

Native HIP API

Miika Komu <miika@iki.fi>

Andrei Gurtov <gurtov@cs.helsinki.fi>

Helsinki Institute for Information Technology

About Host Identity Protocol (HIP)

- HIP base exchange is an authenticated Diffie-Hellman key exchange
- Extensions include rendezvous, mobility and multihoming support
- Introduces a new wedge layer between transport and network layers
- The difference to other key exchange protocols: introduces a cryptographic namespace transport layer and application layers

Terminology

- HI = Host Identifier
- HIT = Host Identity Tag
- LSI = Local Scope Identifier
- Resolver = maps host names to addresses

API Overview

Application Layer	Application		
Socket Layer	IPv4 API	IPv6 API	HIP API
Transport Layer	TCP		UDP
HIP Layer	HIP		
Network Layer	IPv4		IPv6
Link Layer	Ethernet		

Legacy APIs

- Suitable for legacy HIP applications
 - Applications need no or little changes
- The modified resolver gives the application an LSI or HIT instead of an IPv4 or IPv6 address
 - The mapping to the routable IP address is sent to HIP software module
- Connecting directly to a HIT; the mapping to the IP address must be handled by some other means

Native HIP API

- Suitable for new HIP-aware applications
 - Enables application to control the HIP layer better
- Introduces a new socket family: PF_HIP
 - Easy detection of HIP support in the localhost
 - Can be used for communicating user or application specified Host Identifiers
- Introduces a new socket address structure with new identifier: Endpoint Descriptor (ED)
 - Similar to file descriptor, only local significance

Legacy vs. Native HIP API

```
struct addrinfo hints, *res, *try;

char *hello = "hello";

int err, int bytes, sock;
    memset(hints, 0, sizeof(hints));
    hints.ai_flags = AI_HIP;
    hints.ai_family = AF_INET6;
    hints.ai_socktype = SOCK_STREAM;

err = getaddrinfo("www.host.org", "echo",
                  &hints, &res);

sock = socket(res->ai_family, res->ai_socktype,
              res->protocol);

for (try = res; try; try = try->ai_next)
    err = connect(sock, try->ai_addr,
                  try->ai_addrlen);
    bytes = send(sock, hello, strlen(hello), 0);
    bytes = recv(sock, hello, strlen(hello), 0);

err = close(sock);
err = freeaddrinfo(res);
```

```
struct endpointinfo hints, *res, *try;

char *hello = "hello";

int err, int bytes, sock;
    memset(hints, 0, sizeof(hints));
    hints.ei_family = PF_HIP;
    hints.ei_socktype = SOCK_STREAM;

err = getendpointinfo("www.host.org", "echo",
                     &hints, &res);

sock = socket(res->ai_family, res->ai_socktype,
              res->protocol);

for (try = res; try; try = try->ai_next)
    err = connect(sock, try->ai_addr,
                  try->ai_addrlen);
    bytes = send(sock, hello, strlen(hello), 0);
    bytes = recv(sock, hello, strlen(hello), 0);

err = close(sock);
err = freeendpointinfo(res);
```

Application Specified Identifiers

```
int sockfd, err, family = PF_HIP,

    type = SOCK_STREAM;
char *user_priv_key = "/home/mk/hip_host_dsa_key";
struct endpoint *endpoint;
struct sockaddr_ed my_ed;
struct endpointinfo hints, *res = NULL;

err = load_hip_endpoint_pem(user_priv_key, &endpoint);
err = setmyeid(&my_ed, "", endpoint, NULL);
sockfd = socket(family, type, 0);
err = bind(sockfd, (struct sockaddr *) &my_ed, sizeof(my_ed));

memset(&hints, 0, sizeof(&hints));
hints.ei_socktype = type;
hints.ei_family = family;
err = getendpointinfo("www.host.org", "echo", &hints, &res);

/* connect, send and recv normally */
```


Summary

- ED hides the presentation of HITs and HIs
 - Useful e.g. in implementing opportunistic HIP
 - Size of HIT may change in the future
- PF_HIP family can be used e.g. for detecting HIP support in the localhost
- Native HIP API extends HIP architecture by allowing apps to have their own id

References

- Applying a Cryptographic Namespace to Applications [Komu et al]
- draft-mkomu-hip-native-api-00.txt
- Application Programming Interfaces for Host Identity Protocol

Questions / Feedback

Miika Komu <miika@iki.fi>