

# esercizi su vulnerabilità del software e delle reti

# input fidato e non

- per quali dei seguenti software una vulnerabilità rappresenta una minaccia? in quali condizioni?
  - apache: server web
  - il kernel linux
  - exim: server smtp
  - passwd: comando per cambiare la password
  - login: comando che chiede la password e apre una shell per un certo utente
  - crontab: comando per schedulare operazioni periodiche da eseguire anche se l'utente non è loggato
  - ps: comando per vedere i processi attivi
  - ls: comando per vedere la directory corrente

# questo cgi script è vulnerabile?

```
#!/bin/sh
# mostra via web lo spazio libero
# nelle partizioni
echo -e "Content-type: text/html\n\n"
df
```

# questo cgi script è vulnerabile?

```
#!/bin/sh
# mostra un traceroute verso
# un ip passato come parametro
echo -e "Content-type: text/html\n\n"
traceroute $QUERY_STRING
```

- in `$QUERY_STRING` il web server passa tutto ciò che, nell'url, è dopo "?"
  - es. `http://localhost/cgi-bin/tr?193.204.161.1`

# buffer overflow: il seguente programma è vulnerabile?

```
int main()
{ f(); }
void f()
{
char b[30];
getwd(b) ; /* get working directory */
printf("%s", b);
}
```

# buffer overflow: il seguente programma è vulnerabile?

```
int main()
{
    char b[30];
    getwd(b) ; /* get working directory */
    printf("%s", b);
}
```

# buffer overflow: il seguente programma è vulnerabile?

```
int main()
{
  char *b;
  char b2[100];
  b = getenv ("PATH"); /* take the value of
  $PATH */
  ....
  ....
}
```

# buffer overflow: il seguente programma è vulnerabile?

```
int main()
{
  char *b;
  char b2[100];
  b = getenv ("PATH"); /* take the value of
  $PATH */
  strcpy(b2, b) ; /* copy b in b2 */
  ....
}
```

# vulnerabile?

- questo programma è vulnerabile?
- perché?

```
readRecipe (int fd)
```

```
{
```

```
    char str[200];
```

```
    while (readLine (fd, str))
```

```
        printf ("%s\n", str);
```

```
}
```

```
readline (int fd, char* str)
```

```
{
```

```
    int n;
```

```
    do
```

```
    {
```

```
        n = read (fd, str, 1);
```

```
    }
```

```
    while (n > 0 && *str++ != NULL)
```

```
    return (n > 0);
```

```
}
```

# buffer overflow

## stack cresce verso l'alto

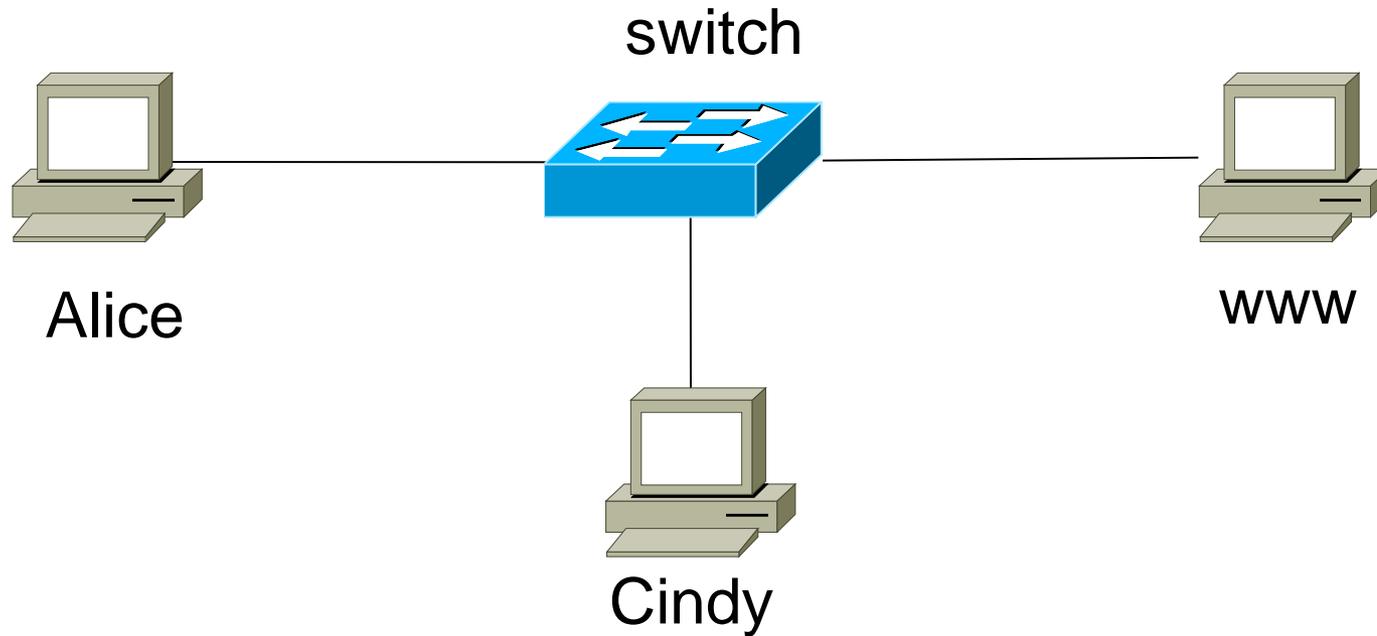
- come si comportano le architetture con stack che cresce verso l'alto rispetto al buffer overflow?
  - sono più o meno esposte al rischio?

```
int main()
{
char b[30];
scanf ("%s", b);
}
```

# buffer overflow: stack cresce verso l'alto

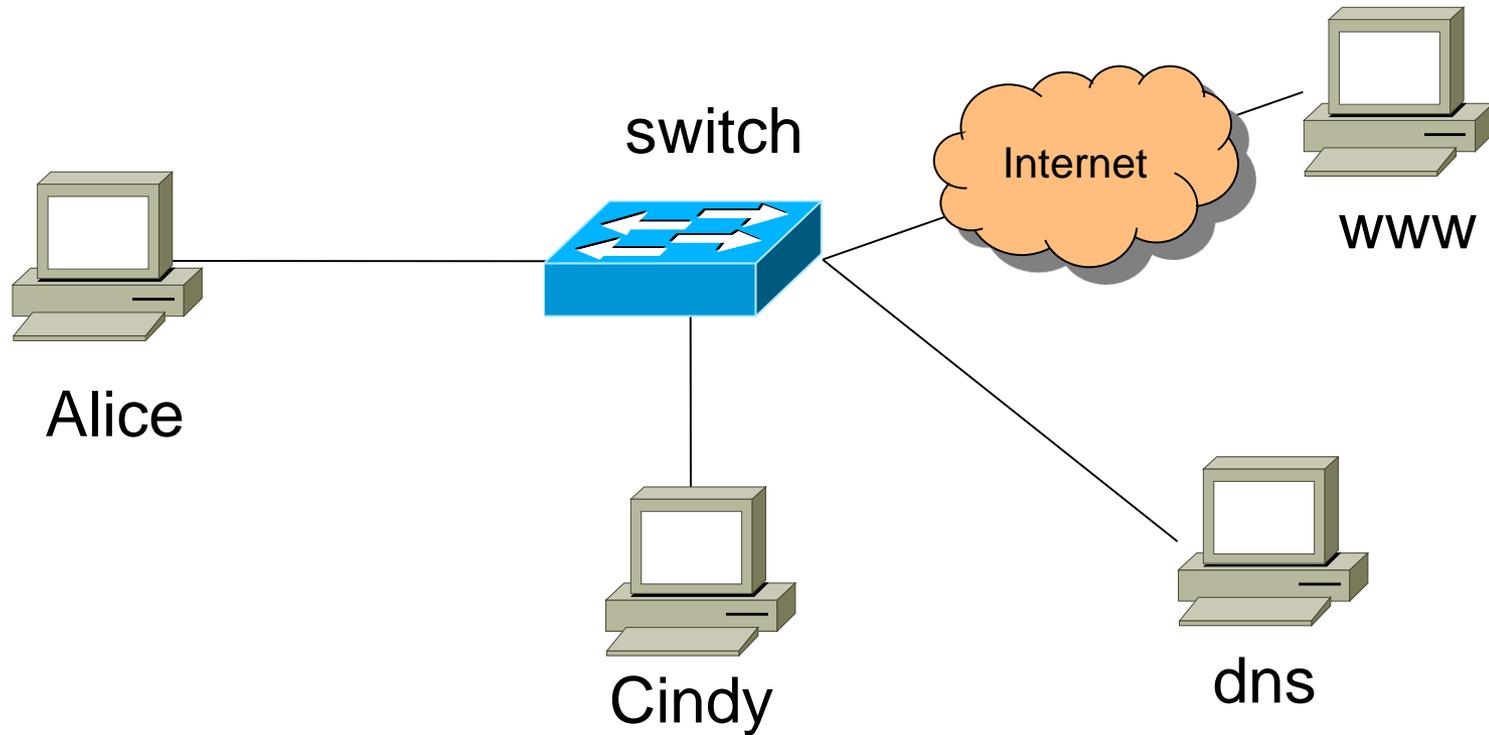
- rivedi tutti i precedenti programmi nel caso di stack che cresce verso l'alto

# vulnerabilità reti



- Alice dialoga con www, http su tcp
- Come fa cindy a sniffare il traffico?
- come fa Alice o l'amministratore ad accorgersi dell'intruso?

# vulnerabilità reti



- Alice accede a www (http su tcp), risolvendo il nome tramite il server dns (non autenticato)
- Come fa cindy a far si che Alice acceda ad un web server sulla sua macchina anziché a www?
- come fa Alice o l'amministratore ad accorgersi dell'intruso?

# metodologia di attacco ad una lan

- Sei un hacker "malintenzionato" e puoi accedere ad una rete locale tramite una presa ethernet. Non hai alcuna informazione sulla rete e sulle macchine connesse. Illustra una (o più) metodologie di attacco che mirano ad ottenere il controllo, se possibile, di una delle macchine connesse alla rete locale.