EXERCISE #33

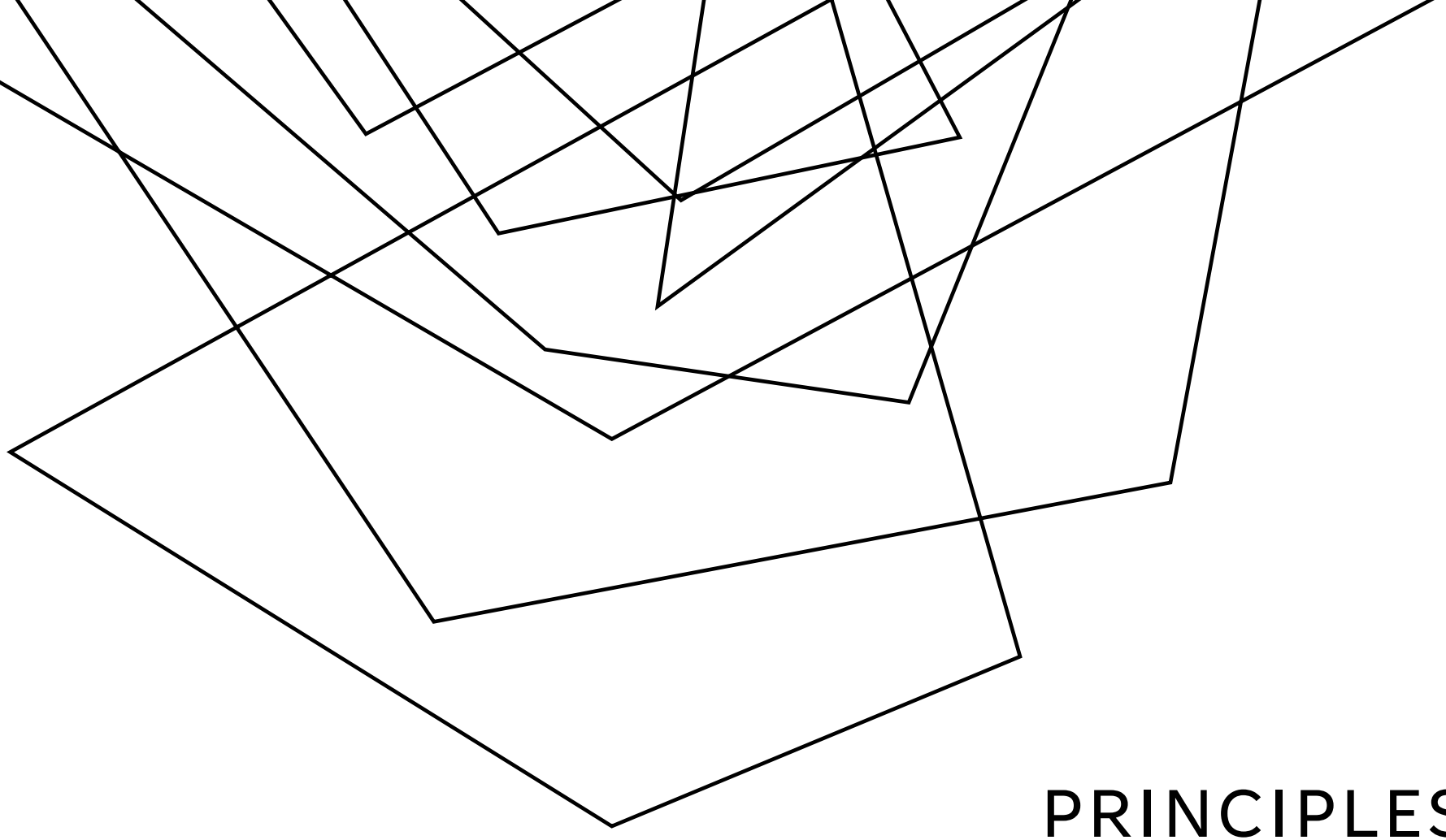
BOOLEAN SATISFIABILITY REVIEW

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

Use the Nelson-Oppen procedure to separate theories in the following formula

$$(a - f(b + c) = 3) \wedge (a - 1 = 0) \wedge (b > 10)$$

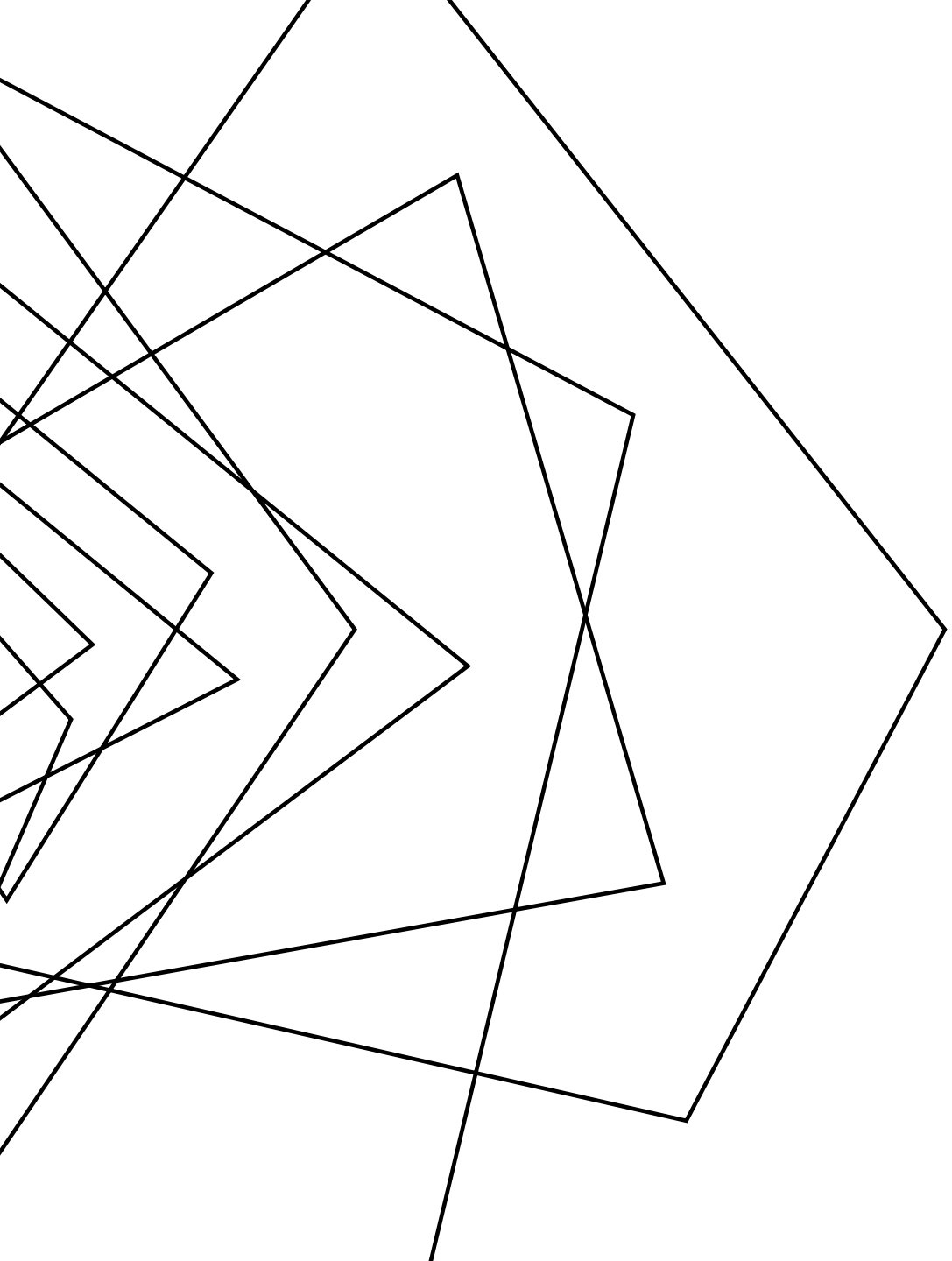
ADMINISTRIVIA AND ANNOUNCEMENTS



PRINCIPLES OF SECURE ENGINEERING

EECS 677: Software Security Evaluation

Drew Davidson



WHERE WE'RE AT

FINISHED UP DYNAMIC ANALYSIS

GRAB-BAG TIME!

PREVIOUSLY: SMT SOLVING

REVIEW LAST LECTURE

SATISFIABILITY BEYOND SIMPLE BOOLEAN EXPRESSIONS

Gets us (closer) to the real programs that we want to analyze

KEY PRINCIPLES

Individual theory solvers

Formulating constraints modularize a concern to a theory

THIS LECTURE
SMT SOLVING

Best Practices

SECURE DESIGN

PRINCIPLES OF SECURE SOFTWARE DEVELOPMENT

Security is Relative

Obviously (from our previous study) it is relative to a threat model

Less obviously (from our topics) is that it should be conceptualized in terms of improvements

Mindsets

Convince yourself that the system is secure

Convince yourself that you are identifying and fixing weaknesses of the system

SECURITY PRINCIPLES

PRINCIPLES OF SECURE SOFTWARE DEVELOPMENT

Simplicity

Open design

Maintainability

Privilege Separation / Least priviledge

Defense in depth / diversity

Complete mediation and fail-safe

THE PRINCIPLE OF LEAST PRIVILEGE

PRINCIPLES OF SECURE SOFTWARE DEVELOPMENT

“ENTITIES SHOULD ONLY HAVE ACCESS TO THE DATA AND RESOURCES NEEDED TO PROVIDE AUTHORIZED TASKS”

At what granularity do we consider an entity?

PRIVILEGE SEPARATION

PRINCIPLES OF SECURE SOFTWARE DEVELOPMENT

Break system into compartments
- Ensure each compartment is isolated
- Ensure each compartment runs with least privilege
- Treat compartment interface as trust boundary

PRIVILEGE SEPARATION

PRINCIPLES OF SECURE SOFTWARE DEVELOPMENT

EXAMPLE: THE OPERATING SYSTEM

Users can execute programs/processes
Processes can access resources

MEMORY ISOLATION

Process should not be able to access another process's memory

RESOURCE ISOLATION

Process should only be able to access certain resources

UNIX ACLS

Permissions to access files are granted based on user IDs
Every user has a unique UID
Access Operations: Read, Write, Execute
Each le has an access control list (ACL)
Grants permissions to users based on UIDs and roles (owner, group, other)
root (UID 0) can access everything

ESTABLISH THE TRUSTED COMPUTING BASE

PRINCIPLES OF SECURE SOFTWARE DEVELOPMENT

WHAT ARE THE SECURITY ASSUMPTIONS ABOUT THE CODE BASE?

What are the components critical for security?

Trusted != secure

RELATED CONCERN: MINIMIZING ATTACK SURFACE

Adversaries can only attack what’s there

SIMPLICITY

PRINCIPLES OF SECURE SOFTWARE DEVELOPMENT

We *have* to trust some components of our system.
In general keeping the Trusted Computing Base small and simple makes it easier to verify.
In theory a hypervisor can be less complex than a full host operating system.
A small OS kernel has less attack surface than one with many features.

Consider the ease of analysis!

FAIL-OPEN VS FAIL-CLOSED

PRINCIPLES OF SECURE SOFTWARE DEVELOPMENT

WHAT DO YOU DO IF YOUR SECURITY MECHANISM BREAKS DOWN?

FAIL-OPEN

Allow anybody access

FAIL-CLOSED

Allow nobody access

MAINTAINABILITY

PRINCIPLES OF SECURE SOFTWARE DEVELOPMENT

DEVELOP GOOD LOGGING / REPORTING

Ensure the state of the system is easy to ascertain

ASSUME EXTENSIONS TO THE SYSTEM MAY EXPOSE INTERNAL FUNCTIONALITY

- Proactive sanity checking / data sanitization

COMPLETE MEDIATION

PRINCIPLES OF SECURE SOFTWARE DEVELOPMENT

EVERY ACCESS REQUEST SHOULD BE SUBJECT TO THE SAME AUTHORIZATION

Subsequent requests should re-check rights regardless of the success of the first check

Simple way to ensure that updates to state do open a security hole

OPEN DESIGN

PRINCIPLES OF SECURE SOFTWARE DEVELOPMENT

THE SECURITY OF THE SYSTEM SHOULD NOT DEPEND ON AN ADVERSARY'S KNOWLEDGE OF THE SYSTEM

Over-approximate adversarial capabilities

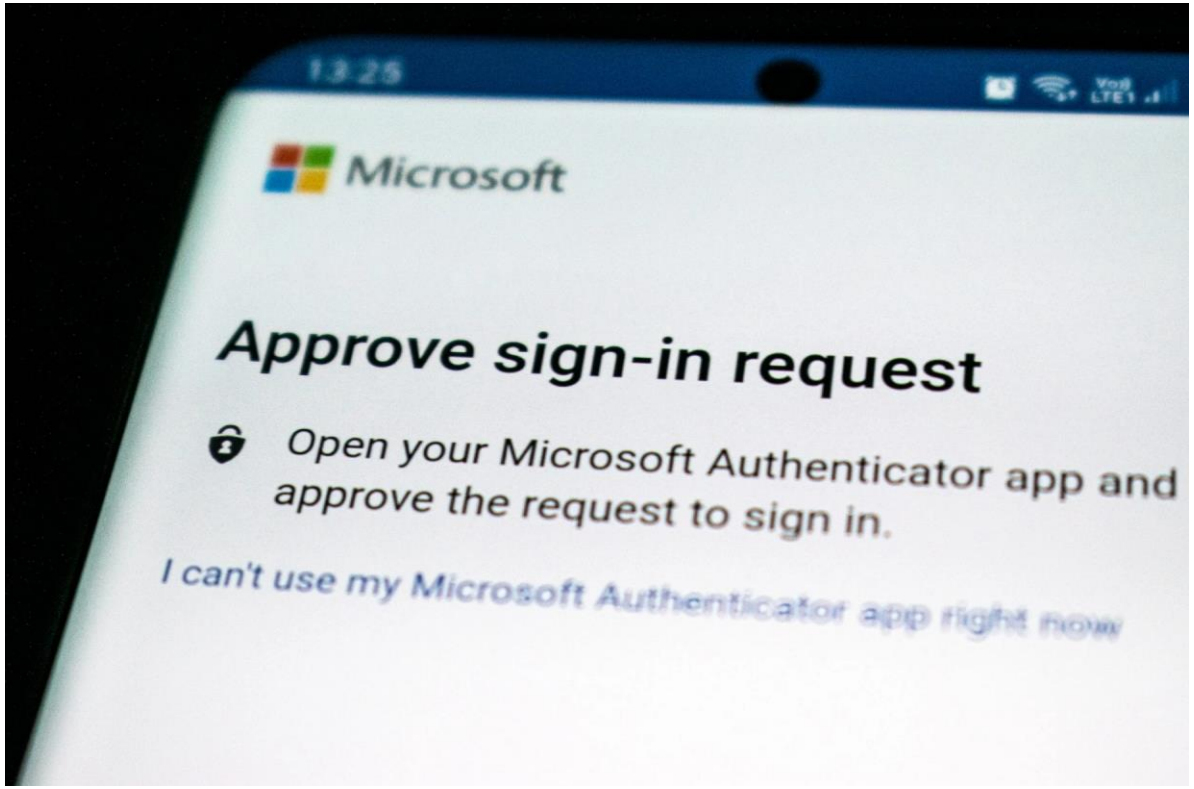
Review of the design should not be considered a security incident

Does not (for better or worse) preclude secret implementation

PSYCHOLOGICAL ACCEPTABILITY

PRINCIPLES OF SECURE SOFTWARE DEVELOPMENT

EASE OF USE AND TRANSPARENCY ARE ESSENTIAL REQUIREMENTS FOR SECURITY



MFA FATIGUE ATTACKS

PRINCIPLES OF SECURE SOFTWARE DEVELOPMENT

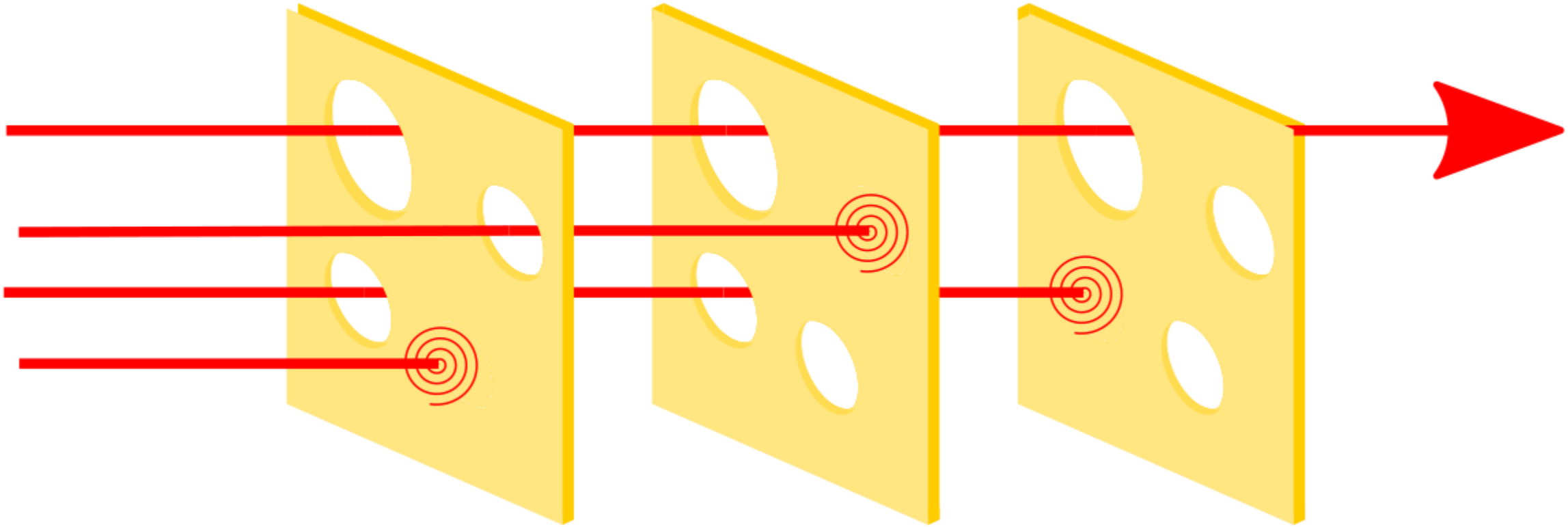
Microsoft Authenticator

Microsoft Corporation

EXAMPLE: 2022 UBER ATTACK BY LAPSUS$

DEFENSE IN DEPTH

PRINCIPLES OF SECURE SOFTWARE DEVELOPMENT



WRAP-UP
SMT SOLVERS

KEEP SECURE DEVELOPMENT IN MIND!

The principles serve as guides and goals to aspire to