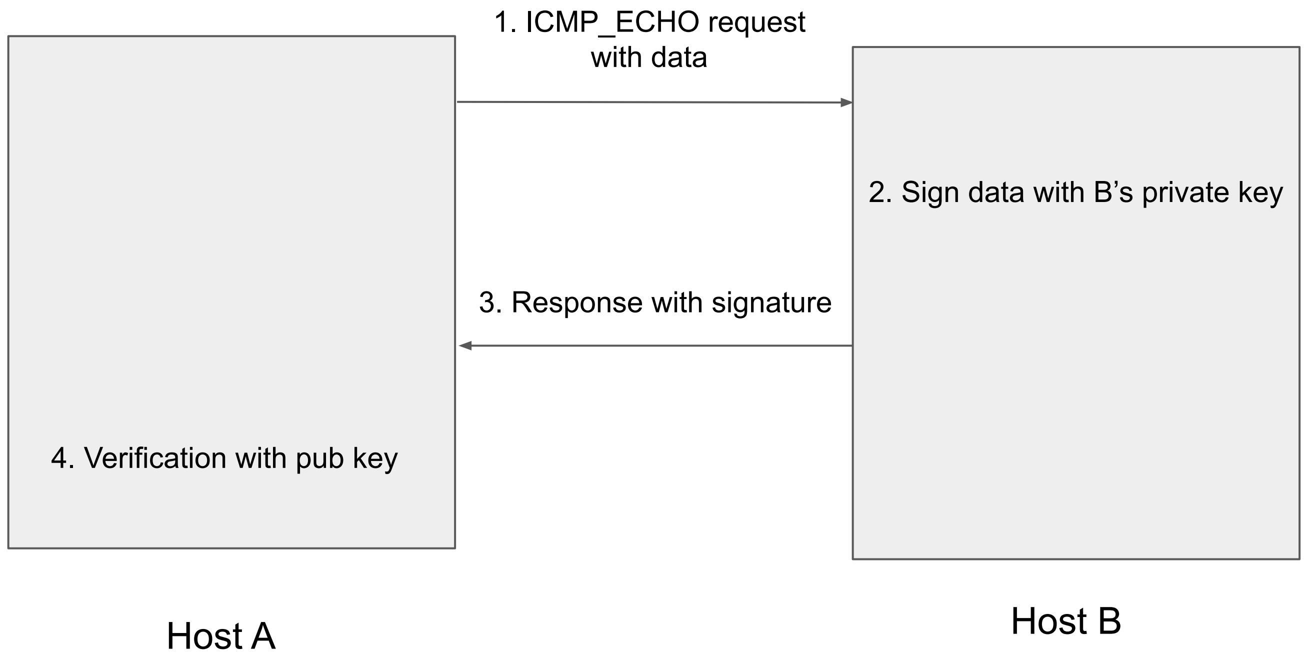
Enhancing ICMP Protocol with Public-Key Signature Verification

2022-20428 이영주

Motivation

- Threats
  - Denial of Service (Ping Flood, Ping of Death)
  - Smurf Attack : forge the source ip to invoke the DoS attack
  - Network Reconnaissance
  - Data Exfiltration : use it as a cover channel
- Goal
  - Adding authentication to the ICMP_ECHO request.

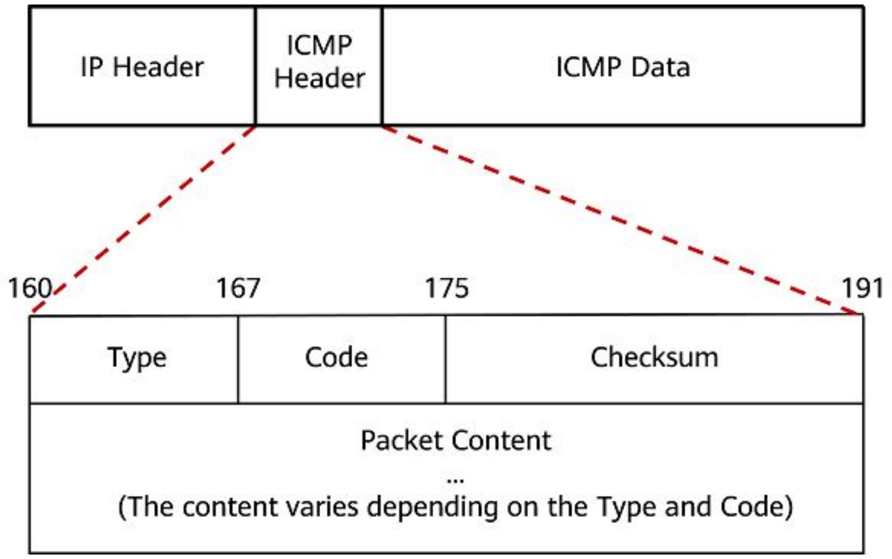
Design


1. ICMP_ECHO request with data
2. Sign data with B’s private key
3. Response with signature
4. Verification with pub key
Host A
Host B

Implementation

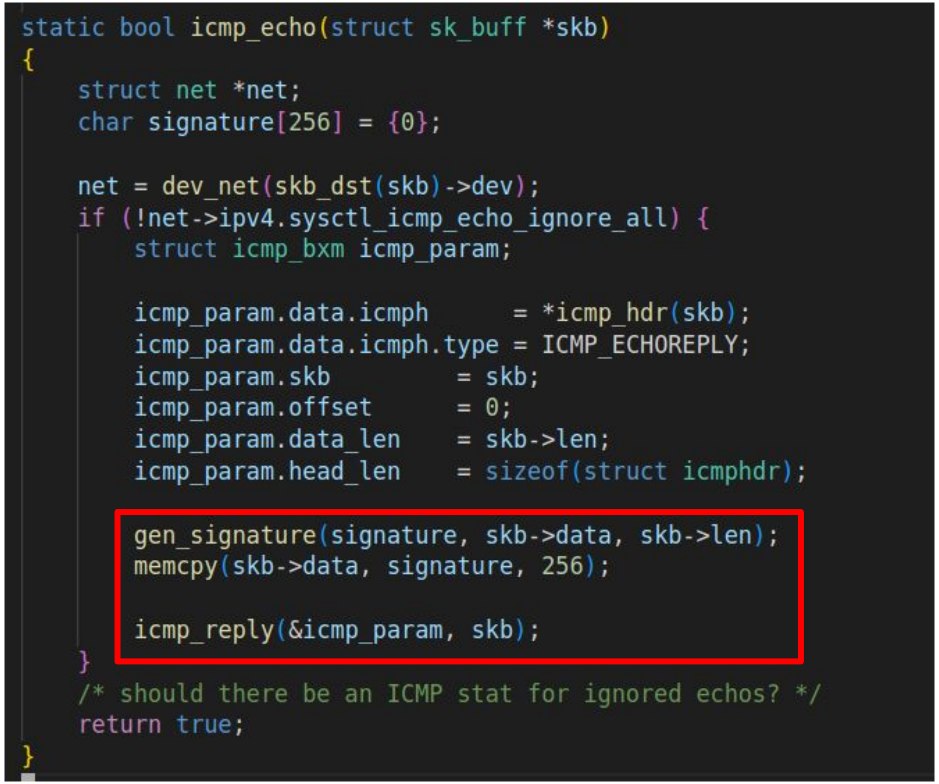
- Environment
  - qemu
  - linux kernel 6.3

- Modification
  - Use ICMP Data section to send a certificate.



Implementation

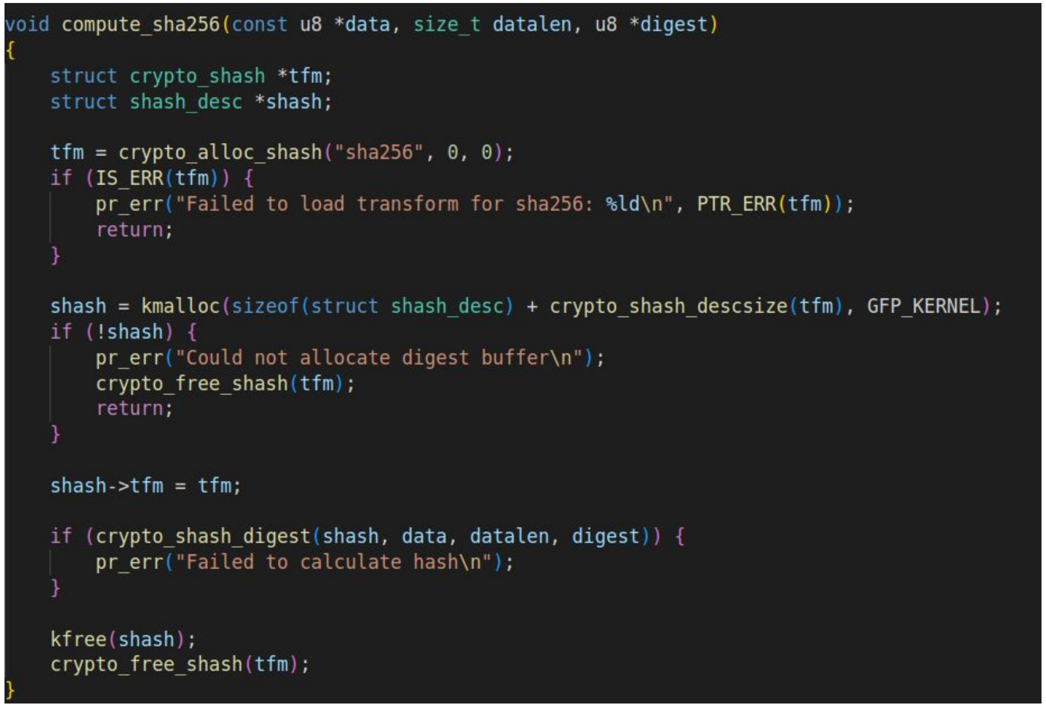
- Modification
    - /net/ipv4/icmp.c



```
static bool icmp_echo(struct sk_buff *skb)
{
    struct net *net;
    char signature[256] = {0};

    net = dev_net(skb_dst(skb)->dev);
    if (!net->ipv4.sysctl_icmp_echo_ignore_all) {
        struct icmp_bxm icmp_param;

        icmp_param.data.icmph      = *icmp_hdr(skb);
        icmp_param.data.icmph.type = ICMP_ECHOREPLY;
        icmp_param.skb             = skb;
        icmp_param.offset          = 0;
        icmp_param.data_len        = skb->len;
        icmp_param.head_len        = sizeof(struct icmphdr);

        gen_signature(signature, skb->data, skb->len);
        memcpy(skb->data, signature, 256);

        icmp_reply(&icmp_param, skb);
    }
    /* should there be an ICMP stat for ignored echos? */
    return true;
}
```

Implementation

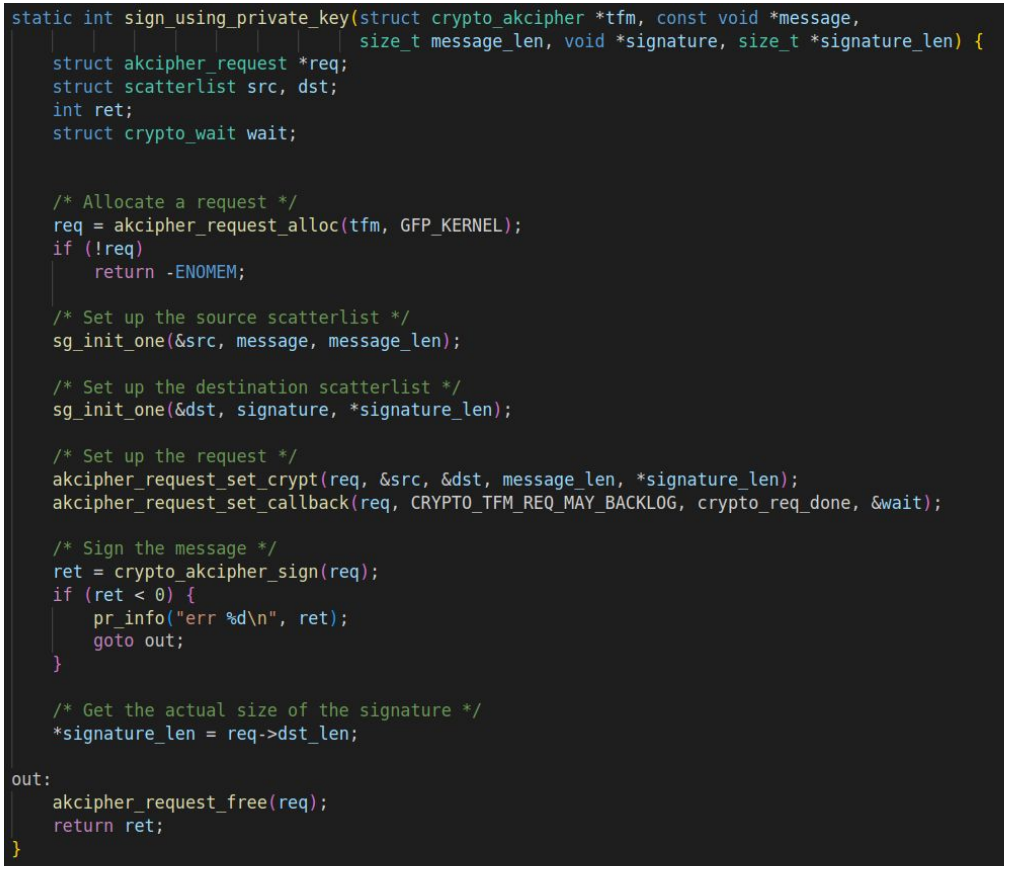
- Hashing
  - sha256
  - Linux kernel crypto api



```c
void compute_sha256(const u8 *data, size_t datalen, u8 *digest)
{
    struct crypto_shash *tfm;
    struct shash_desc *shash;

    tfm = crypto_alloc_shash("sha256", 0, 0);
    if (IS_ERR(tfm)) {
        pr_err("Failed to load transform for sha256: %ld\n", PTR_ERR(tfm));
        return;
    }

    shash = kmalloc(sizeof(struct shash_desc) + crypto_shash_descsize(tfm), GFP_KERNEL);
    if (!shash) {
        pr_err("Could not allocate digest buffer\n");
        crypto_free_shash(tfm);
        return;
    }

    shash->tfm = tfm;

    if (crypto_shash_digest(shash, data, datalen, digest)) {
        pr_err("Failed to calculate hash\n");
    }

    kfree(shash);
    crypto_free_shash(tfm);
}
```

Implementation

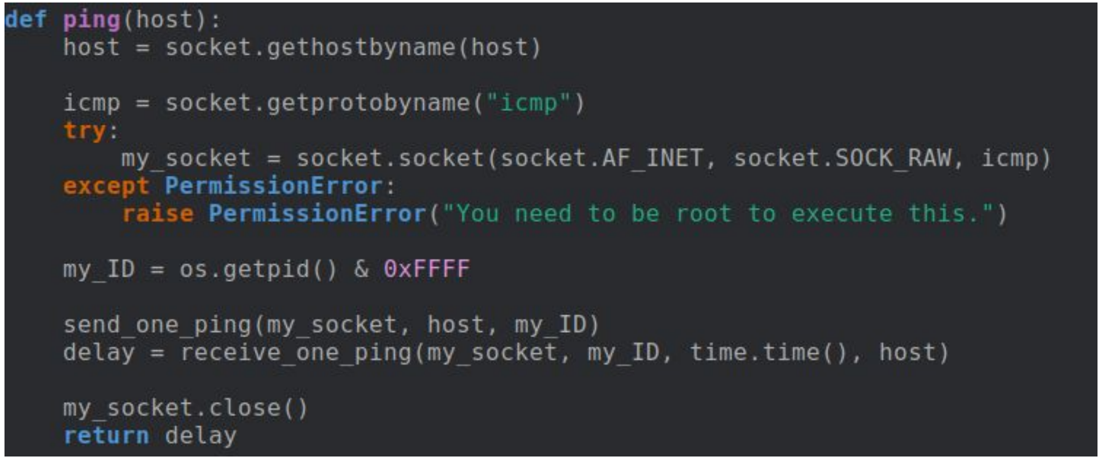
- Signing
  - RSA (pkcs1pad/sha256)
  - Linux kernel crypto api



```c
static int sign_using_private_key(struct crypto_akcipher *tfm, const void *message,
                                  size_t message_len, void *signature, size_t *signature_len) {
    struct akcipher_request *req;
    struct scatterlist src, dst;
    int ret;
    struct crypto_wait wait;


    /* Allocate a request */
    req = akcipher_request_alloc(tfm, GFP_KERNEL);
    if (!req)
        return -ENOMEM;

    /* Set up the source scatterlist */
    sg_init_one(&src, message, message_len);

    /* Set up the destination scatterlist */
    sg_init_one(&dst, signature, *signature_len);

    /* Set up the request */
    akcipher_request_set_crypt(req, &src, &dst, message_len, *signature_len);
    akcipher_request_set_callback(req, CRYPTO_TFM_REQ_MAY_BACKLOG, crypto_req_done, &wait);

    /* Sign the message */
    ret = crypto_akcipher_sign(req);
    if (ret < 0) {
        pr_info("err %d\n", ret);
        goto out;
    }

    /* Get the actual size of the signature */
    *signature_len = req->dst_len;

out:
    akcipher_request_free(req);
    return ret;
}
```

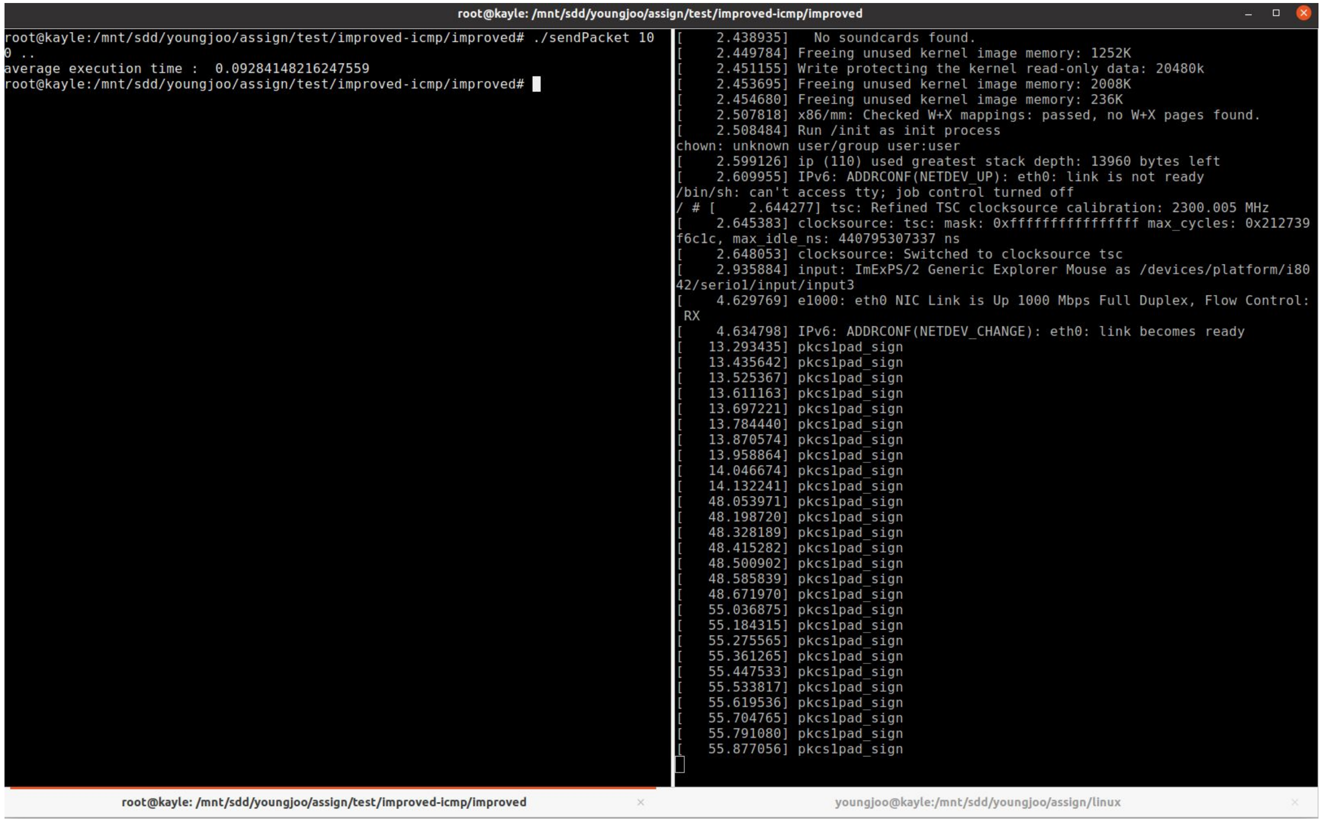
Implementation

- Sending Packet
    - Raw socket



```
def ping(host):
    host = socket.gethostbyname(host)

    icmp = socket.getprotobyname("icmp")
    try:
        my_socket = socket.socket(socket.AF_INET, socket.SOCK_RAW, icmp)
    except PermissionError:
        raise PermissionError("You need to be root to execute this.")

    my_ID = os.getpid() & 0xFFFF

    send_one_ping(my_socket, host, my_ID)
    delay = receive_one_ping(my_socket, my_ID, time.time(), host)

    my_socket.close()
    return delay
```

Evaluation


root@kayle: /mnt/sdd/youngjoo/assign/test/improved-icmp/improved
root@kayle: /mnt/sdd/youngjoo/assign/test/improved-icmp/improved
youngjoo@kayle:/mnt/sdd/youngjoo/assign/linux

```
root@kayle:/mnt/sdd/youngjoo/assign/test/improved-icmp/improved# ./sendPacket 10
0 ..
average execution time :  0.09284148216247559
root@kayle:/mnt/sdd/youngjoo/assign/test/improved-icmp/improved# 
```

```
[    2.438935]   No soundcards found.
[    2.449784] Freeing unused kernel image memory: 1252K
[    2.451155] Write protecting the kernel read-only data: 20480k
[    2.453695] Freeing unused kernel image memory: 2008K
[    2.454680] Freeing unused kernel image memory: 236K
[    2.507818] x86/mm: Checked W+X mappings: passed, no W+X pages found.
[    2.508484] Run /init as init process
chown: unknown user/group user:user
[    2.599126] ip (110) used greatest stack depth: 13960 bytes left
[    2.609955] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
/bin/sh: can't access tty; job control turned off
/ # [    2.644277] tsc: Refined TSC clocksource calibration: 2300.005 MHz
[    2.645383] clocksource: tsc: mask: 0xffffffffffffffff max_cycles: 0x212739
f6c1c, max_idle_ns: 440795307337 ns
[    2.648053] clocksource: Switched to clocksource tsc
[    2.935884] input: ImExPS/2 Generic Explorer Mouse as /devices/platform/i80
42/serio1/input/input3
[    4.629769] e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
 RX
[    4.634798] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[   13.293435] pkcs1pad_sign
[   13.435642] pkcs1pad_sign
[   13.525367] pkcs1pad_sign
[   13.611163] pkcs1pad_sign
[   13.697221] pkcs1pad_sign
[   13.784440] pkcs1pad_sign
[   13.870574] pkcs1pad_sign
[   13.958864] pkcs1pad_sign
[   14.046674] pkcs1pad_sign
[   14.132241] pkcs1pad_sign
[   48.053971] pkcs1pad_sign
[   48.198720] pkcs1pad_sign
[   48.328189] pkcs1pad_sign
[   48.415282] pkcs1pad_sign
[   48.500902] pkcs1pad_sign
[   48.585839] pkcs1pad_sign
[   48.671970] pkcs1pad_sign
[   55.036875] pkcs1pad_sign
[   55.184315] pkcs1pad_sign
[   55.275565] pkcs1pad_sign
[   55.361265] pkcs1pad_sign
[   55.447533] pkcs1pad_sign
[   55.533817] pkcs1pad_sign
[   55.619536] pkcs1pad_sign
[   55.704765] pkcs1pad_sign
[   55.791080] pkcs1pad_sign
[   55.877056] pkcs1pad_sign
```

Evaluation

- Delay
    - without verification : 2 ms per request and reply
    - with verification : 867 ms per request and reply
    - 433.5x slower..

- Why?
    - hashing the data (data to sha256 hash)
    - signing the data
    - verifying the data

Evaluation

- Added Packet Length
  - variable bytes (to sign) + 256 bytes (signature length)