SE-IOT: Internet of Things

IOT Security

Derek Kiong
dkiong@nus.edu.sg

Security Challenge

- Security objectives remain the same
  - Confidentiality
  - Integrity
  - Availability
  - Non-repudiation
- But IOT security is more challenging
  - Pervasive deployment (fully in hands of users)
  - Increasing connectedness (proned to sniffing)
  - Trust/data integrity
  - Data protection/privacy

Remote Server Attack

- Network services
- Web interface
- API/Cloud/mobile app interfaces
- Authentication/Authorisation

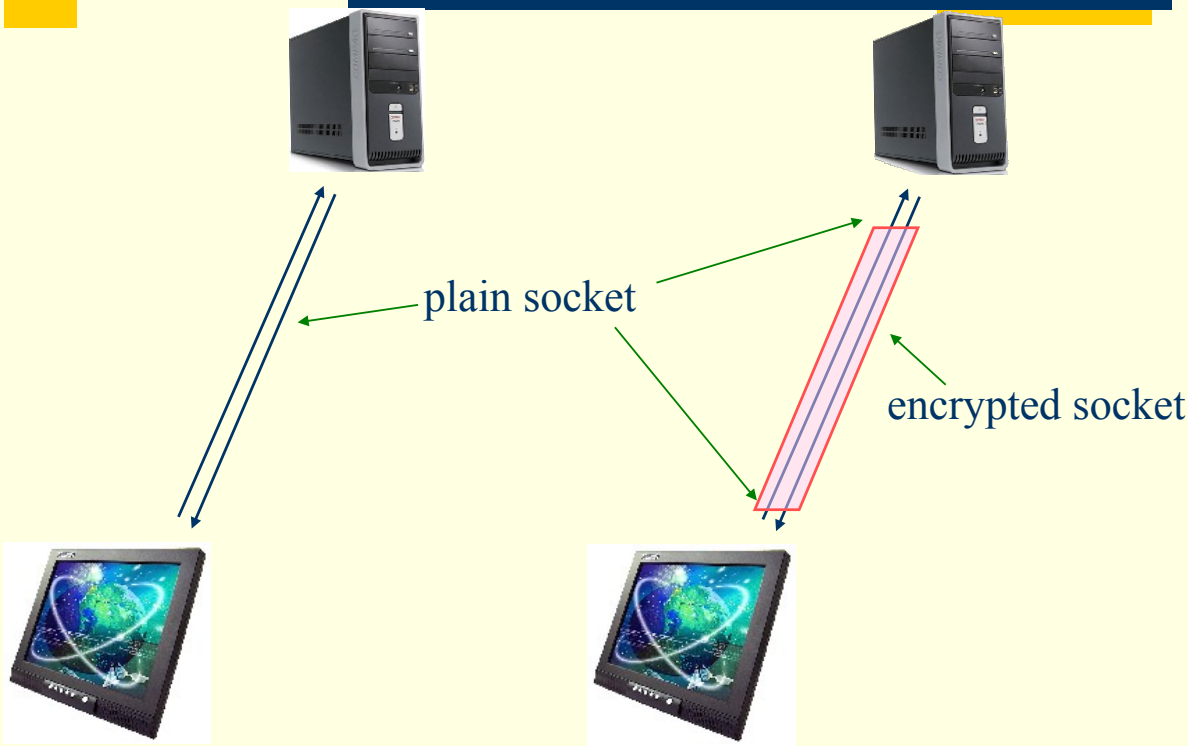
Physical Device Attack

- Hardware sensor
- Device interface (eg, serial/USB ports, memory card)
- Device memory
- Device firmware/update
- Monitor network traffic

Raspberry Pi Defense

- **No** solution to physical attacks
  - file system in SD card is readable
    - unless fully **encrypted** file system
  - Gemalto's Cinterion Secure Element (SE) -- tamper-resistant component
- Use cryptography for confidentiality and authentication protection
  - VPN/tunnel
  - Transport-level security (eg. https/SSL)
  - Application-level security

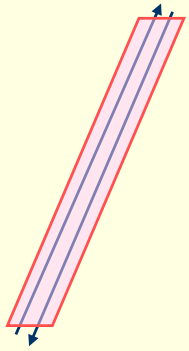
IP Tunnel



X Tunnel

- Remote client provides display – X display server

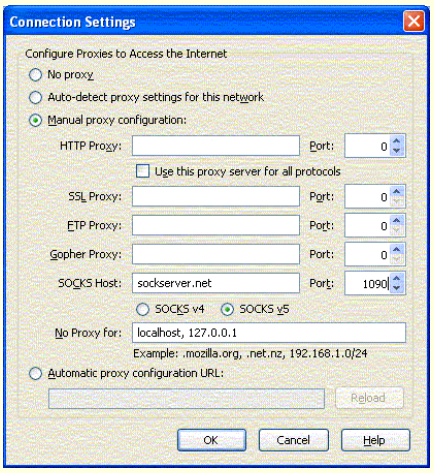
ssh server
server.com



ssh client

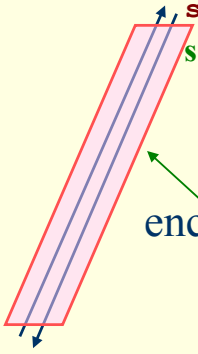
```
$ ssh -X server.com
```


HTTP SOCKS Tunnel



Internet

sshserver.com
ssh server



encrypted socket

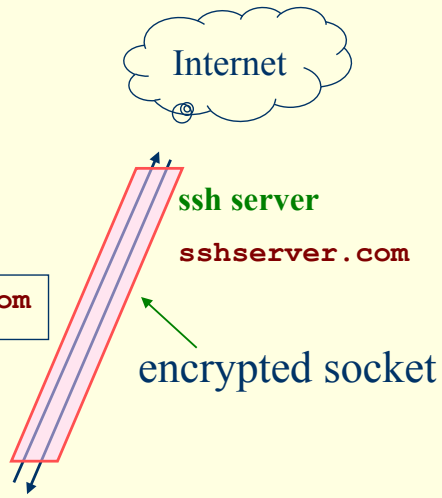
```
$ ssh -D :1090 sshserver.com
```

`ssh` Local Port Forwarding

- Connection to port 1259 of board (local) gets forwarded to port 25 on **`smtphost.com`** on server side

```
$ ssh -L 1259:smtphost.com:25 sshserver.com
```

- *Client proceeds with normal socket connection*



Using SSL

- SSL allows for both confidentiality and authentication
  - (**`ssh`** also allows for public-key authentication)
- Directives in **`nginx`**

```
server {
    listen              443 ssl;
    server_name         www.hostname.com;
    ssl_certificate     www.hostname.com.crt;
    ssl_certificate_key www.hostname.com.key;
}
```


SSL Certificate

- Generating CA and server certificates

```
ca="ca"
server="localhost"    # could be DN or IP
CASUBJ="/C=SG/ST=Singapore/L=Kent Ridge/O=My CA
Pte Ltd/CN=myca.com.sg"
openssl req -subj "$CASUBJ" -x509 -newkey rsa:2048
-days 365 -nodes -keyout "$ca.key" -out "$ca.crt"
SVSUBJ="/C=SG/ST=Singapore/L=Kent Ridge/O=My Host
Pte Ltd/CN=localhost"
openssl req -subj "$SVSUBJ"       -newkey rsa:2048
-days 365 -nodes -keyout "$server.key" -out
"$server.csr"
openssl x509 -req -in "$server.csr" -CA "$ca.crt"
-CAkey "$ca.key" -CAcreateserial -out
"$server.crt" -days 365
```


Using SSL

- **`ssl`** module in Python

```
import socket, ssl
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
ssl_sock = ssl.wrap_socket(s,
                           ca_certs="ca-certificates.crt",
                           cert_reqs=ssl.CERT_REQUIRED)
ssl_sock.connect((host, port))
cert = ssl_sock.getpeercert()
ssl.match_hostname(cert, host)

f = ssl_sock.makefile(mode="rw")
f.write(send)
f.flush()
print(f.read())
f.close()
ssl_sock.close()
```


SSL in HTTPServer

- Same **`ssl.wrap_socket`** style

```
import BaseHTTPServer, SimpleHTTPServer
import ssl

PORT = 8443

Handler = SimpleHTTPServer.SimpleHTTPRequestHandler
httpd = BaseHTTPServer.HTTPServer(("", PORT), Handler)
httpd.socket = ssl.wrap_socket(httpd.socket,
                               certfile='localhost.crt',
                               keyfile='localhost.key',
                               server_side=True)
print "serving at port", PORT
httpd.serve_forever()
```


Application layer protection

- Plaintext/digest http/ftp/smtp authentication
- XML encryption/signatures

Summary

- Authentication with confidentiality incorporated via combination of application layer and transport layer security