DNS Amplification (DNS усиление)

6 мин **96K**

Информационная безопасность*

Не так давно столкнулся с проблемой (и ее решением) учитывая актуальность этой темы в последнее время, а также то, сколько людей сейчас страдают от этой беды, решил объединить информацию в одну статью. Может быть кому-то еще она будет полезной.
Начало

Пару недель назад я заметил странную активность, направленную на мой DNS-сервер. Сразу скажу, что использую шлюз на Linux, соответственно там установлен DNS-сервер bind. Активность заключалась в том, что на порт 53 (DNS) моего сервера сыпалось по несколько UDP пакетов в секунду с различных IP-адресов:

10:41:42.163334 IP 89.149.221.182.52264 > MY_IP.53: 22912+ NS?. (17)
10:41:42.163807 IP MY_IP.53 > 89.149.221.182.52264: 22912 Refused- 0/0/0 (17)

На что, как видно из лога, сервер отвечал отказами. Естественно мне стало интересно, что за IP-адреса долбят мой DNS. Посмотрев несколько адресов через whois я определил, что это крупные хостинговые компании, я написал просьбу прекратить атаку на мой сервер в техподдержку некоторых из них. В ответ я получил отписку, что этот тип атаки относится к тем, что они не могут блокировать, и что они сами они страдают от этой аномальной активности. Было решено со всем разбираться самому.

DNS Amplification (DNS усиление). Теория

Сам тип атаки не нов — о нем было известно еще в 2006 году, подробности на английском языке можно посмотреть здесь , однако активно использовать его начали относительно недавно. В январе-феврале 2009 года несколько интернет-изданий опубликовало информацию о крупномасштабном использовании киберпреступниками этого вида DDOS, о чем можно узнать здесь и на английском языке здесь
Объясняя простым языком, суть усиления заключается в том, что злоумышленник посылает (обычно короткий) запрос уязвимому DNS-серверу, который отвечает на запрос уже значительно большим по размеру пакетом. Если использовать в качестве исходного IP-адреса при отправке запроса адрес компьютера жертвы (ip spoofing), то уязвимый DNS-сервер будет посылать в большом количестве ненужные пакеты компьютеру-жертве, пока полностью не парализует его работу.

Практика

Наиболее эффективен этот тип атаки на старом (непропатченном) или неправильно сконфигурированном DNS-сервере, который, как уже было сказано, отвечает на короткие запросы злоумышленников большими по размеру пакетами.
Вот пример такого взаимодействия (кстати именно такие запросы чаще всего используют атакующие):
Отправляем серверу NS запрос командой

```
#dig @SERVER_IP NS
```

где SERVER_IP — IP-адрес сервера. В результате в логе по 53 порту сервера получаем:
11:08:47.994604 IP MY_IP.47816 > SERVER_IP.53: 5655+ NS?. (17)
т.е. как раз те 17 байт запроса, что мы хотели послать.
В ответ в то же самом логе видим:
11:08:47.995853 IP 192.168.100.254.53 > 192.168.100.100.47816: 5655 13/0/6 NS C.ROOT-SERVERS.NET.,[|domain]
т.е. сервер ответил нам подсказкой в виде адресов корневых DNS-серверов, что составляет уже 360 байт. Это длины DNS запроса и ответа, общая же длина пакетов 60 и 402 байта соответственно. Усиление налицо.

Что делать?

Во-первых конечно же проверить актуальность версии вашего DNS-сервера вне зависимости от того, на какой платформе он работает. Во-вторых, убедиться, что настроен сервер достаточно безопасно и не отвечает на «левые» запросы всем подряд. Об этом в сети можно найти множество мануалов и рекомендаций, упомяну здесь только об одном документе, который нашел не так давно.

Что еще можно сделать?

В моем случае делать было уже особенно нечего. Если посмотреть самый первый лог, который я привел, то видно, что атакующий отправляет запрос длиной 17 байт и получает REJECT той же самой длины (17 байт), т.е. никакого усиления не происходит. Но, видимо, «ddos-еры» особенно не торопятся убирать из своих списков адреса DNS-серверов, не подверженных этой уязвимости, и продолжают беспокоить их своей долбежкой… Эта ситуация меня не устраивала. Неприятно что от моего адреса кто-то получает на свой сервер ненужный трафик (даже пусть я в этом и не виноват).
Поначалу я начал ставить в black-лист адреса отправителей, но не тут-то было, со старых адресов атака прекращалась, но появлялись все новые и новые. Было решено использовать более изощренные методы фильтрации и задействовать на моем Linux-сервере модули iptables, до которых раньше у меня никак руки не доходили. Убить, так сказать, сразу двух зайцев — и атакующим сделать -1 и разобраться с парой модулей iptables.

Модуль String

Закрыть 53 порт (DNS) полностью я конечно же не мог — у меня много клиентов которым он нужен. Я решил фильтровать пакеты по содержимому DNS-запросов и убирать те из них, которые содержат запросы атакующих, а они все как один содержали в себе «NS». Для этой задачи подходит модуль iptables string, который как раз позволяет заглянуть в содержимое пакетов.
Для того, чтобы понять что фильтровать, посмотрим на пакет атакующего через wireshark. Здесь и здесь можно почитать о структуре UDP пакетов и формате кадра DNS соответственно. Для краткости скажу, что первые 14 байт пакета занимают данные протокола Ethernet (на рис. красным), затем идут 20 байт заголовка протокола IP (на рис. синим), затем 8 байт — заголовок UDP (на рис. зеленым), после которого начинаются данные протокола DNS (на рис. желтым). Первые 12 байт кадра DNS занимает заголовок, после которого, наконец, начинается поле DNS Query (т.е. непосредственно сам запрос, на рис. оранжевым). В пакетах, присылаемых атакующим начиная с 54-ого (14+20+8+12) байта идут следующие данные: 00 00 02 00 01 (в шестнадцатеричном коде), что соответствует запросу «NS», о котором я говорил раньше. Таким образом нужно выделить пакеты, которые начиная с 54 байта содержат эти байты. Это будет выглядеть так:

```
 iptables -A INPUT --in-interface eth1 --protocol udp --dport 53 --match state --state NEW --match string --algo kmp --hex-string "|00 00 02 00 01|" --from 40 --to 45 --jump DROP
```

Немного поясню.
--in-interface — указывает на каком интерфейсе отлавливать пакеты. Нужно поставить только внешний интерфейс, на который идет атака (незачем ущемлять пользователей во внутри сети).
--match state --state NEW — отлавливаем пакеты только со состоянием NEW, чтобы не проверять все транзакции подряд, а только инициирующие пакеты (мало ли что может передаваться по 53 порту).
Дальше идет самое интересное — задействуем модуль sting. Мы используем следующие параметры:
--algo — указывает алгоритм поиска, по сути не важен я указал kmp, но можно указать и bm;
--hex-string — записывается та самая строка в шестнадцатеричном виде, которую мы ищем;
--from 40 — ищем начиная с 40 байта (заметьте, не с 54 потому что string начинает поиск, а соответственно и отсчет от первого байта протокола IP, т.е. выбрасывается протокол Ethernet, длина которого 14 байт(на рис. сверху красным). Итого 54 — 14 = 40);
--to 45 — соответственно искать до 45 байта пакета.

Модуль Recent

На этом уже можно было бы остановиться. Пакеты уже не будут доходить до bind, но меня еще не утешала мысль, что я закрыл для BCEX возможность обращаться с запросами NS к моему DNS-серверу, поэтому я решил задействовать еще один модуль iptables — recent. Этот модуль позволяет создавать динамические таблицы IP-адресов в зависимости от определенных условий, а затем устанавливать разрешающие и запрещающие правила для этих таблиц.
Рассмотрим простой пример использования recent, состоящий из двух строк:

```
 iptables -A INPUT --protocol tcp --match state --state NEW --dport 22 --match recent --update --seconds 30 --name SSHT --jump DROP

iptables -A INPUT --protocol tcp --match state --state NEW --dport 22 --match recent --set --name SSHT --jump ACCEPT
```

Начнем разбираться со второго правила. Каждый кто пытается зайти (именно зайти, т.к. мы используем --state NEW) на порт 22 (SSH) пропускается (--jump ACCEPT), но его IP-адрес попадает в таблицу с именем SSHT. Когда с этого адреса пытаются соединиться еще раз, начинает работать первое правило, смысл которого состоит в том, чтобы обновить (--update) запись в таблице и заодно проверить когда эта запись была установлена/обновлена в последний раз. Если запись была установлена/обновлена меньше 30 секунд назад (--seconds 30), то срабатывает --jump DROP и пакет отбрасывается. Таким образом брутфорсеры, пытающиеся долбиться на порт SSH будут отбрасывается, если частота отправки их пакетов будет превышать 1 пакет в 30 секунд.
Попробуем использовать recent для наших нужд:

```
 iptables -A INPUT --in-interface eth1 --protocol udp --dport 53 --match state --state NEW --match string --algo kmp --hex-string "|00 00 02 00 01|" --from 40 --to 45 --match recent --name DNST --update --seconds 600 --jump DROP

iptables -A INPUT --in-interface eth1 --protocol udp --dport 53 --match state --state NEW --match string --algo kmp --hex-string "|00 00 02 00 01|" --from 40 --to 45 --match recent --name DNST --set --jump ACCEPT
```

Таким образом я разрешаю делать запросы NS на внешний интерфейс не чаще, чем 1 раз в 10 минут.

После добавления этих правил в /proc/net/xt_recent моего сервера появился файл DNST, в котором начали записываться IP-адреса атакующих. А DNS-сервер перестал поддаваться на провокации:

```
14:10:19.011818 IP 89.149.221.182.40320 > MY_IP.53: 23508+ NS?. (17)
14:10:25.243499 IP 89.149.221.182.64984 > MY_IP.53: 25306+ NS?. (17)
14:11:08.832827 IP 89.149.221.182.15864 > MY_IP.53: 48029+ NS?. (17)
14:11:15.063058 IP 89.149.221.182.30959 > MY_IP.53: 64444+ NS?. (17)
```

Через несколько дней работы правил количество пакетов со стороны атакующих снизилось в несколько раз. Сейчас я получаю 2-3 пакета в минуту, которые тут же отбрасываются фаерволом.

UPD: В последнем блоке кода поторопился и написал ошибку, уже исправил, спасибо, что заметили

Теги: dnsddosdns amplificationiptablesmod stringmod recentlinuxfirewall

Хабы: Информационная безопасность