### EXERCISE #14

#### **REVIEW: INFORMATION FLOW**

### Write your name and answer the following on a piece of paper

Provide an instance of a program with an implicit information flow from a confidential (high-security) source to an untrusted (low-security) sink

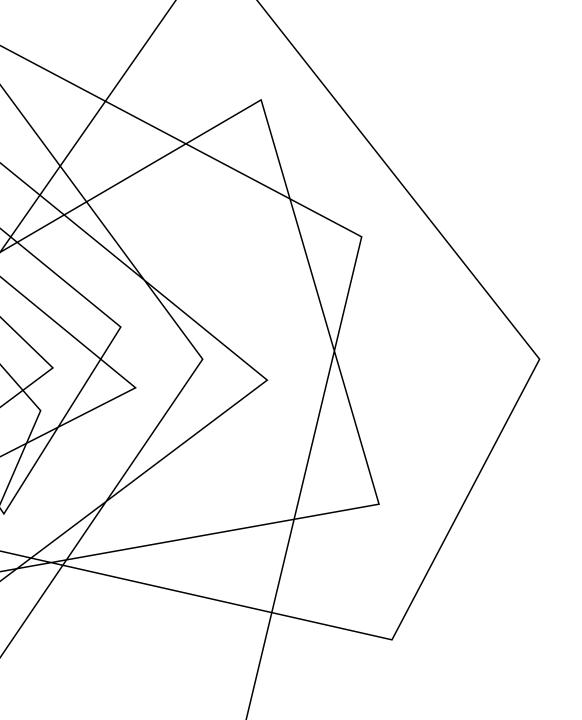
### EXERCISE #14: SOLUTION

**REVIEW: INFORMATION FLOW** 

Provide an instance of a program with an implicit information flow from a confidential (high-security) source to an untrusted (low-security) sink

For EECS777 Students: Paper #1 reading assignment is up!

### ADMINISTRIVIA AND ANNOUNCEMENTS



### **CLASS PROGRESS**

SHOWING SOME APPLICATIONS OF STATIC DATAFLOW

### LAST TIME: DATAFLOW DEPLOYMENT

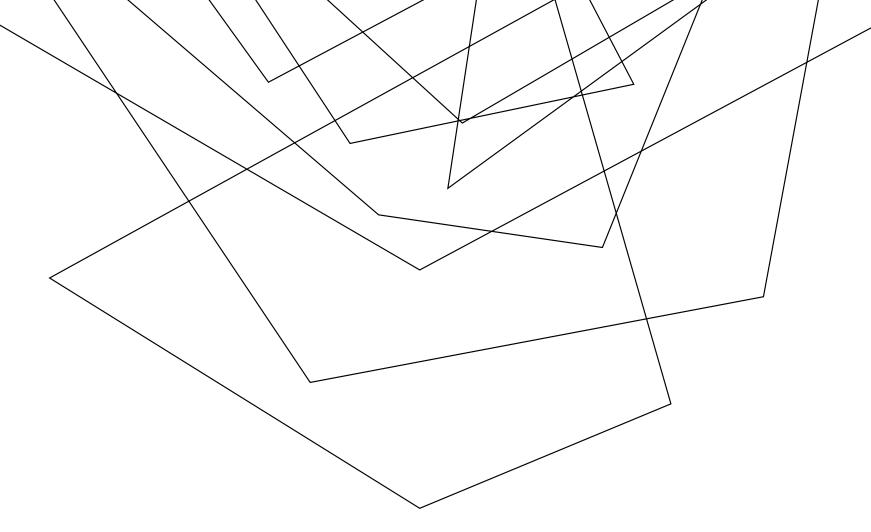
**PREVIOUSLY: INFORMATION FLOW** 

#### USING DATAFLOW IN PRACTICAL CONTEXTS

- Ex. - Looking for secret-holding variables



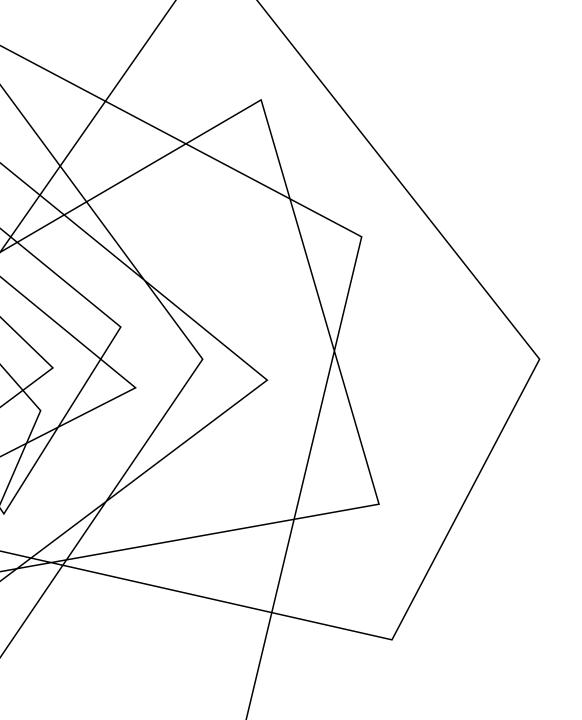
#### **IMPLICIT FLOW** PREVIOUSLY: INFORMATION FLOW



### SIDE CHANNELS

EECS 677: Software Security Evaluation

Drew Davidson

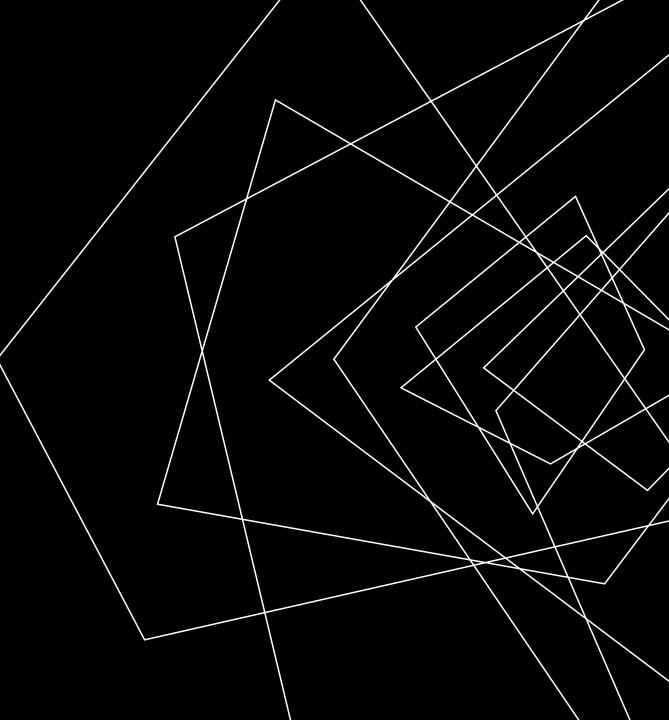


### **OVERVIEW**

CONTEMPLATE OTHER WAYS THAT SNEAKY DATA FLOWS CAN OCCUR

# **LECTURE OUTLINE**

- Threat Models
- Side Channels Overview
- Timing
- A dataflow approach



# THINKING ABOUT ATTACKS

# THERE'S NO SUCH THING AS "ABSOLUTE SECURITY"

 It's always possible to come up with SOME (potentially wacky) scenario where the adversary can subvert a system

### CONSIDER THE VARIOUS ATTACK CLASSES

- **Denial of Service:** Availability is compromised
- **Exfiltration:** Confidentiality policy is compromised
- **Compromise:** Integrity policy is compromised



# THINKING ABOUT ATTACKS

**RECALL: THREAT MODELS** 

# THERE'S NO SUCH THING AS "ABSOLUTE SECURITY"

 It's always possible to come up with SOME (potentially wacky) scenario where the adversary can subvert a system

### CONSIDER THE VARIOUS ATTACK CLASSES

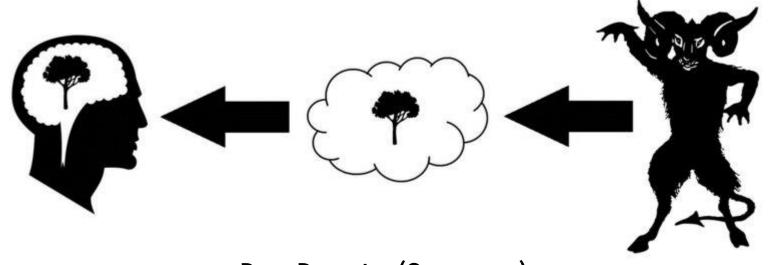
- Denial of Service: Availability is compromised
- **Exfiltration:** Confidentiality policy is compromised
- **Compromise:** Integrity policy is compromised

These assumptions are – captured by a threat model

### UNCONVENTIONAL ADVERSARIES

**RECALL: THREAT MODELS** 

OUR NOTIONS OF COMPLETENESS ARE ULTIMATELY TIED TO OUR ASSUMPTIONS



Deus Deceptor (Game over)

## UNCONVENTIONAL ADVERSARIES

**RECALL: THREAT MODELS** 

### OUR NOTIONS OF COMPLETENESS ARE ULTIMATELY TIED TO OUR ASSUMPTIONS

- An adversary may have the ability to influence (or observe) phenomena that are outside of the threat model
- Anecdote: sensor input spoofing attacks

Side-channels: Extra-semantic observation

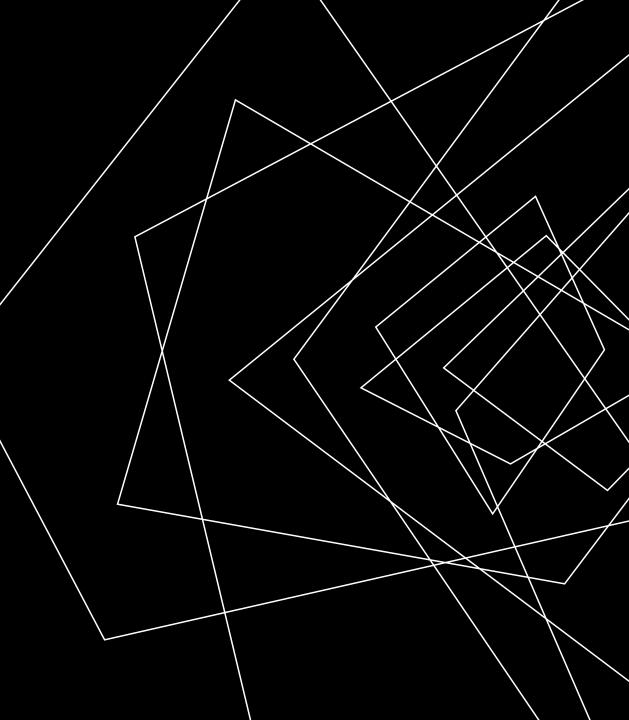
SISA: Extra-semantic influence

# **LECTURE OUTLINE**

• Threat Models

Side Channels – Overview

- Timing
- A dataflow approach



### THE BASIC IDEA OF SIDE CHANNELS

SIDE CHANNELS

#### ABSTRACTION IS A KEY PRINCIPLE OF COMPUTER SCIENCE!

As a programmer, you shouldn't need to know underlying details

#### AS A SECURITY EXPERT, THESE DETAILS MIGHT END UP BEING IMPORTANT!

The way a program accomplishes its tasks are important, especially from a security aspect

- How long does it take for the program to do X ?
- How hot does it make the processor when X happens?
- How much power does it draw when X happens?

### UNCONVENTIONAL ADVERSARIES

**RECALL: THREAT MODELS** 

(SADLY) OUR SOFTWARE NEEDS TO BE MANIFESTED IN HARDWARE

## SIDE CHANNELS – THE BIG IDEA

SIDE CHANNELS - INSTANCES

#### Computation may have effects outside of program semantics

Some operations (internally) take longer based on aspects of the data

### **TEMPEST** SIDE CHANNELS – HISTORY

#### ELECTROMAGNETIC LEAKAGE OF KEYS

- WWII: Bell Telephone discovers electromagnetic leakage in one-time pad teleprinters, detectable at 100-ft radius
- 1951: CIA rediscovers leakage, detectable at 200-ft radius
- **1964:** TEMPEST shielding protocol established



### **TEMPEST** SIDE CHANNELS - HISTORY

#### ELECTROMAGNETIC LEAKAGE OF KEYS

- WWII: Bell Telephone discovers electromagnetic leakage in one-time pad teleprinters, detectable at 100-ft radius
- 1951: CIA rediscovers leakage, detectable at 200-ft radius
- 1964: TEMPEST shielding protocol established



# VAN ECK PHREAKING

# ELECTROMAGNETIC LEAKAGE OF MONITORS

- Pick up the monitor's electromagnetic emanations that differ depending on how the screen lights up
- Originally determined for CRT (1985), also discovered for LCD monitors (2004)

		1
The quick brown for jumps over the lazy dag		-
It is well known that electronic equipment produces electromognetic fields which may cause interference to radio and television reception. The phenomena	-	- 22
underlying this have been thoroughly studied over the past few decades. These studies have resulted in internationally agreed wethods for seasuring the interference produced by equipment. These are needed because the maximum interference levels which equipment may generate have been laid down by law in most countries.	-	- 20
However, interference is not the only problem caused by electromagnetic radiation. It is possible in some cases to obtain information on the signals used inside the equipment when the radiation is picked up and the received	-	- <mark>18</mark>
saignals are decoded. Especially in the case of digital equipment this possibility constitutes a problem, because remote reconstruction of signals inside the equipment may enable reconstruction of the data the equipment is processing.	-	- <mark>16</mark>
This problem is not a new one: defence specialists have been aware of it for over twenty years. Information on the way in which this kind of "envestropping" can be prevented is not freely available. Equipment designed to protect military information will probably be three or four times more expensive than the equipment likely to be used for processing of non military	-	- 14
information. C. RExcerpt From Wim van Eck. Electromognetic Radiation from Video Display Units: An Envendropping Risk? Computers & Security 4 (1985) 269-285.1		12
<pre>'**sii'O**,/0123456789:;&lt;=&gt;?00BCDEFORIJKINK(P0PSTUVAXYZI\I'</pre>		10

Fig. 3. Text signal received from a 440CDX laptop at 10 m distance through two intermediate offices (3 plasterboard walls).

### SIDE CHANNELS – PARTIAL CREDIT

**SIDE CHANNELS - INSTANCES** 

#### EVEN "HINTS" ABOUT SECRET DATA CAN BE PROBLEMATIC

Assume you're trying to guess a password

- knowing even 1 character massively reduces the search space
- knowing the length of the password reduces the search space



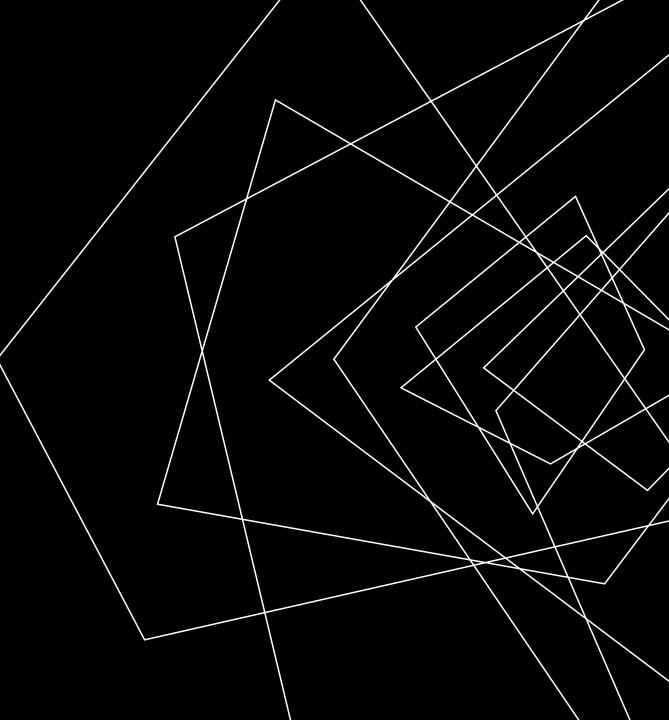


#### Sometimes a Program <u>Wants</u> to Leak data

Exfiltration !

# **LECTURE OUTLINE**

- Threat Models
- Side Channels Overview
- Timing
- A dataflow approach



#### TIMING SIDE CHANNELS SIDE CHANNELS - INSTANCES

#### Some computations take longer than others

Some operations (internally) take longer based on aspects of the data

```
bool checkPW(const char * given){
  const char * expected = "12345";
  int gLen = strlen(given);
  int eLen = strlen(expected);
  if (gLen != eLen) { return false; }
  for (int i = 0; i < eLen; i++) {
    if (given[i] != expected[i]) {
      return false;
    }
  }
  return true;
}</pre>
```

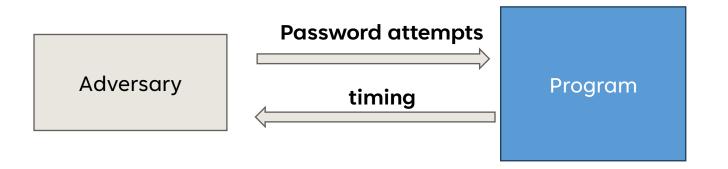
TIMING SIDE CHANNELS SIDE CHANNELS - INSTANCES

#### Some computations take longer than others

Some operations (internally) take longer based on aspects of the data

### THREAT MODEL

Interactive, low-latency\*, black-box access to the program, precise timer



\*: May be overcome with more samples

### TIMING SIDE CHANNELS - FIX

SIDE CHANNELS - INSTANCES

```
bool checkPW(const char * given) { bool checkPW(const char * given) {
  const char * expected = "12345";
  int gLen = strlen(given);
  int eLen = strlen(expected);
  if (gLen != eLen) { return false; } bool ok = true;
  for (int i = 0; i < eLen; i++) { if (gLen != eLen) { ok = false; }</pre>
   if (given[i] != expected[i]) {
     return false;
  return true;
```

```
const char * expected = "12345";
int gLen = strlen(given);
int eLen = strlen(expected);
for (int i = 0; i < eLen; i++) {</pre>
  int gIdx = math.min(gLen - 1, i);
  if (given[gIdx] != expected[i]) {
    ok = false;
return ok;
```

### TIMING SIDE CHANNELS - FIX

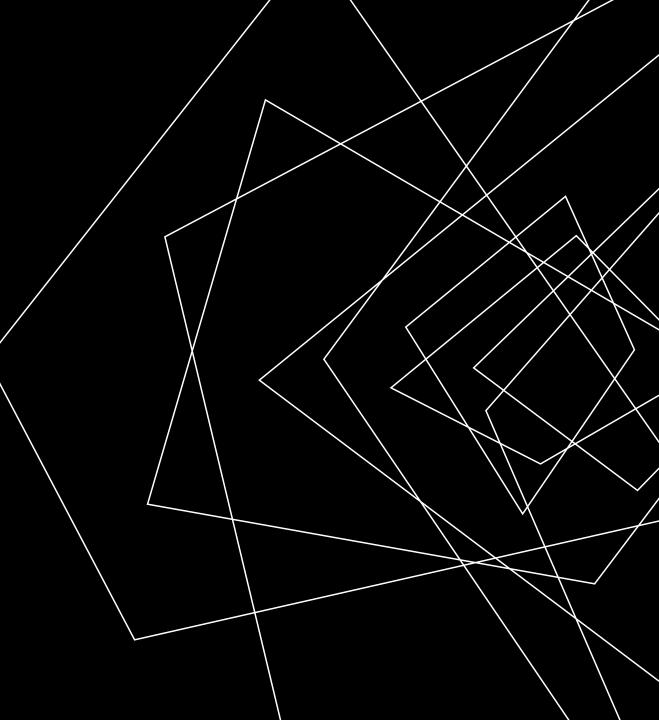
SIDE CHANNELS - INSTANCES

#### LIMITATIONS OF UNIFORM EXECUTION

- Necessarily slow down your computation to the worst case
- May require some pretty precise understanding of timing
- May not always be obvious what the worst-case even is

# **LECTURE OUTLINE**

- Threat Models
- Side Channels Overview
- Instances
- A dataflow approach



### TIMING SIDE CHANNELS - FIX

SIDE CHANNELS - INSTANCES

#### CAN WE FIX THIS ISSUE WITH OUR DATAFLOW APPROACH?

- Instruction transformers: how much time that instruction takes
- Block composition: the sum total of instruction times
- Merge operation: some sort of check that all paths are of comparable time?

## WRAP-UP

