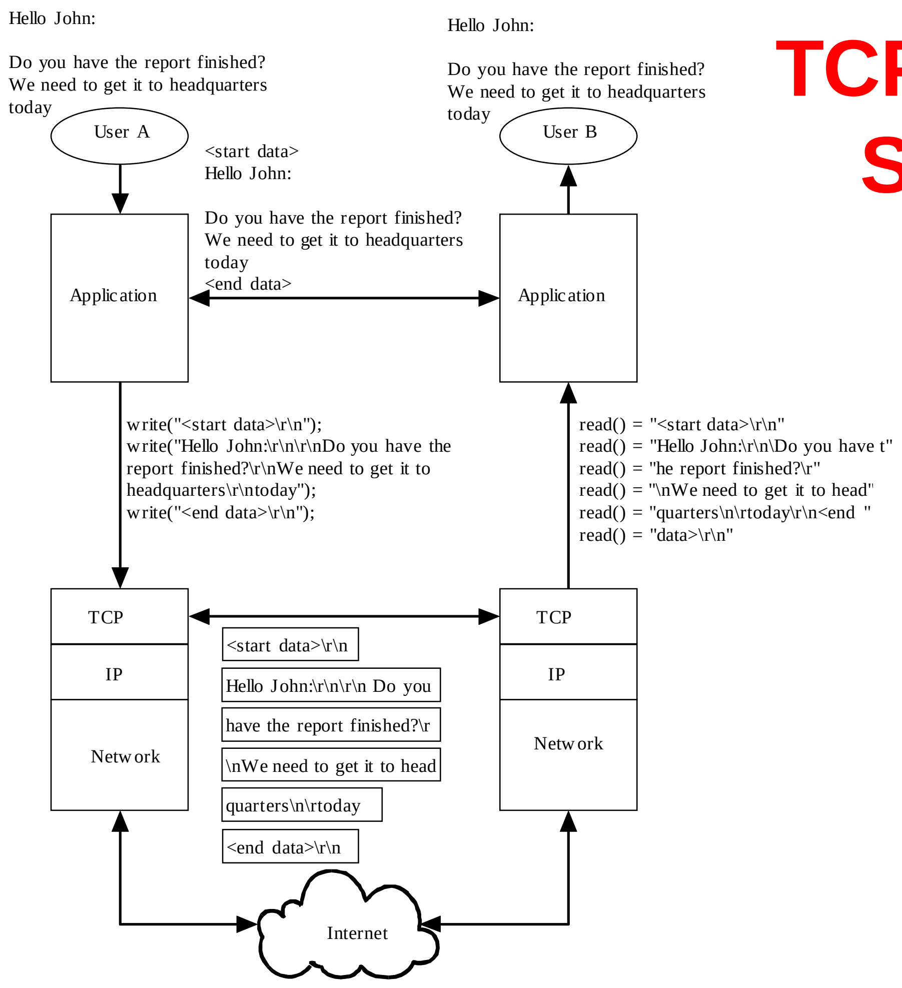
CprE 530

Lecture 16

Application Layer Security

- TCP stream Service
- Socket Layer
- Common Attack Methods

TCP Stream Service



Hello John:

Do you have the report finished?
We need to get it to headquarters
today

```
<start data>
Hello John:

Do you have the report finished?
We need to get it to headquarters
today
<end data>
```

```
write("<start data>\r\n");
write("Hello John:\r\n\r\nDo you have the
report finished?\r\nWe need to get it to
headquarters\r\ntoday");
write("<end data>\r\n");
```

```
<start data>\r\n
Hello John:\r\n\r\n Do you
have the report finished?\r
\nWe need to get it to head
quarters\n\rtoday
<end data>\r\n
```

```
read() = "<start data>\r\n"
read() = "Hello John:\r\n\Do you have t"
read() = "he report finished?\r"
read() = "\nWe need to get it to head"
read() = "quarters\n\rtoday\r\n<end "
read() = "data>\r\n"
```

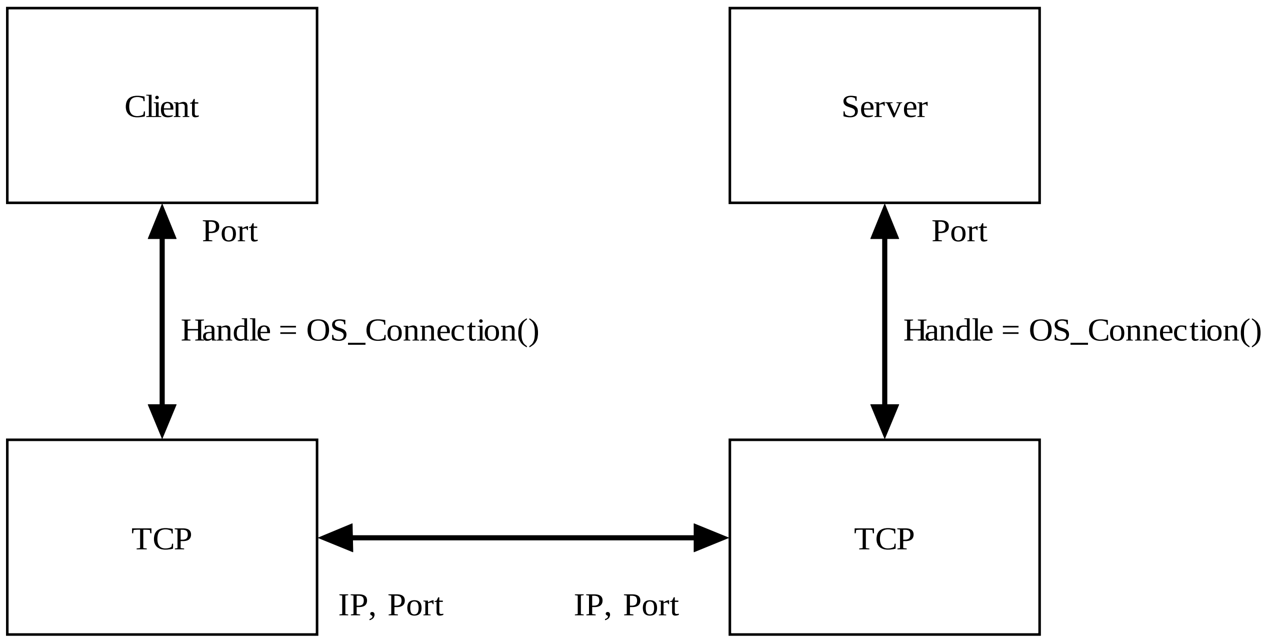
Hello John:

Do you have the report finished?
We need to get it to headquarters
today

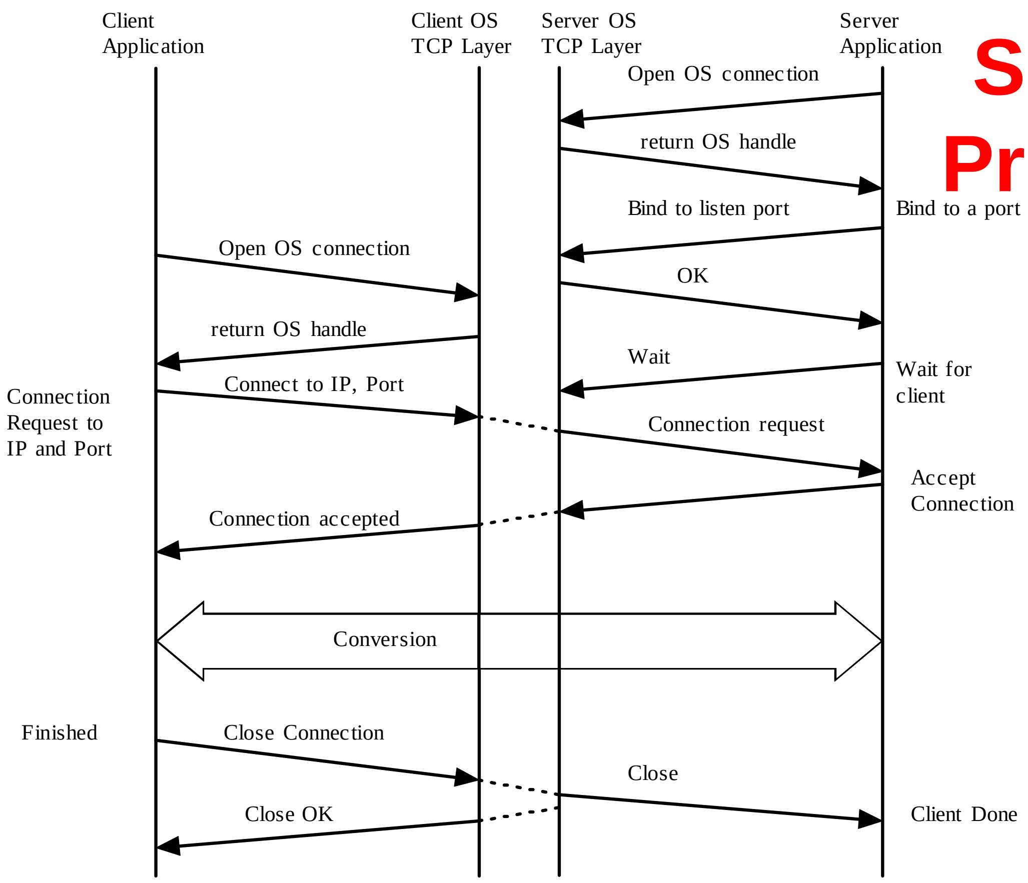
Sockets

- Application to TCP interface
- Socket protocol

Application / TCP Interface



Socket Protocol



Socket Code – Server Side

```
   nsaddr.sin_family = AF_INET;
   nsaddr.sin_addr.s_addr = INADDR_ANY;  // Accept connection from all
/* nsaddr.sin_addr.s_addr = inet_addr("129.186.5.101"); */
   nsaddr.sin_port = htons(2000);
// Open stream port.
   if ((vs = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        printf("socket(SOCK_DGRAM): %d\n",errno);
        exit(1);
   }
// bind stream to port 2000 from any address
   if (bind(vs, (struct sockaddr *)&nsaddr, sizeof(nsaddr)) < 0) {
        printf("bind(vs, %s[%d]) errno = %d\n "
         ,inet_ntoa(nsaddr.sin_addr), ntohs(nsaddr.sin_port),errno);
        perror("bind error");
        exit(1);
   }
   fprintf(stderr,"SERVER: bind(vs, %s[%d]):\n ",
        inet_ntoa(nsaddr.sin_addr), ntohs(nsaddr.sin_port));
```

Socket Code – Server Side

```
   printf("SERVER: listen waiting\n");
// allow 5 pending connection requests to this port
   if ((listen(vs,5)) < 0 ) {
        perror("listen");
        exit(1);
   }
   printf("SERVER: waiting  buf size = %d\n",sizeof(buf));
   from_len = sizeof(from_addr);
// wait for incoming connection
   if ((ns = accept(vs, &from_addr, &from_len)) < 0) perror("accept");
```

Socket Code – Client Side

```
// this calls the DNS system
h_name = gethostbyname("vulcan.ee.iastate.edu");
/* s_name = getservbyname("phone", "udp");*/
/* sin.sin_port     = s_name->s_port; */
sin.sin_family = AF_INET;
sin.sin_port = htons(2000);  // port to connect to
sin.sin_addr.s_addr = *(u_long *)h_name->h_addr;
printf("port = %d  %s\n",ntohs(sin.sin_port),
inet_ntoa(sin.sin_addr));
// open socket
sockFD = socket(AF_INET, SOCK_STREAM, 0);
// open connection to server
if (connect(sockFD, &sin, sizeof(sin)) < 0) {
     perror("connect request");
     (void) close(sockFD);
     exit(1);
}
```

Socket Code – data xfer

```
   strcpy(buf,"from client");
// client sends first
   if (send(sockFD, buf, strlen(buf),0) != strlen(buf)) {
        perror("send request");
        (void) close(sockFD);
        exit(1);
   }
// Client waits for answer
   cp = answer;
   if ((n = recv(sockFD, cp, 100, 0)) < 0){
        perror("SendRequest");
        (void) close(sockFD);
   }
   cp[n] = 0;
   printf("===<%s>===\n",cp);
   (void) close(sockFD);
```

Socket Code – data xfer

```
printf("SERVER: accepted call\n");
// print where the connection came from
fprintf(stderr,"SERVER: from_addr(ns, %s[%d]):\n ",
    inet_ntoa(from_addr.sin_addr), ntohs(from_addr.sin_port));
// get the data from the client
blen = recv(ns,buf,sizeof(buf), 0);
buf[blen] = 0;
printf("SERVER: --<%s>--\n",buf);
strcpy(buf,"hello");
printf("SERVER: sending\n");
// send response to client
if (send(ns, buf, strlen(buf), 0) != strlen(buf)) {
        perror("Sendto");
}
// shutdown connection, leaves socket open
shutdown(ns,2);
```

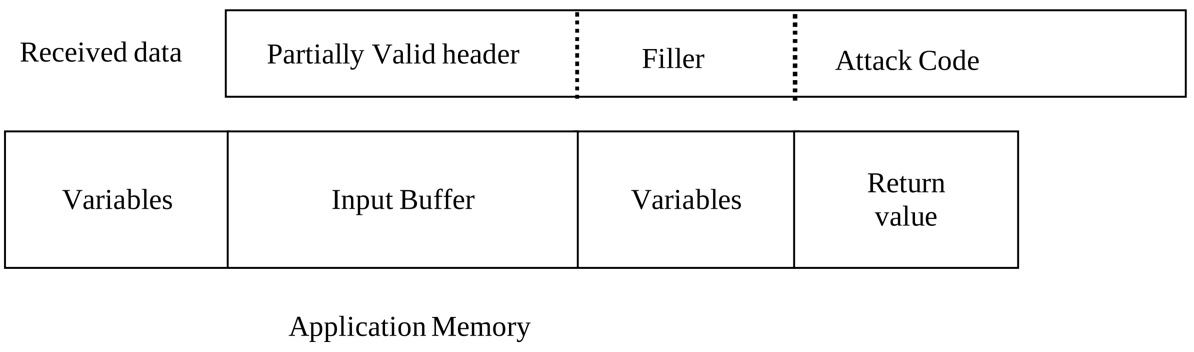
Application Layer Vulnerabilities

- Same four categories
- Applications do have some attacks in common
- Attacks are often limited to the application
- Can allow access to the computer (privileged applications are a common target)

Header-Based

- Common attack against applications
- Most applications have a freeform header which means the header must be parsed
- Buffer overflow is a common form

Buffer overflow



Protocol-Based

- Application specific
- Often part of an authentication attack

Authentication-Based

- The most common type of attack
- Two types
- Direct Attack
  - Using the applications authentication mechanism to gain access (password guessing)
- Indirect Attack
  - Using one of the other attack categories to circumvent authentication (primary focus)

Traffic-Based

- DOS
- Sniffing