Why netStackGo has better security?

**Isolation**
- Application is provided transparent networking service with encryption, authentication, authorization
- Failure in application will not propagate to network service

**Simplicity**
- Simple and few APIs exposed to programmers
- Less chance for programmers to make mistakes

Why Use netStackGo?

- Simple Syntax: As easy as `connect` and `communicate`
- Easy Administration: Hassle-free configuration
- 0 line of application code for security
- Security through careful abstraction and design
- Security embedded into normal process of networking

Security Properties in Comparison

<table>
<thead>
<tr>
<th>Property</th>
<th>netStackGo</th>
<th>Kerberos</th>
<th>TLS/SSL</th>
<th>PKI/SSH</th>
<th>IPsec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public Key</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Authentication Server</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Protection of Private Key</td>
<td>N/A</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Timeliness</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Availability</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Multi-Organization</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Low Complexity</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Isolation</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Authenticate User</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Mutual Authentication</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

- ✔ = yes; ✗ = no; * = partial; N/A = not applicable

Authentication with Strong Trust Model

- Based on Public Key Cryptography
- Administrator can choose which certificate issuers to trust
- Authenticate with multiple organizations

Low Complexity Networking Primitives

```go
comm ← ipc(service, remoteHost)
```  
- Initiate a connection to specified host and service  
- Daemon retrieves host IP and public key  
- Service is analogy to port number in TCP/UDP  
  
```go
ad ← advertise(service)
```  
- Announce to provide a service  
  
```go
conn, user ← ad.import()
```  
- Accept an incoming connection with remote user id  
  
```go
conn.write(data)
data ← conn.read()
```  
- Write to or read from a connection

Abstraction for Security

1. Bob is authenticated and authorized to his daemon to use `ping` service  
2. Bob requests to ipc to host “ping.example.com”  
3. Bob’s daemon looks up IP and host public key of “ping.example.com” to create an encrypted tunnel  
4. Bob’s daemon mutually authenticates remote host  
5. Bob’s daemon uses his private key for authentication  
6. Alice is authenticated and authorized to her daemon to provide `ping` service  
7. Alice’s daemon authenticates remote user Bob and authorizes him to request `ping` service  
8. Alice imports an authenticated, authorized connection, protected by encrypted tunnel  
9. Bob ends up having a connection authenticated and authorized, protected by encrypted tunnel

Configuration

**User**

Generate user public key pair

**Service**

Create directories `/ipc/ping`, `/import/ping`

**Authorize**

- Add user to group `/ipc/ping` to allow use of `ping`, for access to Unix socket `/ipc/ping`  
- Add user to group `/import/ping` to allow to provide `ping`, for access to Unix socket `/import/ping`

Enforcement of Local Security

- Use Linux identity  
- Daemon authenticates user via Unix socket  
- Only daemon can read host/user private keys  
- Operations done on behalf of user  
- Only specified users can provide/use a service  
- Linux user groups for advertise/ipc separate services

Authentication-related libraries

<table>
<thead>
<tr>
<th>Library</th>
<th>Version</th>
<th>C LoC</th>
<th># of APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenSSH</td>
<td>6.2p1</td>
<td>349,000</td>
<td>N/A</td>
</tr>
<tr>
<td>LibSSH2</td>
<td>1.4.3</td>
<td>296,000</td>
<td>166</td>
</tr>
<tr>
<td>OpenSSL</td>
<td>1.0.1e</td>
<td>274,000</td>
<td>227</td>
</tr>
<tr>
<td>GnuTLS</td>
<td>3.1.9</td>
<td>230,000</td>
<td>641</td>
</tr>
<tr>
<td>NSS</td>
<td>3.14.3</td>
<td>586,000</td>
<td>742</td>
</tr>
<tr>
<td>Kerberos v5</td>
<td>1.11.2</td>
<td>306,000</td>
<td>194</td>
</tr>
<tr>
<td>netStackGo</td>
<td>37.000</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

*major required libraries included in LoC

Due to high complexity, even widely used software written by experts suffer from mis-authentication.

Conclusion

netStackGo provides secure networking by lower complexity and strong trust model in authentication.