

CIS/TCOM 551

# Computer and Network Security

Spring 2005

Lecture 11

# Announcements

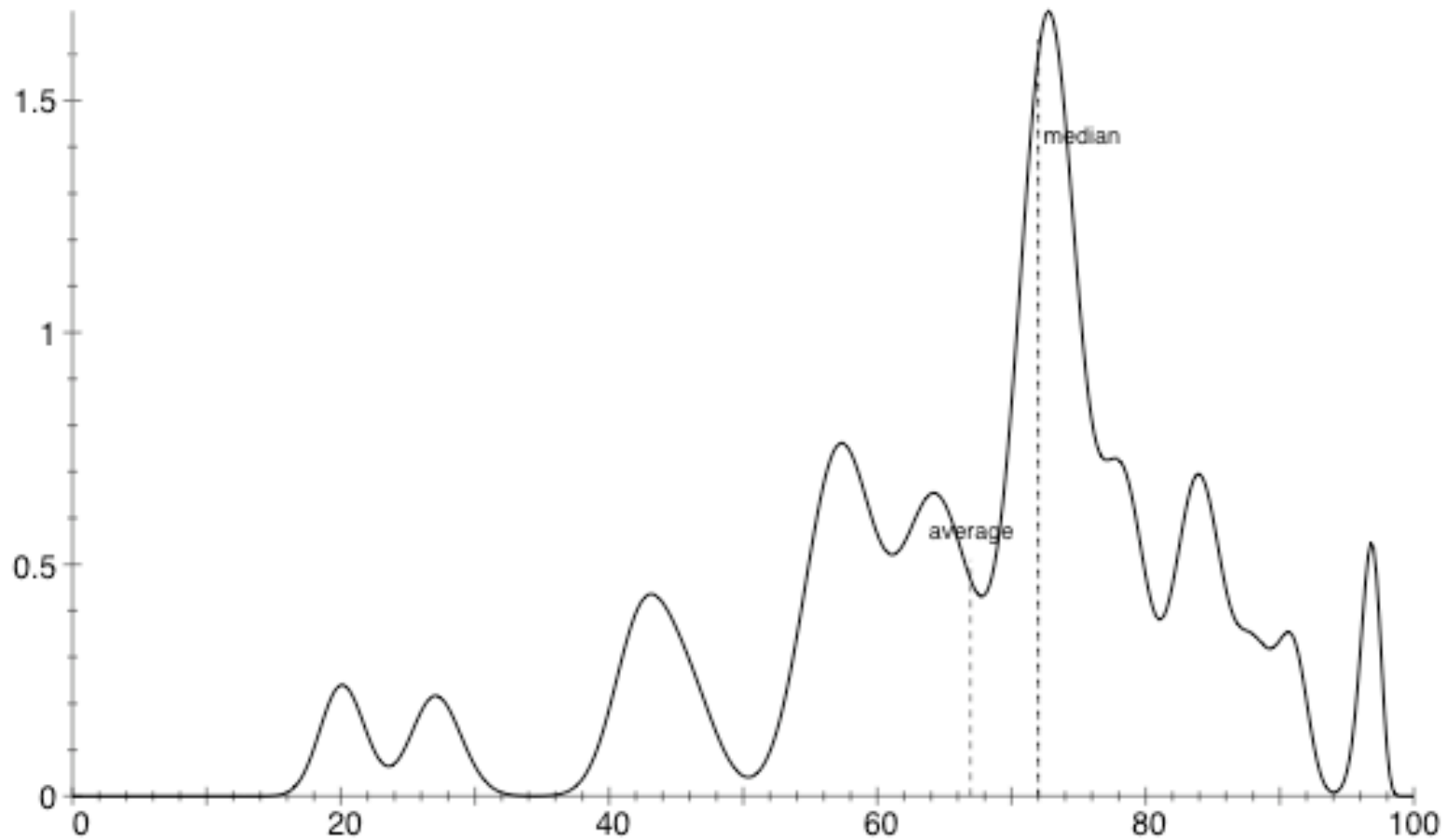
---

- Midterm 2 is graded
  - Average: 67%
  - Std. Dev.: 17.5
- Project 1 grades should be sent out today.
- Final project is on the web site.
  - Due last day of classes: Friday, April 22nd.
  - Groups of 2 or 3 students
    - E-mail Eric Cronin [ecronin@cis.upenn.edu](mailto:ecronin@cis.upenn.edu) by Friday.
- Final exam has been scheduled:
  - Thurs. April 28th 11am - 1pm
  - Chemistry B13

# Midterm 2 Grade Distribution

---

---



# Today's Agenda

---

- Midterm 2
  
- Disguising intrusion traffic
- Covert Channels
  - Examples
- Applications:
  - Watermarking
  - Censorship prevention
  - Anonymity
  - Digital rights management

# Disguising Intrusions

---

- IDS look for known signatures or anomalous behavior to detect intrusions
- There are many ways attacks can be disguised
  - Polymorphic / metamorphic worms & viruses
  - Trickle attacks that take place slowly
  - Disguise traffic by fragmenting/reordering/making invalid packets
    - Most commercial IDS systems do not do proper TCP/IP reconstruction
- Covert channels
  - Sneaky ways of transmitting information, usually by using something in a way that wasn't originally intended.

# Covert Channels

---

- Programs can leak confidential information intentionally via secret channels.
- Not that hard to leak a small amount of data
  - A 64 bit shared key is quite small!
- Even possible to encode relatively large amounts of data!
- Example channels
  - Program behavior
  - Adjust the formatting of output:  
use the “\t” character for “1” and 8 spaces for “0”
  - Vary timing behavior based on key
  - Use "low order" bits to send signals
  - Power consumption
  - Grabbing/releasing a lock on a shared resource

# Applications of Covert Channels

---

- *Steganography*: (covered writing)
  - The process of secretly embedding information into a data source in such a way its very existence is concealed.
- *Digital watermarking*:
  - A short sequence of information embedded in a way that is difficult to erase.
- *Censorship prevention*:
  - Hide communication and communicated data within "legitimate looking" traffic to prevent unwanted filtering.

# Watermarking Basic Idea

---

- Pictures, Video, and Sound
  - Human perception is imperfect
  - There are a lot of “least significant bits”
  - Modifying the least significant bits doesn’t change the picture much



$(R,G,B) = (182,54,89)$



$(R,G,B) = (182,54,90)$

- Encode a signal in the least significant bits.

# Watermarking Example

---



Original Image



Watermarked Image

# Properties of Watermarks

---

- Desirable properties
  - Imperceptible
  - Robust (withstands modifications to the image)
  - High capacity
  - Efficient
  - Hard to remove (some schemes involve cryptographic operations)
- Drawbacks
  - Hard to make tamper proof
  - Can distort image/sound

# Preventing Censorship

---

- *Infranet: Circumventing Web Censorship and Surveillance* N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, D. Karger
- Idea: Use steganography to embed 'censored' web traffic (html data) into the JPEG files returned by 'legal' web sites.
  - Give the end users a way to surf censored parts of the web without revealing that activity.
  - Better than just using SSL because it can't easily be blocked

# Properties of Infranet

---

- Deniability for any Infranet client (requester)
  - Computationally intractable to detect use
- Statistical deniability for any Infranet client
  - Use should affect perceived browsing patterns
- Server (responder) covertness
  - Hard to detect that server is running Infranet
- Communication channel robustness
  - Covert channel should not be easy to interfere with

# Infranet Communication

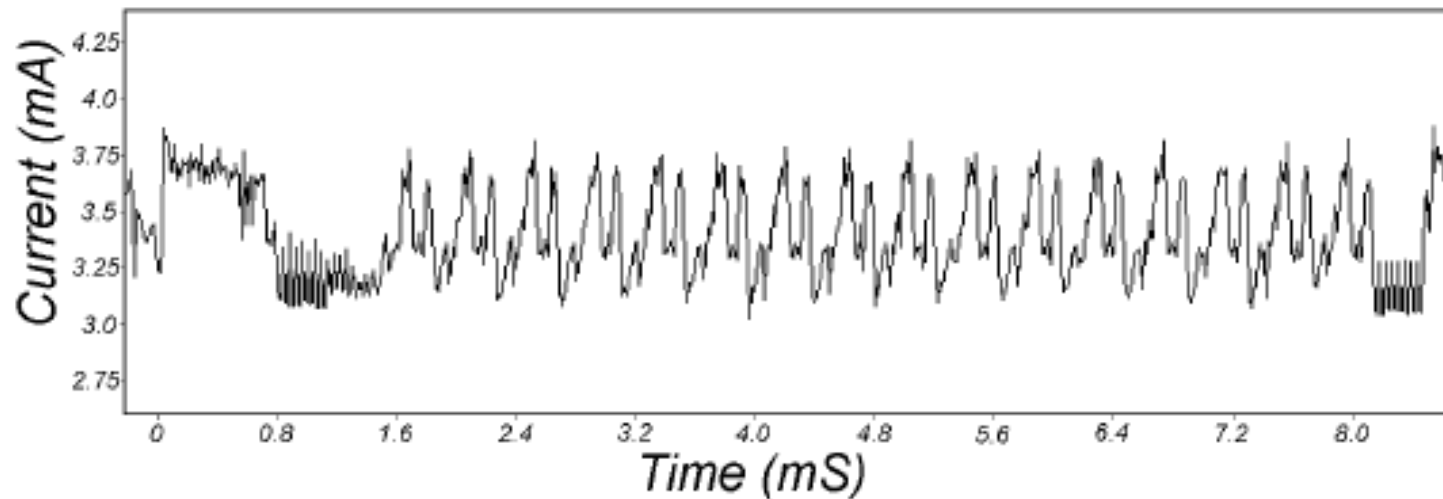
---

- Upstream (client to server) channel:
  - Encode surreptitious requests in the sequence of URLs requested
  - Rate, frequency, ect., are modulated to look like real traffic
- Downstream (server to client) channel:
  - Encode HTML by watermarking a JPEG
- This communication is made cryptographically secure by use of a private key scheme...

# Differential Power Analysis

---

- Read the value of a DES password off of a smartcard by watching power consumption!



- This figure shows simple power analysis of DES encryption. The 16 rounds are clearly visible.

# TEMPEST Security

---

- Transient Electromagnetic Pulse Emanation Standard
  - (Or?) Temporary Emanation and Spurious Transmission
  - Emission security (Van Eck phreaking)
  - computer monitors and other devices give off electromagnetic radiation
  - With the right antenna and receiver, these emanations can be intercepted from a remote location, and then be redisplayed (in the case of a monitor screen) or recorded and replayed (such as with a printer or keyboard).

# TEMPEST

---

- Policy is set in National Communications Security Committee Directive 4
- Guidelines for preventing EM reception
  - Shield the device (expensive)
  - Shield a location (inconvenient?)

# Covert Channels in Software

---

```
int X = some secret;
int Y = 0;

int mask = 1;
for (i=0; i<32; i++) {
    if (X & mask) {
        Y = Y | mask;
    }
    mask = (mask << 1);
}

/* At this point Y == X */
```

# Defenses for Covert Channels

---

- Well specified security policies at the human level
- Auditing mechanisms at the human level
  - Justify prosecution if the attacker is caught
- Code review
  - This is a form of audit
- Automated program analysis
  - Type systems that let programmers specify confidentiality labels like Jif
  - Transform programs so that both branches of a conditional statement take the same amount of time
  - Disallow branches on "secret" information

# Countermeasures

---

- Against timing attacks:
  - Make all operations run in same amount of time
    - Hard to implement!
    - Can't design platform-independent algorithms
    - All operations take as long as slowest one
  - Add random delays
    - Can take more samples to remove randomness
- Against power analysis attacks:
  - Make all operations take the same amount of power
    - Again, hard to implement
  - Add randomness