Race Conditions

- A race condition occurs when an assumption needs to hold true for a period of time, but actually may not
- Bob and Alice example.

Java Example

```
Import java.io.*
Import java.servlet.*
Import java.servlet.http.*
public class Counter extends HttpServlet {
        int count = 0;
        public void doGet(HttpServletRequest in,
HttpServletResponse out) throws ServletException {
                out.setCountentType(“text/plain”);
                Printwriter p = out.getWriter();
                count++;
                p.println(count + “ hits so far!”);
        }
}
```

Java Example

```
Import java.io.*
Import java.servlet.*
Import java.servlet.http.*
public class Counter extends HttpServlet {
        int count = 0;
        public void doGet(HttpServletRequest in,
HttpServletResponse out) throws ServletException {
                int my_count;
                out.setCountentType(“text/plain”);
                Printwriter p = out.getWriter();
                synchronized(this) { my_count = ++count; }
                p.println(my_count + “ hits so far!”);
        }
}
```

Time-of-Check, Time-of-Use

- The check for privilege (access) to a resource should occur at the same time when the resource is used.
- Race conditions always happens when more than one process/thread has access to the same resource.
- Security-critical race conditions happen when the 2 processes has different privileges

Unix file-based Race Condition

- Unix filesystem is shared among all processes
- If a process with privilege writes to a file after checking access, a race condition occur.
- Password file problem in the book, pages 218 - 219

Race Condition in Some Cable Modems

- Users were able to use a race condition in some early version of cable modem software to trick the modem to download a config file from their server instead of the cable company server
- See http://www.securityfocus.com/news/394/

“If cable modem hacking hasn't become a huge problem for service providers, it's probably because the process remains intimidating for non-technical users. The subscriber has to program a DOCSIS configuration file with a special editor, run their own TFTP server, change their IP address and run an DHCP server that tricks the modem into pulling the config file from their host. Dedicated hobbyists have refined the procedure and written tools to automate key portions of it, but pitfalls and caveats abound”

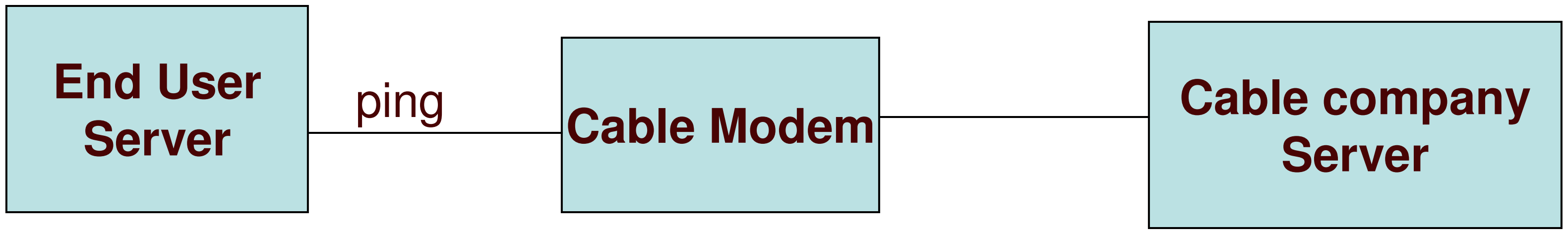
How cable modem works

- Offline.
- Scan for downstream channel.
- Receive Upstream Channel Descriptor (UCD).
- Ranging to find tx level and symbol timing.
- DHCP to get CM IP address and gateway.
- Use TFTP to get config file.
- Initialize Baseline Privacy (BPI).
- Receive Time Of Day (ToD).
- Online.

Cable Modem Config file contains

- Downstream channel identification
- Class of Service settings
- Baseline Privacy settings
- General operational settings
- Network management information
- Software upgrade fields
- Filters
- Vendor specific settings

Using TFTP to get the config file



User server has same IP as the cable company server

- •Cable modem needs to communicate with tftp server based on DHCP parameter
- •Cable modem requests the MAC address of the tftp server if it does not have it already
- •Cable modem starts sending tftp requests/packets to that server
- •The user simply continue to ping the cable modem address from his/her own server to get its own MAC address in the modem memory
- •The cable modem communicates with the user server instead of the company server.
- •This bug has been fixed in most cable modems.

Unix filesystem Race condition

- Race conditions are only a security problem if the process runs as a different user. i.e only a potential security issue if the programs is suid.
- Extra care is needed when writing software that is expected to run as suid under unix. Specially when doing file operations
- Method described on page 220 can be used to avoid race conditions and ensure the correct file was opened.

Unix filesystem Race condition

- Some operations require a filename and cannot operate on a file handles [link(), mkdir(), mknod(), rmdir(), symlink(), unmount(), unlink(), utime()]
- The best solution is to keep files a specific program needs in their own secure dir.
- The software should chdir() to the dir then check it is secure before working on any files in that dir.

Deleting files

- Deleting files securely can only be done if using secure directory approach
- Deleting a file simply remove the reference to it from the dir and frees the blocks. It does not delete the data
- Technology exits to even recover data from disk plates even if data has been overwritten
- The best way to ensure data security is not to write it on disk or use some encryption before writing to disk. Must also ensure that data does not get written to disk by OS by swapping memory out.

Temporary Files

- Writing temp files to a common accessible directory is a bad practice from security point of view.
- If a temporary file must be written to /tmp follow the steps on page 226

File Locking

- File locking in most operating system is discretionary and not mandatory
- To ensure locks are not taken over by potential attackers, file must be secure directory
- Locks on network file systems may not work.

Other Race Conditions

- Cable modem problem described before
- Race condition occurs anytime 2 actions act of the same resource.
- Race condition creates a security problem when the 2 processes has different privileges