11.9 简单的客户端认证¶

问题¶

你想在分布式系统中实现一个简单的客户端连接认证功能，又不想像SSL那样的复杂。

解决方案¶

可以利用 `hmac` 模块实现一个连接握手，从而实现一个简单而高效的认证过程。下面是代码示例：

```
import hmac
import os

def client_authenticate(connection, secret_key):
    '''
    Authenticate client to a remote service.
    connection represents a network connection.
    secret_key is a key known only to both client/server.
    '''
    message = connection.recv(32)
    hash = hmac.new(secret_key, message)
    digest = hash.digest()
    connection.send(digest)

def server_authenticate(connection, secret_key):
    '''
    Request client authentication.
    '''
    message = os.urandom(32)
    connection.send(message)
    hash = hmac.new(secret_key, message)
    digest = hash.digest()
    response = connection.recv(len(digest))
    return hmac.compare_digest(digest,response)
```

基本原理是当连接建立后，服务器给客户端发送一个随机的字节消息（这里例子中使用了 `os.urandom()` 返回值）。客户端和服务器同时利用hmac和一个只有双方知道的密钥来计算出一个加密哈希值。然后客户端将它计算出的摘要发送给服务器， 服务器通过比较这个值和自己计算的是否一致来决定接受或拒绝连接。摘要的比较需要使用 `hmac.compare_digest()` 函数。 使用这个函数可以避免遭到时间分析攻击，不要用简单的比较操作符（==）。 为了使用这些函数，你需要将它集成到已有的网络或消息代码中。例如，对于sockets，服务器代码应该类似下面：

```
from socket import socket, AF_INET, SOCK_STREAM

secret_key = b'peekaboo'
def echo_handler(client_sock):
    if not server_authenticate(client_sock, secret_key):
        client_sock.close()
        return
    while True:

        msg = client_sock.recv(8192)
        if not msg:
            break
        client_sock.sendall(msg)

def echo_server(address):
    s = socket(AF_INET, SOCK_STREAM)
    s.bind(address)
    s.listen(5)
    while True:
        c,a = s.accept()
        echo_handler(c)

echo_server(('', 18000))
```

Within a client, you would do this:

```
from socket import socket, AF_INET, SOCK_STREAM

secret_key = b'peekaboo'

s = socket(AF_INET, SOCK_STREAM)
s.connect(('localhost', 18000))
client_authenticate(s, secret_key)
s.send(b'Hello World')
resp = s.recv(1024)
```

讨论¶

hmac 认证的一个常见使用场景是内部消息通信系统和进程间通信。 例如，如果你编写的系统涉及到一个集群中多个处理器之间的通信， 你可以使用本节方案来确保只有被允许的进程之间才能彼此通信。 事实上，基于 hmac 的认证被 multiprocessing 模块使用来实现子进程直接的通信。

还有一点需要强调的是连接认证和加密是两码事。 认证成功之后的通信消息是以明文形式发送的，任何人只要想监听这个连接线路都能看到消息（尽管双方的密钥不会被传输）。

hmac认证算法基于哈希函数如MD5和SHA-1，关于这个在IETF RFC 2104中有详细介绍。