволят найти злоумышленника;
- Администратор атакованной машины может по ошибке сделать вывод, что *dumb* и есть атакующий (картина в сети будет такая, будто *dumb* выполняет SYN-сканирование);
- Сканирование портов данным методом использует уязвимость не *victim*, а третьей машины. Вне зависимости от того, подвержена ли *victim* этой уязвимости или нет, взломщик может все равно осуществить такую атаку;

- Требуется всего 1 *dumb host* для проведения атаки. А значит, технология будет реализуема, пока ВСЕ машины в Интернете не сменят способ генерации IP ID (если таких машин станет уже критически мало, теоретическая возможность атаки останется, на практике осуществить ее будет тяжело, так как одновременная попытка двух взломщиков провести такую атаку через один *dumb* приведет к тому, что они будут мешать друг другу);
- Ни администратор уязвимой машины, ни разработчик ОС для нее не сильно страдают от такой уязвимости (следовательно, и не испытывают острой необходимости исправлять уязвимость), поскольку *dumb* является релеем, но не жертвой атаки.

Возможности, которые дает IP ID-скан и его модификации

Исследование правил файрвола и обход файрвола при сканировании.

Метод оказывает некоторую помощь в исследовании настроек файрволов. Фактически проверка каждого порта на стороне *victim* выглядит как попытка установления соединения от хоста *dumb*, и результаты сканирования показывают картину *victim* с точки зрения *dumb*. Если *dumb* использует требуемую здесь генерацию IP ID, есть возможность проверить, доступен ли какой-либо порт на *victim* для *dumb*. Для сканирования машин за файрволом требуется только, чтобы у атакующего была возможность послать пакет на *victim* с *saddr=dumb*. Часто это не проблема, так как тяжело определить, кто на самом деле отправил пакет, но если атакующий и *dumb* находятся на разных интерфейсах роутера/файервола жертвы, то правильная настройка файервола может быть серьезным препятствием для такого сканирования.

Сканирование машин с приватными адресами.

Elie Bursztein aka lupin опубликовал advisory, где указал на применимость этого сканирования для случая, когда *dumb* — шлюз в приватную сеть, а *victim* — машина, находящаяся во внутренней сети за *dumb*. Сканирование выполняется тем же путем. Но следует учесть, что возможно это только в случае (см. выше), когда атакующий может послать спуфленный пакет на *victim*. А это возможно, только когда ВСЕ роутеры на пути от атакующего до жертвы пропустят такой пакет, и более того, будут маршрутизировать его в направлении, указанном атакующим. На практике это возможно только в случае, когда *dumb* не имеет элементарной защиты от spoof'инга, а взломщик атакует с машины, непосредственно общающейся с *dumb* (e.g. с роутера провайдера). Но в этом случае, как правило, у атакующего есть и обычная возможность выполнить сканирование *victim*, единственное ограничение, которое в этом случае снимается — нет необходимости в получении атакующим трафика с *victim*. Также следует помнить, что, поскольку в качестве *dumb* выступает машина из той же сети, что и *victim*, для расследования

инцидента могут быть использованы логи трафика на ней — в них будет присутствовать ip атакующего.

Использование IP ID при сканировании UDP-сервисов за файерволом.

С некоторыми модификациями IP ID-сканирование можно проводить и для сканирования UDP-портов. 'Обычное' UDP-сканирование отличается от TCP из-за того, что нет схемы хендшейка на уровне протокола и нет возможности, послав один UDP-пакет на какой либо порт, получить в ответ какое-либо стандартное подтверждение или опровержение гипотезы о наличии открытого порта от UDP-протокола. Но ОС, в случае обращения к несуществующему UDP-порту, посылают ICMP-сообщение об этом. Для сканирования используется метод посылки UDP-пакета на анализируемый порт и ожидание ICMP-сообщения с Type=3 (Destination Unreachable), Code=3 (Port Unreachable). Если сообщение не приходит, порт считается открытым (с некоторой вероятностью), если приходит — порт считается закрытым.

Любой правильно сконфигурированный файервол не пропустит такие сообщения наружу, и для атакующего, использующего ICMP для определения статуса порта, нет возможности узнать, открыт порт или нет. Здесь опять возможно использование IP ID для сканирования, только в качестве *dumb* выступает сама *victim* (если она подходит под это определение). Атакующий следит за изменениями IP ID, при этом сканируя UDP-порты. Как только на *victim* (она же *dumb*) приходит пакет с адресом закрытого UDP-порта, *victim* отсылает ICMP-сообщение об этом, увеличивая тем самым счетчик IP ID. Ответ на следующий TCP-пакет атакующего уже будет нести в себе информацию об этом. saddr может spoof'иться любым значением, в зависимости от конкретной ситуации.

Атакующий должен 'видеть' ответы на свои TCP-запросы, чтобы отслеживать скачки IP ID, следовательно, метод не позволяет сохранить полную анонимность, но UDP-пакеты могут идти с произвольного адреса, так что многие IDS могут зарегистрировать сканирование с подставного адреса. Сложнее, но тоже возможно использовать слежение за IP ID для проведения анонимного UDP-сканирования на известные сервисы. Технология заключается в том, чтобы спровоцировать *victim* послать UDP-сообщение (возможно, некорректное), которое бы вызвало ответную датаграмму на *dumb*. Если атакующий может это сделать — тогда *dumb* может ответить на это icmp port unreachable, что вызовет скачок IP ID в исходящих с него пакетах. Метод позволяет анонимно определить некоторые сервисы на victim, которые можно спровоцировать на исходящее UDP-сообщение отправителю. Например:

```
    hping2 -i u100 -a <DUMB> -D <victim> -p 53 -S -r -sign
123456789012 -d 12 -s 123 -udp
```

Вызывает шторм (100 пакетов в секунду) некорректных DNS-запросов на 53 порт *victim*, что ведет к такому же шторму UDP-сообщений об ошибке с *victim* на *dumb* с флагами 0xb3b4 (Unknown operation response, Not implemented) на порт 123. Каждое такое сообщение, в свою очередь, приведет к

исходящему с *dumb* на *victim* ICMP-сообщению, и главное — к изменению IP ID, что и требуется для подтверждения того, что на victim работает DNS-сервер. Картина при 'hping2 <DUMB> -r' во время вышеописанного шторма DNS:

```
   len=46 ip=<DUMB> flags=RA seq=1 ttl=128 id=+98 win=0 rtt=0.2
ms
   len=46 ip=<DUMB> flags=RA seq=2 ttl=128 id=+101 win=0 rtt=0.4
ms
   len=46 ip=<DUMB> flags=RA seq=3 ttl=128 id=+101 win=0 rtt=0.2
ms
   len=46 ip=<DUMB> flags=RA seq=4 ttl=128 id=+103 win=0 rtt=0.3
ms
```

Метод подходит не для всех сервисов. Syslog, насколько я знаю, вообще невозможно обнаружить таким способом, т.к. он молча проглатывает все входящее по udp/514 к себе в лог. Для неизвестных UDP-сервисов можно пробовать несколько разных пакетов (на будущее) — можно разработать минимальный набор таких 'ошибочных' UDP-датаграмм, чтобы любой достаточно популярный udp-сервис ответил хотя бы на одну из них. Поскольку достаточно просто сделать пакет, который был бы 'плохим' для нескольких сервисов, этот путь мне кажется более перспективным, чем разработка минимального набора корректных датаграмм), благо, что любой ответ *victim* (и хоть один на сколько угодно входящих пакетов) будет устраивать атакующего, так как говорит о том, что порт открыт.

Для определения 'немых' сервисов типа syslog возможно использование косвенных признаков. Например, syslog не посылает ответов на приходящие датаграммы, но резолвит ip адреса, с которых пришли эти сообщения. Если следить за IP ID на DNS-сервере, который будет спрошен в результате ресолвинга отправленной syslog'у датаграммы, можно заметить скачок IP ID на нем, что будет свидетельством того, что порт открыт. В качестве DNS-сервера для этой атаки подойдет тот, который:

1. обладает уязвимой схемой генерации IP ID;
2. не имеет трафика в момент сканирования;
3. будет запрошен в момент получения датаграммы на открытый порт.

Это может быть либо дефолтный неймсервер для сканируемой машины, либо DNS, который держит реверсную зону для ip-адреса, который атакующий использовал в качестве ;

Помехой могут являться 'паразитные бэкрезолверы' — любые программы, которые пытаются выполнять бэкрезолв для адресов в услышанных ими пакетах (снифферы, NIDS, логгеры). Чтобы проверить, есть ли таковые в сканируемой сети, стоит отправить пакет на заведомо (наиболее вероятно) закрытый порт. Отсутствие DNS-запроса в этом случае говорит о благоприятных условиях для сканирования.

Так же следует учесть то, что DNS-записи могут кэшироваться, и второй пакет с того же адреса на открытый и резолвящий сервис может не вызвать DNS-запроса на контролируемый сервер. Поэтому рекомендуется использовать либо разные saddr в разных пакетах, либо сканировать через достаточно большие промежутки времени, чтобы запись проэкспарилась (выбрать хорошую реверсную зону с небольшим TTL).

"2 в степени" (2^) сканирование и временные особенности сканирования.

Используя классическое IP ID-сканирование, атакующий после отсылки пакета на тестируемый порт жертвы ждет либо скачка IP ID, чтобы сделать вывод о том, что порт открыт, либо истечения достаточного времени (назовем его Max RTT) чтобы сделать вывод, что ответа нет и порт закрыт. Для тестирования следующего порта атакующий обязательно должен быть уверен, что нет трафика оставшегося от предыдущего тестирования, так как это может исказить результат: представим ситуацию, что либо пакет от атакующего на открытый порт жертвы, либо SYN|ACK-пакеты жертвы задержались при передаче и атакующий перешел к тестированию следующего порта (ошибочно посчитав этот тестируемый порт закрытым). Отослав пакет на следующий порт жертвы, он видит скачок IP ID на *dumb*, вызванный задержавшимся пакетом от предыдущего тестирования и делает потенциально ошибочный вывод о том, что текущий тестируемый порт открыт. Кроме того, возможно, имеется еще и SYN|ACK-пакет от этого порта, который внесет свои помехи в тестировании следующего, и так далее. Отсюда правило — спешить в IP ID-сканировании не стоит. Сеть может доставлять мегабайты в секунду и иметь очень небольшую величину RTT, но если на ней изредка встречаются достаточно большие задержки, это вынуждает использовать большие значения для MaxRTT. А поскольку при хорошем тестировании проверяется много портов, из которых подавляющее большинство — закрыты, время сканирования может быть равно почти MaxRTT * numports. При таком достаточно длительном времени тестирования высока вероятность появления постороннего трафика, который может сбить результаты тестирования. Отсюда правило — сканировать нужно быстро.

Хорошим решением было бы одновременное сканирование нескольких портов. Но атакующий не знает, ответные пакеты с каких портов вызвали приращение IP ID, но знает величину этого приращения. Если при одновременном сканировании нескольких портов на каждый отсылать разное количество пакетов (по степеням двойки), то по итоговому приращению IP ID можно легко представить, какие порты открыты. Для примера рассмотрим ситуацию, когда одновременно сканируются порты 21,22,25 и 80. Отправляется 1 пакет на 21 порт, 2 на 22, 4 на 25 и 8 на 80. Всего 15 пакетов. Допустим, итоговое приращение составило 7. 7=1+2+4. Делаем вывод, что открыты порты 21,22,25, а 80-ый — закрыт.

В принципе, 2^ можно считать более общей технологией, так как при одновременном сканировании одного порта она вырождается в канонический

вариант IP ID-сканирования. Надежность метода: лишний или потерянный пакет вносят ошибку как при обычном сканировании, так и при "2 в степени" его модификации. Но в последнем случае ошибка несет больший дефект (если бы вместо правильных 7 в приращение оказалось равным 8, результаты были бы абсолютно противоположными). Для надежности рекомендуется повторить сканирование, желательно использовав другие количества пакетов для портов, чем первоначально. Одновременно можно тестировать и больше портов, но следует знать меру — одновременное тестирование 10 портов потребует послать 1023 (очень небольших) пакета. Для ста портов это число становится уже совершенно неприемлемым.

Неприятности от мелких потерь или 'находок' пакетов атакующий может обойти, если одинаково в несколько раз увеличит количество сканирующих пакетов на каждый порт (e.g. 1,2,4,8 -> 5,10,20,40).

Скрытное слежение за изменением IP ID.

Трафик, который происходит между *dumb* и атакующим во время сканирования по классической IP ID-схеме, имеет две неприятных для взломщика особенности: Он не специфичен ни для какой 'легальной' деятельности и вызывает сильные подозрения. Он содержит реальный адрес атакующего, и достаточно квалифицированный специалист по защите информации, способен сконвертировать эти свои подозрения во вполне конкретный адрес взломщика.

Существует возможность для взломщика выполнять слежение за IP ID без генерации такого подозрительного трафика. Все что нужно взломщику — это как раз то, что *dumb* сам отсылает в каждом IP-пакете! По этому следящий трафик можно замаскировать под любой другой, например, под перекачку файлов по ftp. Технология сканирования при скрытном слежении за IP ID:

На атакующем ведется запись (желательно со временем) всех приходящих с *dumb* IP-пакетов (например, с помощью ethereal или tcpdump). Интересует только ID-поле. Запоминается IP ID в первом пакете (IPID First).

Атакующий инициирует какой-либо трафик с *dumb*, например, начинает выкачивание файла через ftp. Выполняется проверка портов *victim* (рассмотрим вариант с использованием "2^" метода).

Через некоторое время (MaxRTT) фиксируется текущее значение IP ID (IPID Last) и общее количество полученных пакетов (от First до Last включительно) — NumPackets.

По простой формуле NumReplies = IPID_Last — IPID_First — NumPackets +1 получаем инкремент IP ID, который был вызван сканированием. По нему, как описывалось выше, можно получить информацию по состоянию портов. После проверки первой группы портов можно перейти к следующей, сбросив NumPackets и заново переинициализировав IPID First. При этом следящий трафик можно не прерывать.

В результате на *dumb* хоть и присутствует трафик с реальным адресом атакующего, он совершенно легален и сам по себе не вызывает подозрений ни у систем обнаружения вторжений, ни у экспертов. И хоть все равно

обнаружить корреляцию между скачиванием файла и сканированием удаленной системы в принципе возможно, "Никто не может быть осужден за скачивание публичного файла".

уязвимые системы и способы проверки

Для простой проверки можно использовать :

```
hping2 <DUMB> -r
```

Ответы с постоянным приращением id на 1:

```
   len=46 ip=<DUMB> flags=RA seq=0 ttl=128 id=23994 win=0
rtt=1.4 ms
   len=46 ip=<DUMB> flags=RA seq=1 ttl=128 id=+1 win=0 rtt=0.3
ms
   len=46 ip=<DUMB> flags=RA seq=2 ttl=128 id=+1 win=0 rtt=0.3
ms
   len=46 ip=<DUMB> flags=RA seq=3 ttl=128 id=+1 win=0 rtt=0.3
ms
```

говорят об уязвимости системы (см. ниже.).

Для прохода за файервол иногда может иметь смысл указание точных значений порта и флагов (по умолчанию hping2 посылает на 0-ой порт). e.g. Если система имеет открытый www-сервер, имеет смысл указать опции -S и -p 80, тогда это будет абсолютно нормальным обращением к www-серверу с точки зрения правил файервола. Поскольку RFC позволяет системам использовать разные значения IP ID для разных соединений, для большей точности стоит проверить, что трафик по другому соединению влияет не IP ID первого соединения. Можно для проверки протестировать несколько портов, статус которых заранее известен. Вот список из нескольких проверенных систем, в моем случае результаты были такими:

```
Linux kernel 2.4.2, 2.4.18 - NOT vulnerable.
Linux kernel 2.2.17 - vulnerable
FreeBSD 4.6-RC - NOT vulnerable
Windows 2000 Server - vulnerable
Windows XP - vulnerable
Cisco PIX Firewall 525 ver.  6.1(2)- vulnerable
Cisco Content Service Switch (CSS 11150) - vulnerable
Cisco router 2600, 3600 - vulnerable
```

Все тесты проводились на открытых портах.

сетевые характеристики атаки

трафик на victim Трафик на *victim* при сканировании с использованием хоста *dumb*, порта 80. Сканировался открытый порт 22 (ssh).

```
   0.489813 dumb -> victim TCP www > ssh [SYN] Seq=3375705111
Ack=0 Win=2048 Len=0
   0.489880 victim -> dumb TCP ssh > www [SYN, ACK] Seq=2194740763
Ack=3375705112 Win=5840 Len=0
   0.490013 dumb -> victim TCP www > ssh [RST] Seq=3375705112
Ack=3375705112 Win=0 Len=0
```

В следующем примере сканировался закрытый порт 23 telnet.

```
   23.097869 dumb -> victim TCP www > telnet [SYN] Seq=619834409
Ack=0 Win=3072 Len=0
   23.097911 victim -> dumb TCP telnet > www [RST, ACK] Seq=0
Ack=619834410 Win=0 Len=0
```

Как видим, нигде не встречается адрес атакующего, а картина аналогична обычному SYN-сканированию. IDS на *victim* работает так же, как и в случае с SYN-сканированием. Невозможно по этому трафику определить, является ли это *dumb* сканом или нет.

трафик на dumb Во-первых, во время атаки на системе *dumb* наблюдается периодический трафик с атакующим, необходимый ему для слежения за изменениями IP ID, во-вторых, иногда приходят сообщения от *victim* (RST либо SYN|ACK в случае TCP-сканирования).

Как IDS в сети *dumb* может обнаружить использование защищаемой машины в качестве релея? Думается, разумно будет считать признаком атаки наличие одновременно следующих факторов:

- трафика вида [victim: SYN|ACK, dumb: RST] и [victim: RST] при отсутствие иного трафика между этими системами. Это говорит о том, что машина victim получает TCP SYN|ACK-пакеты с поддельным *source addr* = *dumb*.
- любого IP-трафика с какой-либо одной машиной (считаемой за атакующего). (т.к. это может быть слежением за IP ID на *dumb*)

Пассивное наблюдение за *dumb* (во время сканирования) показывает, что эта машина уязвима для атаки и во время сканирования ведет себя соответственно — монотонно увеличивая IP ID с каждым отправленным пакетом.

Кроме того, для 2^-модификации первый фактор будет дополнительно характеризоваться тем, что количество пакетов с каждого порта dumb будет являться степенью двойки.

Язык описания сигнатур атак простых IDS типа snort не позволяет обнаруживать такого рода сигнатуру, так что для этого нужно использовать препроцессоры на C. Используемый в NFR IDS язык NCode, насколько я знаю, достаточно гибок для этого. Замечу также, что это НЕ идеальная характеристика атаки, и имеет свои способы обхода и свои false positives.

способы противодействия атаке

Для системы безопасности сети, в которой находится *dumb*, возможны несколько видов реакции: закрытие на файерволе трафика с подозреваемым в атаке. К сожалению, это очень опасная мера, так как возможно сымитировать ситуацию, на которую сработает триггер IDS и невиновный хост окажется заблокированным. установка хорошего stateful файервола, который не пропустит неожидаемый трафик с *victim* на *dumb*, так что изменение IP ID последнего не будет представлять интереса для взломщика.

Добавление в файервол функции изменения IP ID исходящих пакетов. В простом случае можно тривиально затирать их константой (желательно с установлением флага DF), но тогда мы теряем возможность фрагментации. В более серьезном случае файрвол должен сам иметь свой надежный алгоритм генерации IP ID и присваивать пакетам значения из этого генератора.

Для атакуемой сети: Обычные действия специфичные для SYN-сканирования. Увы, закрыть источник атаки (*dumb*) опять же нельзя, так как атаку легко сымитировать. Один из вариантов защиты от SYN-сканирования — на ВСЕ SYN-пакеты отвечать SYN-ACK'ом и при приходе следующего ACK от удаленной стороны устанавливать соединение (если порт на самом деле открыт) или высылать (не обязательно) RST-пакет, если он закрыт (насколько я знаю, эта технология используется). В таком случае перебор портов, который может быть портсканом, можно осуществить только с настоящего адреса, и блокирование его на файрволе менее опасно.

Можно вместе с каждым RST, оправляемым при попытке установления соединения на закрытом порту, отправлять icmp echo request или другой пакет, который бы вызвал исходящий пакет с *dumb*. В этом случае IP ID на *dumb* будет увеличиваться вне зависимости от того, открыт порт на *victim* или нет.