Name-based sockets

A technical overview
Name-based sockets

Presentation overview

- Goals
- A name-based approach
- API
- Name-exchange
- Features made possible – Name-shim6
- Open issues – name exchange & authentication

November 4, 2010
Goals

Make application development easier
By providing a friendly API
and packaging all that difficult network stuff neatly

Make deployment easier
By using a different deployment strategy
Constraints

Be backwards compatible

Do not require new/changed infrastructure

Do not worsen performance

Be configurable
The approach

Host

- New API
- A session layer
- Easy transition
- All features are opt-in

Network

- No changes
- Compatible
- Middle-boxes unaffected

November 4, 2010
How is this different?

SCTP
HIP
LISP

Application
Transport
Network
Outside the host

Name-based sockets

November 4, 2010
## Uni- / Bi-lateral

<table>
<thead>
<tr>
<th>Uni-lateral</th>
<th>Bi-lateral</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Session</td>
</tr>
<tr>
<td>IPv4 / IPv6 agnosticism</td>
<td>Mobility / multi-homing</td>
</tr>
<tr>
<td>NAT-traversal</td>
<td>IPv4 / IPv6 interop.</td>
</tr>
<tr>
<td>Name exchange</td>
<td>Multi-path exploit.</td>
</tr>
</tbody>
</table>

**FQDN**
- Not required
- Does help reachability

---

November 4, 2010
A time-line example

sock( ..., b.right.net, ...);

API
- IPv4 / IPv6 agnosticism
- NAT-traversal
- Name exchange
- Session
- Mobility / multi-homing
- IPv4 / IPv6 interop.
- Multi-path exploit.
Name-based sockets
Presentation overview (functionality)

- API
  - Current prototype status
- Name-exchange
- Name-shim6
  - A name-based example
API – the vision

Use names

Do away with struct sockaddr

Use service names (not ports)

And of course provide some sensible defaults and a set of setsockopt()s

```c
fd = sock(AF_NAME, dest_name, proto, service);
data = read(fd);
write(fd, data);
```
Names can be provided as either FQDN or as IPs. IPs are encoded as strings (\texttt{<ip>.ip6.arpa})

“The IP with which the session started.”

```c
fd = sock(AF_NAME, SOCK_STREAM, IPPROTO_IP);

struct sockaddr_name bind_name;
strcpy(bind_name->sname_addr.name, src_name);
bind(fd, (struct sockaddr *)&bind_name, sizeof(bind_name));

struct sockaddr_name conn_name;
strcpy(conn_name->name, dst_name);
conn_name.sname_port = htons(port);
connect(fd, (struct sockaddr *)&conn_name, &len);
```
Normal operation

Name-exchange – current prototype

OK!
Bilateral mode
Name-exchange – current prototype

Backward compatibility

a.left.org

b.right.net

Append extension header to following packets until a response is received

Timeout!
Unilateral mode

November 4, 2010
Name-shim6 – current prototype

What names can change

By changing the binding from the upper layers to a name we enable a set of possible enhancements.

We chose to apply a name-based approach to shim6.

Nota bene: Name-shim6 is not a required feature by name-based sockets. It is an example on which kind of features might be part of a name-based socket and how a name-based approach might benefit existing solutions.
Name-shim6 – current prototype

What names can change
Name-shim6 – current prototype

What names can change

Vanilla shim6

ULID = Locator
Out of the hosts control
If this locator becomes Invalid the connection must be shut down

Name-shim6

ULID = Name
Host is in control of the validity of the Name.
Multi-homing → Mobility

November 4, 2010
Name-shim6 – current prototype

What names can change
Name-shim6 – current prototype

If ( name == FQDN )
If the name is resolvable to a locator
the ID → locator substrate can be used
as a rendezvous-point.

DNS is already out there!
Name-exchange
Opt-in-ness

Name-based sockets functions should all be opt-in.

The name exchange and authentication of the ID → locator binding SHOULD be done if and when requested by a feature.
Name-based sockets

Re-iteration

- API
  - Name-exchange
    - Name-shim6
  - When should the names be exchanged?
  - When should name → IP be authenticated?
Name-based sockets

Thank you!